АЛЕКСАНДР ЛЕОНЕНКОВ

РЕШЕНИЕ ЗАДАЧ ОПТИМИЗАЦИИ В СРЕДЕ MS EXCEL

Уникальная подборка прикладных задач оптимизации всех классов

Примеры практического решения типовых задач в среде MS Excel

Оригинальные схемы популярных алгоритмов в нотации языка UML

Авторские решения по графическому представлению результатов

Листинги программ на VBA, Delphi и C++, расширяющих функциональность MS Excel



МАСТЕР

Александр Леоненков

Решение задач оптимизации в среде MS Excel

Санкт-Петербург

«БХВ-Петербург»

2005

УДК 681.3.06
ББК 32.973.26-018.2
Л47

Леоненков А. В.

Л47 Решение задач оптимизации в среде MS Excel. — СПб.: БХВ-Петербург, 2005. — 704 с.: ил.

ISBN 5-94157-503-3

Рассматриваются методы и алгоритмы практического решения типовых задач оптимизации всех основных классов. Подробно описываются теоретические основы и практические особенности постановки и решения соответствующих задач. Для типовых задач оптимизации предлагаются несколько способов их решения и приводятся рекомендации по выбору наиболее эффективного из них. Представлены пошаговые инструкции по выполнению практических действий, связанных с подготовкой исходных данных и последующего решения прикладных задач оптимизации всех основных классов. Содержится доступное введение в программирование на языке VBA и приводятся листинги программ, расширяющих функциональность MS Excel.

Для широкого круга пользователей

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. гл. редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Татьяна Лапина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 20.05.05.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 56,76.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-503-3

© Леоненков А. В., 2005
© Оформление, издательство "БХВ-Петербург", 2005

Оглавление

Предисловие **1**

Структура книги 3
Рекомендации по изучению книги 5
Благодарности 5

ЧАСТЬ I. ЗАДАЧИ ОПТИМИЗАЦИИ И ИХ ОСНОВНЫЕ СВОЙСТВА **7**

Глава 1. Общая характеристика задач оптимизации **9**

1.1. Природа и особенности задач оптимизации 9
1.2. Примеры типовых задач оптимизации 14
1.2.1. Задача о коробке максимального объема 14
1.2.2. Задача о пожарном ведре 15
1.2.3. Задача об оптимальной диете 16
1.2.4. Транспортная задача 16
1.2.5. Задача о минимальном пути в графе 18
1.2.6. Задача коммивояжера 19
1.2.7. Задача о рюкзаке 20
1.2.8. Задача о назначении 21
1.2.9. Задача о минимальном покрывающем дереве в графе 23
1.2.10. Задача о максимальном потоке в сети 24
1.2.11. Задача водопроводчика 26
1.3. Методология системного моделирования 27
1.4. Процесс постановки и решения задач оптимизации 32
1.4.1. Анализ проблемной ситуации 33
1.4.2. Построение математической модели 34
1.4.3. Анализ модели 35
1.4.4. Выбор метода и средства решения 36
1.4.5. Выполнение численных расчетов 38

1.4.6. Анализ результатов расчетов ... 38
1.4.7. Применение результатов расчетов ... 39
1.4.8. Коррекция и доработка модели ... 40
1.5. Математическая модель задач оптимизации ... 40
1.5.1. Понятие математической модели и ее основные элементы ... 41
1.5.2. Характеристика переменных ... 41
1.5.3. Характеристика ограничений ... 42
1.5.4. Характеристика целевой функции ... 44
1.5.5. Общая классификация задач оптимизации ... 46
1.6. Основные подходы к решению задач оптимизации ... 48
1.6.1. Понятие оптимального решения задачи оптимизации ... 48
1.6.2. Проблема существования и единственности решения задач оптимизации ... 49
1.6.3. Понятие о методах и алгоритмах решения задач оптимизации ... 53
1.6.4. Структура описания задач оптимизации ... 56

Глава 2. Основные приемы практической работы в среде MS Excel ... 58

2.1. Общая характеристика программы электронных таблиц MS Office Excel 2003 ... 59
2.2. Основные элементы рабочего интерфейса MS Office Excel 2003 ... 60
2.2.1. Главное меню ... 61
2.2.2. Стандартная панель инструментов ... 62
2.2.3. Панель инструментов Форматирование ... 65
2.2.4. Строка ввода и редактирования формул ... 67
2.2.5. Область рабочего листа ... 67
2.2.6. Область задач ... 68
2.3. Основные приемы работы с электронной таблицей ... 69
2.3.1. Ввод и форматирование данных ... 71
2.3.2. Копирование и перенос данных ячеек и рабочих листов ... 75
2.3.3. Ввод, редактирование и копирование формул ... 77
2.4. Основные виды диаграмм в программе MS Excel и приемы их построения ... 87
2.4.1. Построение графика функции одной переменной ... 88
2.4.2. Построение графика функции двух переменных ... 93

ЧАСТЬ II. ЗАДАЧИ НЕПРЕРЫВНОЙ ОПТИМИЗАЦИИ ... 101

Глава 3. Задачи нелинейного программирования ... 103

3.1. Общая характеристика задачи нелинейного программирования ... 103
3.1.1. Математическая постановка задачи нелинейного программирования ... 104

3.1.2. Основные методы решения задач нелинейного программирования ... 105
3.2. Задача о коробке максимального объема ... 107
3.2.1. Математическая постановка задачи о коробке максимального объема ... 107
3.2.2. Решение задачи о коробке максимального объема с помощью программы MS Excel ... 109
3.2.3. Аналитическое решение задачи о коробке максимального объема ... 118
3.3. Задача о пожарном ведре ... 119
3.3.1. Математическая постановка задачи о пожарном ведре ... 119
3.3.2. Решение задачи о пожарном ведре максимального объема с помощью программы MS Excel ... 121
3.3.3. Аналитическое решение задачи о пожарном ведре ... 125
3.4. Задача о строительстве универсама ... 126
3.4.1. Содержательная постановка задачи о строительстве универсама ... 126
3.4.2. Математическая постановка задачи о строительстве универсама ... 127
3.4.3. Решение задачи о строительстве универсама с помощью программы MS Excel ... 128
3.5. Тестовые задачи нелинейного программирования ... 131
3.5.1. Задача оптимизации с целевой функцией Розенброка и ее решение с помощью программы MS Excel ... 131
3.5.2. Задача оптимизации с целевой функцией Пауэлла и ее решение с помощью программы MS Excel ... 134
3.5.3. Задача оптимизации с двумерной экспоненциальной целевой функцией и ее решение с помощью программы MS Excel ... 135
3.6. Упражнения ... 137
3.6.1. Задача Тартальи ... 137
3.6.2. Задача Ферма ... 138
3.6.3. Задача Кеплера ... 138
3.6.4. Обобщенная задача Кеплера ... 138
3.6.5. Задача Евклида ... 138
3.6.6. Обобщенная задача Евклида ... 138
3.6.7. Задача Зенодора ... 138
3.6.8. Задача Архимеда ... 138
3.6.9. Задача Герона ... 139
3.6.10. Задача Аполлония для эллипса ... 139
3.6.11. Задача Аполлония для параболы ... 139
3.6.12. Задача Аполлония для гиперболы ... 139
3.6.13. Задача о вписанном прямоугольнике ... 139

3.6.14. Задача о вписанном треугольнике ... 139
3.6.15. Задача о вписанном конусе ... 139
3.6.16. Задача о вписанном тетраэдре ... 139
3.6.17. Задача о треугольнике ... 139
3.6.18. Задача об угле и точке ... 140
3.6.19. Задача о трех точках ... 140
3.6.20. Обобщенная задача о точках ... 140

Глава 4. Задачи линейного программирования ... 141

4.1. Общая характеристика задачи линейного программирования ... 142
4.1.1. Математическая постановка задачи линейного программирования ... 142
4.1.2. Основные методы решения задач линейного программирования ... 145
4.2. Задача об оптимальной диете ... 146
4.2.1. Математическая постановка задачи об оптимальной диете ... 147
4.2.2. Решение задачи об оптимальной диете с помощью программы MS Excel ... 148
4.3. Задача о производстве красок ... 154
4.3.1. Общая постановка задачи производственного планирования ... 154
4.3.2. Математическая постановка задачи о производстве красок ... 156
4.3.3. Графическое решение задачи о производстве красок ... 157
4.3.4. Решение задачи о производстве красок с помощью симплекс-метода ... 165
4.4. Двойственная задача линейного программирования ... 175
4.4.1. Математическая формулировка двойственной задачи линейного программирования ... 176
4.4.2. Математическая постановка двойственной задачи о красках ... 177
4.4.3. Решение двойственной задачи о красках с помощью программы MS Excel ... 178
4.5. Транспортная задача линейного программирования ... 181
4.5.1. Математическая постановка транспортной задачи ... 181
4.5.2. Решение транспортной задачи с помощью программы MS Excel ... 183
4.5.3. Решение транспортной задачи с помощью метода потенциалов ... 188
4.6. Упражнения ... 199
4.6.1. Задача о производстве клея ... 199
4.6.2. Задача об оптимальной диете ... 200
4.6.3. Транспортная задача ... 201

ЧАСТЬ III. ЗАДАЧИ ДИСКРЕТНОЙ И КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ 203

Глава 5. Задачи целочисленного линейного программирования 205

5.1. Общая постановка задачи целочисленного линейного программирования 205
5.1.1. Математическая постановка задачи целочисленного линейного программирования 206
5.1.2. Основные методы решения задач целочисленного линейного программирования 209
5.2. Задача о рюкзаке 211
5.2.1. Математическая постановка одномерной задачи о рюкзаке 211
5.2.2. Решение одномерной задачи о рюкзаке с помощью программы MS Excel 212
5.2.3. Аналитическое решение одномерной задачи о рюкзаке 219
5.3. Задача об изготовлении часов 221
5.3.1. Математическая постановка задачи об изготовлении часов 221
5.3.2. Графическое решение задачи об изготовлении часов 222
5.4. Задача о планировании перевозок пассажиров 227
5.4.1. Математическая постановка задачи о планировании перевозок пассажиров 228
5.4.2. Решение задачи о планировании перевозок пассажиров с помощью программы MS Excel 229
5.5. Задача об изготовлении стержней 232
5.5.1. Содержательная постановка задачи 232
5.5.2. Математическая постановка задачи об изготовлении стержней 234
5.5.3. Решение задачи об изготовлении стержней с помощью программы MS Excel 235
5.6. Транспортная задача целочисленного линейного программирования 238
5.6.1. Математическая постановка транспортной задачи 239
5.6.2. Решение многопродуктовой целочисленной транспортной задачи с помощью программы MS Excel 240
5.7. Упражнения 247
5.7.1. Задача о погрузке автомобиля 247
5.7.2. Задача об изготовлении обуви 248
5.7.3. Задача об изготовлении мебели 249
5.7.4. Многопродуктовая транспортная задача 249

Глава 6. Задачи оптимизации с булевыми переменными 252

6.1. Общая постановка задачи оптимизации с булевыми переменными 252
6.1.1. Математическая постановка задачи оптимизации с булевыми переменными 253
6.1.2. Основные методы решения задач оптимизации с булевыми переменными 254
6.2. Задача о рюкзаке с булевыми переменными 256
6.2.1. Математическая постановка одномерной задачи о рюкзаке с булевыми переменными 256
6.2.2. Решение одномерной задачи о рюкзаке с булевыми переменными с помощью программы MS Excel 257
6.2.3. Решение задачи о рюкзаке с помощью метода динамического программирования 262
6.3. Задача водопроводчика 266
6.3.1. Математическая постановка задачи водопроводчика 266
6.3.2. Решение задачи водопроводчика с помощью программы MS Excel 268
6.3.3. Аналитическое решение задачи водопроводчика 273
6.4. Задача о назначении 275
6.4.1. Математическая постановка задачи о назначении 275
6.4.2. Решение задачи о назначении с помощью программы MS Excel 277
6.4.3. Решение задачи о назначении с помощью венгерского метода 283
6.5. Упражнения 291
6.5.1. Двумерная задача о рюкзаке 291
6.5.2. Задача водопроводчика 292
6.5.3. Задача о назначении 293

Глава 7. Задачи оптимизации на графах 294

7.1. Общая характеристика задач оптимизации на графах 295
7.1.1. Математическая постановка задачи оптимизации на графах 295
7.1.2. Основные методы решения задач оптимизации на графах 296
7.2. Задача о минимальном покрывающем дереве в графе 297
7.2.1. Математическая постановка задачи 297
7.2.2. Решение задач о минимальном и максимальном покрывающем дереве в графе с помощью программы MS Excel 300
7.2.3. Решение задачи о максимальном покрывающем дереве в графе с помощью программы MS Excel 305
7.2.4. Решение задач о максимальном и минимальном покрывающем дереве с помощью жадного алгоритма 308
7.3. Задача о минимальном пути в графе 313

7.3.1. Математическая постановка задачи ... 313
7.3.2. Решение задачи о минимальном пути в ориентированном графе с помощью программы MS Excel ... 315
7.3.3. Решение задачи о минимальном пути в графе с помощью алгоритма пометок Дейкстры ... 321
7.4. Задача нахождения максимального пути в ориентированном графе ... 328
7.4.1. Содержательная постановка задачи нахождения критического пути выполнения бизнес-процесса ... 328
7.4.2. Математическая постановка задачи ... 331
7.4.3. Решение задачи нахождения критического пути в сетевом графе с помощью программы MS Excel ... 332
7.4.4. Решение задачи нахождения критического пути в сетевом графе с помощью алгоритма расстановки постоянных пометок ... 336
7.5. Задача о максимальном потоке в сети ... 342
7.5.1. Математическая постановка задачи ... 342
7.5.2. Решение задачи о максимальном потоке в сети с помощью программы MS Excel ... 343
7.5.3. Решение задачи о максимальном потоке в сети с помощью алгоритма пометок Форда — Фалкерсона ... 350
7.6. Упражнения ... 356
7.6.1. Задача о минимальном и максимальном покрывающем дереве в графе ... 356
7.6.2. Задача о минимальном и максимальном пути в ориентированном графе ... 356
7.6.3. Задача о максимальном потоке в сети ... 357

Глава 8. Задачи комбинаторной оптимизации ... 358

8.1. Общая характеристика задач комбинаторной оптимизации ... 358
8.1.1. Математическая постановка задачи комбинаторной оптимизации ... 359
8.1.2. Основные методы решения задач комбинаторной оптимизации ... 360
8.2. Задача коммивояжера ... 361
8.2.1. Математическая постановка задачи ... 362
8.2.2. Решение задачи коммивояжера с помощью программы MS Excel ... 365
8.2.3. Решение задачи коммивояжера с помощью алгоритма динамического программирования ... 372
8.3. Задача о разбиении ... 377
8.3.1. Содержательная постановка задачи ... 377
8.3.2. Математическая постановка задачи ... 378

8.3.3. Решение задачи о разбиении с помощью программы MS Excel 380
8.3.4. Решение задачи о разбиении с помощью алгоритма динамического программирования 388
8.4. Упражнения 394
8.4.1. Задача коммивояжера 395
8.4.2. Задача о разбиении 395

ЧАСТЬ IV. ЗАДАЧИ МНОГОКРИТЕРИАЛЬНОЙ ОПТИМИЗАЦИИ 397

Глава 9. Задачи многокритериального линейного и целочисленного программирования 399

9.1. Общая характеристика задач многокритериальной оптимизации 399
9.1.1. Математическая постановка задачи многокритериальной оптимизации 400
9.1.2. Основные подходы и методы решения задач многокритериальной оптимизации 404
9.1.3. Метод уступок для решения задач многокритериальной оптимизации 407
9.1.4. Метод минимального отклонения от идеальной точки 410
9.2. Задача об оптимальной диете с двумя целевыми функциями 412
9.2.1. Математическая постановка задачи и подходы к ее решению 412
9.2.2. Решение многокритериальной задачи об оптимальной диете с помощью программы MS Excel методом уступок 413
9.2.3. Решение двухкритериальной задачи о диете с помощью программы MS Excel методом минимального отклонения 421
9.2.4. Решение двухкритериальной задачи о диете с помощью программы MS Excel методом аддитивной свертки 426
9.3. Задача о производстве красок с двумя целевыми функциями 428
9.3.1. Математическая постановка двухкритериальной задачи о производстве красок 429
9.3.2. Графический способ построения множества Парето для двухкритериальной задачи о производстве красок 430
9.4. Двухкритериальная задача о рюкзаке 436
9.4.1. Математическая постановка двухкритериальной задачи о рюкзаке 436
9.4.2. Решение двухкритериальной задачи о рюкзаке с помощью программы MS Excel методом уступок 437
9.4.3. Решение двухкритериальной задачи о рюкзаке с помощью программы MS Excel методом минимального отклонения 443

9.4.4. Решение двухкритериальной задачи о рюкзаке с помощью программы MS Excel методом аддитивной свертки 447
9.5. Упражнения 449
9.5.1. Двухкритериальная задача о производстве клея 450
9.5.2. Двухкритериальная задача о погрузке автомобиля 450
9.5.3. Двухкритериальная задача об изготовлении обуви 451

Глава 10. Задачи многокритериальной булевой оптимизации 452

10.1. Общая характеристика задач многокритериальной оптимизации с булевыми переменными 452
10.2. Задача водопроводчика с двумя целевыми функциями 453
10.2.1. Математическая постановка двухкритериальной задачи водопроводчика 454
10.2.2. Решение двухкритериальной задачи водопроводчика с помощью программы MS Excel методом уступок 455
10.2.3. Решение двухкритериальной задачи водопроводчика с помощью программы MS Excel методом минимального отклонения 461
10.2.4. Решение двухкритериальной задачи водопроводчика с помощью программы MS Excel методом аддитивной свертки 465
10.3. Двухкритериальная задача о назначении 468
10.3.1. Математическая постановка двухкритериальной задачи о назначении 468
10.3.2. Решение двухкритериальной задачи о назначении с помощью программы MS Excel методом уступок 470
10.3.3. Решение двухкритериальной задачи о назначении с помощью программы MS Excel методом минимального отклонения 477
10.3.4. Решение двухкритериальной задачи о назначении с помощью программы MS Excel методом аддитивной свертки 481
10.4. Двухкритериальная задача о наборе высоты и скорости 484
10.4.1. Содержательная постановка индивидуальной задачи о наборе высоты и скорости летательным аппаратом 484
10.4.2. Математическая постановка двухкритериальной задачи о наборе высоты и скорости 485
10.4.3. Решение двухкритериальной задачи о наборе высоты и скорости с помощью программы MS Excel методом уступок 487
10.4.4. Решение двухкритериальной задачи о наборе высоты и скорости с помощью программы MS Excel методом минимального отклонения 494
10.4.5. Решение двухкритериальной задачи о наборе высоты и скорости с помощью программы MS Excel методом аддитивной свертки 499

10.5. Упражнения ... 502
10.5.1. Двухкритериальная задача о рюкзаке ... 502
10.5.2. Двухкритериальная задача водопроводчика ... 503
10.5.3. Двухкритериальная задача о назначении ... 504

ЧАСТЬ V. ПРОГРАММИРОВАНИЕ ЗАДАЧ ОПТИМИЗАЦИИ В СРЕДЕ EXCEL ... 505

Глава 11. Алгоритмы и программы решения задач оптимизации на графах ... 507

11.1. Особенности разработки пользовательских программ в среде MS Excel ... 508
11.1.1. Среда и язык программирования Visual Basic For Applications ... 508
11.1.2. Создание пользовательской функции для вычисления двумерной экспоненциальной функции ... 517
11.1.3. Построение графика функции двух переменных ... 520
11.1.4. Программа изображения структуры неориентированного графа ... 523
11.2. Минимальное покрывающее дерево графа и его графическое изображение ... 529
11.2.1. Программа нахождения минимального покрывающего дерева графа ... 530
11.2.2. Программа изображения минимального покрывающего дерева графа ... 534
11.3. Максимальное покрывающее дерево графа и его графическое изображение ... 539
11.3.1. Программа нахождения максимального покрывающего дерева графа ... 539
11.3.2. Программа изображения максимального покрывающего дерева графа ... 542
11.4. Путь минимальной длины и его графическое изображение ... 546
11.4.1. Программа нахождения минимального пути в ориентированном графе ... 546
11.4.2. Программа изображения минимального пути в ориентированном графе ... 551
11.5. Путь максимальной длины и его графическое изображение ... 556
11.5.1. Программа нахождения критического пути в сетевом графе ... 556
11.5.2. Программа изображения критического пути в сетевом графе ... 559
11.6. Упражнения ... 565
11.6.1. Максимальный поток в сети ... 565
11.6.2. Графическое изображение максимального потока в сети ... 565

Глава 12. Алгоритмы и программы решения задач комбинаторной оптимизации ... 566

12.1. Задача коммивояжера и ее решение с помощью VBA ... 566
12.1.1. Алгоритм приближенного решения задачи коммивояжера ... 567
12.1.2. Программа приближенного решения задачи коммивояжера ... 569
12.1.3. Программа изображения полного замкнутого пути в ориентированном графе ... 575
12.2. Задача о разбиении и ее решение с помощью VBA ... 580
12.2.1. Алгоритм приближенного решения задачи о разбиении ... 580
12.2.2. Программа приближенного решения задачи о разбиении ... 584
12.3. Использование программ на языке VBA в книгах MS Excel ... 591
12.3.1. Экспорт и импорт модулей с текстами программ на VBA ... 591
12.3.2. Использование шаблонов с текстами программ на VBA ... 592
12.3.3. Создание и использование надстроек пользователя с текстами программ на VBA ... 594
12.4. Внешние программы и их использование в среде MS Excel ... 595
12.4.1. Разработка внешней функции в среде Borland Delphi и ее использование в среде MS Excel ... 596
12.4.2. Разработка внешней функции в среде MS Visual Studio .NET и ее использование в среде MS Excel ... 603
12.4.3. Разработка функции нахождения минимального пути в среде Borland Delphi и ее использование в среде MS Excel ... 611
12.5. Упражнения ... 618
12.5.1. Модификация программы приближенного решения задачи коммивояжера ... 618
12.5.2. Модификация программы приближенного решения задачи о разбиении ... 618
12.5.3. Разработка программы нахождения максимального покрывающего дерева ... 619
12.5.4. Разработка программы нахождения критического пути ... 619

ПРИЛОЖЕНИЯ ... 621

Приложение 1. Основные понятия теории множеств, теории графов и комбинаторного анализа ... 623

Множество и способы его задания ... 623
Основные теоретико-множественные операции ... 629
Булеан или множество всех подмножеств ... 636
Отношения и способы их задания ... 637

Операции над бинарными отношениями 647
Отображение 649
Свойства бинарных отношений, заданных на одном базисном множестве..... 650
Некоторые специальные виды бинарных отношений, заданных на одном базисном множестве 652
Отношение строгого частичного порядка 653
Отношение толерантности 653
Отношение эквивалентности 654
Перестановка 655
Сочетание..... 656
Размещение..... 656

Приложение 2. Назначение операций главного меню программы электронных таблиц MS Office Excel 2003 658

Приложение 3. Назначение операций главного меню редактора Visual Basic пакета MS Office System 2003..... 671

Список литературы..... 683

Предметный указатель..... 689

Предисловие

Книга посвящена рассмотрению основ теории и практики решения задач оптимизации в среде электронных таблиц MS Excel®. Чем же обусловлена необходимость изучения задач оптимизации, разработки методов и программ их решения? Насколько это актуально при ведении современного бизнеса, когда зачастую проблемы планирования и внутрифирменного управления отступают на второй план по сравнению с юридическими и административными?

С одной стороны, можно было бы упомянуть известное утверждение о том, что вся история человечества неразрывно связана с явным или неявным решением задач оптимизации. С другой стороны, часто отмечают и бытовой контекст принятия оптимальных решений. Действительно, кто из читателей хотя бы раз не задавал себе вопроса о том, как с максимальной для себя пользой потратить некую сумму имеющуюся в его распоряжении? Или как побывать в нескольких точках города, затратив на это минимальное время? Наконец, существуют прагматический интерес изучения данной тематики, когда студентам нужно подготовить и сдать соответствующую курсовую работу или домашнее задание.

При составлении плана оптимального использования семейного бюджета на текущий месяц большинство из нас поступают на основе здравого смысла или в лучшем случае ограничиваются листом бумаги и ручкой. В несколько более сложных ситуациях может потребоваться калькулятор. Однако уже при планировании работы предприятия мелкого или среднего бизнеса калькулятора оказывается недостаточно. Требуется что-то еще, о чем современные менеджеры стараются не задумываться, поскольку для многих из них оперативность принятия решения оказывается важнее точности оптимального решения. Как в этой связи не вспомнить известный совет: "Всегда довольствуйтесь третьим по качеству решением, ибо второе приходит слишком поздно, а лучшее не приходит никогда".

Тем самым суть проблемы смещается в иную плоскость — если имеются средства выполнения быстрых расчетов для получения оптимальных решений,

то почему бы ими не воспользоваться? Ведь опыт развития современной экономики неизменно свидетельствует, что умение эффективно расходовать имеющиеся ресурсы является одним из решающих факторов конкурентных преимуществ в любой сфере бизнеса. И в этом кроется один из важнейших потенциалов выживания современных фирм и компаний.

В дополнение к этому существуют задачи оптимизации, связанные с проектированием новых технических устройств типа самолетов, морских судов, электростанций, где наряду с новыми конструкторскими и технологическими идеями присутствуют расчеты. Решением возникающих в этой области задач заняты специалисты: инженеры и математики, которые используют для этой цели либо специализированные математические программы типа MATLAB® и MathCAD®, либо пишут собственные, ориентированные на узкий класс задач.

У читателей, возможно, сложилось впечатление, что для решения математических задач лучше всего использовать именно специализированные математические программы. И с этим трудно не согласиться. Однако когда речь идет о решении задач оптимизации, то ситуация представляется не столь однозначной. С одной стороны, среди подобных задач существуют такие, которые не могут быть напрямую решены специализированными программами типа MATLAB и Mathcad. Среди них программы, которые не содержат функций решения задач комбинаторной оптимизации и задач оптимизации на графах. И как это ни покажется парадоксальным, электронные таблицы MS Excel здесь оказываются более предпочтительными, поскольку обладают возможностью постановки и решения таких задач.

С другой стороны, офисный пакет MS System Office® стал общепризнанным стандартом для подготовки деловой документации. Этот пакет устанавливается на компьютер пользователя сразу после установки операционной системы от Microsoft. Тем самым программа электронных таблиц MS Excel всегда оказывается под рукой, так же, как и электронный калькулятор, функции которого эта программа часто с успехом выполняет в работе обычных пользователей. Рабочие листы MS Excel легко встраиваются в другие офисные документы, что позволяет разрабатывать сложные комплекты электронной документации. Пользователю нет необходимости не только приобретать специализированную математическую программу, но и затрачивать время на ее изучение. Что же касается знания MS Excel, то это представляется само собой разумеющимся для всех, кто пишет в своем резюме: "Опытный пользователь ПК".

Наконец, имеющиеся возможности MS Excel могут быть расширены за счет использования встроенного в офисный пакет языка программирования VBA. И здесь снова пользователь избавлен от необходимости приобретать специализированную среду программирования и осваивать порядок работы с ней.

Достаточно изучить один раз язык VBA и применять его для всех офисных программ. Впрочем, для тех, кто уже владеет языками программирования и соответствующими средами Borland Delphi® и MS Visual C++®, электронные таблицы MS Excel тоже идут навстречу, предоставляя им возможность вызова внешних функций из библиотек динамической компоновки, разработанных самими пользователями.

Есть и еще один аспект проблемы, который не только программисты, но и многие математики часто упускают. Речь идет о том, что появление современных высокопроизводительных персональных компьютеров создает обманчивую иллюзию потенциальной возможности решения с их помощью всех без исключения вычислительных задач. И снова задачи оптимизации вносят коррекцию в этот тезис, поскольку точное решение некоторых из них за приемлемое время оказывается невозможным не только для современных компьютеров, но и тех, которые могут появиться в обозримом будущем. Именно поэтому поиск новых методов и разработка алгоритмов решения отдельных задач оптимизации продолжает оставаться актуальным направлением не только прикладной математики, но и программирования.

Цель книги — помочь читателям не только овладеть приемами и способами практического решения задач оптимизации в среде MS Excel, но и освоить основы технологии разработки математических моделей, которые могут быть использованы для самостоятельного решения соответствующих задач.

Материал книги рассчитан на широкую аудиторию читателей. Так, например, студенты при выполнении курсовых работ и домашних заданий могут ограничиться изучением отдельных глав первых трех частей книги. Математики и программисты в дополнение к этому найдут для себя информацию к размышлению в последних двух частях книги. Инженеров и бизнес-аналитиков, возможно, заинтересуют некоторые вычислительные аспекты точного и приближенного решения отдельных классов задач оптимизации. Преподаватели смогут использовать изложенный материал при проведении практических занятий по соответствующим учебным дисциплинам.

Структура книги

В основу книги положены две основные идеи. С одной стороны, рассмотреть базовые возможности электронных таблиц MS Excel, которые могут быть эффективно использованы для нахождения решения типовых задач оптимизации. С другой стороны, донести до читателя основы методологии построения и анализа математических моделей поиска оптимальных решений, без понимания которой вряд ли возможно адекватно самостоятельно решать задачи оптимизации самых различных классов.

Материал книги делится на пять частей.

I часть знакомит с типовыми задачами оптимизации и основными теоретическими понятиями, которые необходимы в процессе постановки и решения этих задач. Здесь также приводится описание рабочего интерфейса и базовые приемы практической работы в среде MS Excel.

II часть посвящена рассмотрению задач непрерывной оптимизации, в рамках которых рассматриваются задачи нелинейного и линейного программирования. Данные задачи оптимизации традиционно составляют базовый курс инженерного, экономического и математического образования.

III часть содержит описание особенностей задач дискретной и комбинаторной оптимизации, которым значительно меньше уделяют внимания при обучении студентов, но которые не менее часто встречаются на практике. При этом алгоритмы решения соответствующих задач оптимизации могут служить основой для формирования так называемого алгоритмического мышления, которое представляется непременным атрибутом современных программистов и системных аналитиков.

IV часть содержит описание особенностей постановки и решения задач многокритериальной оптимизации. Эти задачи часто встречаются в практике современного бизнеса, однако традиционно считаются сложными для адекватной постановки и решения. Здесь приводится описание подходов и способов решения, которые продолжают оставаться недостаточно известными.

V часть ориентирована на читателей, знакомых с программированием, и содержит описание возможностей расширения функциональности программы MS Excel в контексте решения задач оптимизации. Здесь приводятся листинги программ на языках VBA, Pascal Delphi и MS Visual C++, предназначенные для решения задач комбинаторной оптимизации и задач оптимизации на графах. Использование данных программ позволяет избежать рутинных операций при подготовке исходных данных большого объема в процессе решения реальных задач оптимизации.

В приложениях содержится вспомогательный материал, необходимый для понимания формальных особенностей постановки и решения задач оптимизации. В *приложении 1* рассматриваются основные понятия теории множеств, теории графов и комбинаторного анализа, которые используются в различных главах книги. В *приложении 2* описывается назначение операций главного меню программы электронных таблиц MS Office Excel 2003, являющейся последней версией на момент написания книги. В *приложении 3* приводится назначение операций главного меню редактора Visual Basic® пакета MS Office System 2003.

Рекомендации по изучению книги

Книга ориентирована на специалистов различной квалификации. Основной материал предназначен для системных аналитиков, бизнес-аналитиков, экономистов и инженеров, которым вовсе не требуется знание языков программирования для решения представленных в книге задач оптимизации. Материал *V части* может рассматриваться для них как дополнительный, поскольку ориентирован на программистов, которые заняты в проектах, связанных с офисным программированием. Приводимые в тексте описания различных методов и схем алгоритмов решения задач оптимизации отдельных классов предназначены для математиков, у которых нет абсолютной веры в количественные результаты, получаемые с помощью программы MS Excel. В этом контексте уместно заметить, что квалификация математика пропорциональна количеству методов, которыми он может решить ту или иную задачу. И задачи оптимизации не являются исключением.

Наконец, студенты и преподаватели найдут для себя достаточно много интересных задач оптимизации, которые помогут скрасить однообразие математических формул занимательным характером содержательных постановок. Именно по этой причине в книге отсутствуют формальные постановки задач оптимизации без содержательной интерпретации исходных данных. При желании такие задачи могут быть найдены в дополнительной литературе, список которой приведен в конце книги.

Для понимания основного материала книги, связанного с решением типовых задач оптимизации, достаточно общей эрудиции и некоторого знакомства с программой MS Excel. Однако для творческого овладения методологией решения задач оптимизации необходимо знание особенностей процесса математического моделирования и одного из языков программирования. Соответствующий теоретический материал приводится в *приложении 1* и используется в тексте при изложении конкретных методов и алгоритмов.

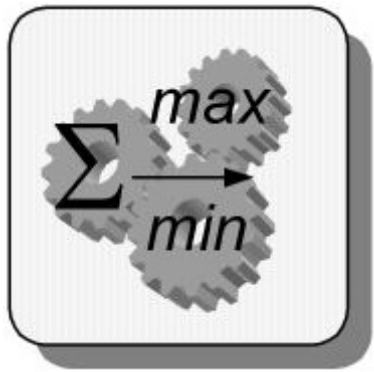
Благодарности

Автор искренне благодарит Е. В. Кондукову, Е. В. Строганову, А. М. Коновалова и В. Л. Натанзона за предоставленные в разное время материалы, которые были использованы при написании книги. Автор искренне признателен директору Школы IT-менеджмента АНХ при Правительстве РФ (**www.itmane.ru**) А. И. Соколову, а также Л. А. Ермакову, В. А. Перекрестову и В. В. Фамильнову за оказанную поддержку в процессе работы над книгой.

Написание современной книги немыслимо без использования ресурсов Интернета. В этой связи хотелось бы выразить особую признательность директору Междисциплинарного Центра СПбГУ (**www.icape.nw.ru**) профессору Н. В. Борисову за предоставленную возможность электронной коммуникации.

Хотя все результаты количественных расчетов были неоднократно проверены, значительный объем изложенного материала не позволяет дать абсолютных гарантий отсутствия погрешностей и ошибок. В связи с этим автор и редакция не несут ответственности за возможный ущерб, связанный с конкретным применением изложенных в книге методик и рекомендаций. В то же время автор будет признателен за все отзывы и конструктивные предложения, связанные с содержанием книги и проблематикой решения задач оптимизации, которые можно отправлять по адресу: **alex@lngs.lukoil.com** или в редакцию издательства БХВ-Петербург.

Все названия программных продуктов и конкретных технологий, которые упоминаются в книге, являются зарегистрированными торговыми марками соответствующих компаний и фирм.



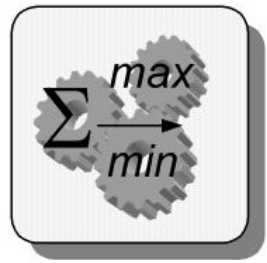
Часть I

Задачи оптимизации и их основные свойства

Глава 1. Общая характеристика задач оптимизации

Глава 2. Основные приемы практической работы в среде MS Excel

Принято считать, что с той или иной задачей оптимизации каждый человек хотя бы однажды сталкивался в своей жизни. Если эта встреча имела явный характер, то, по всей видимости, приходилось выполнять некоторые численные расчеты. Если же проблема возникала в неявной форме, то и решение принималось интуитивно, на основе здравого смысла или опыта. При этом часто по прошествии времени оказывалось, что принятое решение приводит вовсе не к тому результату, какой хотелось бы получить. Как избежать разочарований, а то и печальных последствий от далеко не лучших вариантов развития событий, предусмотреть, по возможности, все аспекты и альтернативы, выработать систематический взгляд на способы решения задач оптимизации — обо всем этом рассказывается на страницах этой книги.



Глава 1

Общая характеристика задач оптимизации

Настоящая глава является введением в проблематику постановки и решения задач оптимизации. В ней рассматриваются и особенности природы подобных задач, приводятся примеры типовых задач оптимизации, которые являются предметом детального изучения в следующих главах книги. Далее в этой главе дается краткая характеристика основных этапов процесса постановки и решения задач оптимизации и приводится общая классификация этих задач. В заключение представлены основные подходы к решению задач оптимизации, которые нашли отражение в книге.

1.1. Природа и особенности задач оптимизации

Начиная разговор о задачах оптимизации, обычно всегда упоминают об исключительно широком распространении этих задач, а также о том, что их история восходит к зарождению человеческой цивилизации. Действительно, трудно найти человека, который хотя бы раз не попадал в ситуацию необходимости выбора одной из нескольких возможностей. Вполне очевидно, что в подобной ситуации всегда следует выбирать наилучший вариант. Более сложный вопрос заключается в точном определении того, какой смысл следует вкладывать в понятие "наилучшее решение".

Как оптимально распорядиться семейным бюджетом или за минимальное время добраться до нужной точки в городе, как наилучшим образом спланировать деловые встречи и минимизировать риски капитальных вложений, есть ли смысл воспользоваться страховым полисом или это неоправданные финансовые затраты, как наиболее эффективно организовать работу персонала компании или определить оптимальные запасы сырья на складе — это лишь

небольшая часть проблем, в которых желательно принять не просто какое-то решение, а наилучшее из всех потенциально возможных решений.

В более простых, на первый взгляд, ситуациях, когда, например, речь заходит о покупке новой вещи в быту, вряд ли у кого не возникает желание иметь вещь наилучшего качества, надежности, внешнего вида или комфорта. Однако, с одной стороны, в подобных случаях выбор может быть ограничен имеющимися в наличии доступными средствами. Тем самым на принятие решения оказывают влияние некоторые ограничивающие обстоятельства, которые из всех потенциально возможных вариантов исключают не удовлетворяющие определенным ограничениям. С другой стороны, введение в рассмотрение нескольких характеристик для оценки наилучшего варианта приводит к задачам оптимизации в многокритериальной постановке, которые с концептуальной точки зрения считаются наиболее трудными для решения.

Что касается истории систематического изучения задач оптимизации, то ее начало относится к эпохе зарождения математики как науки. Известно, что одними из первых математических задач, которые были сформулированы и решены античными математиками, были задачи нахождения геометрических фигур максимальной площади и тел максимального объема при ограничениях на периметр и площадь поверхности, соответственно.

Задачи, в которых требуется найти значения одной или нескольких переменных и при этом максимизируется или минимизируется значение некоторой непрерывной функции, занимают в математике особое место. Для их решения был разработан классический метод, связанный с нахождением нулей первой производной целевой функции и проверки их на экстремум. Этот метод, получивший название *аналитического* способа решения задач оптимизации, вполне подходит для решения простых задач. Его характерной особенностью является установление или нахождение решения в форме некоторой функциональной зависимости между исходными данными и решением.

Интенсивное развитие экономики и производства в XX в. привело к появлению целого ряда новых типов задач оптимизации, которые не могли быть решены классическими методами и потребовали разработки специальной математической теории — теории решения задач оптимизации. В рамках этой теории были разработаны не только методологические основы постановки и анализа задач оптимизации, но и вычислительные методы и алгоритмы их решения. Хотя по-прежнему некоторые неклассические задачи оптимизации могут быть решены аналитически, однако акцент исследований смещается в сторону поиска *вычислительного* способа решения для целых классов задач оптимизации.

Таким образом, для задач оптимизации характерно наличие нескольких возможностей для выбора альтернативных решений и наличие некоторой

оценочной функции для количественного выражения наилучшего выбора. В случае отсутствия альтернатив или соответствующей оценочной функции задача оптимизации теряет свой смысл. Что касается наличия некоторых дополнительных, ограничивающих выбор условий, то последние не являются обязательным атрибутом классических задач оптимизации. Однако большинство, если не все неклассические задачи оптимизации имеют дополнительные ограничения, формирующие в своей совокупности множество допустимых альтернатив.

Несколько слов о самом подходе к решению задач оптимизации. Ранее уже были упомянуты аналитический и вычислительный способы. Однако из всех методов решения задач оптимизации с конечным множеством допустимых альтернатив имеется один, который представляется универсальным средством решения задач любого класса. Речь идет о методе *полного перебора* и сравнения альтернативных возможностей с целью выбора наилучшего из них. Появление высокопроизводительных персональных компьютеров создает у многих пользователей ложное впечатление о потенциальной возможности решения любой задачи оптимизации простым перебором всех возможностей для выбора из них наилучшей.

Казалось бы, для этого вполне достаточно написать соответствующую компьютерную программу, которая будет последовательно рассчитывать значения оценочной функции для каждой из возможных альтернатив, сохранять текущий рекорд, а после окончания своей работы сообщит сохраненный результат пользователю. Этот способ представляется настолько многообещающим, что для значительной части современных программистов был потерян сам смысл изучения других вычислительных алгоритмов, разработанных для решения задач оптимизации.

Увы, это впечатление настолько обманчиво, что даже не требует серьезных аргументов для опровержения. Достаточно взять реальную задачу комбинаторной оптимизации, например, задачу коммивояжера с 11—12 городами, чтобы на многие часы, а может быть и сутки, ввести Pentium-IV в состояние глубочайшей задумчивости со 100%-ной загрузкой процессора. А ведь в реальных ситуациях может потребоваться решить подобную задачу с гораздо большим количеством городов для обхода. Данный класс задач и эффективные методы их решения рассматриваются в *главах 8* и *12*. Применение метода полного перебора для непрерывных задач оптимизации вообще теряет смысл, поскольку в подобных задачах множество исходных альтернатив бесконечно по своей природе.

Другая распространенная ошибка среди пользователей и аналитиков — считать какие-нибудь новомодные методы, например, генетические или нейросетевые алгоритмы, универсальным средством решения задач оптимизации.

Об этом можно прочитать в рекламных проспектах релизов тех или иных программ. Однако сознательно или нет, разработчики соответствующих программных пакетов не указывают, что эти алгоритмы, действительно обладая известной универсальностью, по своему характеру являются всего лишь *приближенными* методами поиска оптимальных решений. Это значит, что в общем случае нет никаких оснований считать полученное с их помощью решение действительно наилучшим из всех возможных. Полученное решение будет всего лишь *близким* к оптимальному, насколько же близко — это уже другой вопрос, точный ответ на который соизмерим со сложностью решения исходной задачи оптимизации.

Наконец, несколько слов следует сказать о средствах решения задач оптимизации. В простейших случаях для этого может оказаться достаточным наличие ручки и листа бумаги. А вот в более сложных… Здесь уже у аналитика появляется выбор, который зависит от наличия персонального компьютера. Действительно, существуют специальные математические программы, например, MATLAB и Mathcad, которые имеют встроенные средства решения задач оптимизации. Очевидное неудобство для пользователя может быть связано с необходимостью дополнительных затрат на их приобретение и времени на изучение их особенностей. А если ни того, ни другого в данный момент нет?

Оказывается, в большинстве случаев для решения типовых задач оптимизации может оказать эффективную помощь программа электронных таблиц MS Excel. Неоспоримое ее достоинство заключается в том, что она не требует дополнительных затрат на приобретение, поскольку офисный пакет от Microsoft практически всегда инсталлируется сразу после установки операционной системы MS Windows®. Что касается времени на изучение программы электронных таблиц MS Excel, то оно существенно меньше за счет единообразного рабочего интерфейса офисных программ. С другой стороны, от большинства пользователей, которые пишут в своем резюме "опытный пользователь ПК", требуется умение работать в среде MS Excel на уровне редактирования рабочих листов и манипулирования содержимым ячеек. Значит, изучение материала книги для таких читателей послужит приобретению дополнительных навыков практической работы с одной из самых "математических" программ офисного пакета.

Примечание

Как это ни покажется парадоксальным, использование для решения задач оптимизация других технических средств — калькулятора или ставшей уже реликтом логарифмической линейки — может потребовать от аналитика более серьезных усилий и квалификации. Действительно, чтобы правильно выполнить все расчетные действия с помощью этих инструментов, необходимо

хорошо знать детали того или иного вычислительного алгоритма. Учитывая высокую вероятность ошибки пользователя при выполнении промежуточных расчетов, а также невысокую точность второго инструмента, оба эти средства вряд ли могут считаться базовыми при решении задач оптимизации. В то же время нельзя их полностью исключить из рабочего арсенала аналитика, поскольку они по-прежнему используются для ручной проверки правильности вычислительных алгоритмов в простейших случаях.

Конечно, программа электронных таблиц MS Excel, содержащая встроенные средства для решения задач оптимизации, является далеко не универсальной. Современные задачи оптимизации настолько разнообразны по своему характеру, что разработка универсального метода или средства для их решения представляется практически невозможной. Однако существуют типовые классы задач оптимизации, которые могут быть успешно решены с помощью программы электронных таблиц MS Excel. Именно рассмотрению этих задач и особенностей методов их решения уделяется основное внимание на страницах книги.

В то же время отдельные классы интересных задач оптимизации либо не могут быть вовсе решены с помощью программы MS Excel, либо их решение связано со значительным неудобством и затратами времени. В подобных случаях пользователь или аналитик оказывается в ситуации выбора. С одной стороны, обладая некоторым опытом программирования, можно воспользоваться встроенным языком VBA (Visual Basic for Applications) для написания собственных программ, реализующих те или иные алгоритмы оптимизации.

С другой стороны, электронные таблицы MS Excel позволяют вызывать пользовательские функции из внешних библиотек динамической компоновки (DDL), которые, в свою очередь, могут быть созданы с помощью специальных сред визуального программирования, таких как MS Visual C++.NET® и Borland Delphi 7® и их предыдущих версий. Подобные задачи оптимизации, методы и алгоритмы их программного решения рассматриваются в заключительной части книги.

Примечание

На всем протяжении книги под программой электронных таблиц MS Excel будет подразумеваться последняя на момент написания книги локализованная версия MS Office Excel 2003. В то же время, практически весь изложенный материал оказывается справедливым для предыдущих версий этой программы, начиная с версии MS Excel 97. Более того, как это ни покажется парадоксальным, возможностью вызова пользовательских функций из библиотек динамической компоновки реально обладает именно версия MS Excel 97. Поскольку эта возможность используется только в заключительной части книги, на указа-

ние версии программы электронных таблиц MS Excel в этом случае будет акцентировано дополнительное внимание.

Таким образом, для правильной постановки и решения практических задач оптимизации необходимо знать особенности отдельных классов этих задач и возможности программы электронных таблиц MS Excel. Далее рассматриваются примеры типовых задач оптимизации, которые послужили прообразом появления целых классов задач оптимизации и источником разработки специальных алгоритмов для их решения.

1.2. Примеры типовых задач оптимизации

Понятие типовой задачи оптимизации является в некоторой степени условным и определяется исключительно соображениями удобства анализа и классификации соответствующих задач. Из всего разнообразия задач оптимизации и их модификаций были выбраны только те, которые имеют наглядный характер и не требуют для понимания своих особенностей специальных знаний из той или иной предметной области. В следующих главах книги рассматриваются математические формулировки этих задач и методы их практического решения.

1.2.1. Задача о коробке максимального объема

Данная задача формулируется следующим образом. Имеется квадратная заготовка из некоторого гибкого материала, например, картона или жести, причем размеры этой заготовки фиксированы для конкретной ситуации (рис. 1.1, *а*). Из этой заготовки следует вырезать четыре равных квадрата по ее углам, а полученную фигуру (рис. 1.1, *б*) согнуть так, чтобы получилась коробка без верхней крышки (рис. 1.1, *в*). При этом необходимо так выбрать размер вырезаемых квадратов, чтобы получилась коробка максимального объема.

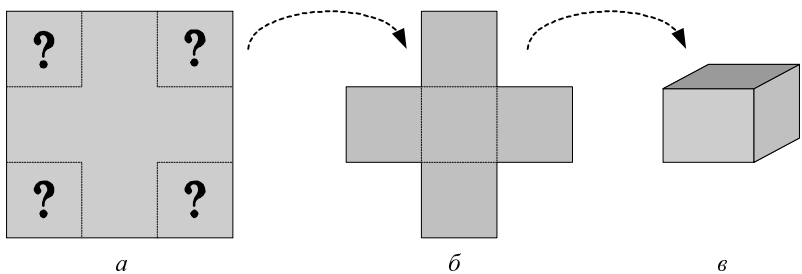


Рис. 1.1. Схема изготовления коробки из прямоугольной заготовки фиксированного размера

На примере данной задачи можно проиллюстрировать все элементы постановки задач оптимизации. Оценочной функцией в данной задаче служит объем изготовленной коробки. Проблема выбора заключается в выборе размера вырезаемых квадратов. Действительно, если размер вырезаемых квадратов будет слишком мал, то будет получена широкая коробка малой высоты, а значит, и ее объем окажется невелик. С другой стороны, если размер вырезаемых квадратов будет слишком большой, то будет получена узкая коробка большой высоты, а значит, и ее объем также окажется невелик.

В то же время на выбор размера вырезаемых квадратов оказывает влияние ограничение размера исходной заготовки. Действительно, если вырезать квадраты со стороной, равной половине стороны исходной заготовки, то задача теряет смысл. Сторона вырезаемых квадратов также не может превышать половину стороны исходной заготовки, поскольку это невозможно из практических соображений. Из этого следует, что в постановке данной задачи должны присутствовать некоторые ограничения.

Задача о коробке максимального объема имеет не только наглядную интерпретацию, но и достаточно простое аналитическое решение. Она относится к классу задач нелинейной оптимизации, методы решения которых подробно рассматриваются в *главе 3*.

1.2.2. Задача о пожарном ведре

Данная задача по своему характеру похожа на предыдущую задачу о коробке и формулируется следующим образом. Имеется заготовка в форме круга из некоторого гибкого и влагостойкого материала, например, жести или алюминия, причем размеры заготовки фиксированы для конкретной ситуации (рис. 1.2, *а*). Из заготовки следует вырезать некоторый сектор так, чтобы, согнув полученную фигуру (рис. 1.2, *б*) и сварив шов, можно было бы получить конус без основания. Если к этому конусу добавить ручку из согнутого стержня или проволоки, то получится ведро, которое вполне можно использовать для тушения пожара (рис. 1.2, *в*), откуда и произошло название задачи. Необходимо так выбрать размер вырезаемого сектора, чтобы пожарное ведро оказалось максимального объема.

На примере данной задачи также можно проиллюстрировать все элементы постановки задач оптимизации. Оценочная функция и ограничивающие условия оказываются аналогичными предыдущей задаче. Хотя данная задача имеет наглядную интерпретацию, однако ее аналитическое решение уже не является простым. Она относится к классу задач нелинейной оптимизации и более подробно рассматривается в *главе 3*.

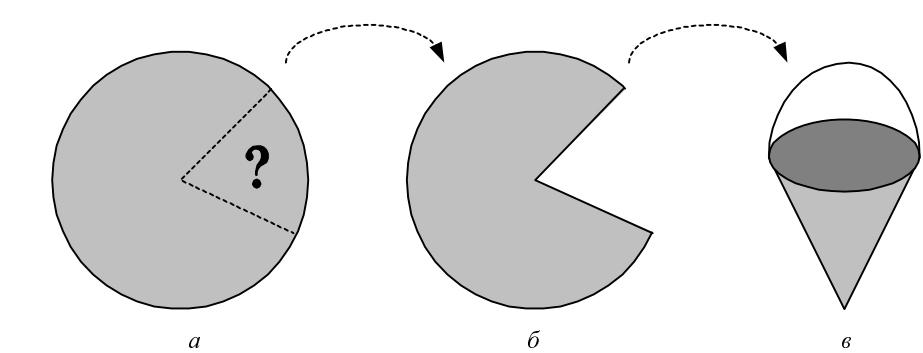


Рис. 1.2. Схема изготовления ведра из круглой заготовки фиксированного размера

1.2.3. Задача об оптимальной диете

Данная задача не имеет столь наглядного характера, как предыдущие, и допускает некоторую неоднозначность в своей постановке. Однако это не влияет на особенности ее анализа и решения. В общем случае задача об оптимальной диете формулируется следующим образом.

Имеется конечное число видов продуктов питания, например, хлеб, крупа, мясо, рыба, в которых содержится конечное число типов питательных веществ, например, белки, жиры, углеводы, витамины А, Б, С. В каждом виде продуктов питания содержится известное количество питательных веществ каждого из типов. Задана минимальная суточная потребность человека, например, спортсмена или пациента больницы, в каждом из видов питательных веществ. Задана также калорийность каждого типа продукта.

Требуется определить такой состав рациона питания, чтобы каждое питательное вещество содержалось в нем в необходимом количестве, обеспечивающем суточную потребность человека, а при этом суммарная калорийность рациона была минимальной.

Данная задача известна в литературе также как задача об оптимальной смеси или рационе питания. Оценочной функцией в ней является суммарная калорийность рациона, а ограничениями служат минимальная суточная потребность человека в каждом из видов питательных веществ. Задача об оптимальной диете является одной из классических задач линейного программирования, методы решения которых подробно рассматриваются в *главе 4*.

1.2.4. Транспортная задача

В некотором географическом регионе имеется фиксированное число пунктов производства и хранения некоторого однородного продукта и конечное число

пунктов потребления этого продукта (рис. 1.3). В качестве продукта может выступать, например, нефть, уголь, песок, цемент и т. п. Для каждого из пунктов производства и хранения известен объем производства продукта или его запаса. Для каждого пункта потребления задана потребность в продукте в этом пункте потребления. Известна стоимость перевозки или транспортировки одной единицы продукта из каждого пункта производства в любой из пунктов потребления.

Требуется определить оптимальный план перевозок продукта, так чтобы потребности во всех пунктах потребления были удовлетворены, а суммарные затраты на транспортировку всей продукции были минимальными.

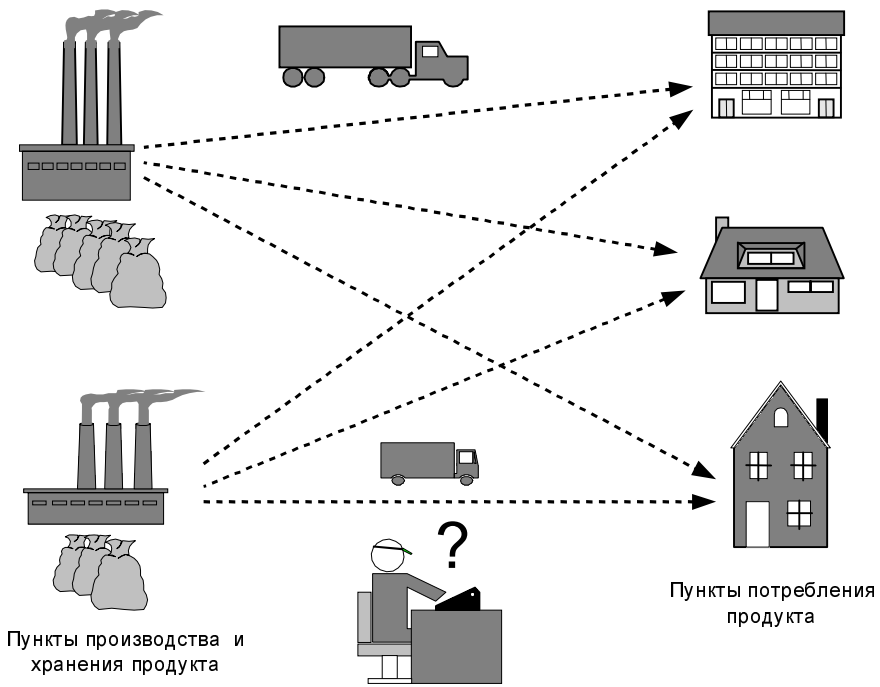


Рис. 1.3. Иллюстрация транспортной задачи для двух пунктов производства и трех пунктов потребления

Очевидно, оценочной функцией в данной задаче являются суммарные затраты на транспортировку всей продукции, а ограничениями служат объемы производства продукта в каждом пункте производства и потребности в продукте в каждом пункте потребления.

Данная задача также является одной из классических задач линейного программирования, методы ее решения рассматриваются в *главе 4*. В бизнес-

приложениях эта задача известна как задача о перемещении товаров со складов на торговые точки или задача о планировании цепочек поставок. В случае штучного товара, например, телевизоры, компьютеры, пылесосы, автомобили и пр., соответствующая транспортная задача относится к классу задач целочисленного программирования, методы решения которых рассматриваются в *главе 5*.

1.2.5. Задача о минимальном пути в графе

Данная задача в чем-то похожа на предыдущую задачу, поскольку в ней тоже идет речь о перемещении в некотором географическом регионе, однако суть ее иная. Водителю автомобиля необходимо проехать из некоторого исходного населенного пункта *А* в конечный населенный пункт *Б*, используя магистральные шоссейные дороги (рис. 1.4). Каким образом выбрать кратчайший маршрут перемещения, минуя различные промежуточные пункты?

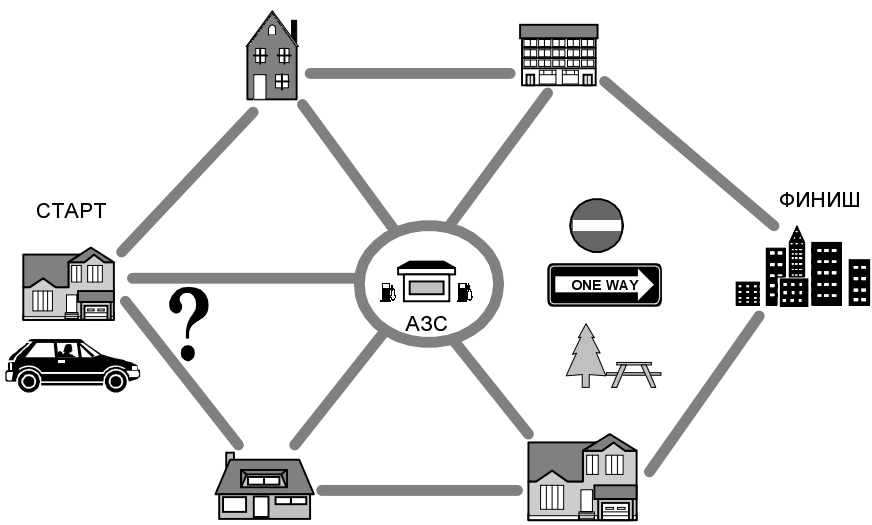


Рис. 1.4. Иллюстрация задачи о минимальном пути в графе

Воспользуемся атласом автомобильных дорог для соответствующего географического района. Очевидно, каждый участок дороги в этом случае будет соответствовать наличию в атласе автомобильной дороги между двумя соседними населенными пунктами или перекрестками. Из атласа можно узнать длину соответствующих участков, соединяющих промежуточные населенные пункты или перекрестки.

Задача поиска кратчайшего маршрута в этом случае формулируется следующим образом. Из всех возможных маршрутов перемещения, связывающих

исходный населенный пункт с конечным, необходимо найти такой маршрут, суммарная длина участков дорог которого минимальна.

Оценочной функцией в данной задаче является суммарная длина пути, связывающего пункты *А* и *Б*, а ограничениями служат наличие или отсутствие дорог между отдельными населенными пунктами соответствующего географического региона.

Данная задача в общем случае может быть сформулирована как задача о минимальном пути в графе. Для соответствующего географического района может быть построен граф, вершины которого интерпретируются как точки пересечения дорог или населенные пункты. Каждое ребро графа в этом случае будет соответствовать наличию автомобильной дороги между двумя соседними населенными пунктами или перекрестками. В качестве веса ребра следует принять длину соответствующего участка дороги в *км*.

Таким образом, задача поиска кратчайшего маршрута формулируется как задача нахождения минимального пути в графе между двумя фиксированными парами вершин. Данная задача относится к классу задач оптимизации на графах, методы решения которых рассматриваются в *главе 7*.

1.2.6. Задача коммивояжера

Данная задача очень похожа по своей постановке на предыдущую задачу, однако по своей сложности принципиально от нее отличается. Суть проблемы заключается в следующем.

Бродячему торговцу или коммивояжеру необходимо обойти фиксированное число городов, начиная с города, в котором он находится, и закончить свой маршрут, вернувшись в исходный город, не побывав нигде дважды. Пусть для определенности это будут города: *Москва*, *Париж*, *Нью-Йорк*, *Рио-де-Жанейро*, *Сидней* и *Токио* (рис. 1.5). Известна стоимость перелета между любой парой указанных городов. При этом вовсе не очевидно, что наиболее короткий маршрут будет обладать минимальной стоимостью перелета, которая, в свою очередь, реально зависит от авиакомпании, типа рейса и авиалайнера.

Требуется определить маршрут или последовательность посещения городов, которые обладают минимальной суммарной стоимостью перелета среди всех возможных маршрутов.

Оценочной функцией в данной задаче является суммарная длина полного пути, начинающегося и оканчивающегося в некотором городе *А* (например, в Москве), а ограничениями служат наличие или отсутствие авиарейса между отдельными городами рассматриваемого списка, а также необходимость посещения всех этих городов.

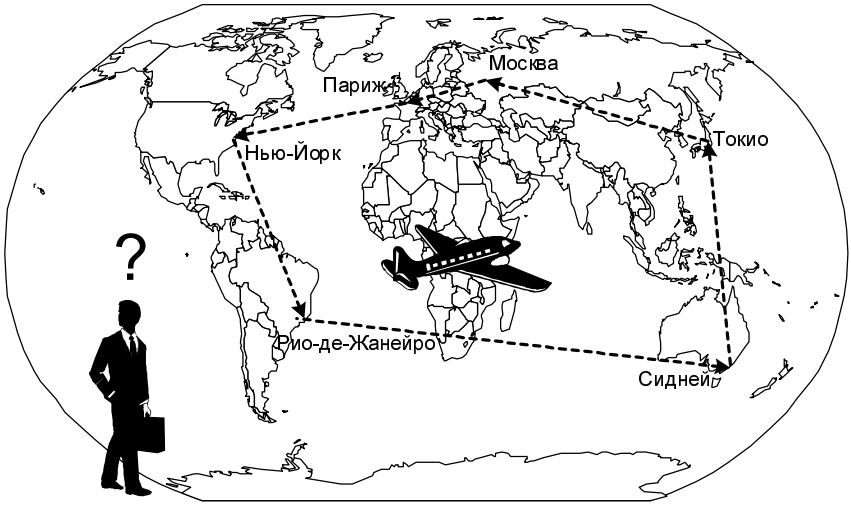


Рис. 1.5. Иллюстрация задачи коммивояжера

Примечание

Задача коммивояжера (TSP — travelling salesman problem) представляет собой пример классической задачи комбинаторной оптимизации, которая выглядит очень простой по своей постановке, но требует самых серьезных усилий для нахождения точного решения. В бизнес-приложениях встречается также другой вариант этой задачи, известный как задача о переналадке оборудования или задача составления оптимального плана.

Хотя в общем случае задача коммивояжера может быть сформулирована как задача нахождения замкнутого или незамкнутого контура (пути, проходящего через все вершины) минимальной длины в графе, однако ряд особенностей позволяют отнести ее к классу задач комбинаторной оптимизации. Для решения задач этого класса разработаны специальные методы, которые рассматриваются в *главе 9*.

1.2.7. Задача о рюкзаке

Следующая задача покажется знакомой всем, кто хоть однажды собирался в дальнее путешествие с рюкзаком за плечами. Итак, турист готовится к длительному переходу в горах или в лесу. В рюкзаке он может нести груз, масса которого не должна превышать некоторого фиксированного значения. Сам по себе груз может включать конечное число видов предметов, например, пакеты с продуктами, баллоны с газом для горелки и пр. Для каждого вида

предмета известна его масса, например, в *кг*. При этом каждый вид предмета имеет для туриста определенную эвристическую или объективную ценность на все время перехода.

Сколько предметов каждого вида турист должен положить в рюкзак, чтобы суммарная ценность предметов была максимальной, а их общая масса не превысила допустимого фиксированного значения?

Очевидно, что оценочной функцией в данной задаче является суммарная ценность предметов, а ограничением служит требование, чтобы общая масса выбираемых для похода предметов не превысила допустимого фиксированного значения.

Примечание

В бизнес-приложениях встречаются многочисленные варианты этой задачи, известные как задача о складировании товаров, задача о погрузке транспортного судна, трейлера или самолета. Хотя содержание подобных задач может существенно отличаться друг от друга, все они являются типовой задачей о рюкзаке. Профессионализм аналитика заключается в том, чтобы за внешним различием содержания подобных задач правильно выполнить их классификацию. Это необходимо для выбора наиболее эффективного метода решения задач соответствующего класса.

В исходной постановке задача о рюкзаке относится к классу задач целочисленного линейного программирования (дискретной оптимизации), методы решения которых рассматриваются в *главе 5*. Если же в постановке задачи каждый вид предмета присутствует в единственном экземпляре, то это условие изменяет не только характер задачи, но и ее сложность. Соответствующая задача о рюкзаке перейдет в разряд задач булева программирования, методы решения которых рассматриваются в *главе 6*.

1.2.8. Задача о назначении

Хотя данная задача по своему содержанию мало похожа на предыдущую задачу, однако обе они относятся к одному и тому же классу.

Имеется конечное число видов работ, которые могут быть выполнены потенциальными кандидатами. При этом каждого кандидата можно назначить на выполнение только одной работы, а каждая работа, в свою очередь, должна выполняться только одним кандидатом (рис. 1.6). Известна эффективность выполнения каждой работы любым из потенциальных кандидатов. Требуется распределить всех кандидатов по работам, так чтобы общая эффективность выполнения всех работ была наибольшей.

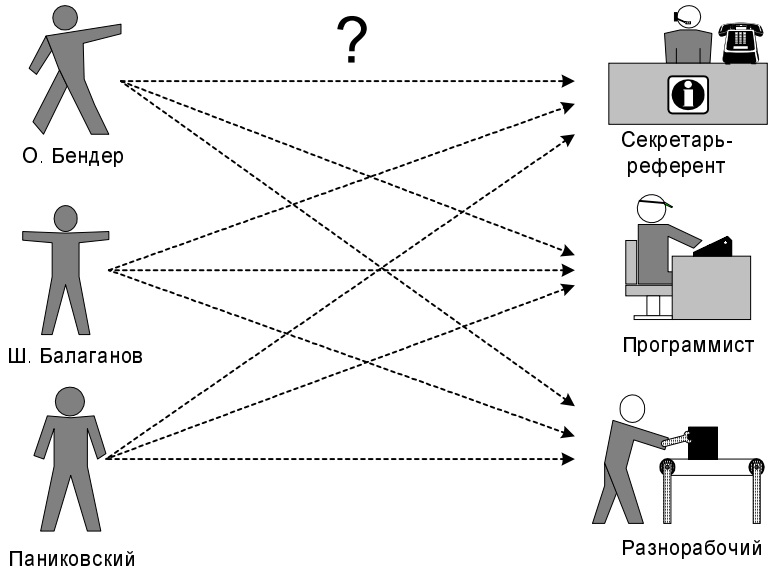


Рис. 1.6. Иллюстрация задачи о назначении

Оценочной функцией в данной задаче является общая эффективность выполнения всех работ, а ограничениями служат дополнительные условия на выполнение каждой работы только одним кандидатом и участие каждого кандидата в выполнении только одной работы.

Примечание

В бизнес-приложениях данная задача встречается в форме задачи по подбору персонала на замещение должностей в компании или фирме. В литературе данная задача встречается также в форме поиска оптимальных сочетаний пар для заключения браков. Хотя принято считать, что браки заключаются на небесах, в земных условиях существуют фирмы по оказанию соответствующих услуг населению. Впрочем, автор не берется утверждать, что в своей практике они используют методы решения рассматриваемой задачи оптимизации. Хорошо это или плохо — судить читателю.

Классическая задача о назначении — это *симметричная* задача, когда общее число видов работ равно количеству потенциальных кандидатов. В противном случае задача о назначениях называется *несимметричной*. Оба варианта этой задачи относятся к классу задач булева программирования, методы решения которых рассматриваются в *главе 6*.

1.2.9. Задача о минимальном покрывающем дереве в графе

Данная задача служит характерным примером задач, которые имеют не совсем очевидную постановку, но довольно просто решаются даже в случае большого количества исходных данных.

Один из возможных вариантов этой задачи может быть сформулирован следующим образом. Необходимо разработать проект транспортной сети, которая должна соединить конечное число населенных пунктов в некотором географическом районе (рис. 1.7). Известна стоимость прокладки автодороги между двумя соседними населенными пунктами. Из экономических соображений требуется, чтобы общая стоимость реализации проекта была минимальной, при этом должно быть выполнено обязательное условие — из любого населенного пункта по построенным автодорогам можно было бы попасть в любой другой населенный пункт рассматриваемого географического района.

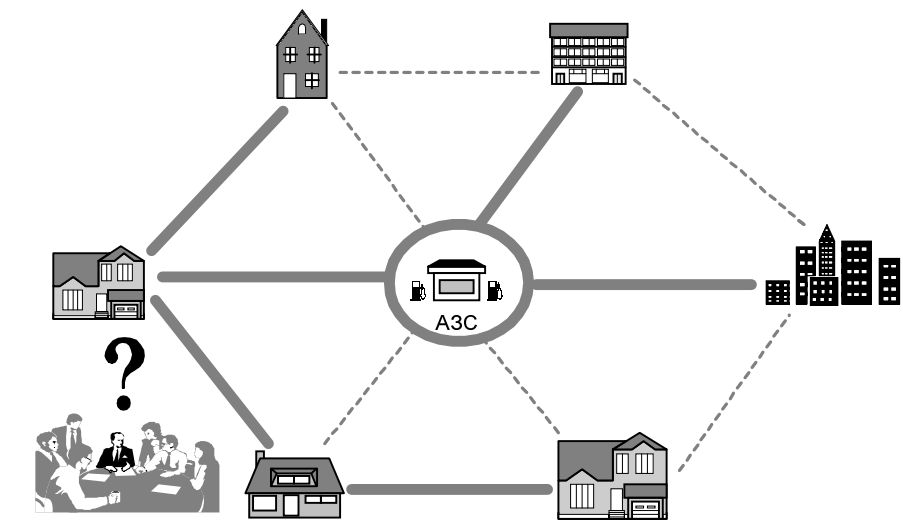


Рис. 1.7. Иллюстрация задачи о минимальном покрывающем дереве

Оценочной функцией в данной задаче является общая стоимость реализации проекта, а ограничением служит требование достижимости из произвольного населенного пункта любого другого.

В бизнес-приложениях встречается и другой вариант постановки данной задачи. Суть его заключается в следующем. Провайдеру телекоммуникационных услуг необходимо разработать проект кабельной сети связи, которая

должна соединить конечное число абонентов в некотором географическом районе. Считается, что два абонента соединены между собой, если существует транзитный канал связи между ними. Известна стоимость прокладки канала связи между двумя любыми абонентами. Требуется определить вариант построения сети связи с минимальной проектной стоимостью прокладки кабелей. Нетрудно увидеть, что за различным содержанием этих двух задач скрывается один и тот же математический смысл, который становится очевидным при переходе к обобщенному рассмотрению этих задач в форме задачи о минимальном покрывающем дереве в графе.

Задача о минимальном покрывающем дереве в графе в общем случае формулируется следующим образом. Для соответствующего географического района строится граф, вершины которого интерпретируются как населенные пункты этого района или абоненты связи соответствующих двух частных задач. Каждое ребро графа в этом случае будет соответствовать потенциально возможной автомобильной дороге между двумя соседними населенными пунктами или прокладке кабеля между двумя абонентами. Вес ребра примем равным стоимости прокладки соответствующего участка дороги или кабеля. Таким образом, задача разработки оптимального проекта транспортной сети и задача проектирования кабельной сети в общем случае соответствуют задаче нахождения покрывающего дерева в графе минимальной общей стоимости.

Задача о минимальном покрывающем дереве в графе легко преобразуется к варианту задачи о максимальном покрывающем дереве в графе, при этом характер задачи и ее сложность не изменяются. В литературе она также известна как задача о минимальном или максимальном остовном дереве. Все эти варианты данной задачи относится к классу задач оптимизации на графах, методы решения которых рассматриваются в *главе 7*.

1.2.10. Задача о максимальном потоке в сети

Данная задача имеет множество возможных вариантов постановки, один из которых может быть сформулирован следующим образом. Имеется система магистральных трубопроводов, связывающих источник добычи нефти или газа с предприятием по его промышленной переработке (рис. 1.8). Отдельные участки трубопроводов оснащены компрессорными установками для поддержания требуемого давления, необходимого для транспортировки продукта. Известны предельные значения пропускной способности каждого участка рассматриваемой системы. В предположении, что источник обладает достаточными запасами продукта, требуется определить количество транспортируемого продукта по каждому из участков трубопроводной системы, так чтобы количество доставленного на предприятие переработки продукта было максимальным.

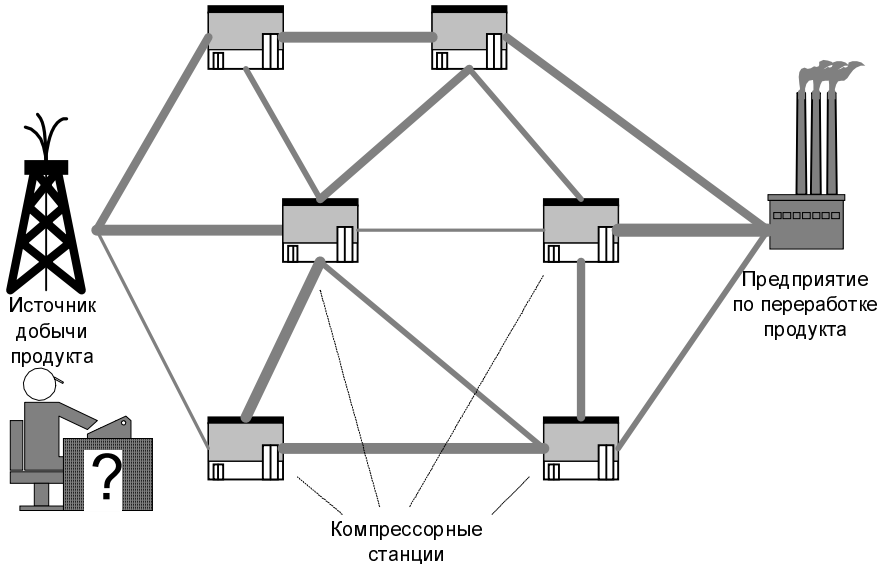


Рис. 1.8. Иллюстрация задачи о максимальном потоке в сети

Оценочной функцией в данной задаче является количество продукта, доставленного на предприятие переработки, а ограничениями служат предельные значения пропускной способности каждого участка рассматриваемой системы.

Примечание

В бизнес-приложениях встречаются самые разнообразные варианты данной задачи, например, нахождение максимального пассажиропотока в транспортной системе мегаполиса, регулирование транспортных потоков на автомобильных магистралях и пр. Для наиболее эффективного анализа и решения подобных задач, их следует рассматривать в обобщенной постановке в форме задачи о максимальном потоке в сетевом графе или сети.

Чтобы в общем случае сформулировать задачу о максимальном потоке в сети, для рассматриваемой системы магистральных трубопроводов построим граф. В данном графе одна из вершин, называемая *источником*, будет соответствовать источнику добычи продукта, другая, называемая *стоком*, — предприятию по переработке этого продукта. Остальные вершины графа интерпретируются как компрессорные станции. Каждое ребро графа в этом случае будет соответствовать наличию участка трубопровода между парой компрессорных станций. Вес ребра равен пропускной способности соответствующего участка трубопровода.

Таким образом, задача о максимальном потоке в сети формулируется как задача нахождения переменных значений величин потока по каждому ребру графа, которые не превышают весов соответствующих ребер и максимизируют общий поток от источника к стоку. Данная задача также относится к классу задач оптимизации на графах, методы решения которых излагаются в *главе 7*.

1.2.11. Задача водопроводчика

Данная задача достаточно часто встречается при производстве дорожно-строительных и ремонтных работ, ее называют также задачей временного удаления плит с поверхности грунта. Сущность задачи водопроводчика заключается в следующем.

Водопроводчик получил наряд на установку вентильных перекрытий или шлюзовых затворов на нескольких водопроводных трубах, проложенных под землей и покрытых тяжелыми бетонными плитами прямоугольной или квадратной формы. Имеется конкретный план прокладки труб под землей в пределах района производства работ (рис. 1.9). Согласно технологическим правилам выполнения работ допускается установление вентильного перекрытия

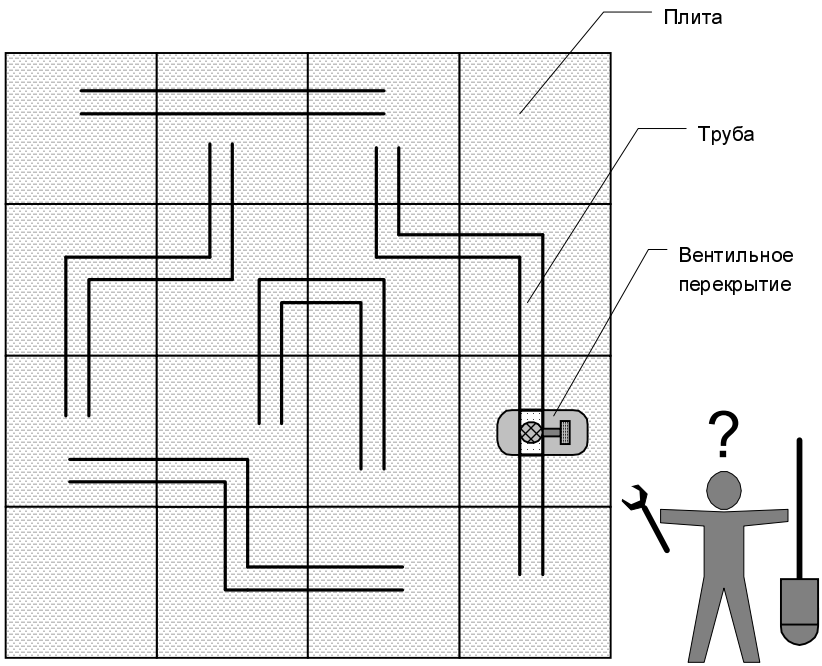


Рис. 1.9. Геометрическая интерпретация задачи водопроводчика

в любом месте трубы, но лишь по одному шлюзовому затвору на каждой трубе. С целью минимизации трудозатрат водопроводчику необходимо определить минимальное число плит, которые требуется приподнять, чтобы установить по одному вентильному перекрытию на каждой трубе.

По своей постановке задача водопроводчика может быть отнесена к классу задач *геометрического программирования*. Многие задачи этого класса, после некоторых предварительных преобразований, могут быть сформулированы и эффективно решены с помощью модели булева программирования. Методы решения излагаются в *главе 6*.

После рассмотрения содержания типовых задач оптимизации следует дать краткую характеристику базовой методологии, которая определяет общие принципы и подходы к анализу и решению данных задач. Незнание этой методологии или ее неверная интерпретация, к сожалению, приводят к многочисленным ошибкам и выбору неадекватных средств при решении практических задач оптимизации.

1.3. Методология системного моделирования

Общей методологией постановки и решения задач оптимизации является системный анализ. Применительно к решению прикладных задач эта методология получила название *системного моделирования*.

Центральным понятием системного моделирования является собственно понятие *системы*, под которым понимается совокупность объектов, компонентов или элементов произвольной природы, образующих некоторую целостность в том или ином контексте. Определяющим принципом рассмотрения некоторой совокупности объектов как системы является появление у нее новых свойств, которых не имеют составляющие ее элементы. Этот принцип получил специальное название — принцип *эмерджентности* (от *англ.* emergence — появление, выявление).

Системы различной физической природы окружают нас повсеместно — это конкретные предметы и объекты: солнечная система, человек, персональный компьютер, автомобиль, самолет, аэропорт. Характерным признаком системного мышления является рассмотрение абстрактных сущностей, таких как алгоритм, компьютерная программа, естественный язык, коммерческая фирма, культура, политика, наука, экономика как система.

Примечание

Наиболее ортодоксальная точка зрения заключается в том, что все окружающие нас предметы и категории мышления являются системами. Не вдаваясь

в философские аспекты данной проблемы, которая далека от своего удовлетворительного решения, в рамках системного моделирования понятие системы ограничено возможностью представления реальных объектов в форме некоторой модели. При этом концептуальная сложность проблемы смещается в область модельных представлений, базовым или главным из которых является математическая модель.

При рассмотрении той или иной системы исходным этапом при построении ее модели является определение ее *границы*. Речь идет о необходимости разделения всех элементов на два класса: принадлежащих и не принадлежащих системе. При этом те сущности или объекты, которые собственно принадлежат системе, и будут являться ее элементами. Напротив, не принадлежащие системе объекты, но оказывающие на нее то или иное влияние, образуют *среду* или внешнюю по отношению к системе предметную область (рис. 1.10).

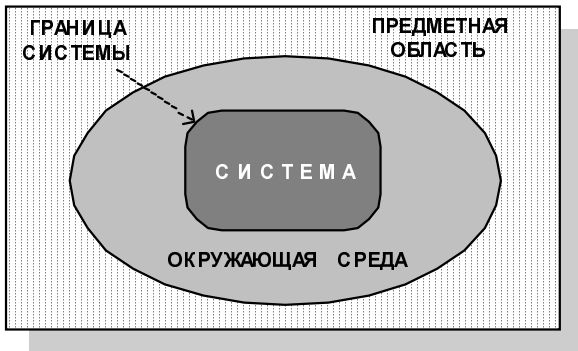


Рис. 1.10. Общее представление системы и окружающей среды в контексте традиционного системного анализа

Важнейшими характеристиками любой системы являются ее структура и процесс функционирования. Под *структурой системы* понимают устойчивую во времени совокупность взаимосвязей между ее элементами или компонентами. Именно структура связывает воедино все элементы и препятствует распаду системы на отдельные компоненты. Структура системы может отражать самые различные взаимосвязи, в том числе, и вложенность элементов одной системы в другую. В этом случае принято называть более мелкую или вложенную систему *подсистемой*, а более крупную систему — *метасистемой*.

Процесс функционирования тесно связан с изменением свойств системы или отдельных ее элементов во времени. При этом важной характеристикой системы является ее *состояние*, под которым понимается совокупность свойств или признаков, которые в каждый момент времени отражают наиболее существенные особенности поведения системы.

Структура системы может быть описана с разных точек зрения. Наиболее общее представление о структуре дает схема устройства той или иной системы. При этом взаимодействие элементов может носить не только механический, электрический или биологический характер, но и информационный, что характерно для современных бизнес-систем. Состояние системы также можно рассматривать с различных точек зрения, наиболее общей из которых является рассмотрение особенностей функционирования или эксплуатации той или иной системы.

Процесс функционирования системы отражает поведение системы во времени и может быть представлен как последовательное изменение ее состояний. При изменении состояния системы говорят о том, что система *переходит* из одного состояния в другое. Совокупность признаков или условий изменения состояний системы в этом случае называется *переходом*. Для системы с дискретными состояниями процесс функционирования может быть представлен в виде последовательности состояний с соответствующими переходами.

Методология системного моделирования служит концептуальной основой системно-ориентированной структуризации предметной области. В этом случае исходными компонентами концептуализации являются системы и взаимосвязи между ними. Результатом системного моделирования является построение некоторой модели системы и соответствующей предметной области, которая описывает важнейшие с точки зрения решаемой проблемы аспекты системы.

В общем случае под *моделью* понимается некоторое представление о системе, отражающее наиболее существенные закономерности ее структуры и процесса функционирования и зафиксированное на некотором языке или в некоторой форме. Применительно к контексту задач оптимизации имеют интерес только такие аспекты построения моделей, которые связаны с информационным или логическим моделированием систем.

Примерами моделей являются не только известные физические модели (аэродинамическая модель гоночного автомобиля или проектируемого самолета), но и абстрактные или логические модели различных систем (математическая модель колебательной системы, аналитическая модель системы электроснабжения региона, информационная модель избирательной компании и другие).

Общим свойством всех моделей является их подобие некоторому реальному объекту или системе-оригиналу. Важность построения моделей заключается в возможности их использования для получения информации о свойствах структуры или поведении системы-оригинала. При этом сам процесс построения и последующего применения моделей для получения информации о системе-оригинале является основным содержанием системного моделирования.

Наиболее общей информационной моделью системы является так называемая модель "черного ящика". В этом случае система представляется в виде прямоугольника, внутреннее устройство которого скрыто от системного аналитика или вообще неизвестно. Однако система не является полностью изолированной от внешней среды, поскольку последняя оказывает на систему некоторые информационные или материальные воздействия. Такие воздействия получили название *входных воздействий* или *входных параметров*, *входных переменных*.

Среди входных воздействий выделяют специальный класс — так называемых *управляющих воздействий (переменных)*. Последние предназначены для того, чтобы оказывать на систему целенаправленное воздействие, предназначенное для достижения системой некоторой цели (целей) или желаемого поведения. В свою очередь система также оказывает на среду или другие системы определенные информационные или материальные воздействия, которые получили название *выходных воздействий (параметров, переменных)*. Графически данная модель может быть изображена следующим образом (рис. 1.11).

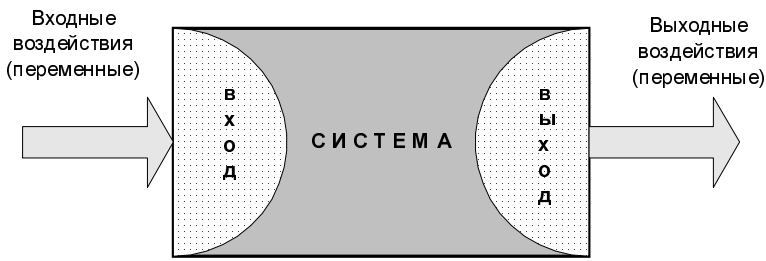


Рис. 1.11. Графическое изображение модели системы в виде "черного ящика"

Ценность моделей, подобных модели "черного ящика", весьма условна. Основное ее назначение состоит в том, чтобы структурировать исходную информацию относительно самой системы и внешней по отношению к ней среды. Поэтому эта модель, прежде всего, фиксирует упоминавшиеся ранее границы системы. В дополнение к этому модель специфицирует воздействия, на которые реагирует система, и как проявляется эта реакция на окружающие объекты и системы. При этом, в случае количественного описания входных (выходных) воздействий, их иногда называют входными (выходными) переменными. В рамках системного моделирования разработаны определенные методологические средства, позволяющие выполнить дальнейшую структуризацию или концептуализацию этой наиболее общей модели системы.

В методологии системного моделирования выделяются сложные системы, исследование которых представляет наибольший интерес в контексте постановки и решения задач оптимизации. При этом *сложность* системы и, соответственно, ее модели может быть рассмотрена с различных точек зрения. Прежде всего, можно выделить сложность структуры системы, которая характеризуется большим количеством элементов и различными типами взаимосвязей между ними.

Так, например, если количество элементов системы превышает некоторое пороговое значение, которое, вообще говоря, не является строго фиксированным, то такая система может быть названа сложной. Например, если программная система управления базой данных насчитывает более 100 отдельных форм ввода и вывода информации, то многие программисты сочтут ее сложной. Если исходные данные некоторой задачи оптимизации содержат несколько сотен переменных и ограничений, то есть все основания считать подобную задачу и соответствующую ей систему сложной. Транспортные и энергетические системы современных мегаполисов, макроэкономика государства или отдельных отраслей также могут служить примерами сложных систем, состоящих из десятков и сотен отдельных подсистем или элементов с нетривиальной структурой взаимосвязей между ними.

Вторым аспектом сложности является сложность процесса функционирования системы или отдельных ее подсистем. Это может определяться как непредсказуемым характером поведения системы, так и невозможностью формального представления правил преобразования входных воздействий в выходные. Этот важный аспект сложности системы может быть связан с наличием неопределенности в описании процесса поведения системы-оригинала.

Так, например, процесс поведения участников некоторого рынка товаров или услуг в определенной степени непредсказуем или характеризуется неопределенностью состояний своих элементов. Процесс функционирования современных операционных систем также характеризуется сложностью поведения, поскольку их надежность и безопасность не всегда удовлетворяют требованиям различных категорий пользователей.

При анализе структуры и поведения сложных систем, как правило, присутствуют различные факторы неопределенности, которые могут быть учтены и адекватно представлены в процессе построения информационно-логических моделей в рамках нового направления системного моделирования — нечеткого моделирования. Однако анализ и решение задач оптимизации с неопределенностью в настоящей книге не рассматриваются.

1.4. Процесс постановки и решения задач оптимизации

В общем случае процесс постановки и решения задач оптимизации может быть представлен в форме взаимосвязанных этапов, на каждом из которых

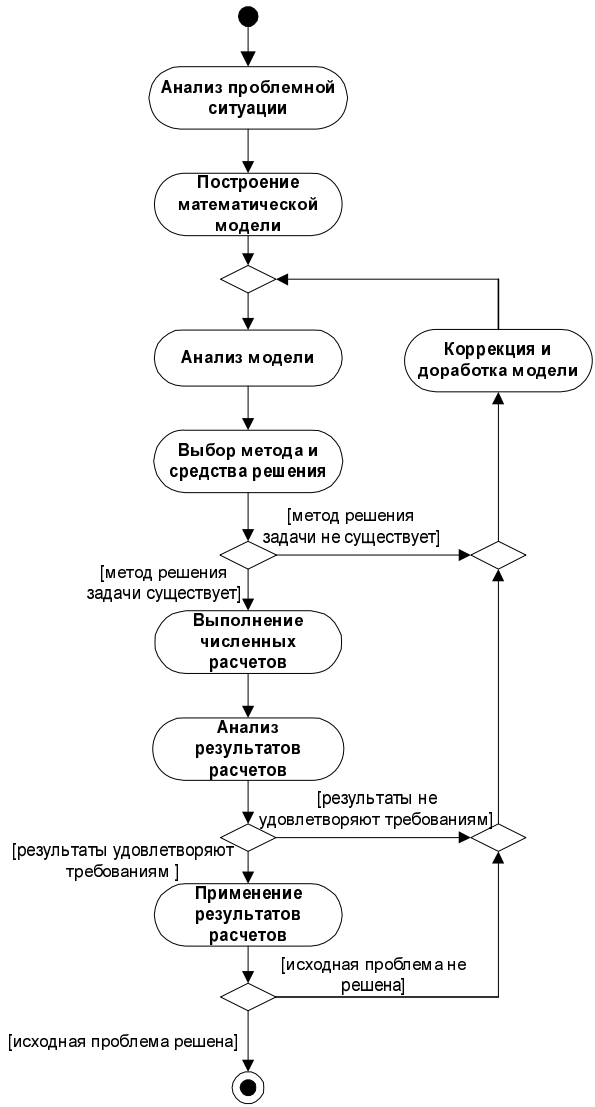


Рис. 1.12. Общая схема процесса постановки и решения задач оптимизации в форме диаграммы деятельности языка UML

выполняются определенные действия, направленные на построение и последующее использование информационно-логических моделей систем (рис. 1.12). Характерной особенностью данного процесса является его циклический или итеративный характер, который отражает современные требования к анализу и проектированию сложных систем.

Таким образом, отдельными этапами процесса постановки и решения задач оптимизации являются:

1. Анализ проблемной ситуации.
2. Построение математической модели.
3. Анализ модели.
4. Выбор метода и средства решения.
5. Выполнение численных расчетов.
6. Анализ результатов расчетов.
7. Применение результатов расчетов.
8. Коррекция и доработка модели.

Далее дается краткая характеристика каждого из этапов, конкретное содержание которых зависит от специфических особенностей решаемых задач оптимизации в той или иной проблемной области. При этом каждый новый цикл процесса постановки и решения задачи инициируется этапом анализа проблемной ситуации, в чем проявляется реализация требования проблемно-ориентированного подхода к построению и использованию информационно-логических моделей систем для решения задач оптимизации.

1.4.1. Анализ проблемной ситуации

Одним из основных принципов системного моделирования является проблемная ориентация процессов построения и использования моделей. Другими словами, та или иная модель конкретной системы строится в контексте решения некоторой проблемы или достижения некоторой цели. Главное назначение первого этапа — логическое осмысление конкретной проблемы в контексте методологии системного моделирования. При этом выполняется анализ всех доступных ресурсов (материальных, финансовых, информационных и других), необходимых для построения модели, ее использования и реализации полученных результатов с целью решения имеющейся проблемы. В случае отсутствия требуемых ресурсов на данном этапе может быть принято решение либо о сужении (уменьшении масштаба) решаемой проблемы, либо вообще об отказе от использования средств системного моделирования.

На этом этапе также выполняется анализ требований, предъявляемых в той или иной форме к результату решения проблемы.

Первоначальный анализ решаемой проблемы и соответствующей проблемной области является наименее формализуемым с точки зрения использования известных аналитических подходов и средств. Поэтому на данном этапе рекомендуется применять так называемые *эвристические* или неформальные методы системного анализа. К ним относятся:

- методы построения логических сценариев или повествовательных историй на естественном языке для анализа возможных способов и альтернативных путей решения проблемы;
- методы мозговой атаки или штурма для генерации новых идей и нестандартных подходов к решению проблемы;
- методы морфологического и концептуального анализа для достижения требуемой полноты рассмотрения исходной проблемы;
- методы построения и анализа дерева целей и задач, которые позволяют разбить исходную проблему на ряд более частных или более простых подпроблем.

1.4.2. Построение математической модели

Целью данного этапа является построение адекватной модели проблемной ситуации и соответствующей проблемной области в наиболее общем контексте решения исходной проблемы. Структуризация проблемной области предполагает определение и последующее уточнение ее границ, а также установление границ и состава систем, которые потенциально могут участвовать в решении исходной проблемы. Соответствующая информация представляется в форме модели системы или проблемной области в целом на некотором формально-логическом языке.

При этом представляется важным, чтобы вся доступная и существенная информация о решении проблемы была зафиксирована в виде некоторой информационно-логической модели системы. При этом модель должна удовлетворять принципу адекватности отражения основных особенностей системы-оригинала. Другими словами, модель не должна быть ни поверхностной или неполной, не учитывающей существенные аспекты структуры или поведения системы-оригинала; ни излишне сложной или избыточной, в рамках которой разработчики пытаются учесть даже несущественные с точки зрения исходной проблемы детали системы-оригинала.

Данный этап построения информационно-логической модели предполагает выполнение следующих действий:

1. Построение концептуальной или информационной модели системы и проблемной области, которая содержит наиболее общую информацию и отражает структурные взаимосвязи системы с другими объектами окружающей среды.
2. Построение аналитической или математической модели системы, которая детализирует отдельные аспекты структуры и поведения системы-оригинала в форме текста с использованием специальной математической нотации и символики.
3. Построение имитационной или программной модели системы, которая непосредственно реализует информационно-логическую модель в форме, специально предназначенной для ее исследования с использованием компьютеров.

Примечание

Один из принципов системного моделирования заключается в том, что для построения адекватной модели сложной системы может потребоваться не одна, а несколько моделей системы-оригинала. В этом случае каждая из подобных моделей будет являться отдельным представлением сложной системы, а полная модель системы будет состоять из комплекса взаимосвязанных моделей. Этот принцип получил специальное название — принцип *многомодельности* системного моделирования. Важно, что с точки зрения системного аналитика все частные модели системы равноправны, поэтому корректно вести речь лишь об их адекватности. При этом выбор типа модели должен зависеть от характера решаемой проблемы, а не от профессиональной специализации прикладных математиков и системных аналитиков, участвующих в решении проблемы.

Процесс разработки адекватных моделей и их последующего конструктивного применения требует не только знания общей методологии системного анализа, но и наличия соответствующих изобразительных средств или языков для фиксации результатов моделирования и их документирования. Очевидно, что естественный язык не вполне подходит для этой цели, поскольку обладает неоднозначностью и неопределенностью. Поэтому для построения моделей используются формально-логические методы, основанные на дальнейшем развитии математических и логических средств моделирования.

1.4.3. Анализ модели

В общем случае формально-логическая модель системы разрабатывается для получения некоторой новой информации о системе-оригинале с целью решения

исходной проблемы. При решении задач оптимизации для этой цели строится некоторая математическая модель, анализ которой предполагает установление характерных свойств отдельных элементов этой модели. Такими элементами являются: переменные, ограничения и целевая функция модели, множество допустимых альтернатив и его математические свойства. После анализа свойств элементов математической модели оказывается возможным соотнести решаемую задачу оптимизации с одним из классов данных задач, что имеет принципиальное значение для выбора метода и средств для ее последующего решения.

Анализ математической модели решаемой задачи необходимо выполнять в контексте общей классификации задач оптимизации, которая рассматривается в *разд. 1.6*.

1.4.4. Выбор метода и средства решения

Хотя для отдельных задач оптимизации существует решение в аналитической форме, это является скорее исключением из общего правила. Сколь-нибудь интересные практические задачи оптимизации, встречающиеся в современных бизнес-приложениях, как правило, не имеют аналитического решения в форме расчетных формул или номограмм. Именно поэтому становится актуальным выбор вычислительного метода и программного средства для их практического решения.

На выбор метода и средства оказывает влияние характер математической модели и математические свойства множества допустимых альтернатив. В первом случае класс, к которому относится рассматриваемая математическая модель, как правило, предопределяет выбор метода и алгоритма решения соответствующей задачи оптимизации. Во втором случае такая характеристика множества допустимых альтернатив, как, например, размерность исходных данных, оказывает принципиальное влияние на возможность получения точного или приближенного решения.

На выбор программного средства для решения задач оптимизации оказывают влияние следующие соображения. Как уже упоминалось, на рынке программ существуют математические пакеты, например, MATLAB и Mathcad, которые специально ориентированы на решение математических задач. Их основное достоинство заключается в наличии сотен и тысяч встроенных математических функций и десятков вычислительных алгоритмов для выполнения практических расчетов. Очевидный недостаток — дополнительные финансовые затраты на их приобретение и время — на изучение. В пользу выбора этих программ говорит весьма специфический характер отдельных задач оптими-

зации, которые требуется решить, например, из области цифровой обработки сигналов, распознавания изображений или звука и т. д.

Наряду с этим на компьютерах практически всех пользователей присутствует программа электронных таблиц MS Excel, которая, устанавливается в составе офисного пакета, как правило, сразу после инсталляции ОС от Microsoft. Вполне очевидно желание аналитика воспользоваться возможностями этой программы для решения задач оптимизации. В пользу выбора MS Excel в качестве программного средства служит наличие встроенных функций и нескольких алгоритмов поиска решения. При этом никаких дополнительных затрат от пользователя не требуется. Недостаток связан с отсутствием возможности решения некоторых классов задач оптимизации, о чем дополнительно будет указано далее в книге.

Примечание

Однако отмеченный недостаток может встретиться и при решении отдельных задач оптимизации с помощью специализированных математических пакетов. В этом случае выбор аналитика предопределен — либо разработка на одном из языков программирования собственной программы, которая реализует тот или иной алгоритм решения подобной задачи, либо поиск другой программы, обладающей требуемой функциональностью. Особенности программирования отдельных вычислительных алгоритмов при решении задач оптимизации рассматриваются в *главах 11* и *12*.

При выполнении данного этапа может сложиться ситуация, когда для рассматриваемой задачи оптимизации не существует адекватного метода решения. Это может потребовать выполнения дополнительной коррекции и доработки модели (что отражено на рис. 1.11) либо вообще отказа от решения исходной проблемы (что не отражено на рис. 1.11). Поскольку последний случай характерен для крайнего пессимизма и, как правило, соответствует низкой квалификации аналитика, он не рассматривается как серьезная альтернатива первому. Вывод об отказе решения исходной проблемы следует принимать лишь в случае невозможности коррекции и доработки модели или сужения исходной проблемы.

При наличии метода решения рассматриваемой задачи оптимизации в сформулированной постановке для разработанной математической модели и выбранном программном средстве следует перейти к ее практическому решению в форме выполнения численных расчетов.

1.4.5. Выполнение численных расчетов

Реализация данного этапа в контексте методологии системного моделирования означает выполнение серии экспериментов с программной моделью системы на той или иной вычислительной платформе. В нашем случае, это решение конкретной задачи оптимизации для фиксированной совокупности исходных данных средствами программы электронных таблиц MS Excel. При этом возможна следующая последовательность действий, отражающая содержание собственно процесса планирования экспериментов:

1. Формирование конкретных значений исходных данных (значений коэффициентов ограничений и целевой функции) и их ввод в специальном формате на отдельный рабочий лист MS Excel.
2. Задание свойств алгоритма расчета и параметров поиска решения MS Excel.
3. Выполнение расчетов с целью получения решения в форме конкретных значений переменных модели.
4. Представление результатов расчетов в графической форме для их наглядной интерпретации.

В отдельных случаях средство поиска решения MS Excel не позволяет получить решение задачи, о чем программа любезно сообщает пользователю. Причиной этого сообщения чаще всего являются ошибки при задании параметров поиска решения, например, неверный знак ограничений, что может привести к их несовместимости. Однако причина может быть и более сложной, связанной с особенностями встроенных вычислительных алгоритмов MS Excel, о чем дополнительно указывается в соответствующих главах книги при рассмотрении отдельных классов задач оптимизации.

1.4.6. Анализ результатов расчетов

Цель данного этапа заключается в анализе точности и правильности полученных результатов, если поиск решения MS Excel закончился успешно. При этом возможна следующая последовательность действий:

1. Оценка точности и верификация полученных результатов на основе проверки согласованности отдельных компонентов вычислительных расчетов с использованием аналитической модели и ручного просчета.
2. Интерпретация полученных результатов в форме управляющих воздействий или альтернатив решения исходной проблемы.
3. Оценка потенциальной возможности реализации полученных результатов применительно к системе-оригиналу.

Для удобства выполнения указанных действий результаты расчетов могут быть представлены в графическом виде в форме соответствующих диаграмм и графиков. Наличие данной возможности у программных средств также оказывают влияние на его выбор.

Примечание

В отдельных случаях для проверки или верификации полученных результатов может потребоваться привлечение в качестве дополнительных специальных математических программ. Как правило, уверенность в правильном решении практических задач оптимизации приходит после того, как она решена с помощью различных средств. Отдельные аспекты верификации результатов расчетов также нашли отражение в книге при рассмотрении методов решения типовых задач оптимизации.

Если анализ полученных результатов показывает их недостаточную точность или потенциальную невозможность их реализации применительно к системе-оригиналу, то следует перейти к этапу коррекции и доработки модели. Если полученные результаты удовлетворяют всем предъявляемым к ним требованиям, то можно перейти к этапу их применения к системе-оригиналу для решения исходной проблемы.

1.4.7. Применение результатов расчетов

Содержанием данного этапа является материальное или информационное воздействие на систему-оригинал с целью решения исходной проблемы. При этом может потребоваться дополнительное планирование организационных мероприятий по реализации подобных воздействий и контроль их выполнения.

После реализации рекомендаций выполненных исследований и после окончания этапа вычислительных экспериментов с моделью, вообще говоря, может сложиться одна из двух ситуаций:

- исходная проблема полностью решена — тем самым цели системного моделирования достигнуты. В этом случае можно перейти к решению очередной проблемы из данной проблемной области или ожидать эффекта от решения исходной проблемы;
- исходная проблема не решена или решена не полностью — тем самым цели системного моделирования не достигнуты. В этом случае необходимо тщательно проанализировать сложившуюся ситуацию и причины неудачи. После этого можно перейти либо к коррекции исходной модели системы, либо вообще отказаться от построенной модели и реализовывать цикл системного моделирования заново.

Следует заметить, что процесс решения сложных проблем оптимизации занимает достаточно продолжительное время, в течение которого, вообще говоря, может измениться как само содержание исходной проблемы, так и наличие необходимых для ее решения ресурсов. Эти особенности зачастую не учитываются при реализации сложных проектов, что является источником их неудачного завершения. Именно для исключения или ослабления негативного влияния данных факторов на схеме процесса постановки и решения задач оптимизации должен быть предусмотрен отдельный этап — коррекция или доработка модели, который может начать выполняться с любого момента изменения исходной ситуации или в результате возникновения признаков неадекватности модели на любом из рассмотренных ранее этапов.

1.4.8. Коррекция и доработка модели

Цель данного этапа неявно была уже сформулирована ранее, а именно — внесение изменений в существующую модель, которые направлены на обеспечение ее адекватности решаемой проблеме. Речь может идти как о включении в состав или исключении из состава исходной модели дополнительных компонентов, так и о радикальном изменении структуры и содержания модели. Важно отметить проблемно-ориентированный характер этих изменений, т. е. коррекция или доработка модели должны выполняться в непосредственном контексте с решаемой проблемой.

Результатом выполнения этого этапа может служить упрощение исходной задачи оптимизации, например, заменой нелинейных ограничений или целевой функции их линейными аналогами, а также сокращение количества переменных или ограничений модели. С другой стороны, данный этап может быть связан и с усложнением исходной модели посредством введения дополнительных переменных и ограничений для обеспечения необходимой адекватности модели решаемой проблеме.

При рассмотрении отдельных классов задач оптимизации приводятся также рекомендации и практические советы по выполнению этого этапа. Для правильного выполнения этапов построения и анализа математической модели следует рассмотреть определения и свойства основных элементов собственно математической модели.

1.5. Математическая модель задач оптимизации

При рассмотрении процесса постановки и решения задач оптимизации ключевую роль играет понятие математической модели задач оптимизации и свойства ее основных элементов. Именно от свойств математической модели

зависит возможность решения отдельной задачи оптимизации и выбор наиболее эффективного алгоритма и способа для этой цели. Игнорирование или незнание особенностей математических моделей задач оптимизации служат источником ошибок и неудач в решении данных задач.

1.5.1. Понятие математической модели и ее основные элементы

В общем случае под *математической моделью* задачи оптимизации будем понимать специальную запись постановки и условий решения типовой задачи оптимизации с использованием понятий математики и математической символики. Применительно к конкретной задаче оптимизации математическая модель соответствует *математической постановке* данной задачи.

Как уже отмечалось ранее, при постановке задачи оптимизации должны быть определены или специфицированы следующие ее базовые компоненты:

- характеристика переменных, фиксированный набор значений которых характеризует отдельное решение задачи;
- набор ограничивающих условий, исключающих из рассмотрения отдельные решения по причине их физической или логической невозможности;
- оценочная функция, позволяющая количественно сравнивать различные решения с целью выбора наилучшего из них.

Каждый из базовых компонентов математической модели задачи оптимизации имеет свои особые свойства, которые зависят от свойств соответствующих математических объектов. Далее рассматриваются лишь основные свойства этих компонентов, используемых при общей классификации задачи оптимизации.

1.5.2. Характеристика переменных

Понятие *переменной* является одним из фундаментальных в математике, характерным свойством которой служит множество принимаемых ею значений. В зависимости от специфических свойств этого множества *(см. приложение 1)* в задачах оптимизации используются три основных типа переменных:

- *непрерывные*, множество принимаемых значений которых имеет мощность континуума и, как правило, совпадает с множеством всех неотрицательных вещественных чисел или является его подмножеством;
- *целочисленные* или *дискретные*, множество принимаемых значений которых имеет счетную или даже конечную мощность и, как правило, совпа-

дает с множеством всех неотрицательных целых чисел или является его подмножеством;

- *булевы*, множество принимаемых значений которых имеет всего лишь два значения: 0 и 1.

Как в математике, так и в математической модели задач оптимизации переменные традиционно принято обозначать латинскими буквами: без индексов — x, y, z; с одним индексом — x_i, y_j, z_k; с двумя индексами — x_{ij}, y_{ij}, z_{kl}. Важно понимать, что каждая переменная должна иметь реальную интерпретацию или физический смысл в постановке задачи, а набор их значений — определять некоторое потенциальное решение исходной проблемы. При этом выбор той или иной буквы для обозначения переменной не играет принципиального значения. Фиксированный набор значений отдельных переменных задачи оптимизации называют *альтернативой*.

Традиционно в математике множество всех вещественных чисел обозначают через $\boldsymbol{R}^1$ или просто $\boldsymbol{R}$, множество всех целых чисел — через $\boldsymbol{Z}^1$ или просто $\boldsymbol{Z}$, множество из двух чисел: 0 и 1 — через $\boldsymbol{B}^1$ или $\{0, 1\}$. С учетом этих обозначений требование непрерывности переменных принято символически обозначать в виде: $x, y, z \in \boldsymbol{R}^1$; требование целочисленности переменных — в виде: $x, y, z \in \boldsymbol{Z}^1$; требование булевости переменных — в виде: $x, y, z \in \{0, 1\}$ или, если не возникает недоразумений, $x, y, z \in \boldsymbol{B}^1$.

1.5.3. Характеристика ограничений

Большинство, если не все, практические задачи оптимизации имеют некоторый набор ограничивающих условий, которые исключают из рассмотрения отдельные решения по причине их физической или логической невозможности. Данные условия получили название *ограничений*, которые в своей совокупности определяют или специфицируют *множество допустимых альтернатив* рассматриваемой задачи оптимизации.

В общем случае ограничение представляет собой некоторую функциональную зависимость, которая связывает отдельные значения переменных друг с другом. В общем случае подобная функциональная зависимость записывается в виде некоторой функции, например $g(x, y, z)$, от исходных переменных задачи оптимизации. При этом предполагается, что данная функция, как правило, является непрерывной. Очевидно, что в различных задачах оптимизации может присутствовать различное число подобных функций.

Математическая запись ограничения дополнительно предполагает требование, что значения данной функции или функций ограничены некоторым ин-

тервалом или числом. В зависимости от знака ограничения задач оптимизации используются два основных типа ограничений:

- *равенства*, которые символически записываются в виде: $g(x, y, z) = a$ или $g(x, y, z) = 0$;
- *неравенства*, которые символически записываются в виде: $g(x, y, z) \le (\ge) a$ или $g(x, y, z) \le (\ge) 0$.

Здесь a — некоторое вещественное число, которое принимает заданное значение в контексте рассматриваемой задачи оптимизации. Саму функцию $g(x, y, z)$ в этом случае называют *правой частью* ограничения, а значение a — *левой частью* ограничения. Поскольку возможен случай, когда $a = 0$, то само значение левой части ограничения не имеет принципиального характера.

Примечание

Внимательные читатели могли бы заметить, что при рассмотрении типа ограничений не указан случай строгих неравенств $g(x, y, z) < (>) a$ или $g(x, y, z) < (>) 0$. Действительно, этот вариант теоретически возможен, однако практически не рассматривается. Причина заключается в том, что, с одной стороны, подобные ограничения изменяют характер соответствующей теории их анализа и решения, превращая ее в набор предельных теорем и экзотических методов, а с другой стороны, в реальных задачах оптимизации подобные ограничения практически не встречаются. Именно поэтому они вообще исключены из рассмотрения в книге.

В зависимости от дополнительных свойств функции $g(x, y, z)$ в задачах оптимизации используются следующие основные типы ограничений.

- *Линейные*, в которых все функции $g(x, y, z)$ является линейными относительно всех своих переменных.
- *Нелинейные*, в которых все функции $g(x, y, z)$ является нелинейными относительно всех своих переменных. В последнем случае иногда дополнительно рассматривают следующие типы ограничений:
  - *выпуклые*, в которых все функции $g(x, y, z)$ является выпуклыми относительно всех своих переменных;
  - *невыпуклые*, в которых все функции $g(x, y, z)$ является невыпуклыми относительно всех своих переменных.

Набор ограничений каждой задачи оптимизации сужает исходное множество ее решений или альтернатив. В простейших случаях это может соответствовать сужению множества $\boldsymbol{R}^1$ до интервала значений, как, например, в задачах о коробке максимального объема или пожарном ведре. В более сложных

случаях это может соответствовать сужению множества $\boldsymbol{R}^k$ до k-мерного симплекса, как, например, в задачах линейного программирования.

В общем случае *множество допустимых альтернатив*, удовлетворяющих всем ограничениям рассматриваемой задачи оптимизации, принято обозначать через Δ_β или просто Δ. Математические свойства этого множества полностью определяются характером переменных и ограничений задачи оптимизации. Тот факт, что некоторый набор значений переменных удовлетворяет всей совокупности ограничений задачи, а соответствующая альтернатива принадлежит множеству допустимых альтернатив, символически записывается в виде: $x, y, z \in \Delta_\beta$.

1.5.4. Характеристика целевой функции

Целевой или *критериальной* функцией задачи оптимизации называется некоторая оценочная функция, предназначенная для количественного сравнения альтернатив с целью выбора наилучшей. Целевая функция определяется как некоторая математическая функция, функционал или оператор, что, в общем случае, записывается в виде: $f(x, y, z)$, где f: $\Delta_\beta \rightarrow \boldsymbol{R}^1$.

В зависимости от свойств функции $f(x, y, z)$ в задачах оптимизации используются следующие основные типы целевых функций.

- *Линейные*, в которых функция $f(x, y, z)$ является линейной относительно всех своих переменных.
- *Нелинейные*, в которых функция $f(x, y, z)$ является нелинейной относительно всех своих переменных. В последнем случае иногда дополнительно рассматривают:
  - *выпуклые (квадратичные)*, в которых функция $f(x, y, z)$ является выпуклой (соответственно, квадратичной) относительно своих переменных;
  - *невыпуклые*, в которых функция $f(x, y, z)$ является невыпуклой относительно своих переменных.

В зависимости от количества целевых функций рассматриваются два основных типа задач оптимизации:

- *однокритериальная* задача оптимизации, в математической модели которой присутствует единственная целевая функция;
- *многокритериальная* задача оптимизации, в математической модели которой присутствует несколько целевых функций.

В контексте математической модели задач оптимизации требование нахождения наилучшего решения конкретизируется в требование *максимизации* или *минимизации* целевой функции. Данное требование может быть записано

символически в виде: $f(x, y, z) \to max$ или $f(x, y, z) \to min$. При этом максимум (минимум) целевой функции находится только среди множества допустимых альтернатив. Решением задачи оптимизации является некоторый допустимый набор значений переменных, который доставляет максимальное или минимальное значение целевой функции на множестве допустимых альтернатив.

С учетом введенных обозначений общая математическая модель однокритериальной задачи оптимизации может быть записана символически в следующем виде:

$$f(x,y,z) \to \max_{x,y,z \in \Delta_\beta} \text{ или } f(x,y,z) \to \min_{x,y,z \in \Delta_\beta}, \tag{1.5.1}$$

$$\text{где} \quad \Delta_\beta = \{\Delta \mid g_k(x,y,z) \le (=)0\}, \qquad (k \in \{1,2,...,m\}). \tag{1.5.2}$$

Здесь через Δ_β обозначено множество допустимых альтернатив, которое формируется посредством сужения исходного множества альтернатив Δ с помощью совокупности ограничений, записанных в произвольной форме: $g_k(x, y, z) \le 0$ или $g_k(x, y, z) = 0$. В качестве исходного множества альтернатив Δ выступает одно из рассмотренных ранее множеств: множество действительных чисел $\boldsymbol{R}^1$, множество целых чисел $\boldsymbol{Z}^1$ или множество из двух чисел: $\boldsymbol{B}^1$. Выбор этого множества определяется типом переменных, которые используются в постановке соответствующей задачи оптимизации.

С учетом введенных обозначений общая математическая модель многокритериальной задачи оптимизации может быть записана символически в следующем виде:

$$f_i(x,y,z) \to \max_{x,y,z \in \Delta_\beta} \text{ или } f_i(x,y,z) \to \min_{x,y,z \in \Delta_\beta}, \qquad (\forall i \in \{1,2,...,n\}), \tag{1.5.3}$$

$$\text{где} \quad \Delta_\beta = \{\Delta \mid g_k(x,y,z) \le (=)0\}, \qquad (k \in \{1,2,...,m\}). \tag{1.5.4}$$

Здесь через Δ_β также обозначено множество допустимых альтернатив, которое формируется посредством сужения исходного множества альтернатив Δ. В случае $n = 2$ соответствующие задачи оптимизации называются *двухкритериальными*, $n = 3$ — *трехкритериальными* и т. д. Натуральное число m определяет общее количество ограничений задачи оптимизации. В математических моделях типовых задач оптимизации явно указывают исходные множества альтернатив для точной спецификации типа переменных.

Рассмотренные свойства базовых компонентов математической модели задач оптимизации позволяют выполнить общую классификацию этих задач, знание которой необходимо для правильного анализа и выбора метода для решения конкретных задач оптимизации.

1.5.5. Общая классификация задач оптимизации

Построение исчерпывающей классификации задач оптимизации, которая учитывала бы все возможные модификации типовых задач и нюансы математических свойств их базовых компонентов, а также удовлетворяла математиков и системных аналитиков, практически вряд ли возможно. В то же время рассмотренных свойств базовых компонентов оказывается вполне достаточно для общей классификации задачи оптимизации, используемой для анализа и решения большинства конкретных задач, встречающихся на практике.

Для удобства общая классификация задач оптимизации представлена в табличном виде (табл. 1.1).

Таблица 1.1. *Общая классификация задач оптимизации*

Характеристика переменных	**Характеристика ограничений**	**Характеристика целевой функции**	**Класс задач оптимизации и их рассмотрение в книге**
Непрерывные	Линейные	Одна, линейная	Линейное программирование, *глава 4*
Непрерывные	Нелинейные или линейные	Одна, нелинейная	Нелинейное программирование, *глава 3*
Целочисленные	Линейные	Одна, линейная	Целочисленное программирование, *глава 5*, а также — *7*, *8*
Целочисленные	Нелинейные или линейные	Одна, нелинейная	Целочисленное нелинейное программирование, не рассматривается в книге
Булевы	Линейные	Одна, линейная	Булево программирование, *глава 6*
Булевы	Нелинейные или линейные	Одна, нелинейная	Булево нелинейное программирование, не рассматривается в книге
Непрерывные	Линейные	Несколько, линейные	Многокритериальное линейное программирование, *глава 9*
Непрерывные	Нелинейные или линейные	Несколько, нелинейные	Многокритериальное нелинейное программирование, не рассматривается в книге

Таблица 1.1 *(окончание)*

Характеристика переменных	Характеристика ограничений	Характеристика целевой функции	Класс задач оптимизации и их рассмотрение в книге
Целочисленные	Линейные	Несколько, линейные	Многокритериальное целочисленное программирование, *глава 9*
Целочисленные	Нелинейные или линейные	Несколько, нелинейные	Многокритериальное целочисленное нелинейное программирование, не рассматривается в книге
Булевы	Линейные	Несколько, линейные	Многокритериальное булево программирование, не рассматривается в книге
Булевы	Нелинейные или линейные	Несколько, нелинейные	Многокритериальное булево нелинейное программирование, не рассматривается в книге

Примечание

Следует заметить, что попытка рассмотрения смешанных вариантов свойств базовых компонентов математической модели задач оптимизации приводит не только к усложнению общей классификации, но и усложнению методов анализа и решения соответствующих задач. Такие смешанные задачи оптимизации являются предметом специальных теоретических исследований и поэтому не включены в данную классификацию.

Общая классификация задач оптимизации, основанная на рассмотрении характерных свойств базовых компонентов математической постановки данных задач, служит концептуальной основой для адекватного выбора метода решения конкретных задач того или иного класса. Далее приводится краткая характеристика основных подходов к решению задач оптимизации, которая в последующем уточняется при рассмотрении конкретных методов и алгоритмов решения типовых задач.

1.6. Основные подходы к решению задач оптимизации

При решении задач оптимизации необходимо найти наилучшее решение из всех допустимых. Формализация оценочной функции в форме целевой функции математической модели, ограничивающих условий — в форме ограничений позволяют также дать строгое определение понятию "наилучшее решение". Таким является оптимальное решение.

1.6.1. Понятие оптимального решения задачи оптимизации

В общем случае под *оптимальным решением* однокритериальной задачи оптимизации в математической постановке (1.5.1 и 1.5.2) понимается такой набор значений переменных: x^*, y^*, $z^* \in \Delta_\beta$, которые доставляют максимум или минимум целевой функции $f(x, y, z)$ среди всех допустимых решений множества Δ_β. Другими словами, характерным признаком оптимального решения задачи оптимизации является выполнение следующего условия:

$$f(x^*, y^*, z^*) \geq f(x, y, z), \qquad \forall x, y, z \in \Delta_\beta; \tag{1.6.1}$$

$$f(x^*, y^*, z^*) \leq f(x, y, z), \qquad \forall x, y, z \in \Delta_\beta. \tag{1.6.2}$$

При этом условие (1.6.1) должно выполняться для задач максимизации, а условие (1.6.2) — для задач минимизации. Говоря о решении той или иной задачи оптимизации, всегда понимают нахождение ее оптимального решения, которое соответствует понятию наилучшего решения в содержательной постановке.

Примечание

Применительно к многокритериальным задачам оптимизации условия (1.6.1 и 1.6.2) обобщаются на весь набор целевых функций, а собственно понятие их решения имеет весьма специальный характер и подробно обсуждается в *главах 9* и *10*.

Обобщая задачи максимизации и минимизации, часто говорят о нахождении *экстремума* задачи оптимизации, а саму теорию решения задач оптимизации называют теорией решения *экстремальных задач*. Не вдаваясь в семантические детали и нюансы этих терминов, будем считать понятия оптимума и экстремума синонимами, что никак не отразится на качестве и корректности решения практических задач.

Примечание

В связи с понятием теории экстремальных задач как не вспомнить о похожем на него, но никак с ним не связанным *экстремальным программированием* (XP). Хотя в книге довольно часто встречается термин "программирование" в совокупности с терминами линейное, нелинейное, целочисленное, в этом контексте экстремальное программирование — это нечто совершенно иное и поэтому в книге не рассматривается. Что касается термина "линейное программирование" и аналогичных ему, то это сложившаяся традиция, связанная с историей возникновения и изучения задач соответствующего класса. Конечно, для решения задач подобных классов вовсе не нужно быть программистом. Более того, настоящая книга содержит всю необходимую информацию, которая позволяет непрограммистам эффективно решать практические задачи оптимизации.

Что касается других синонимов, то зачастую задачи целочисленного программирования называют задачами целочисленной или дискретной оптимизации, нелинейного программирования — нелинейной оптимизации и наоборот. Как правило, если не возникает недоразумения, эти понятия также предполагаются взаимозаменяемыми на всем протяжении книги.

1.6.2. Проблема существования и единственности решения задач оптимизации

В связи с анализом математической модели типовых задач оптимизации в контексте нахождения оптимального решения (или просто решения) встает два теоретических вопроса: существования решения и его единственности. Не вдаваясь в детали обсуждения этих проблем, отметим лишь их основные особенности.

Первая из проблем — проблема *существования* оптимального решения — рассматривается для типовых задач оптимизации в конкретной математической постановке. Имеется два подхода к ее решению: формальный и неформальный. *Формальный* подход предполагает использование математических теорем существования для того или иного множества допустимых альтернатив и того или иного вида целевой функции задачи оптимизации. Главный вывод соответствующих теорем существования — *если множество допустимых альтернатив замкнуто, а целевая функция непрерывна на этом множестве, то решение соответствующей задачи оптимизации существует.*

Примечание

Условие замкнутости множества допустимых альтернатив обеспечивается нестрогим характером ограничений или их записью в форме равенств, а в реаль-

ных задачах оптимизации целевая функция практически всегда непрерывна. Именно по этим причинам строгие формулировки теорем существования для задач оптимизации не рассматриваются в настоящей книге. Заинтересованные в формальном решении проблемы существования решений читатели могут обратиться к специальным работам, указанным в списке литературы.

Неформальный подход к решению проблемы существования предполагает установление физической или логической осуществимости некоторых из возможных решений. Речь идет о том, что в контексте содержательной постановки задачи оптимизации выполняется эвристический анализ допустимости некоторых из решений. Если подобный анализ заканчивается успешно, то можно приступать к поиску решения задачи. В противном случае может потребоваться коррекция и доработка математической модели. С учетом сделанного ранее замечания, неформального анализа существования решения для типовой задачи оптимизации может оказаться вполне достаточно для ее корректного решения.

Примечание

Пояснить неформальный анализ существования решения можно следующим образом. Для этого обратимся, например, к типовой задаче о минимальном пути в графе *(см. разд. 1.2.5)*. В этом случае имеется исходный граф, ребра которого соответствуют наличию отдельных автодорог между населенными пунктами. Если при рассмотрении данного графа аналитику удается отыскать хотя бы один возможный путь между исходным и конечным пунктами, то данный факт, наряду с конечным множеством всех возможных путей, в общем случае, свидетельствует о существовании оптимального решения. Этот вывод может быть обобщен на все задачи целочисленного программирования с конечным множеством допустимых альтернатив, включая наиболее сложные из них — задачи комбинаторной оптимизации.

Вторая из проблем — проблема *единственности* оптимального решения — рассматривается в контексте вида целевых функций типовых задач оптимизации. Имеется также два подхода к ее решению: формальный и неформальный. *Формальный* подход предполагает использование математических теорем, гарантирующих единственность решения для выпуклых целевых функций выпуклых множеств допустимых альтернатив. Главный вывод соответствующих теорем — *если множество допустимых альтернатив выпукло и целевая функция на этом множестве выпукла (или вогнута), то существует единственное решение соответствующей задачи оптимизации.*

К сожалению, данные условия характерны только для задач с непрерывными переменными, имеют слишком жесткий характер и не выполняются для целых классов типовых задач оптимизации. В частности, подобные теоремы

непосредственно не применимы к задачам целочисленного и булева программирования. Сложность последних задач возрастает с многоэкстремальным характером соответствующих целевых функций. Тем самым следует признать, что формальный способ установления единственности оптимального решения имеет весьма ограниченную область применения.

Неформальный подход к решению проблемы единственности предполагает практическое нахождение нескольких допустимых решений и выбора из них наилучшего. Для этого используется один или несколько способов или методов поиска решения. Если найденные решения совпадают, то этот факт будет свидетельствовать о единственности решения. В противном случае решение может оказаться неединственным. Подобного неформального анализа единственности полученного решения для типовой задачи оптимизации также может оказаться вполне достаточно для ее корректного решения. При этом на характер получаемого решения оказывает влияние не только конкретная задача оптимизации, но и применяемый для ее решения метод или алгоритм.

Таким образом, следует различать два принципиально различных подхода к решению задач оптимизации:

- *точное* решение задачи оптимизации, которое соответствует нахождению единственного или нет оптимального решения x^*, y^*, $z^* \in \Delta_\beta$, гарантирующего выполнение формальных условий (1.6.1) или (1.6.2);
- *приближенное* решение задачи оптимизации, которое соответствует нахождению некоторого допустимого решения x^*, y^*, $z^* \in \Delta_\beta$, не гарантирующего выполнение формальных условий (1.6.1) или (1.6.2).

Говоря о приближенном решении задач оптимизации, часто условия (1.6.1 и 1.6.2) заменяют их более слабым аналогом:

$$f(x^*, y^*, z^*) \geq f(x, y, z), \qquad \forall x, y, z \in \Theta_\varepsilon; \tag{1.6.3}$$

$$f(x^*, y^*, z^*) \leq f(x, y, z), \qquad \forall x, y, z \in \Theta_\varepsilon. \tag{1.6.4}$$

где через Θ_ε обозначена некоторая окрестность решения x^*, y^*, $z^* \in \Delta_\beta$ на множестве допустимых альтернатив. В этом случае решение задачи оптимизации x^*, y^*, $z^* \in \Delta_\beta$, удовлетворяющее только условию (1.6.3) — для задачи максимизации или (1.6.4) — для задачи минимизации, называют *локально-оптимальным* решением. В данном контексте решение, удовлетворяющее условиям (1.6.1 и 1.6.2), называют также *глобально-оптимальным* решением.

Не вызывает сомнения тот факт, что наиболее эффективный подход к решению задач оптимизации связан с нахождением *точного глобально-оптимального решения*. Однако на этом пути встречаются как теоретические, так и практические трудности. Теоретические трудности связаны с формальным

решением проблемы существования и единственности для отдельных классов задач оптимизации, а также разработкой и наличием вычислительных алгоритмов для нахождения такого решения.

Практические трудности связаны с общим количеством переменных и ограничений конкретных задач оптимизации. В последнем случае при большом числе этих компонентов практически не может быть найдено точное решение отдельной задачи оптимизации за приемлемое время, даже при наличии точных вычислительных методов ее решения.

Примечание

Существует еще один аспект проблемы нахождения точного решения задачи оптимизации, связанный с многоэкстремальным характером целевой функции. Для отдельных задач нелинейного программирования и практически для всех задач целочисленного и булева программирования целевые функции являются по своему характеру не выпуклыми и многоэкстремальными. Опуская строгое определение этого понятия, ограничимся лишь выводом о том, что многоэкстремальный характер целевой функции делает исключительно сложным практическое нахождение точного решения той или иной задачи оптимизации.

В случае теоретической или практической невозможности нахождения точного решения задачи оптимизации следует попытаться найти некоторое приближенное решение, которое обеспечивает значение целевой функции, близкое к глобально-оптимальному. Если и это невозможно по тем или иным причинам, то остается вариант нахождения нескольких (сколько — это отдельный вопрос) локально-оптимальных решений и выбора из них наилучшего, т. е. соответствующего максимальному или минимальному значению целевой функции. В отдельных случаях может и повезти — найденное решение может оказаться глобально-оптимальным или близким к нему.

В крайнем случае, ограничиваются единственным найденным локально-оптимальным решением, принимая его за окончательный результат. При этом важно понимать, что найденное решение может весьма значительно отличаться от точного глобально-оптимального решения.

Примечание

Как в связи с этим не вспомнить известный совет системному аналитику: "Всегда довольствуйтесь третьим по качеству решением, ибо второе по качеству решение приходит слишком поздно, а первое не приходит никогда".

На первый взгляд может показаться, что с понятием точного и приближенного решения также связана проблема выполнения вычислений в конкретной системе команд той или иной вычислительной платформы. Действительно,

современные компьютеры по своей природе являются цифровыми, поэтому при решении любых задач, включая задачи оптимизации с непрерывными переменными, будет всегда получено некоторое цифровое значение, точность которого ограничена разрядностью системы команд компьютера и количеством цифр для представления действительных чисел.

Хотя данная проблема серьезно обсуждалась в первые годы появления компьютеров, сейчас ее актуальность существенно упала. Это связано с тем, что точности 32- или 64-разрядной системы команд платформы WinIntel вполне достаточно для обеспечения приемлемого уровня точности при решении большинства практических задач. Что касается специальных классов задач, в которых необходима повышенная точность расчетов, то они служат темой специальных исследований с применением профессиональных математических программ.

В общем случае с проблемой поиска точных и приближенных решений тесно связана проблема разработки соответствующих конструктивных средств их практического нахождения. Применительно к решению задач оптимизации соответствующие конструктивные средства получили название методов и алгоритмов их точного или приближенного решения.

1.6.3. Понятие о методах и алгоритмах решения задач оптимизации

История математики и теории решения задачи оптимизации тесно связана с разработкой тех или иных алгоритмов решения актуальных для своей эпохи задач. Само понятие алгоритма является одним из центральных в вычислительной и конструктивной математике. В общем случае, под *алгоритмом* понимают формальное предписание выполнить точно определенную последовательность действий, направленных на достижение заданной цели или решение поставленной задачи.

Примечание

Принято считать, что сам термин алгоритм, а точнее его более ранний вариант — *алгорифм*, происходит от имени математика Аль-Хорезми, который в 825 г. описал правила выполнения арифметических действий в десятичной системе счисления. В настоящее время понятие алгоритма и связанные с ним проблемы вычислимости являются предметом специальной теории — *теории алгоритмов*. Со временем содержание этой теории стало настолько абстрактным, что понимание соответствующих результатов и их применение в прикладных задачах стало доступно только узкому кругу профессиональных математиков. По этой причине в задачах оптимизации используются только общие выводы данной теории.

При рассмотрении алгоритмов выделяют три основных свойства, которым должна удовлетворять та или иная последовательность действий, чтобы быть алгоритмом:

- *детерминированность* — характеризует точную фиксацию следующего действия после выполнения предыдущего. Другими словами, при рассмотрении алгоритмов должны быть исключены случаи неопределенности продолжения процесса выполняемых действий;
- *массовость* — характеризует применение алгоритма для решения задач целого класса, в рамках которого он должен позволить решить любую конкретную задачу с заданными значениями исходных данных;
- *результативность* — характеризует завершение процесса выполнения действий алгоритма за конечное число шагов или конечный интервал времени. В результате завершения процесса выполнения действий алгоритм либо должен решить поставленную задачу, либо сообщить о том, что по той или иной причине процесс решения не может быть продолжен.

При решении задач оптимизации выбранный или разработанный алгоритм должен позволять решать ту или иную конкретную типовую задачу оптимизации за конечное время или сообщить пользователю о невозможности ее решения. В общем случае для описания алгоритмов используется естественный язык с элементами математической символики, который может дополняться структурной схемой алгоритма в некоторой графической нотации.

Примечание

Для графического представления алгоритмов решения задач оптимизации весьма удобным оказывается унифицированный язык моделирования — язык UML (Unified Modeling Language). Обладая интуитивно понятными и эффективными выразительными средствами визуализации вычислительных процессов, язык UML позволяет наглядно представлять алгоритмы в форме диаграмм деятельности. Именно эти диаграммы используются на всем протяжении книги для изображения схем алгоритмов, как, например, на рис. 1.11. Более подробно с нотацией языка UML можно познакомиться по одной из книг автора, приводимых в списке литературы.

Отдельные действия алгоритма при его описании получили название *шагов* алгоритма, а их совокупность, выполняемая в рамках некоторого цикла, называется *итерацией* алгоритма. Поскольку эти понятия служат вспомогательным целям и используются, главным образом, при описании конкретных алгоритмов, более подробно они рассматриваются в соответствующих главах книги.

В контексте задач оптимизации рассматривается более общее и менее формальное понятие *метода* решения задачи оптимизации, под которым будем

понимать некоторое обобщенное описание вычислительного процесса в форме рекомендаций по выполнению некоторых действий, также направленных на достижение заданной цели или решение поставленной задачи.

Таким образом, метод решения задачи оптимизации отличается от алгоритма более общим характером описания и, как следствие, является более универсальным по сравнению с последним. В то же время менее формальный способ спецификации метода позволяет говорить о его конкретизации в рамках того или иного алгоритма.

Для решения типовой задачи оптимизации могут быть применены различные методы и алгоритмы. Так, например, наиболее известными и получившими широкое распространение являются: метод динамического программирования и метод ветвей и границ. Они, в свою очередь, могут быть реализованы в различных вариациях, что позволяет различать соответствующие алгоритмы, ориентированные, например, на решение задач целочисленного и комбинаторного программирования.

В то же время для специальных классов задач, например, для задач оптимизации на графах, предложены специальные алгоритмы, такие как алгоритмы Дейкстры или Краскала, которые имеют весьма узкую область применения. В этом случае может вообще отсутствовать общий метод, от которого наследует свойства рассматриваемый алгоритм. Указанные взаимосвязи между методом и алгоритмом решения задач оптимизации могут быть изображены графически (рис. 1.13).

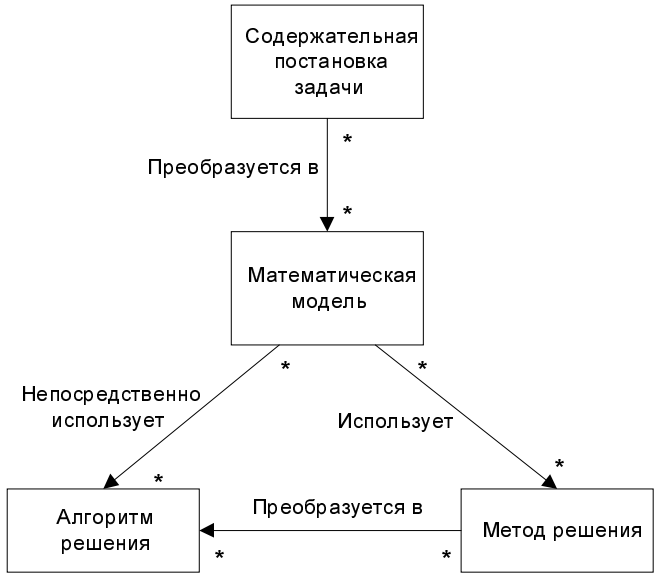


Рис. 1.13. Общая схема взаимосвязей методов и алгоритмов решения задач оптимизации в форме диаграммы классов языка UML

Что касается методов и алгоритмов решения задач многокритериальной оптимизации, то эта проблематика рассматривается в *части IV* книги. В заключение данной главы приведем общую схему описания задач оптимизации, которая может быть использована при выполнении соответствующих проектов.

1.6.4. Структура описания задач оптимизации

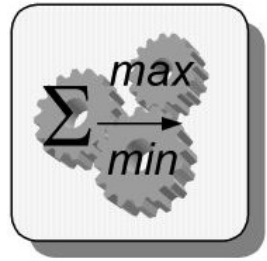
При решении практических задач оптимизации в рамках конкретных проектов следует придерживаться некоторого шаблона, позволяющего достичь требуемого уровня унификации и систематизации, необходимого для адекватного понимания особенностей решаемой проблемы и выбора наиболее эффективных средств ее решения. Общее описание задачи оптимизации должно включать в себя следующую информацию:

- название задачи, позволяющее соотнести решаемую задачу с одним из известных классов задач оптимизации;
- содержательная постановка задачи, которая в повествовательной форме или в форме сценария описывает сущность исходной проблемы, подлежащей решению;
- математическая постановка задачи, которая представляет исходную проблему в форме математической модели. Дальнейший анализ данной модели служит основой для выбора средств ее решения;
- метод решения, который может быть использован для решения рассматриваемой задачи оптимизации применительно к разработанной математической модели;
- алгоритм решения, который может быть использован для практического решения рассматриваемой задачи оптимизации применительно к конкретному набору исходных данных;
- средство или программа решения, которые позволяют реализовать вычислительный процесс поиска оптимального решения применительно к разработанной математической модели и к конкретному набору исходных данных;
- примеры численных расчетов, иллюстрирующих правильность выбранных методов, алгоритмов и средств решения исходной проблемы.

Как нетрудно заметить, предложенный шаблон позволяет фиксировать информацию при выполнении отдельных этапов процесса постановки и решения

задач оптимизации (см. рис. 1.12). При рассмотрении особенностей решения отдельных задач оптимизации весьма удобно придерживаться данного шаблона, который позволяет достичь необходимого уровня систематизации при изучении столь широкой проблематики, какой на сегодня является область задач оптимизации.

После краткого знакомства с особенностями типовых задач оптимизации и процессом их постановки и решения следует рассмотреть основные возможности базового программного средства, которое будет использоваться для их практического решения — программой электронных таблиц MS Office Excel 2003. Именно эта тематика является предметом изложения *главы 2*.



Глава 2

Основные приемы практической работы в среде MS Excel

Рассмотреть в одной главе все или даже большинство приемов практической работы в среде MS Excel вряд ли возможно, поскольку этой теме посвящены отдельные руководства. С одной стороны, можно было бы сделать предположение, что читатели хорошо знакомы с программой MS Excel и умеют выполнять все практические действия, необходимые для решения задач оптимизации. С другой стороны, можно просто сослаться на учебную литературу по программе MS Excel, а ссылки на отдельные издания поместить в список литературы.

Однако ни тот ни другой способ изложения не свободен от недостатков. Именно по этой причине в настоящей главе приводится базовый материал, необходимый для выполнения всех практических действий при решении задач оптимизации с помощью программы MS Excel. Этого материала достаточно, чтобы, не обращаясь к дополнительной литературе, выполнить все рассмотренные примеры и упражнения.

При отборе материала в настоящую главу, главным образом, преследовалась методическая цель — сконцентрировать внимание на действительно важных практических приемах, уверенное владение которыми служит необходимым залогом успеха при решении практических задач оптимизации. Именно поэтому здесь не рассматриваются вопросы справочного характера, такие как исчерпывающее описание форматов ячеек, встроенных функций, особенностей печати и публикации в Интернете и многие другие. При необходимости соответствующую информацию можно получить с помощью справочной системы программы MS Excel, которая в локализованной версии содержит большое количество статей на русском языке с возможностью контекстного поиска информации по ключевым словам. В то же время отдельные вопросы

практической работы в среде MS Excel, не вошедшие в настоящую главу, рассматриваются в следующих главах при решении типовых классов задач оптимизации.

2.1. Общая характеристика программы электронных таблиц MS Office Excel 2003

Программа электронных таблиц MS Office Excel 2003 или кратко — MS Excel, как уже следует из ее названия, представляет собой специальное средство организации и манипулирования данными в форме электронных таблиц. Эта программа поставляется в составе офисного пакета MS Office System 2003 и может быть установлена на компьютер пользователя либо вместе с другими программами этого пакета, либо отдельно. Процесс установки программы электронных таблиц MS Office Excel 2003 описывается в соответствующей документации по установке и поэтому не рассматривается в книге.

Из документации также можно узнать о версиях операционных систем, которые поддерживают установку программы электронных таблиц MS Office Excel 2003. На этот момент хотелось бы обратить внимание пользователей еще до приобретения пакета MS Office System 2003, поскольку как офисный пакет в целом, так и отдельно программа электронных таблиц MS Office Excel 2003 не могут быть установлены на компьютер с ОС MS Windows 98.

Программа электронных таблиц MS Office Excel 2003 позволяет производить вычисления с различными данными, выполнять математические расчеты и их анализ, а также представлять исходные данные и результаты расчетов в графическом виде. Благодаря наличию мощного арсенала математических, финансовых, статистических и логических функций, возможностей программы MS Excel бывает достаточно для профессиональной работы бизнесменов, бухгалтеров, менеджеров, системных и финансовых аналитиков, математиков и инженеров, студентов и преподавателей.

По мере появления новых версий MS Office совершенствуется не только пользовательский интерфейс программы MS Excel, но расширяется ее функциональность. Так, например, версия программы MS Office Excel 2003 поддерживает новые форматы ячеек и редактирование схем и изображений, возможности использования смарт-тегов и дополнительных средств задания формул, распознавание речи и рукописного ввода на некоторых языках.

Примечание

Перечень новых возможностей программы электронных таблиц MS Office Excel 2003 может быть продолжен, однако не все из них имеют непосредст-

венное отношение к решению задач оптимизации. Большинство рассмотренных в книге способов и приемов практических действий решения соответствующих задач могут быть без изменения использованы в предыдущих версиях этой программы, начиная с версии MS Excel 97. Именно по этой причине в тексте книги часто упоминается краткое название программы MS Excel. В то же время все примеры и упражнения иллюстрируются применительно к версии программы электронных таблиц MS Office Excel 2003, являющейся последней на момент написания книги.

Как отмечается многими экспертами, при выходе новых версий программы MS Excel часть функциональности предыдущих версий исчезает. Так произошло с возможностью вызова пользовательских функций из внешних библиотек, которая имеет принципиальное значение при рассмотрении материала в *IV части*. Не вдаваясь в причины подобного решения разработчиков, при решении задач оптимизации соответствующим способом целесообразно обратиться к версии MS Excel 97, в которой эта возможность реализована корректным образом. В любом случае изложенная в заключительной части информация представляет интерес для тех пользователей, которые являются программистами и заинтересованы в расширении возможностей программы MS Excel.

Если для адекватного понимания особенностей решения задач оптимизации необходимо знание теоретических основ математических моделей соответствующих задач, то для нахождения оптимальных решений конкретных задач в среде MS Excel — умение выполнять основные практические действия в среде MS Windows. Именно по этой причине предполагается, что читатели знакомы с такими действиями, как щелчок левой или правой кнопкой мыши, использование полос прокрутки, позиционирование курсора и одновременное нажатие некоторых клавиш клавиатуры. Хотя материал настоящей главы по полноте изложения не может заменить самоучитель, однако описанных действий вполне достаточно для решения практических задач.

2.2. Основные элементы рабочего интерфейса MS Office Excel 2003

В программе электронных таблиц MS Office Excel 2003 реализован ставший общепринятым стандарт на рабочий интерфейс, подобно другим программам офисного пакета Microsoft Office System 2003. После установки MS Office Excel 2003 на компьютер пользователя, что практически не вызывает трудностей у пользователей, запуск программы электронных таблиц в среде MS Windows 200x/XP приводит к появлению на экране соответствующего рабочего интерфейса (рис. 2.1).

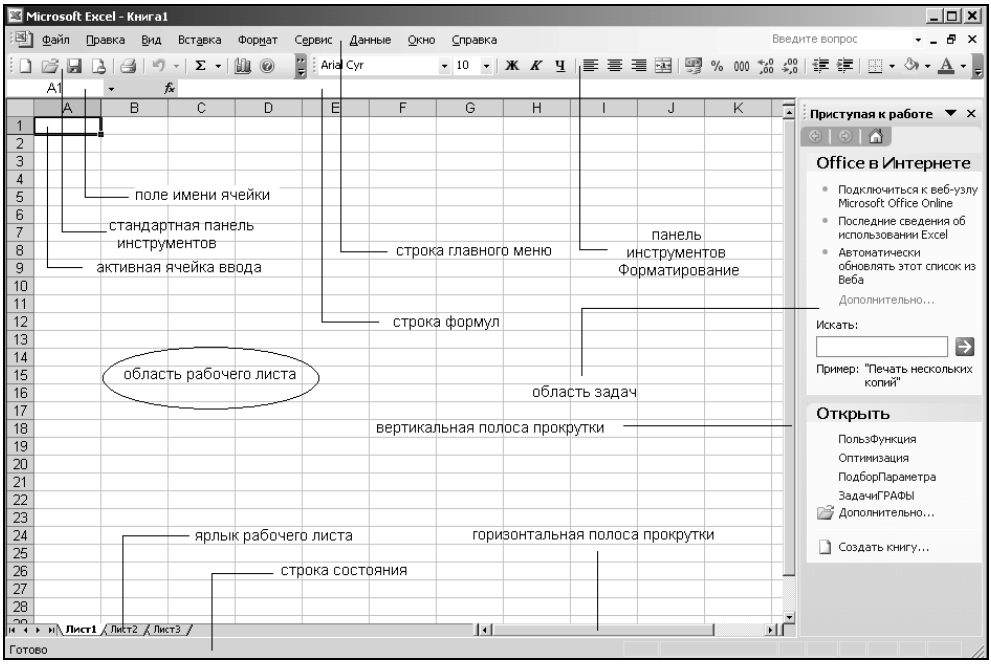


Рис. 2.1. Общий вид рабочего интерфейса программы электронных таблиц MS Office Excel 2003

Рабочий интерфейс программы электронных таблиц MS Office Excel 2003 состоит из различных элементов, основными из которых являются:

- ❒ главное меню;
- ❒ стандартная панель инструментов;
- ❒ панель инструментов Форматирование;
- ❒ строка ввода и редактирования формул;
- ❒ область рабочего листа;
- ❒ область задач;
- ❒ строка состояния.

Рассмотрим кратко назначение и основные функции каждого из этих элементов.

2.2.1. Главное меню

Главное меню программы электронных таблиц MS Office Excel 2003 выполнено в общепринятом стандарте и имеет следующий вид (рис. 2.2).



Рис. 2.2. Внешний вид главного меню программы

Отдельные пункты меню объединяют сходные операции, относящиеся ко всему документу, который применительно к программе электронных таблиц MS Office Excel 2003 получил специальное название — *книга*. Некоторые из пунктов главного меню содержат хорошо знакомые функции (открытие документа, вывод на печать листов и диаграмм, копирование в буфер и вставка из буфера различных фрагментов рабочего листа). Другие настолько специфичны, что требуют дополнительных усилий на изучение (опции генерации программного кода, проверка согласованности моделей, подключение дополнительных модулей).

2.2.2. Стандартная панель инструментов

Стандартная панель инструментов располагается ниже строки главного меню и имеет следующий вид (рис. 2.3). Стандартная панель инструментов содержит набор кнопок, которые дублируют наиболее часто выполняемые операции главного меню. Некоторые из кнопок могут быть недоступны для новой книги. Стандартная панель инструментов обеспечивает быстрый доступ к тем командам меню, которые, по мнению разработчиков программы электронных таблиц MS Office Excel 2003, выполняются пользователями наиболее часто.



Рис. 2.3. Внешний вид стандартной панели инструментов

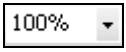
Пользователь может настроить внешний вид этой панели по своему усмотрению. Для этого следует выполнить операцию главного меню: **Сервис | Настройка** или воспользоваться самой крайней справа кнопкой этой панели: **Параметры панелей инструментов**. Данная кнопка позволяет отображать кнопки стандартной панели и панели форматирования на одной строке или на двух отдельных строках, а также добавить или удалить кнопки со стандартной панели инструментов. Последний способ представляется наиболее удобным для того, чтобы показать или скрыть различные кнопки данной и других панелей инструментов.

Назначение отдельных кнопок стандартной панели инструментов приводится в табл. 2.1.

Таблица 2.1. *Назначение кнопок стандартной панели инструментов*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	**Создать**	Создает новый документ (книгу) программы электронных таблиц MS Office Excel 2003, которому по умолчанию присваивается имя Книга 1
	**Открыть**	Вызывает стандартное диалоговое окно открытия документа с возможностью выбора файла для дальнейшего редактирования
	**Сохранить**	Сохраняет редактируемый документ в файле под текущим именем и стандартным расширением xls
	**Разрешение (Неограниченный доступ)**	Позволяет управлять правами доступа к редактируемому документу, требует установки дополнительного компонента – клиента управления правами Windows (WRM)
	**Печать (Fax)**	Позволяет распечатать текущий рабочий лист на принтере или отправить его содержание по факсу
	**Предварительный просмотр**	Позволяет визуализировать текущий рабочий лист в дополнительном окне, что рекомендуется выполнять перед его печатью на принтере
	**Орфография**	Проверяет содержание текущего рабочего листа на наличие орфографических ошибок
	**Справочные материалы**	Позволяет вызвать окно с поиском контекстно-зависимой справкой для заданной фразы
	**Вырезать**	Удаляет содержание выделенных ячеек и помещает их в буфер обмена
	**Копировать**	Копирует содержание выделенных ячеек и помещает их в буфер обмена
	**Вставить**	Позволяет вставлять содержание буфера обмена в выделенную область рабочего листа в выбранном формате и выполнять специальную вставку
	**Формат по образцу**	Позволяет форматировать выделенные ячейки рабочего листа по заданному образцу

Таблица 2.1 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	**Отменить**	Отменяет выполнение последнего действия по редактированию рабочего листа
	**Вернуть**	Возвращает рабочий лист в состояние до отмены выполнения последнего по редактированию рабочего листа
	**Добавить гиперссылку**	Позволяет добавить в ячейку ссылку на внешний файл в форме гиперссылки
	**Автосумма**	Позволяет быстро вызвать наиболее часто используемые функции (суммирования, среднее, максимум, минимум и др.)
	**Сортировка по возрастанию**	Сортирует выделенный диапазон значений по возрастанию их содержимого
	**Сортировка по убыванию**	Сортирует выделенный диапазон значений по убыванию их содержимого
	**Мастер диаграмм**	Вызывает Мастера построения диаграмм
	**Рисование**	Отображает или скрывает панель инструментов Рисование
	**Масштаб**	Позволяет изменять масштаб изображения текущего рабочего листа
	**Справка: Microsoft Excel**	Открывает дополнительное окно с возможностью поиска справочной информации по ключевой фразе
	**Параметры панелей инструментов**	Позволяет изменять стиль отображения кнопок панелей инструментов на одной строке или на двух отдельных строках, а также добавить или удалить кнопки со стандартной панели инструментов

Выполнение отдельных операций с использованием тех или иных кнопок стандартной панели инструментов рассматривается далее при редактировании ячеек и рабочих листов.

2.2.3. Панель инструментов Форматирование

Панель инструментов Форматирование по умолчанию располагается в одной строке со стандартной панелью инструментов. Однако это представляется не совсем удобным. Поэтому можно воспользоваться крайней справа кнопкой **Параметры панелей инструментов** и расположить данную панель отдельно под стандартной панелью инструментов. В этом случае она будет иметь следующий вид (рис. 2.4).



Рис. 2.4. Внешний вид панели инструментов Форматирование

Панель инструментов Форматирование содержит набор кнопок, которые дублируют наиболее часто выполняемые операции главного меню, выполняемые при форматировании выделенных ячеек. Назначение отдельных кнопок панели инструментов Форматирование приводится в табл. 2.2.

Таблица 2.2. *Назначение кнопок панели инструментов Форматирование*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
Arial Cyr	**Шрифт**	Позволяет выбрать шрифт текста из вложенного списка
10	**Размер**	Позволяет выбрать размер шрифта из вложенного списка
Ж	**Полужирный**	Изображает содержимое выделенных ячеек полужирным шрифтом
К	**Курсив**	Изображает содержимое выделенных ячеек курсивом
Ч	**Подчеркнутый**	Изображает содержимое выделенных ячеек подчеркнутым шрифтом
	По левому краю	Выравнивает содержимое выделенных ячеек по левому краю
	По центру	Выравнивает содержимое выделенных ячеек по центру
	По правому краю	Выравнивает содержимое выделенных ячеек по правому краю

Таблица 2.2 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	**Объединить и поместить в центре**	Объединяет выделенные ячейки и размещает содержимое верхней левой ячейки по центру в нижней части образованной области
	**Денежный формат**	Устанавливает финансовый формат для выделенных ячеек
	**Процентный формат**	Устанавливает процентный формат для выделенных ячеек
	**Формат с разделителями**	Устанавливает финансовый формат с разделителями для выделенных ячеек
	**Увеличить разрядность**	Позволяет увеличить разрядность на один десятичный знак для выделенных ячеек
	**Уменьшить разрядность**	Позволяет уменьшить разрядность на один десятичный знак для выделенных ячеек
	**Уменьшить отступ**	Позволяет уменьшить отступ текста на установленный интервал от левого края для выделенных ячеек
	**Увеличить отступ**	Позволяет увеличить отступ текста на установленный интервал от левого края для выделенных ячеек
	**Границы**	Позволяет изменить изображения границ выделенных ячеек
	**Цвет заливки (желтый)**	Позволяет изменить цвет фона выделенных ячеек
	**Цвет шрифта (красный)**	Позволяет изменить цвет шрифта выделенных ячеек
	**Параметры панелей инструментов**	Позволяет изменять стиль отображения кнопок панелей инструментов на одной строке или на двух отдельных строках, а также добавить или удалить кнопки с панели инструментов Форматирование

Выполнение отдельных операций с использованием тех или иных кнопок панели инструментов Форматирование также рассматривается далее при редактировании ячеек и рабочих листов.

2.2.4. Строка ввода и редактирования формул

Строка ввода и редактирования формул располагается ниже панели инструментов Форматирование и имеет следующий вид (рис. 2.5).

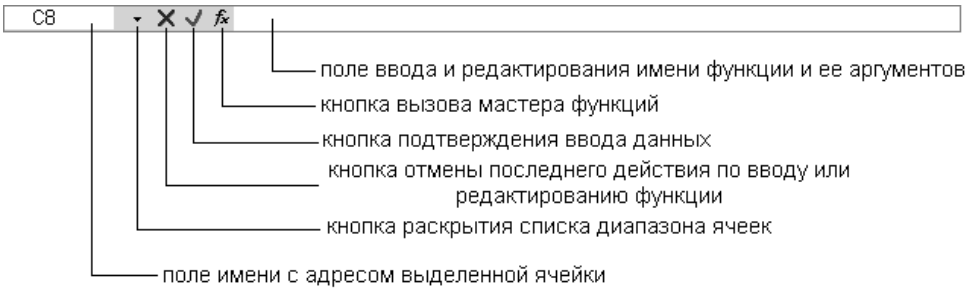


Рис. 2.5. Внешний вид строки ввода и редактирования формул

Левую часть этой строки занимает поле с адресом выделенной ячейки, в которую предполагается поместить формулу. Сама формула записывается в поле, расположенном в правой части строки. Непосредственно слева от поля ввода и редактирования формул располагается кнопка вызова мастера функций (**Вставка функции**). Нажатие этой кнопки приводит к появлению специального диалогового окна, позволяющего выбрать нужную функцию из предложенного списка и задать для нее все необходимые аргументы и параметры. Эта возможность программы электронных таблиц MS Office Excel 2003 является одной из наиболее важных ее особенностей, которая будет постоянно использоваться на всем протяжении книги.

Более подробные рекомендации по вводу и редактированию формул приводятся далее в этой главе при рассмотрении операций по вводу и форматированию данных в ячейках.

2.2.5. Область рабочего листа

Область рабочего листа занимает основную часть рабочего интерфейса программы электронных таблиц MS Office Excel 2003, в которой визуализируются различные представления модели проекта. По умолчанию окно диаграммы располагается в правой части рабочего интерфейса, однако его расположение и размеры также можно изменить. При разработке нового проекта, если не был использован мастер проектов, окно диаграммы представляет собой чистую область, не содержащую никаких элементов модели (рис. 2.6).

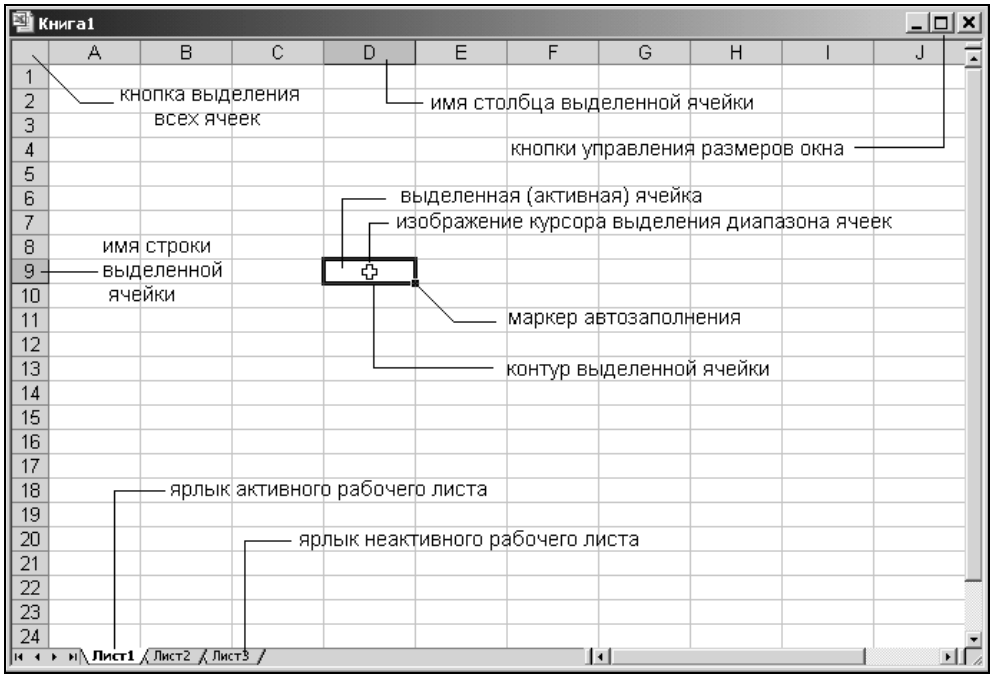
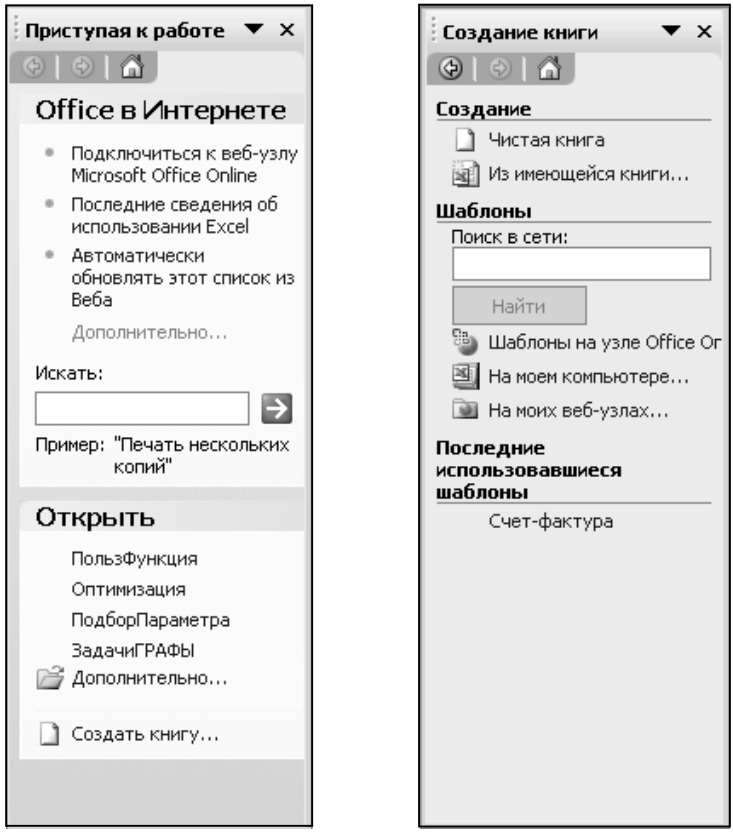


Рис. 2.6. Внешний вид области рабочего листа

Название диаграммы, которая располагается в данном окне, указывается в строке заголовка программы (самая верхняя строка программы) или, если окно не развернуто во весь экран, в строке заголовка окна диаграммы. Одновременно в окне диаграммы могут присутствовать несколько диаграмм, однако активной может быть только одна из них. Например, на рис. 2.6 активной является диаграмма последовательности, хотя имеются и другие диаграммы. Переключение между диаграммами можно осуществить выбором нужного представления на стандартной панели инструментов либо через пункт меню **Window** (Окно). При активизации отдельного вида диаграммы изменяется внешний вид специальной панели инструментов, которая настраивается под конкретный вид диаграммы.

2.2.6. Область задач

Область задач по умолчанию присутствует в окне рабочего интерфейса после запуска программы электронных таблиц MS Office Excel 2003 (рис. 2.7). Для увеличения области рабочего листа область задач зачастую убирается с экрана. При необходимости область задач может быть восстановлена посредством выполнения операции главного меню: **Вид** | **Область задач** или нажатием управляющих клавиш: <Ctrl>+<F1>.



а *б*

Рис. 2.7. Внешний вид окна документации

Область задач способна изменять свой внешний вид в зависимости от выполняемых в текущий момент действий пользователя. Если пользователь только приступает к работе, то в этой области ему предлагается выбрать некоторые стандартные операции по обновлению программы электронных таблиц MS Office Excel 2003 или один из ранее редактируемых документов (рис. 2. 7, *а*). При создании новой книги область задач будет иметь другой внешний вид (рис. 2.7, *б*).

2.3. Основные приемы работы с электронной таблицей

После запуска программы MS Office Excel по умолчанию создается новый документ с именем Книга 1 и пустым содержанием. Поэтому в среде MS Office Excel каждый отдельный документ называется *книгой*. В дальнейшем

эти два понятия будут использоваться как синонимы. Каждая книга состоит из *рабочих листов* или просто листов, которые по умолчанию имеют последовательно возрастающие имена: Лист 1, Лист 2, Лист 3. Максимально возможное количество листов в книге равно 255. Однако по умолчанию для каждой книги задаются три пустых рабочих листа, переход между которыми осуществляется посредством щелчка на ярлыке соответствующей вкладки в нижней части области рабочего листа.

Примечание

Для изменения количества рабочих листов, заданного по умолчанию, следует выполнить операцию главного меню: **Сервис** | **Параметры**, в появившемся диалоговом окне перейти на вкладку **Общие** и установить нужное значение в поле **Листов в новой книге**.

Каждый рабочий лист состоит из *ячеек*, которые изображаются в форме небольших прямоугольников. Тем самым область рабочего листа становится похожей на таблицу, откуда и происходит название программы электронных таблиц MS Excel. Каждая ячейка рабочего листа имеет свой *адрес*, который по умолчанию состоит из двух частей: буквы и цифры. Буква соответствует горизонтальному расположению ячейки на листе, а цифра — ее вертикальному расположению. Самая верхняя слева ячейка имеет адрес **A1**, справа от нее находится ячейка **B1**, далее следует ячейка с адресом **C1** и т. д. Все они располагаются последовательно до ячейки **IV1**, которая по умолчанию скрыта от пользователя. Аналогично вниз от ячейки **A1** располагаются ячейки с адресами: **A2**, **A3**, **A4** и т. д. Все они располагаются последовательно до ячейки **A65536**, которая также по умолчанию скрыта от пользователя.

Примечание

Заинтересованные читатели могут убедиться, что при наличии 256 горизонтальных адресов и 65 536 вертикальных общее количество ячеек на каждом рабочем листе составляет 256 × 65 536 = 16 777 216. Этого числа оказывается вполне достаточно для решения большинства, если не всех, практических задач. Для тех, у кого возникают сомнения в этом, можно порекомендовать оценить общее время заполнения всех ячеек одного рабочего листа, в предположении, например, что пользователь на ввод данных в одну ячейку тратит 1 секунду (весьма и весьма оптимистичное предположение). В этой связи особую значимость приобретают такие возможности программы MS Excel, как автозаполнение ячеек и копирование диапазона ячеек.

Кроме ячеек на рабочих листах могут размещаться *диаграммы*, содержащие изображения данных в различной графической форме. В частности, данные

могут быть представлены в форме графиков функций, что оказывается весьма удобным для визуального анализа результатов решения некоторых задач оптимизации. Особенности построения диаграмм рассматриваются далее в этой главе.

Таким образом, общая структура документов программы электронных таблиц MS Excel может быть представлена в следующей графической форме:

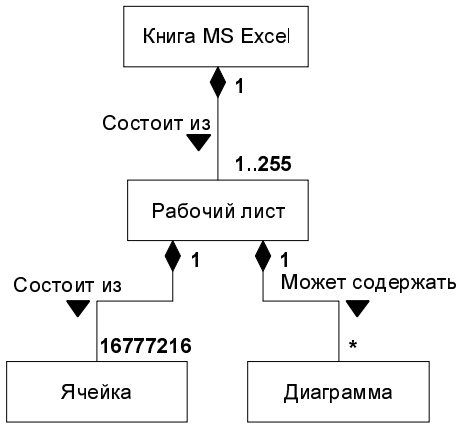


Рис. 2.8. Общая схема структуры документов программы MS Excel в форме диаграммы классов языка UML

Заметим, что изображенный на рис. 2.8 тип отношений соответствует композиции классов языка UML, которая служит для представления взаимосвязи типа "часть-целое".

2.3.1. Ввод и форматирование данных

Для ввода тех или иных данных в ячейку ее необходимо предварительно активизировать или выделить посредством щелчка на ней левой кнопкой мыши или с помощью клавиш перемещения (←, ↑, →, ↓) на клавиатуре. *Выделенная* или *активная* ячейка имеет на рабочем листе полужирную рамку (см. рис. 2.6), а ее адрес указывается в поле в левой части строки ввода и редактирования формулы (см. рис. 2.5). При этом каждая ячейка рабочего листа может содержать либо некоторые данные, либо формулу. В последнем случае она является вычислимой. Ввод данных в ячейку осуществляется непосредственно с клавиатуры, при этом форма представления данных определяется *форматом* соответствующей ячейки.

По умолчанию для всех ячеек используется формат Общий, который предназначен для отображения данных как текстовых, так и числовых значений

произвольного типа. Изменить формат отображения информации ячейки можно с помощью операции главного меню: **Формат | Ячейки**, с помощью аналогичного пункта контекстного меню, вызываемого щелчком правой кнопкой мыши, или с помощью комбинации управляющих клавиш: <Ctrl>+<1>.

Примечание

Поскольку для решения большинства рассматриваемых в книге задач оптимизации оказывается вполне достаточно общего формата отображения содержимого ячеек, более детальное описание возможностей других форматов ячеек выходит за пределы настоящей тематики. Заинтересованные читатели при необходимости могут обратиться за более подробной информацией к встроенной справочной системе программы MS Excel.

При необходимости изменить информацию в той или иной ячейке нужно выделить данную ячейку и начать ввод требуемых данных. После окончания ввода или редактирования данных следует нажать клавишу <Ввод> на клавиатуре. Редактируемый рабочий лист сохраняется в рамках всего документа во внешнем файле аналогичным для всех программ офисного пакета способом.

При непосредственном вводе данных в ячейку указанным ранее способом вся содержащаяся в ней информация заменяется вводимой. Это бывает не всегда удобно, особенно в случаях, когда требуется изменить один или несколько символов данных в ячейке. Для редактирования отдельных значений содержащейся в ячейке информации следует либо щелкнуть левой кнопкой мыши в поле ввода формулы, которое дублирует содержание данной ячейки, либо нажать клавишу <F2>. При этом содержимое ячейки сохранится и будет доступно для последующего редактирования.

Кроме единственной ячейки программа MS Excel позволяет выделять множество ячеек, которое называется *диапазоном* ячеек. Диапазон ячеек представляется в форме указания адресов первой и последней ячеек, разделенных двоеточием. Например, диапазон вертикально расположенных ячеек, первой из которых является ячейка с адресом A1, а последняя — ячейка с адресом A10, записывается в виде: A1:A10. Очевидно, ячейка с адресом A4 входит в указанный диапазон, а ячейка с адресом A12 — нет. Кроме одномерного диапазона ячеек можно выделить и двумерный диапазон. В этом случае первой ячейкой диапазона будет крайняя левая верхняя ячейка, а последней — крайняя правая нижняя ячейка.

Выделить диапазон ячеек в программе MS Excel можно как с помощью мыши, так и клавиатуры. Для выделения диапазона ячеек с помощью *мыши* необходимо выделить первую (или последнюю) ячейку диапазона и, удерживая

нажатой левую кнопку мыши, переместить курсор в область последней (или первой) ячейки диапазона. После чего левую кнопку мыши следует отпустить.

Для выделения диапазона ячеек с помощью *клавиатуры* необходимо выделить первую (или последнюю) ячейку диапазона и, удерживая нажатой клавишу <Shift>, с помощью клавиш перемещения (←, ↑, →, ↓) на клавиатуре переместить курсор в область последней (или первой) ячейки диапазона. Выделенный диапазон ячеек имеет более темный цвет их заполнения (рис. 2.9).

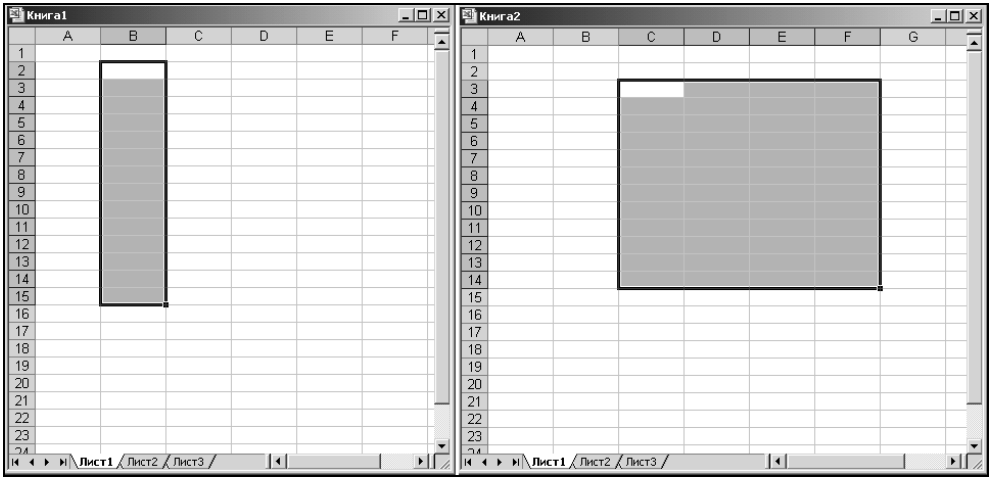


Рис. 2.9. Результат выделения одномерного диапазона ячеек **B2:B15** (Книга1) и двумерного диапазона ячеек **C3:F14** (Книга 2)

Для выделения всех ячеек некоторого столбца или некоторой строки следует щелкнуть на имени соответствующего столбца или строки. Для выделения всех ячеек текущего рабочего листа следует щелкнуть на самой верхней слева кнопке рабочей области (см. рис. 2.6.).

При вводе данных, имеющих некоторую регулярную структуру, например, одинаковые значения или последовательный ряд чисел, следует воспользоваться возможностью автозаполнения ячеек. Существует несколько способов, простейший из них основан на выполнении следующих действий.

Способ автозаполнения ячеек № 1:

1. Выделить первую (или последнюю) ячейку диапазона автозаполнения.
2. Ввести в первую ячейку некоторые данные, которые должны быть помещены во все ячейки диапазона (например, число 10).

3. После окончания ввода данных в первую (или последнюю) ячейку следует позиционировать курсор мыши на маркере автозаполнения данной ячейки (см. рис. 2.6).
4. Выполнить щелчок левой кнопкой мыши на маркере автозаполнения и, удерживая нажатой левую кнопку мыши, переместить курсор в область последней (или первой) ячейки диапазона. После чего левую кнопку мыши следует отпустить.

В результате выполнения указанной последовательности действий все ячейки диапазона будут заполнены данными первой (или последней) ячейки диапазона, например, числом 10.

Для автозаполнения диапазона ячеек данными с регулярной структурой, например, последовательный ряд чисел, необходимо выполнить следующие действия.

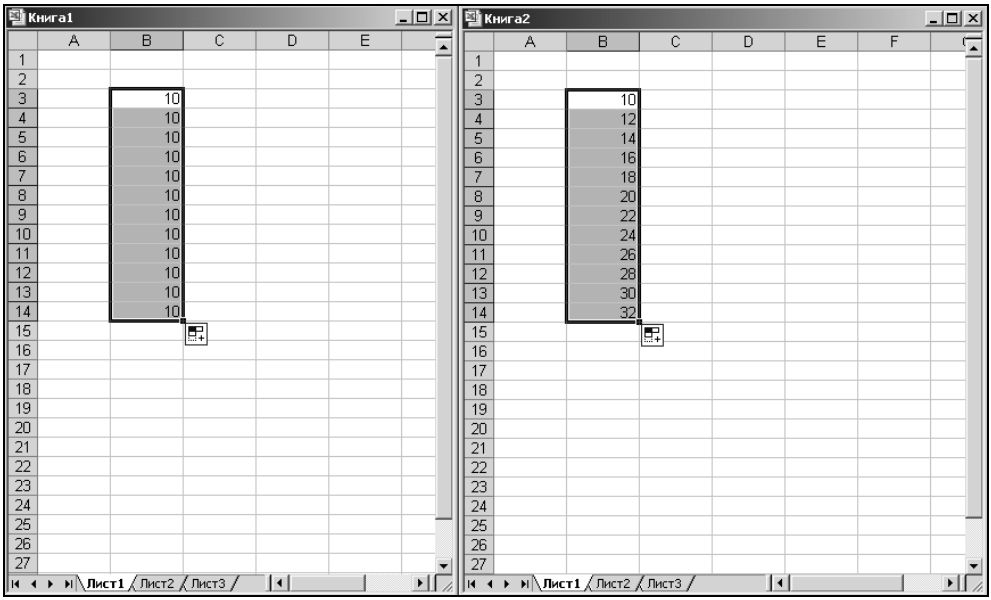
Способ автозаполнения ячеек № 2:

1. Выделить первую (или последнюю) ячейку диапазона автозаполнения.
2. Ввести в первую ячейку некоторые данные, которые являются первым значением из некоторого последовательного диапазона (например, число 10).
3. Выделить соседнюю горизонтальную или вертикальную ячейку диапазона автозаполнения.
4. Ввести во вторую ячейку некоторые данные, которые являются следующим значением из некоторого последовательного диапазона (например, число 12).
5. Выделить обе заполненные ячейки одним из указанных ранее способом.
6. Позиционировать курсор мыши на маркере автозаполнения выделенной пары ячеек.
7. Выполнить щелчок левой кнопкой мыши на маркере автозаполнения и, удерживая нажатой левую кнопку мыши, переместить курсор в область последней (или первой) ячейки диапазона. После чего левую кнопку мыши следует отпустить.

В результате выполнения указанной последовательности действий все ячейки диапазона будут заполнены данными из некоторого последовательного диапазона от числа 10 до числа 32 (рис. 2.10).

Небольшой маркер в нижней части выделенного диапазона, который получил специальное название *смарт-тега*, позволяет выбрать эти и другие режимы автозаполнения. Для этого достаточно позиционировать курсор в графической области смарт-тега и щелкнуть левой кнопкой мыши на небольшой стрелке в правой его части. После чего выбрать необходимый режим автозаполнения

или копирования из предложенного списка. Другие возможности операции автозаполнения ячеек будут рассмотрены далее по мере необходимости выполнения тех или иных действий в процессе решения задач оптимизации.



а *б*

Рис. 2.10. Результат автозаполнения одномерного диапазона ячеек **B3:B14** в Книге 1 числом 10 (*а*) и последовательностью чисел от 10 до 32 в Книге 2 (*б*)

Для удаления содержимого некоторой ячейки или диапазона ячеек следует выделить требуемую ячейку или диапазон и нажать клавишу <Delete>. В этом случае вся информация будет удалена. Для восстановления удаленной из ячеек информации следует воспользоваться операцией главного меню: **Правка** | **Отменить очистку**. Аналогичного результата можно достичь с помощью соответствующей кнопки стандартной панели инструментов (см. табл. 2.1) или комбинации клавиш: <Ctrl>+<Z>.

2.3.2. Копирование и перенос данных ячеек и рабочих листов

Программа MS Excel позволяет копировать данные из одной ячейки или целого диапазона ячеек. Для копирования данных из одной ячейки ее необходимо выделить. Далее можно воспользоваться одним из нескольких способов. Для копирования данных с помощью мыши можно щелкнуть на кнопке

Копировать стандартной панели инструментов, выделить ячейку, в которую необходимо вставить данные, и нажать кнопку **Вставить** этой же панели (см. табл. 2.1). Для копирования данных с помощью клавиатуры следует нажать комбинацию клавиш <Ctrl>+<C> или <Ctrl>+<Insert>, выделить ячейку, в которую необходимо вставить данные, и нажать комбинацию клавиш <Ctrl>+<V> или <Shift>+<Insert>. После копирования содержимого выделенной ячейки в буфер обмена MS Windows данная ячейка будет выделена мерцающей пунктирной линией.

Примечание

При необходимости удаления информации из выделенной ячейки вместо кнопки **Копировать** стандартной панели инструментов следует воспользоваться кнопкой **Вырезать** этой же панели либо комбинацией клавиш: <Ctrl>+<X> или <Shift>+<Delete>.

Для копирования данных диапазона ячеек предварительно одним из описанных ранее способов необходимо выделить исходный диапазон ячеек. После этого с помощью кнопок стандартной панели инструментов, операции главного меню или комбинации клавиш содержимое ячеек выделенного диапазона копируется в буфер обмена MS Windows, о чем будет свидетельствовать мерцающая пунктирная линия или "змейка", ограничивающая исходный диапазон ячеек. Далее следует выделить первую ячейку или весь требуемый диапазон ячеек, куда нужно поместить копируемые данные. В завершение следует вставить данные из буфера обмена в требуемый диапазон ячеек.

Примечание

Условие точного равенства количества ячеек при выполнении операции копирования исходного и конечного диапазонов не является обязательным. Однако корректное копирование происходит именно в этом случае, поскольку в других ситуациях программа MS Excel либо заполняет новый диапазон повторяющимися данными, либо ограничивает диапазон вставляемых данных размерами исходного диапазона. В случае удаления данных из исходного диапазона ячеек вместо операции **Копировать** следует выполнить операцию **Вырезать** или воспользоваться соответствующими комбинациями клавиш.

Существует также дополнительный способ переноса данных выделенных ячеек рабочего листа с помощью мыши. Для этого следует выделить некоторый диапазон ячеек одним из ранее описанных способов и позиционировать курсор мыши на внешней рамке выделенного диапазона. В этом случае курсор должен изменить свою форму и превратиться в широкое перекрестие с небольшими расходящимися стрелками. После этого следует нажать левую

кнопку мыши и, удерживая ее нажатой, перетащить выделенный диапазон ячеек в новое место рабочего листа. После чего отпустить левую кнопку мыши. В результате выполнения этих действий, данные из исходного выделенного диапазона ячеек будут удалены и помещены в новый диапазон ячеек.

Примечание

Если при выполнении переноса данных способом перетаскивания удерживать нажатой клавишу <Ctrl>, то будет выполнено копирование данных выделенного диапазона ячеек. Очевидно, данный способ копирования или переноса данных удобно использовать в пределах одного рабочего листа.

Копирование данных отдельных строк, столбцов или всего рабочего листа осуществляется аналогично, за исключением того, что выделение исходного диапазона выполняется с помощью щелчка на имени соответствующей строки, столбца или всех ячеек текущего рабочего листа. Если требуемая ячейка или диапазон ячеек находится на другом рабочем листе этой же книги, то предварительно следует открыть соответствующий рабочий лист. Если рабочий лист находится в другом документе (книге), то, соответственно, предварительно следует открыть данный документ и нужный рабочий лист.

2.3.3. Ввод, редактирование и копирование формул

Базовым средством обработки содержащихся в таблицах данных и их анализа являются формулы, которые позволяют выполнять разнообразные математические, логические, финансовые, статистические и другие операции. Без знания и использования формул и встроенных функций программа MS Excel превращается в обычный редактор. С другой стороны, именно формулы расширяют базовые возможности программы MS Excel, делая ее мощным средством выполнения сложных расчетов.

В общем случае *формула* в программе MS Excel представляет собой некоторое выражение, определяющее действия над данными той или иной ячейки рабочего листа. Каждая ячейка рабочего листа может содержать данные или формулу. В последнем случае ячейку часто называют *вычислимой*, тем самым подчеркивая принципиальное отличие между данными и формулой для их обработки. В свою очередь программа MS Excel по умолчанию настроена таким образом, что в вычислимых ячейках отображается результат выполнения записанной в ней формулы в форме данных.

При записи формул в вычислимые ячейки можно использовать в качестве аргументов или расчетных параметров данные, которые хранятся в других обычных или расчетных ячейках. С этой целью в формулах могут быть

использованы ссылки на адреса тех или иных ячеек текущего или другого рабочего листа, а также редактируемой или другой существующей книги. Одной из наиболее интересных особенностей программы MS Excel является возможность копирования содержимого вычислимых ячеек с автоматическим изменением относительных адресов используемых в формулах ячеек.

Существуют два основных способа задания в вычислимой ячейке некоторой формулы. Первый способ связан с использованием специального *мастера функций*, который позволяет выбрать одну из встроенных функций из предлагаемого пользователю списка и специфицировать ее аргументы. Второй способ предполагает непосредственный ввод в вычислимую ячейку выражения для формулы, которое должно удовлетворять определенным синтаксическим правилам.

Примечание

Существуют также дополнительные способы задания некоторых функций, которые могут быть выполнены, например, с помощью кнопки **Автосумма** стандартной панели инструментов. Поскольку эти способы дублируют основные и служат для удобства пользователя, они подробно не рассматриваются.

Для определенности рассмотрим пример вычисления суммы первых 10 натуральных чисел двумя основными способами. Для этого на отдельном рабочем листе предварительно в ячейки **A1:A10** с использованием второго способа автозаполнения следует ввести натуральные числа от 1 до 10. После этого необходимо выполнить следующую последовательность действий.

Способ задания формулы № 1:

1. Выделить вычислимую ячейку, в которую предполагается ввести некоторую формулу. В рассматриваемом примере пусть это будет ячейка **A11**.
2. Нажать кнопку **Вставка функции**, расположенную в строке ввода и редактирования формул (см. рис. 2.5), или выполнить операцию главного меню: **Вставка** | **Функция**, в результате чего будет открыто диалоговое окно мастера функций (рис. 2.11).

   При необходимости можно выполнить поиск нужной функции или выбрать категорию функций из вложенного списка. Применительно к рассматриваемому примеру требуемая функция присутствует в списке с именем **Выберите функцию**. Такой является функция с именем СУММ, которая стоит первой в списке. Дополнительно можно получить справку по выделенной функции, нажав на соответствующей ссылке в нижней части окна. В результате появляется дополнительное окно с информацией о данной функции с примерами правильной записи ее аргументов (рис. 2.12).

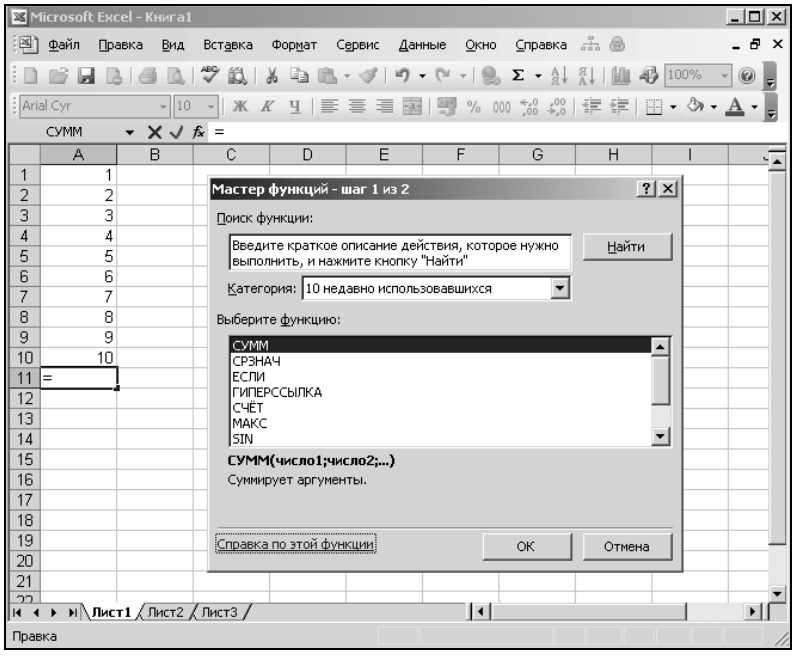


Рис. 2.11. Задание формулы с помощью мастера функций (шаг 1 из 2)

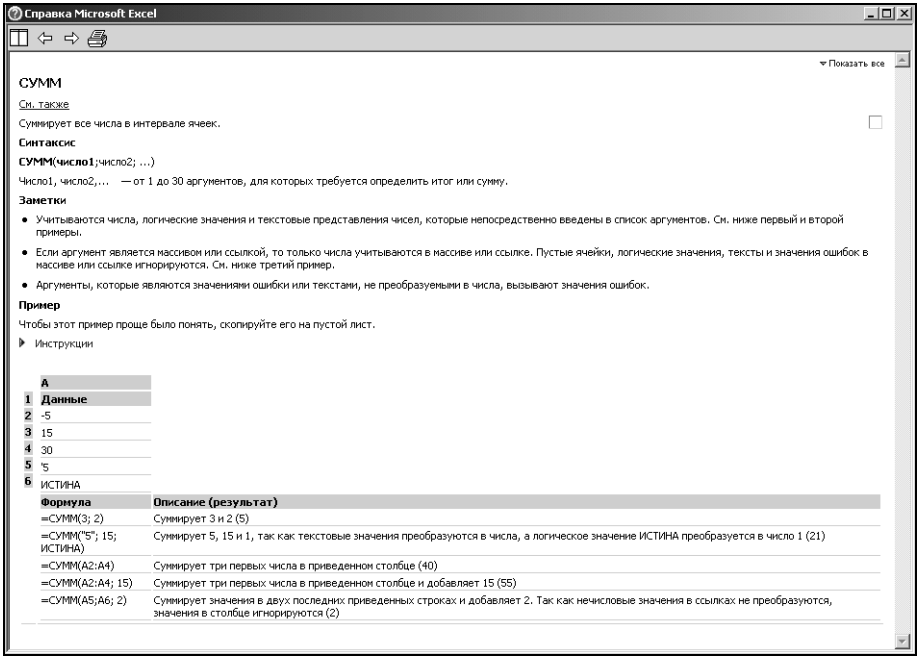


Рис. 2.12. Окно со справочной информацией по функции СУММ

3. Выбрать необходимую функцию из предлагаемого списка и нажать кнопку **ОК**. Внешний вид окна мастера функций изменится (рис. 2.13). В отдельных полях ввода появившегося окна следует выбрать ячейки, в которых содержатся данные, являющиеся аргументами выбранной функции.

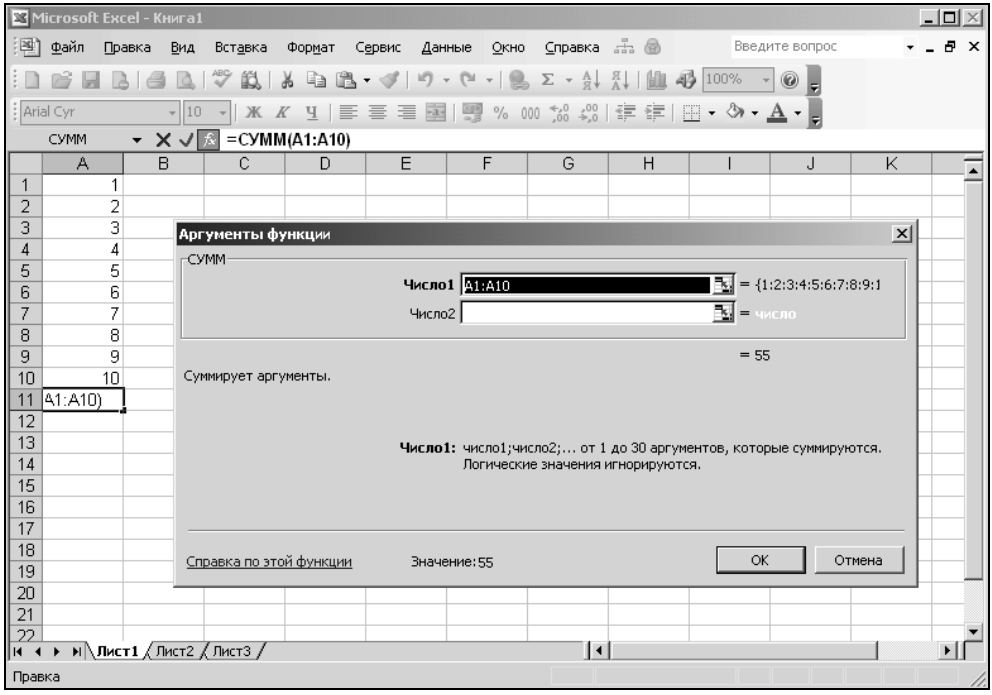
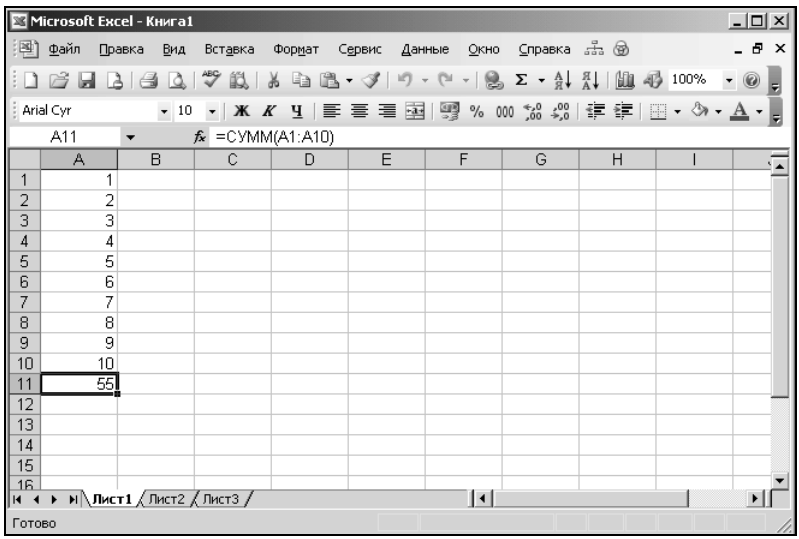


Рис. 2.13. Задание аргументов формулы с помощью мастера функций (шаг 2 из 2)

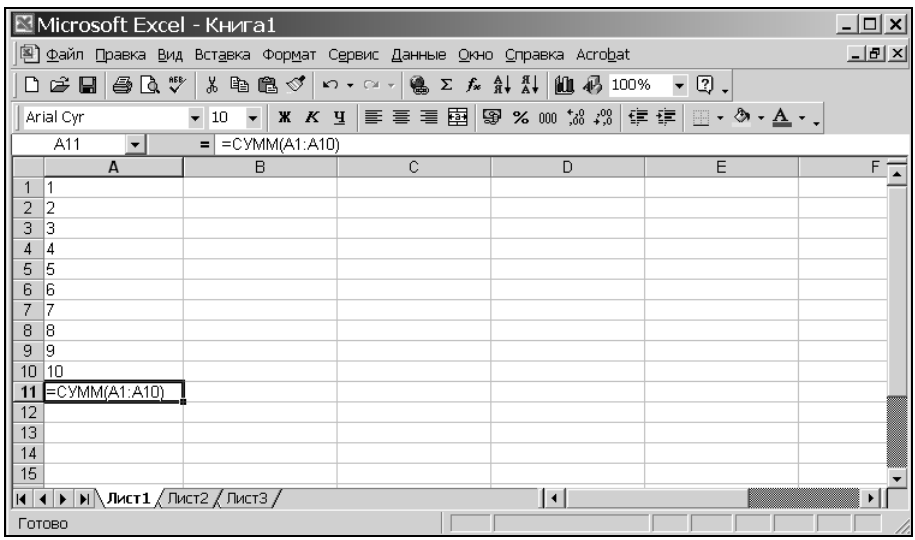
Применительно к рассматриваемому примеру можно согласиться с диапазоном аргументов, предлагаемым программой MS Excel по умолчанию, поскольку аргументами функции суммирования СУММ является диапазон ячеек **A1:A10**. В этом же окне сразу отображается результат суммирования — число 55. Вводимая формула отображается также в строке ввода и редактирования формул. Для завершения ввода формулы следует нажать кнопку **ОК**. После ввода функции суммирования СУММ в вычислимую ячейку **A11** рабочий лист будет иметь следующий вид (рис. 2.14, *а*).

Хотя в вычислимой ячейке **A11** хранится соответствующая формула, которая отображается в строке ввода и редактирования формул, по умолчанию в самой ячейке отображается результат выполнения этой формулы. В отдельных случаях для проверки правильности задания формул в группе ячеек желательно отобразить в них сами формулы. Для отображения формул в ячейках

рабочего листа следует выполнить операцию главного меню: **Сервис | Параметры** и в открывшемся диалоговом окне на вкладке **Вид** отметить флажком строку выбора **формулы** в группе **Параметры окна**. После изменения способа отображения формул указанным способом этот же рабочий лист будет иметь следующий вид (рис. 2.14, *б*).



а



б

Рис. 2.14. Результат ввода функции суммирования в ячейку **A11** при отображении результата выполнения формулы (*а*) и собственно формулы (*б*)

Заданная ранее формула может быть изменена либо вводом новой формулы, либо редактированием существующей. Редактировать существующую формулу можно либо в строке ввода и редактирования, либо непосредственно в вычислимой ячейке.

Для редактирования формулы в строке ввода и редактирования формул достаточно выполнить щелчок в поле ввода и редактирования имени функции и ее аргументов. При редактировании программа MS Excel выполняет цветовое выделение ячеек с аргументами функции, а также позволяет визуально контролировать количество открывающих и закрывающих скобок.

Для непосредственного редактирования формулы в вычислимой ячейке достаточно выделить эту ячейку и нажать клавишу <F2>. В этом случае изменится вид соответствующей ячейки — вместо данных в ней будет отображаться соответствующая формула, а форма курсора укажет на возможность ее редактирования.

При задании более сложных формул требуется явно задавать диапазоны ячеек с аргументами соответствующих функций. Для этой цели служит кнопка второго диалогового окна мастера функций, расположенная в правой части поля ввода аргументов (см. рис. 2.13). После нажатия кнопки задания аргумента функции появится отдельное изображение строки ввода аргументов функции (рис. 2.15).

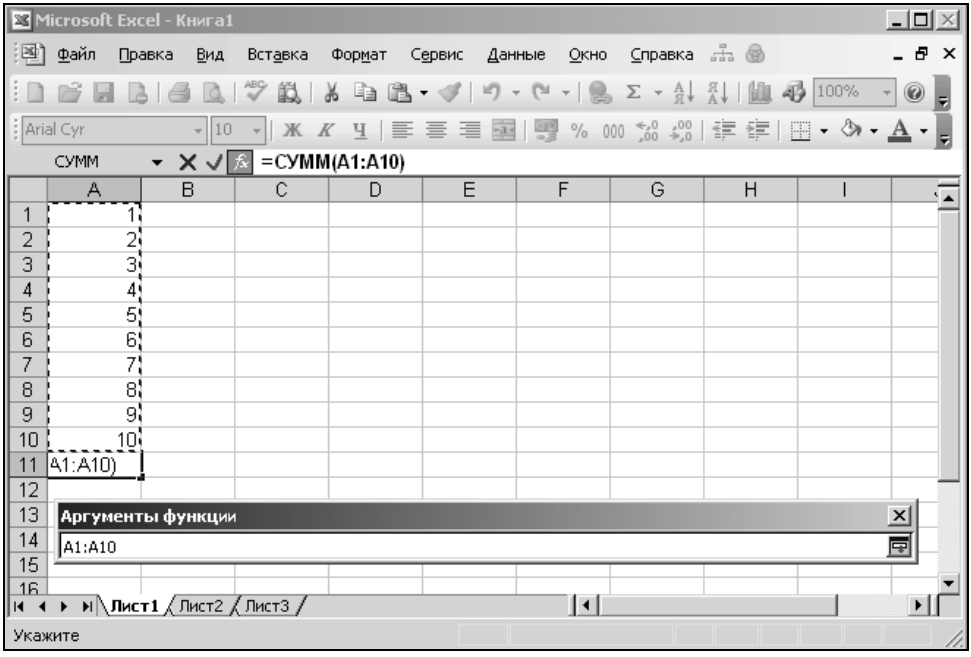


Рис. 2.15. Задание аргументов формулы
с помощью специального окна мастера функций (шаг 2 из 2)

Для задания отдельной ячейки в качестве аргумента функции достаточно просто ее выделить на рабочем листе и нажать клавишу <Ввод>. Для задания диапазона ячеек необходимо выделить соответствующий диапазон с помощью мыши или клавиатуры и также нажать клавишу <Ввод>. Если диапазон ячеек не является непрерывным, следует выделять отдельные его ячейки, удерживая нажатой клавишу <Crtl>. Рамка выделяемого в качестве аргументов диапазона ячеек приобретет вид мерцающей пунктирной линии или "змейки" аналогично выполнению операций копирования и вырезания.

Второй способ задания формул в вычислимые ячейки основан на непосредственном вводе выражения, служащего для выполнения соответствующих расчетов. Для определенности рассмотрим в качестве примера ввод следующей функции: $f(x) = 2x^2 - 3x + 10$, которая используется для расчета значения функции одного из значений независимой переменной x. С этой целью на существующем или отдельном листе выделим ячейку **A2**, предполагая, что отдельное значение независимой переменной будет вводиться в ячейку **A1**.

Способ задания формулы № 2:

1. Выделить вычислимую ячейку, в которую предполагается ввести некоторую формулу. В рассматриваемом примере пусть это будет ячейка **A2**.
2. Начать ввод формулы в ячейку **A2**, для чего в качестве первого символа следует ввести знак равенства (=). Знак равенства в программе MS Excel служит признаком того, что в выделенную ячейку вводятся не данные, а некоторая формула.
3. В качестве выражения для формулы в ячейку **A2** следует ввести следующую строку символов: `=2*A1^2-3*A1+10`. Рабочий лист с вводом формулы данным способом будет иметь следующий вид (рис. 2.16).

В данном случае все математические операции должны быть указаны явно с использованием принятых в программе MS Excel обозначений. В качестве аргумента функции записывается адрес соответствующей ячейки, в которую предполагается вводить те или иные данные. Применительно к рассматриваемому примеру это адрес ячейки **A1**. После окончания ввода выражения для формулы следует нажать клавишу <Ввод>. В результате будет завершен ввод формулы, а в ячейке **A2** при отсутствии ошибок будет указан результат выполнения введенной формулы — число 10. Этот результат получен в предположении, что в ячейке **A1** содержится число 0, хотя это число не вводилось ранее. Если в эту ячейку ввести любое другое число, например, число 5, то после его ввода изменится и значение вычислимой ячейки **A2**, которое станет равно 45.

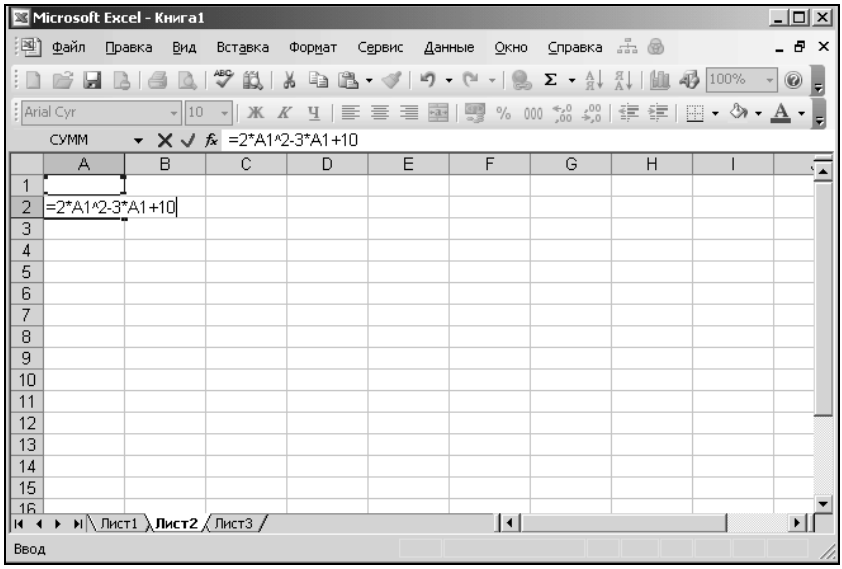


Рис. 2.16. Задание формулы с помощью непосредственного ввода выражения в вычислимую ячейку

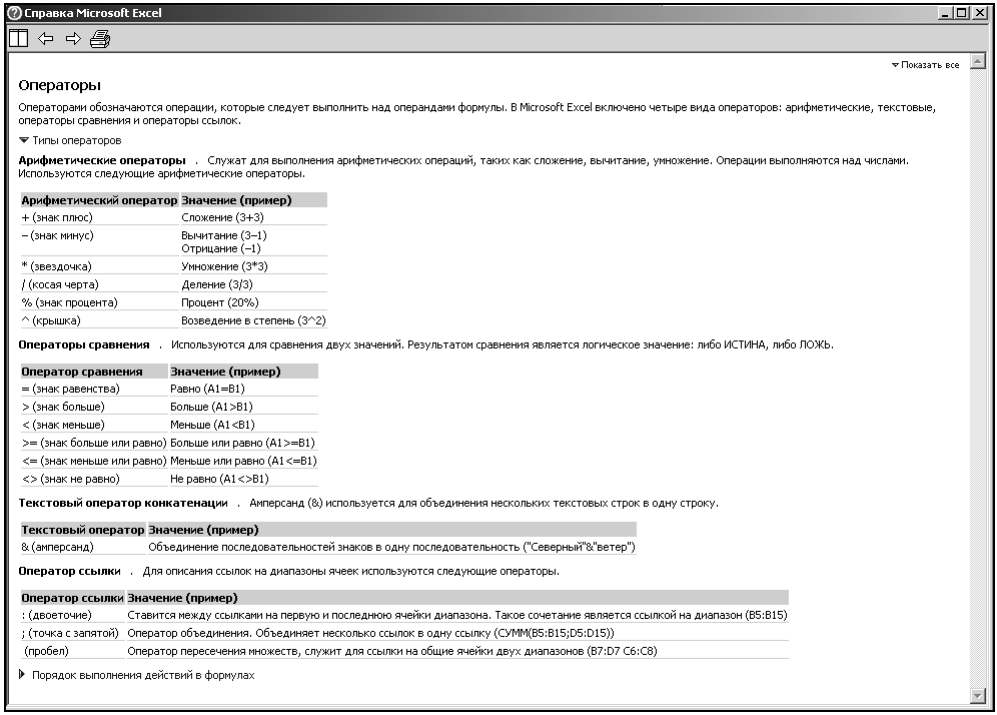


Рис. 2.17. Окно со справочной информацией по типам операторов программы MS Excel

Отдельно следует остановиться на наиболее распространенных ошибках при вводе формул вторым способом. Первая группа ошибок связана с некорректной записью пользователем математических и логических операций. Чтобы убедиться в правильном вводе данных операций, можно воспользоваться справочной системой программы MS Excel. С этой целью ввести в поле поиска окна документации, вызванного с помощью клавиши <F1>, ключевое слово "операторы". В результате будет открыто отдельное окно справочной системы программы MS Excel, в котором следует выбрать информацию из группы "Типы операторов", нажав на ссылке с соответствующим именем (рис. 2.17).

Здесь же можно получить необходимую справочную информацию о порядке выполнения действий в формулах, нарушение которого тоже является распространенной ошибкой при вводе формул.

Наконец следует отметить на возможные ошибки, связанные с вводом адресов ячеек символами кириллицы. Хотя локализованная версия программы MS Excel допускает ввод имен встроенных функций символами кириллицы, при попытке ввода адресов ячеек символами кириллицы в вычислимой ячейке появится сообщение об ошибке: #ИМЯ? В этом случае следует сменить язык ввода символов на английский, а при вводе адресов ячеек в формулу проверять корректность ввода с помощью цветового выделения. Отсутствие последнего при вводе формул свидетельствует об ошибочных действиях со стороны пользователя.

Примечание

Задать формулу вторым способом можно также в поле ввода и редактирования функции и ее аргументов. В этом случае следует активизировать данное поле, выполнив на нем щелчок левой кнопкой мыши. После чего следует выполнить действия, аналогичные указанным в пп. 2 и 3 второго способа задания формулы.

Копирование формул осуществляется аналогично копированию содержимого обычных ячеек с данными. Отличие заключается в изменении адресации ячеек, которые используются в качестве аргументов соответствующих функций. Речь идет о том, что в случае обычной адресации ячеек, которая называется *относительной*, при копировании формул программа MS Excel автоматически изменяет адреса тех ячеек, которые используются в качестве аргументов и имеют относительную адресацию. Так, например, при копировании обычным способом содержимого ячейки **A2** в ячейку **B2** вместо ячейки **A1** в качестве аргумента рассматриваемой функции будет подставлен адрес ячейки **B1**. Соответственно изменится и расчетное значение функции в ячейке **B2** (рис. 2.18).

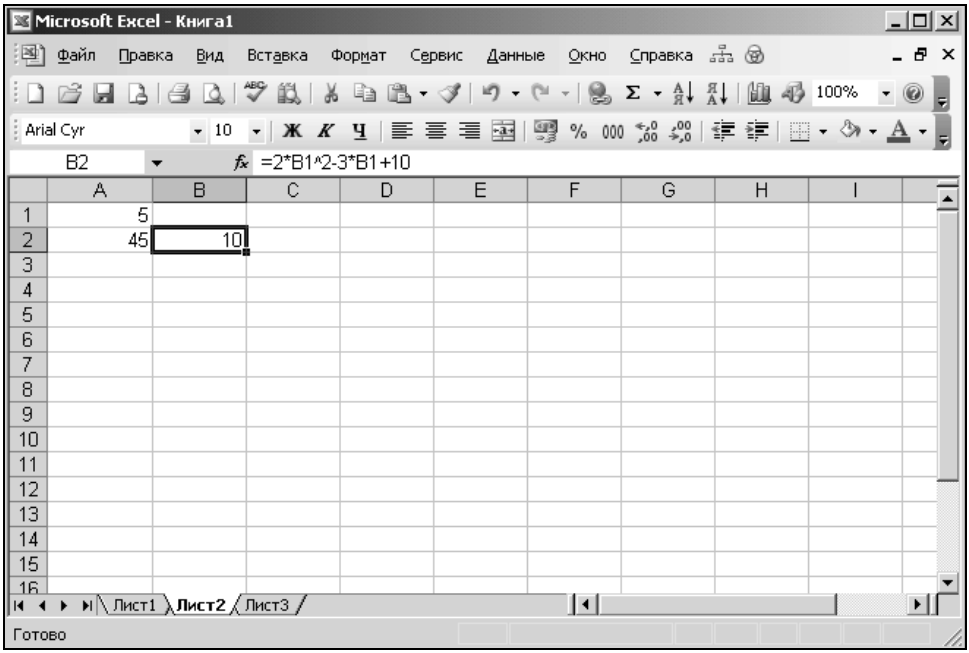


Рис. 2.18. Копирование формулы с относительной адресацией ячеек-аргументов

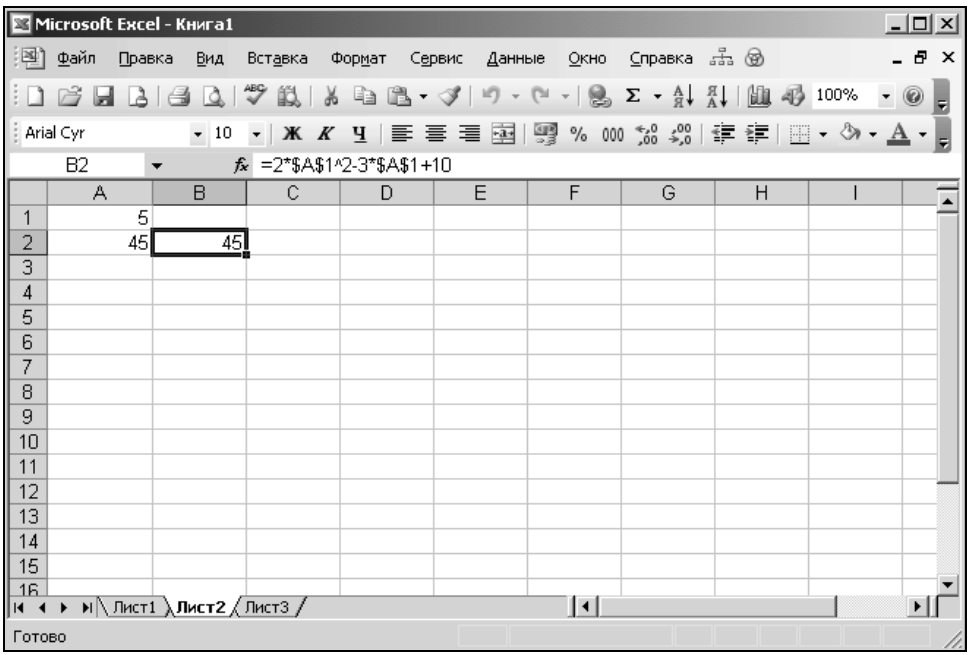


Рис. 2.19. Копирование формулы с абсолютной адресацией ячеек-аргументов

Если необходимо копировать содержимое вычислимых ячеек, не изменяя ячеек-аргументов соответствующих функций, то следует использовать так называемую *абсолютную* адресацию ячеек. При абсолютной адресации ячеек перед ее именем указывается символ доллара — $. При этом он может быть указан как перед именем строки, так и перед именем столбца.

Для того чтобы при копировании функции вычислимой ячейки рассматриваемого примера всегда указывать в качестве ее аргумента ячейку **A1**, следует при первоначальном вводе этой функции или последующем ее редактировании записать соответствующее выражение функции в виде: `=2*$A$1^2-3*$A$1+10`. В этом случае при копировании содержимого вычислимой ячейки **A2** в ячейку **B2** адрес ячейки-аргумента в соответствующей формуле не изменится (рис. 2.19).

Более сложные случаи адресации ячеек в формулах, когда совместно используется относительная и абсолютная адресация ячеек, а также ссылки на ячейки других рабочих листов и книг, рассматриваются далее по мере необходимости. В частности подобная комбинированная адресация используется при построении графиков функций двух переменных *(см. разд. 2.4.2)*.

Все рассмотренные действия по копированию формул оказываются справедливыми для выполнения их переноса. Убедиться в справедливости полученных результатов предлагается читателям самостоятельно в качестве упражнения.

2.4. Основные виды диаграмм в программе MS Excel и приемы их построения

Программа электронных таблиц MS Office Excel 2003 обладает удобной возможностью визуализации и графического представления самых разнообразных данных в форме кривых, поверхностей и диаграмм на плоскости и в трехмерном пространстве. При этом могут быть использованы различные системы координат, стили и способы цветового выделения изображений, что обеспечивает высокий уровень наглядности получаемых рисунков. Оценить графические возможности программы MS Excel можно по изображенным в книге графикам и диаграммам, полученным с помощью данной программы.

Детально рассмотреть все возможные типы диаграмм и особенности их построения в среде MS Excel не представляется возможным. В книге преследуется более скромная цель — познакомиться с основными средствами программы MS Excel, которые удобно использовать для визуализации графиков целевых функций и отдельных ограничений с целью повышения наглядности представления исходных данных и анализа результатов решения задач оптимизации.

2.4.1. Построение графика функции одной переменной

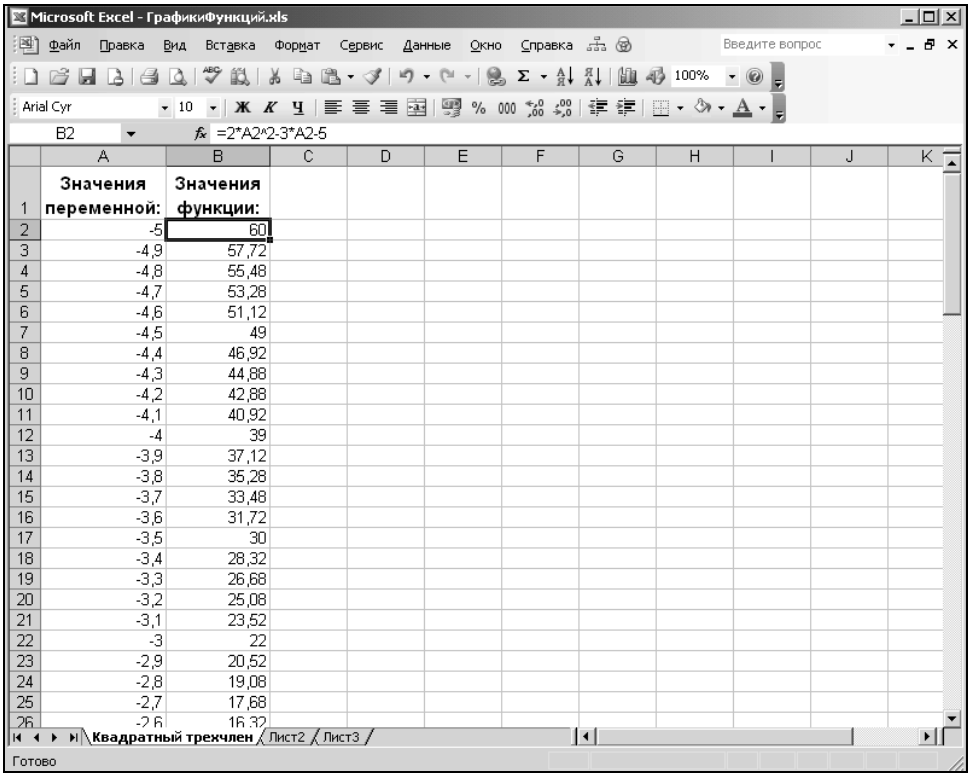
Для построения графика функции одной переменной, прежде всего, необходимо задать множество (диапазон) значений независимой переменной и соответствующее множество значений зависимой (функциональной) переменной. После этого следует воспользоваться мастером построения диаграмм программы MS Excel для изображения графика заданной функциональной зависимости. При этом задание значений независимой переменной удобно выполнить с помощью рассмотренной ранее операции автозаполнения. Ниже приводится описание последовательности практических действий для примера построения графика функции: $f(x) = 2x^2 - 3x - 5$ в интервале изменения независимой переменной: $x \in [-5, 5]$.

Для начала рекомендуется создать новую книгу с именем Графики Функций. На отдельном листе в ячейку **A1** введем текст "Значения переменной:", а в ячейку **B1** введем текст "Значения функции:". Хотя исходная независимая переменная принимает значения из непрерывного интервала действительных чисел [–5, 5], для построения графика необходимо для этой независимой переменной задать дискретные значения. Для наших целей вполне достаточно рассмотреть последовательный диапазон значений от –5 до 5 с интервалом их изменения, равным 0,1.

Для задания этого диапазона значений в ячейку **A2** введем наименьшее значение интервала изменения независимой переменной: число –5, а в ячейку **A3** — число –4,9. После чего вторым способом автозаполнения запишем диапазон значений независимой переменной в ячейки **A2:A102**. Далее в ячейку **B2** введем формулу: `2*A2^2-3*A2-5`, которую с помощью первого способа автозаполнения запишем в ячейки **B2:B102**. Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 2.20).

Для построения графика необходимо воспользоваться мастером диаграмм, который может быть вызван с помощью кнопки стандартной панели инструментов или операции главного меню: **Вставка** | **Диаграмма** (рис. 2.21). Мастер диаграмм представляет собой самостоятельное диалоговое окно с набором вкладок, обеспечивающих задание отдельных свойств элементам графического изображения.

На первой из вкладок содержится список стандартных диаграмм, которые могут быть использованы для графического представления рядов данных. На второй вкладке пользователю предлагается выбор диаграммы из списка нестандартных, часть которых появилась в последних версиях программы MS Excel.



	A	B
1	Значения переменной:	Значения функции:
2	-5	60
3	-4,9	57,72
4	-4,8	55,48
5	-4,7	53,28
6	-4,6	51,12
7	-4,5	49
8	-4,4	46,92
9	-4,3	44,88
10	-4,2	42,88
11	-4,1	40,92
12	-4	39
13	-3,9	37,12
14	-3,8	35,28
15	-3,7	33,48
16	-3,6	31,72
17	-3,5	30
18	-3,4	28,32
19	-3,3	26,68
20	-3,2	25,08
21	-3,1	23,52
22	-3	22
23	-2,9	20,52
24	-2,8	19,08
25	-2,7	17,68
26	-2,6	16,32

Рис. 2.20. Исходные данные для построения графика функции одной переменной

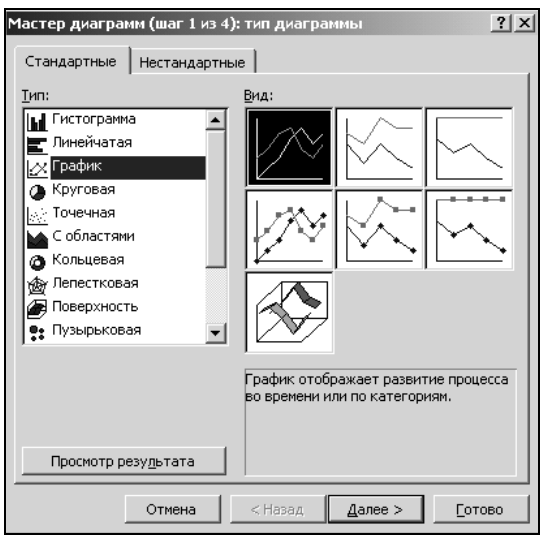
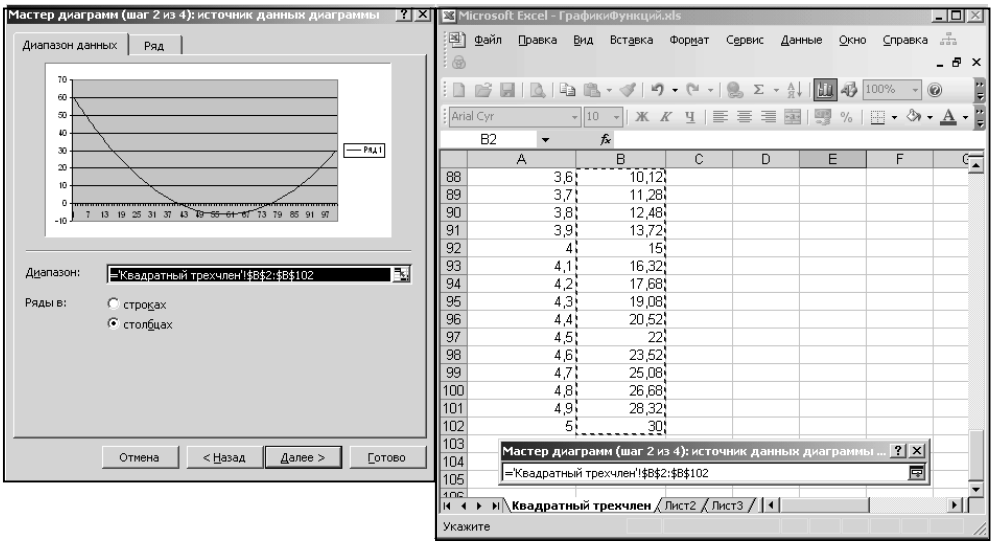


Рис. 2.21. Внешний вид диалогового окна мастера диаграмм (шаг 1 из 4)

На первом шаге построения диаграммы необходимо выбрать тип диаграммы и ее вид. С этой целью следует выделить на вкладке **Стандартные** в левом списке тип диаграммы **График**, а в правом списке с графическими миниатюрами — первый вид графика, который отражает развитие процесса во времени или по категориям.

После выбора типа и разновидности диаграммы следует нажать кнопку **Далее** и перейти ко второму шагу построения диаграммы с помощью мастера диаграмм (рис. 2.22).



а *б*

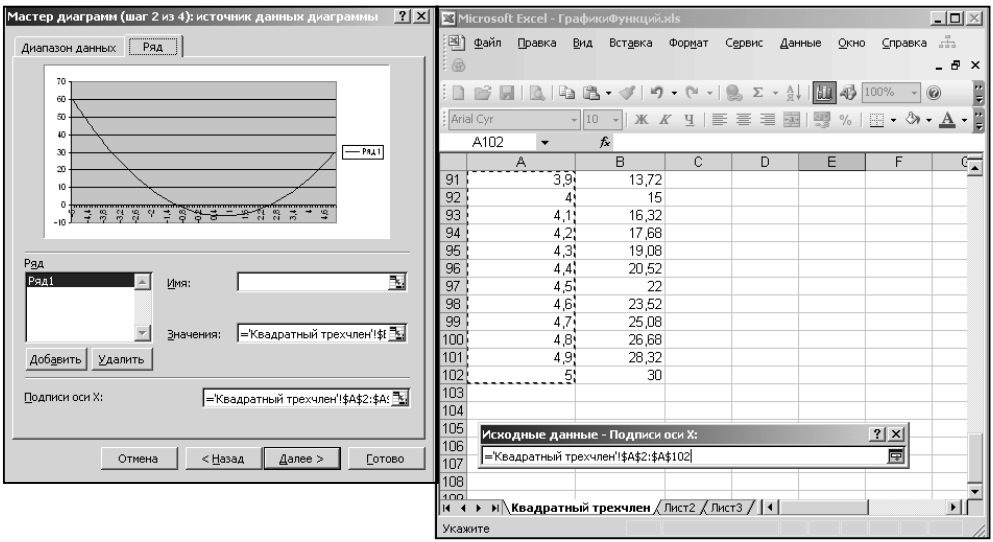
Рис. 2.22. Спецификация источника данных для построения графика с помощью мастера диаграмм (шаг 2 из 4)

Первая из вкладок (рис. 2.22, *а*) служит для задания диапазона значений рядов данных, которые будут отображены на диаграмме. С этой целью на втором шаге построения диаграммы необходимо выбрать ячейки с данными, которые должны быть отображены на соответствующем графике. Применительно к рассматриваемому примеру это значения функции, которые содержатся в диапазоне ячеек **B2:B102**. Для указания этих значений следует установить переключатель **Ряды в:** в положение **столбцах**. После этого выполнить щелчок на кнопке, расположенной в правой части поля ввода **Диапазон**.

В результате будет открыто дополнительное небольшое окно, в единственной строке которого необходимо указать источник данных диаграммы. Для указания источника данных следует на рабочем листе Квадратный трехчлен

с помощью мыши или клавиатуры выделить диапазон значений функции **B2:B102**. Он будет иметь специальную рамку в форме мерцающей пунктирной линии, а в соответствующей строке появится надпись с указанием имени рабочего листа и абсолютных адресов ячеек этого диапазона (рис. 2.22, *б*).

Далее на этом же шаге работы мастера диаграмм следует задать подписи по горизонтальной оси. С этой целью необходимо перейти на вкладку **Ряд** (рис. 2.23, *а*) и выполнить щелчок на кнопке, расположенной в правой части поля ввода с именем **Подписи оси *X***.

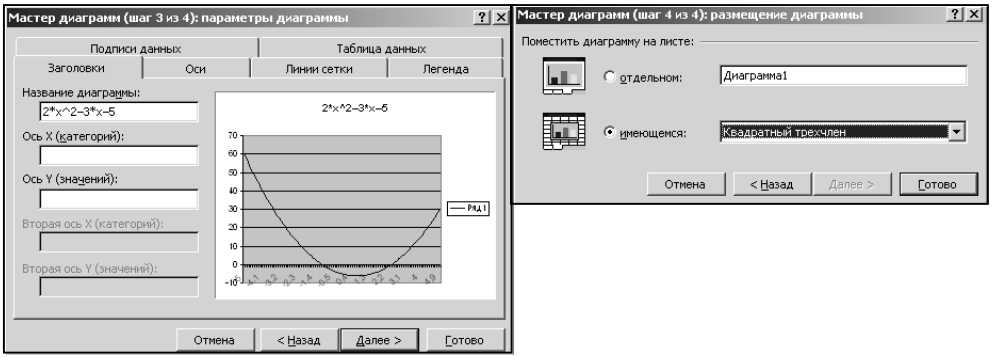


а *б*

Рис. 2.23. Спецификация подписи горизонтальной оси *X* для построения графика с помощью мастера диаграмм (шаг 2 из 4)

В результате будет открыто дополнительное небольшое окно, в единственной строке которого необходимо указать источник данных для независимой переменной графика. Для указания соответствующего источника данных следует на рабочем листе Квадратный трехчлен с помощью мыши или клавиатуры выделить диапазон значений функции **A2:A102**. Выделенный диапазон также приобретет специальную рамку в форме мерцающей пунктирной линии, а в соответствующей строке появится надпись с указанием имени рабочего листа и абсолютных адресов ячеек этого диапазона (рис. 2.23, *б*).

После выбора спецификации рядов данных и подписей оси *X* для графика следует нажать кнопку **Далее** и перейти к третьему шагу построения диаграммы с помощью мастера диаграмм (рис. 2.24, *а*).



а *б*

Рис. 2.24. Спецификация названия диаграммы (*а*) и выбор варианта ее размещения (*б*) с помощью мастера диаграмм (шаги 3 и 4 из 4)

На третьем шаге мастера диаграмм на отдельных вкладках можно задать заголовки и названия диаграммы и ее осей, подписи данных и линии сетки. Применительно к рассматриваемому примеру можно указать название графика в соответствующем поле ввода. После спецификации параметров диаграммы следует нажать кнопку **Далее** и перейти к последнему четвертому шагу построения диаграммы с помощью мастера диаграмм (рис. 2.24, *б*).

На этом шаге следует выбрать один из двух вариантов размещения диаграммы: на отдельном рабочем листе или на текущем рабочем листе, где содержатся ячейки с данными диаграммы. При необходимости можно изменить имя диаграммы, предлагаемой мастером по умолчанию. После этого следует нажать кнопку **Готово**. В результате будет построена диаграмма в форме графика кривой квадратного трехчлена (рис. 2.25).

После построения диаграммы в случае необходимости можно редактировать отдельные ее свойства. Доступ к редактированию свойств построенной диаграммы можно получить либо с помощью контекстного меню диаграммы, либо с помощью операций главного меню **Формат**. Удобным оказывается способ получения доступа к выделенным элементам диаграммы с помощью комбинации клавиш: <Ctrl>+<1> или двойного щелчка левой кнопкой мыши.

Так, например, для изменения цвета фона построенного графика квадратного трехчлена можно выполнить двойной щелчок левой кнопкой мыши на фоне соответствующего рисунка. В появившемся диалоговом окне выбрать желаемый цвет фона графика из предлагаемой палитры цветов. Аналогично можно изменить цвет линии кривой графика и интервал отображения промежуточных значений независимой и зависимой переменных. Изображенный на рис. 2.25 график функции получен после редактирования его свойств в результате из-

менения цвета фона диаграммы, линии кривой графика и интервала отображения промежуточных значений независимой переменной.

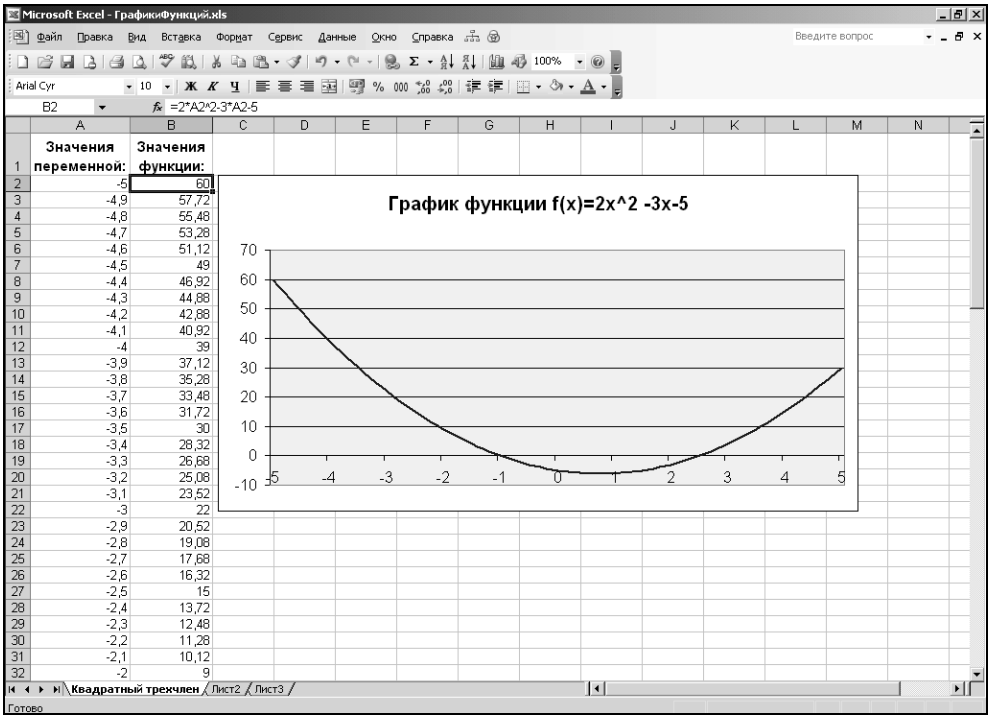


Рис. 2.25. Результат построения графика функции: $f(x) = 2x^2 - 3x - 5$ в интервале изменения независимой переменной: $x \in [-5, 5]$

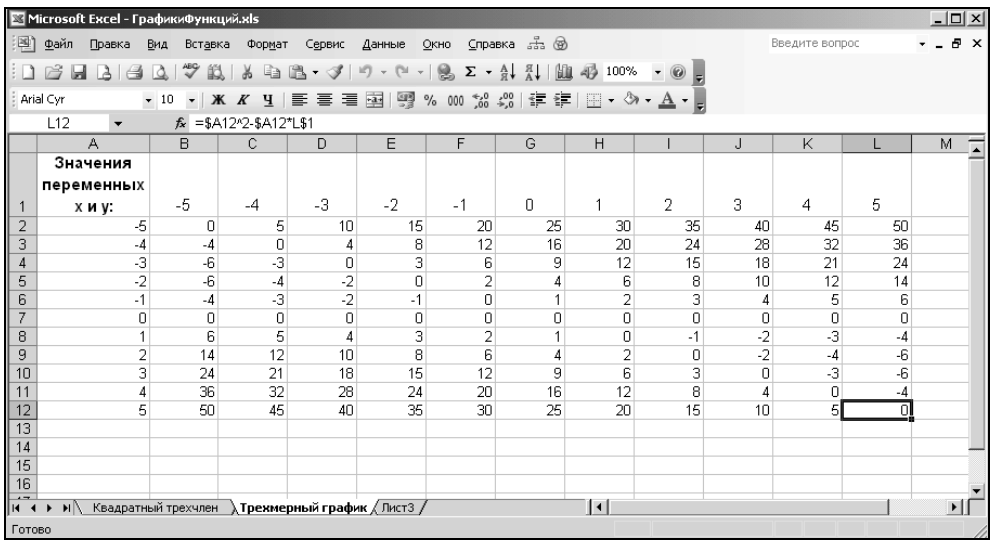
2.4.2. Построение графика функции двух переменных

Для построения графиков трехмерных поверхностей и кривых, представляющих собой функции двух переменных, также необходимо определить множество (диапазон) значений независимых переменных и соответствующее множество значений зависимой (функциональной) переменной. После этого также следует воспользоваться мастером построения диаграмм программы MS Excel для изображения графика заданной функциональной зависимости. Далее приводится описание последовательности практических действий для примера построения графика функции двух переменных: $f(x, y) = x^2 - xy$ в интервале изменения независимых переменных: $x, y \in [-5, 5]$.

С этой целью на отдельном рабочем листе с именем Графики функций этой же книги в ячейку **A1** введем текст "Значения переменных x и y:". Для по-

строения графика вполне достаточно рассмотреть последовательный диапазон значений независимых переменных от –5 до 5 с интервалом их изменения, равным 1. Для этого с помощью операции автозаполнения зададим значения первой переменной x в ячейках **A2:A12**, а значения второй переменной y — в ячейках **B1:L1**.

Далее в ячейку **B2** введем формулу: `=$A2^2-$A2*B$1`, которую с помощью первого способа автозаполнения скопируем в ячейки **C2:L2**. После этого, выделив диапазон ячеек **B2:L2**, также с помощью первого способа автозаполнения скопируем соответствующие данные в ячейки **B3:L12**. Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 2.26).

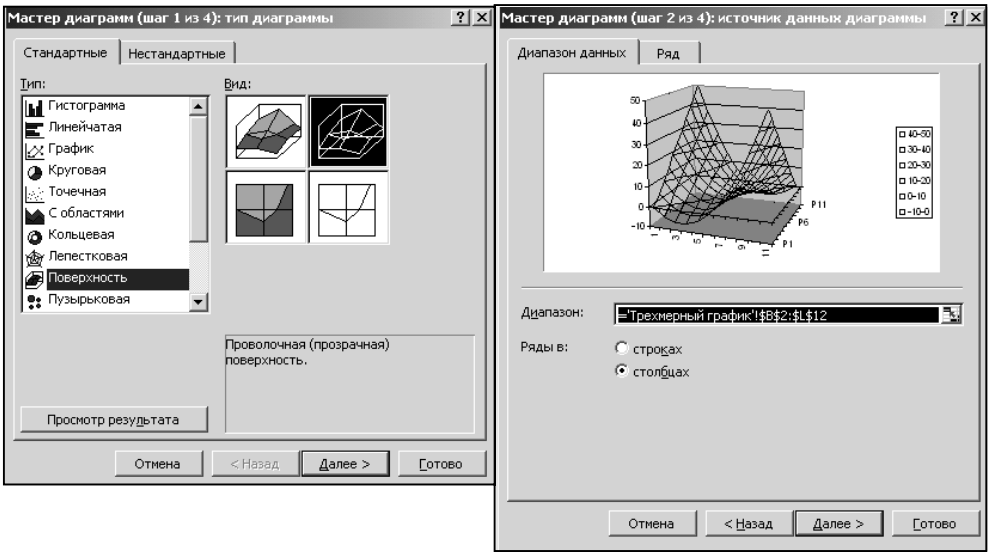


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Значения переменных x и y:	-5	-4	-3	-2	-1	0	1	2	3	4	5	
2	-5	0	5	10	15	20	25	30	35	40	45	50	
3	-4	-4	0	4	8	12	16	20	24	28	32	36	
4	-3	-6	-3	0	3	6	9	12	15	18	21	24	
5	-2	-6	-4	-2	0	2	4	6	8	10	12	14	
6	-1	-4	-3	-2	-1	0	1	2	3	4	5	6	
7	0	0	0	0	0	0	0	0	0	0	0	0	
8	1	6	5	4	3	2	1	0	-1	-2	-3	-4	
9	2	14	12	10	8	6	4	2	0	-2	-4	-6	
10	3	24	21	18	15	12	9	6	3	0	-3	-6	
11	4	36	32	28	24	20	16	12	8	4	0	-4	
12	5	50	45	40	35	30	25	20	15	10	5	0	
13													
14													
15													
16													

Рис. 2.26. Исходные данные для построения графика функции двух переменных

Для построения графика функции двух переменных также следует воспользоваться мастером диаграмм, который может быть вызван с помощью кнопки стандартной панели инструментов (см. табл. 2.1) или операции главного меню: **Вставка** | **Диаграмма**. В результате выполнения этой операции появится диалоговое окно мастера диаграмм (рис. 2.27, *а*).

На первом шаге построения диаграммы следует выделить на вкладке **Стандартные** в левом списке тип диаграммы **Поверхность**, а в правом списке с графическими миниатюрами — второй вид графика, который соответствует проволочной (прозрачной) поверхности. После выбора типа и разновидности диаграммы следует нажать кнопку **Далее** и перейти ко второму шагу построения диаграммы с помощью мастера диаграмм (рис. 2.27, *б*).



а *б*

Рис. 2.27. Внешний вид диалоговых окон мастера диаграмм при построении трехмерной поверхности (шаги 1 и 2 из 4)

Первая из вкладок второго окна мастера служит для задания диапазона значений рядов данных, которые будут отображены на диаграмме. Применительно к рассматриваемому примеру — это значения функции, которые содержатся в диапазоне ячеек **B2:L12**. Для указания этих значений следует установить переключатель **Ряды в:** в положение **столбцах**. После этого выполнить щелчок на кнопке, расположенной в правой части поля ввода с именем **Диапазон**.

В результате этого действия будет открыто дополнительное небольшое окно, в единственной строке которого необходимо указать источник данных диаграммы. Для указания источника данных следует на рабочем листе Трехмерный график с помощью мыши или клавиатуры выделить диапазон значений функции **B2:L12**. Выделенный диапазон будет иметь специальную рамку в форме мерцающей пунктирной линии, а в соответствующей строке появится надпись с указанием имени рабочего листа и абсолютных адресов ячеек этого диапазона (рис. 2.28).

Далее на этом же шаге работы мастера диаграмм следует задать подписи по горизонтальным осям. С этой целью необходимо перейти на вкладку **Ряд** (рис. 2.29) и выполнить щелчок на кнопке, расположенной в правой части поля ввода с именем **Подписи оси *X***.

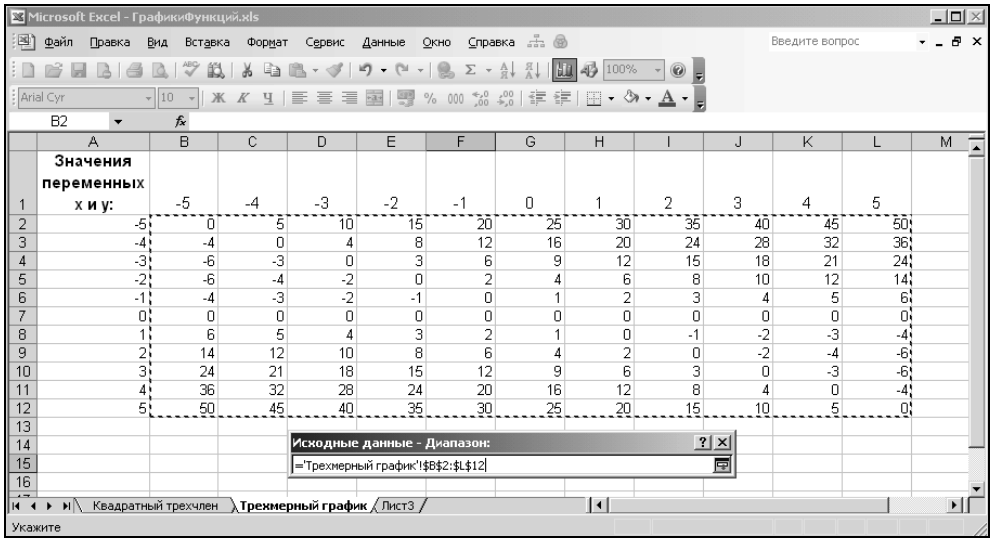
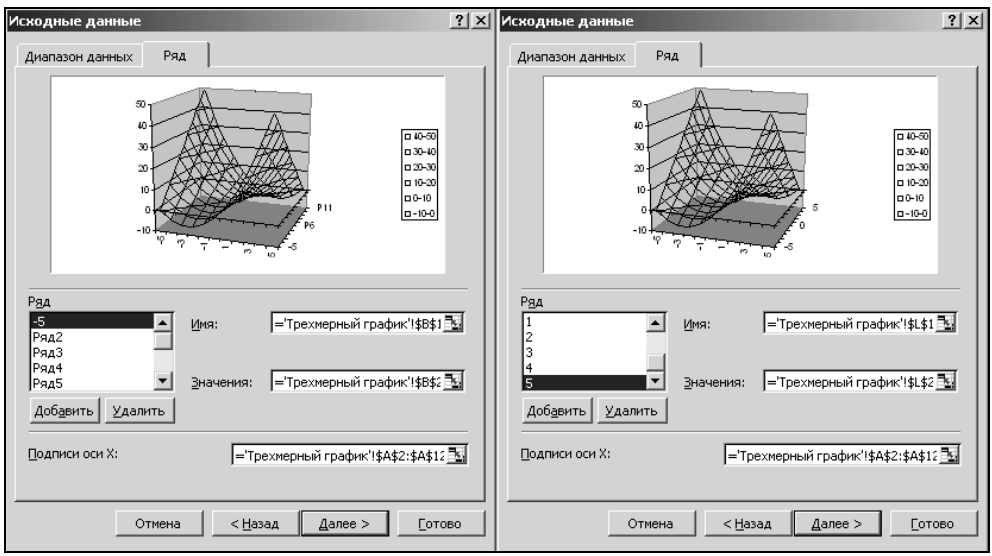


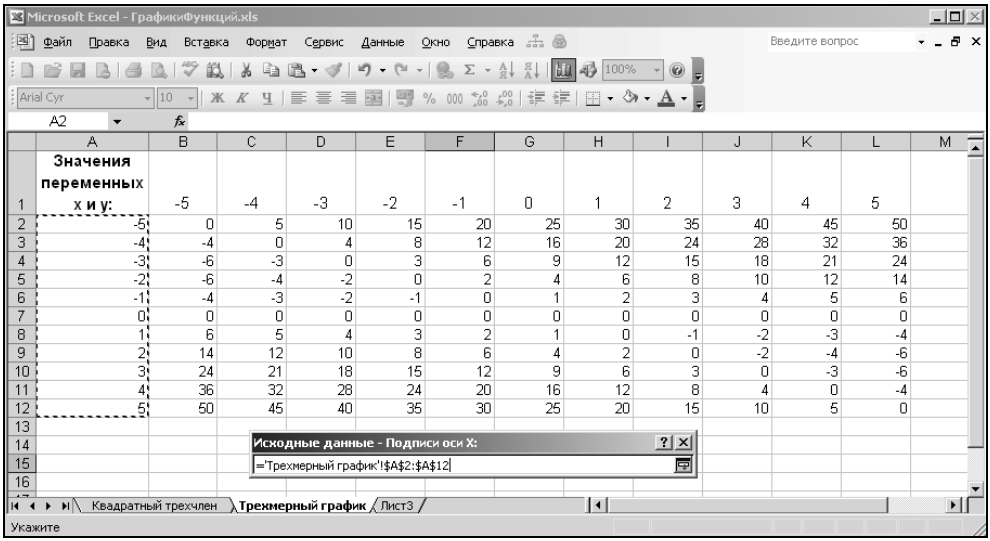
Рис. 2.28. Спецификация источника данных для построения графика с помощью мастера диаграмм (шаг 2 из 4)



а *б*

Рис. 2.29. Спецификация подписи горизонтальной оси X (*а*) и оси Y (*б*) для построения графика с помощью мастера диаграмм (шаг 2 из 4)

В дополнительном окне указывается источник данных для первой независимой переменной графика — x. Для указания соответствующего источника данных следует на рабочем листе Трехмерный график с помощью мыши или клавиатуры выделить диапазон значений функции **A2:A12**. Выделенный диапазон также приобретет специальную рамку в форме мерцающей пунктирной линии, а в соответствующей строке появится надпись с указанием имени рабочего листа и абсолютных адресов ячеек этого диапазона (рис. 2.30).



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Значения переменных x и y:	-5	-4	-3	-2	-1	0	1	2	3	4	5	
2	-5	0	5	10	15	20	25	30	35	40	45	50	
3	-4	-4	0	4	8	12	16	20	24	28	32	36	
4	-3	-6	-3	0	3	6	9	12	15	18	21	24	
5	-2	-6	-4	-2	0	2	4	6	8	10	12	14	
6	-1	-4	-3	-2	-1	0	1	2	3	4	5	6	
7	0	0	0	0	0	0	0	0	0	0	0	0	
8	1	6	5	4	3	2	1	0	-1	-2	-3	-4	
9	2	14	12	10	8	6	4	2	0	-2	-4	-6	
10	3	24	21	18	15	12	9	6	3	0	-3	-6	
11	4	36	32	28	24	20	16	12	8	4	0	-4	
12	5	50	45	40	35	30	25	20	15	10	5	0	
13													
14													
15													
16													

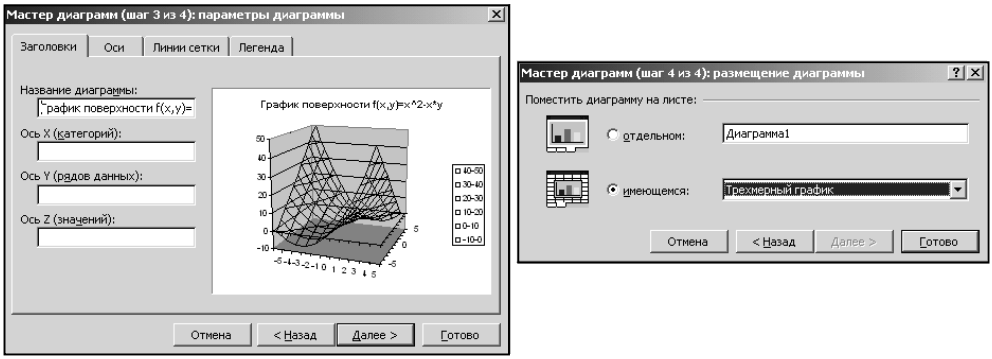
Рис. 2.30. Спецификация подписи горизонтальной оси X с помощью дополнительного диалогового окна мастера диаграмм (шаг 2 из 4)

Далее на этом же шаге работы мастера диаграмм следует задать подписи по второй горизонтальной оси — переменной y. С этой целью на вкладке **Ряд** следует в поле **Ряд** выделить строку **Ряд 1** и в дополнительном диалоговом окне выбрать первое значение второй независимой переменной в ячейке **B1**: –5. Данную операцию следует последовательно выполнить для всех ячеек диапазона **B1:L1** (рис. 2.29, *б*).

После выбора спецификации рядов данных и подписей осей X и Y для графика можно нажать кнопку **Далее** и перейти к третьему шагу построения диаграммы (рис. 2.31, *а*).

Как уже отмечалось ранее, на третьем шаге мастера диаграмм на отдельных вкладках можно задать заголовки и названия диаграммы и ее осей, подписи данных и линии сетки. Применительно к рассматриваемому примеру можно указать название графика поверхности в соответствующем поле ввода. После спецификации параметров диаграммы следует нажать кнопку **Далее** и перейти

к последнему четвертому шагу построения диаграммы с помощью мастера диаграмм (рис.2.31, *б*).



а *б*

Рис. 2.31. Спецификация названия диаграммы (*а*) и выбор варианта ее размещения (*б*) с помощью мастера диаграмм (шаги 3 и 4 из 4)

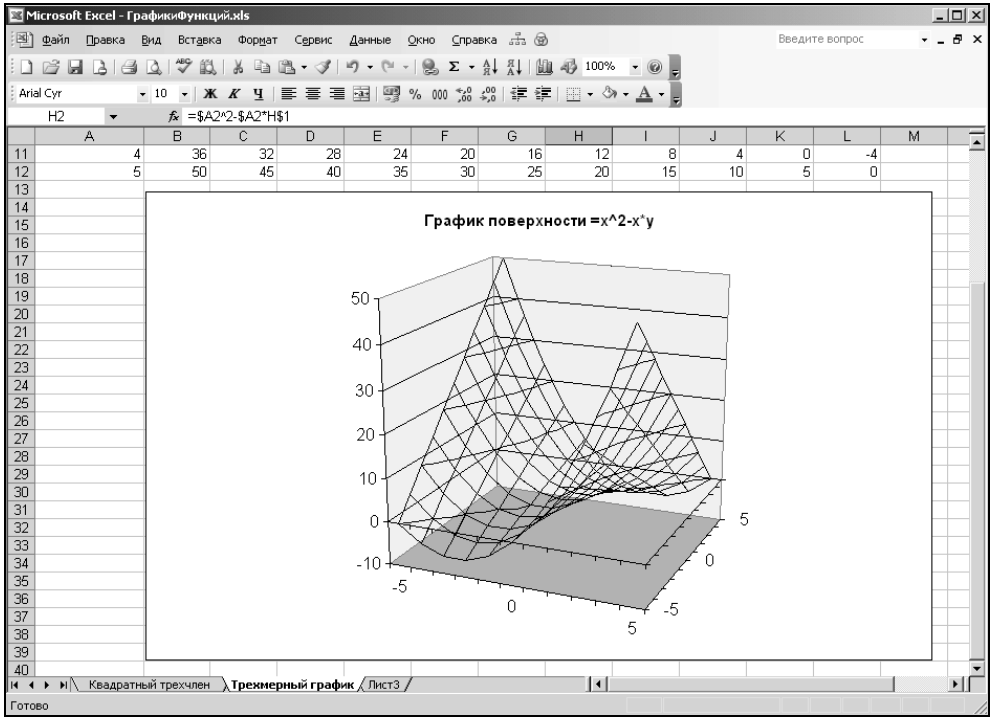


Рис. 2.32. Результат построения графика функции: $f(x, y) = x^2 - xy$ в интервале изменения независимых переменных: $x, y \in [-5, 5]$

Оставив без изменения предложенные мастером диаграмм свойства, следует нажать кнопку **Готово**. В результате будет построена диаграмма в форме графика поверхности рассматриваемой функции двух переменных (рис. 2.32).

Изображенный на рис. 2.32 график функции получен после редактирования его свойств, а именно — изменения цвета фона диаграммы. Наиболее удобный способ доступа к редактированию свойств построенной диаграммы можно получить с помощью двойного щелчка левой кнопкой мыши на выделенном элементе диаграммы.

Важной особенностью трехмерных графиков в среде MS Excel является возможность их поворота вокруг каждой их 3-х осей. Для этого следует выделить область рисунка диаграммы, в результате чего должны появиться концевые маркеры рисунка в форме небольших квадратов по углам осей. Позиционировав курсор мыши на одном из этих маркеров и удерживая нажатой левую кнопку мыши, можно повернуть построенную диаграмму в нужном направлении (рис. 2.33).

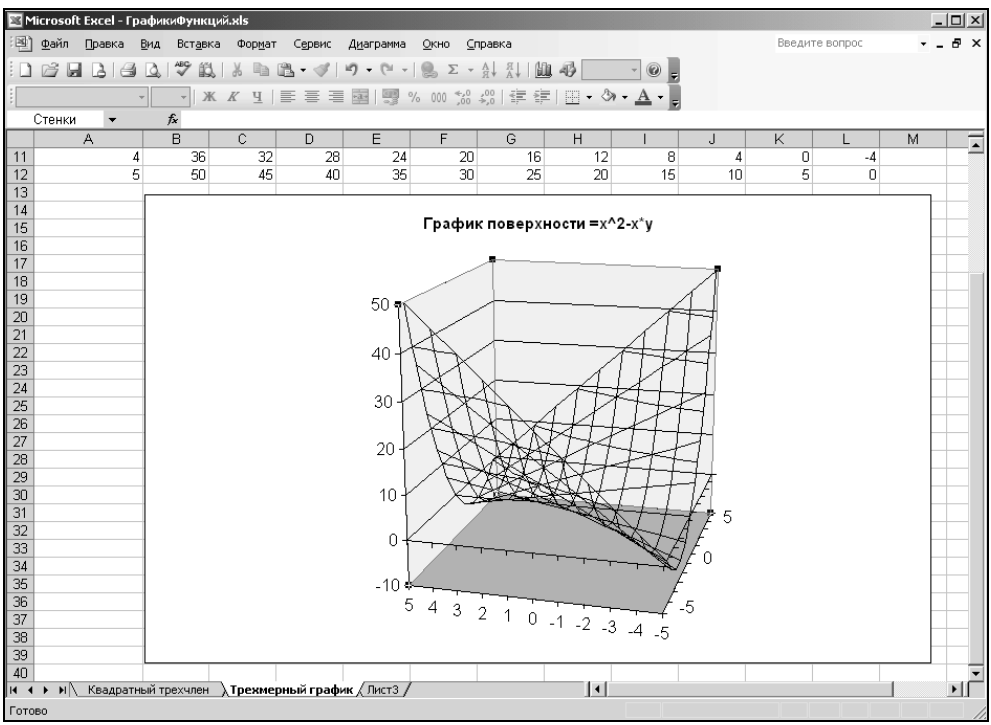
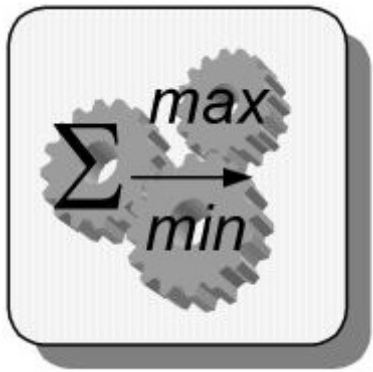


Рис. 2.33. График функции: $f(x, y) = x^2 - xy$ в интервале изменения независимых переменных: $x, y \in [-5, 5]$ после поворота на 180°

Посредством вращения построенной диаграммы можно добиться не только интересных графических эффектов, но что наиболее важно — рассмотреть под различными углами график поверхности соответствующей функции с целью визуального анализа ее особенностей. Так, например, визуальный анализ функции $f(x, y) = x^2 - xy$ показывает, что она имеет явно выраженный "седловой" характер. Ввиду того, что подобные функции имеют специальное применение в теории решения задач оптимизации, они так и называются *седловыми* функциями, примером является рассматриваемая функция двух переменных.

В следующих главах книги будут рассмотрены способы построения некоторых других типов диаграмм, которые целесообразно использовать для визуального анализа математических моделей отдельных задач оптимизации. В качестве упражнения можно построить аналогичным образом графики рассмотренных функций в форме других типов диаграмм. Рассмотренные типы графиков наиболее часто используются для визуального анализа одномерных и двумерных целевых функций при решении соответствующих задач оптимизации.



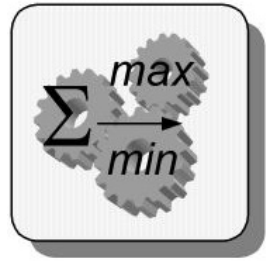
Часть II

Задачи непрерывной оптимизации

Глава 3. Задачи нелинейного программирования

Глава 4. Задачи линейного программирования

После рассмотрения содержательных особенностей типовых задач и общей математической модели задач оптимизации можно приступить к систематическому изложению методов их решения. В данной части книги представлены методы решения задач непрерывной оптимизации, т. е. таких задач, в которых все переменные являются непрерывными. Это означает, что множество значений, которые могут принимать подобные переменные, имеет мощность континуума и, как правило, совпадает с множеством всех вещественных чисел или является его подмножеством. Среди задач непрерывной оптимизации наибольшее распространение получили задачи нелинейного и линейного программирования, которые и рассматриваются в данной части книги.



Глава 3

Задачи нелинейного программирования

К классу задач нелинейного программирования относятся такие задачи непрерывной оптимизации, в которых целевая функция является нелинейной функцией своих аргументов, а ограничения могут быть представлены как в форме линейных, так и нелинейных функций. Задачи нелинейного программирования также называют задачами *нелинейной оптимизации*. Исторически данный класс задач оптимизации был изучен ранее остальных, а аналитические методы решения экстремальных задач входят в современные учебные программы подготовки профессиональных математиков.

Собственно термин *нелинейное программирование* (от *англ.* nonlinear programming) возник в середине XX столетия в США в связи с разработкой специальных методов решения задач планирования боевых действий и оптимизации военных операций. Поскольку в то время первые ЭВМ только появлялись, термин *программирование* просто означал последовательность этапов решения некоторой сложной задачи и не имел непосредственного отношения к решению задач с использованием компьютеров. В последующем произошла трансформация термина *программирование* в современное его представление как процесса написания компьютерных программ на некотором языке программирования. В то же время понятие нелинейного программирования продолжает использоваться в своем первоначальном смысле, о чем следует помнить, изучая методы решения задач оптимизации.

3.1. Общая характеристика задачи нелинейного программирования

К классу задач нелинейного программирования или нелинейной оптимизации относятся такие задачи однокритериальной оптимизации, в которых переменные являются непрерывными, целевая функция является нелинейной

функцией своих аргументов, а ограничения могут быть представлены как в форме линейных, так и нелинейных функций. Более строгое определение данного класса задач можно получить на основе рассмотрения общей математической постановки задачи нелинейного программирования.

3.1.1. Математическая постановка задачи нелинейного программирования

В общем случае математическая постановка задачи нелинейного программирования с одной переменной может быть сформулирована в следующем виде:

$$f(x) \to \max_{x \in \Delta_\beta} \text{ или } f(x) \to \min_{x \in \Delta_\beta}, \tag{3.1.1}$$

где

$$\Delta_\beta = \{\Delta \mid g_k(x) \le (=) 0\}, \ (k \in \{1, 2, ..., m\}). \tag{3.1.2}$$

При этом вводятся в рассмотрение некоторые общие, но принципиальные предположения о характере целевой функции и левых частей ограничений. А именно, целевая функция $f(x)$ предполагается нелинейной, а левые части ограничений $g_k(x)$ могут быть как линейными, так и нелинейными функциями относительно единственного аргумента x. При этом сама переменная x принимает свои значения из множества действительных чисел $\boldsymbol{R}^1$, т. е. $x \in \boldsymbol{R}^1$.

В случае отсутствия ограничений (3.1.2) задача нелинейного программирования называется задачей нелинейной оптимизации *без ограничений*. В этом случае поиск решения задачи осуществляется на всем множестве задания переменной $x \in \boldsymbol{R}^1$.

Примечание

Математическая постановка задачи нелинейного программирования с несколькими переменными формулируется аналогично (3.1.1) и (3.1.2) и поэтому здесь не приводится. Заинтересованным читателям предлагается самостоятельно записать ее постановку в качестве упражнения.

Среди задач нелинейного программирования наибольшую известность получили задачи оптимизации с некоторыми дополнительными предположениями о характере целевой функции и ограничений. В частности, если предположить непрерывность и дифференцируемость целевой функции по всем аргументам, то может оказаться возможным найти аналитическое решение задачи нелинейной оптимизации.

Говоря о задачах нелинейного программирования, часто вводят в рассмотрение дополнительные предположения о выпуклости или вогнутости целевой функции и ограничений. Соответствующие задачи образуют отдельный под-

класс задач *выпуклой* оптимизации. Если целевая функция может быть представлена в форме квадратичной функции, то соответствующие задачи рассматриваются в рамках отдельного подкласса задач *квадратичного программирования*. Аналогично сепарабельный характер целевой функции служит основанием для введения в рассмотрение отдельного подкласса задач *сепарабельного программирования*.

Примечание

Более подробно с отдельными подклассами задач нелинейного программирования можно познакомиться в специальной литературе, перечень которой приводится в конце книги. Следует также отметить, что в общем случае теоретическим базисом для анализа и решения всех задач нелинейного программирования служат соответствующие теоремы о существовании и единственности решения задач того или иного подкласса. Поскольку прикладной характер настоящей книги не позволяет детально рассмотреть теоретические аспекты анализа задач оптимизации, заинтересованные в соответствующих вопросах читатели могут обратиться к специальной литературе.

3.1.2. Основные методы решения задач нелинейного программирования

Как уже отмечалось в *главе 1*, в общем случае существует два подхода к решению задач оптимизации. С одной стороны, та или иная задача оптимизации может быть решена аналитически. При этом под *аналитическим решением* задачи оптимизации понимают установление некоторой функциональной зависимости между исходными данными задачи и точным ее решением, зафиксированной в форме аналитической функции, т. е. функции, допускающей вычисление своего результата по известным значениям аргументов.

С точки зрения математики, получение аналитического решения для той или иной задачи оптимизации, а еще лучше — для целого класса или подкласса, является целью теоретического анализа и изучения задач оптимизации. В этом контексте развитие математики неразрывно связано с нахождением аналитических решений экстремальных задач самых различных классов.

Базовым способом нахождения аналитических решений задач оптимизации с одной переменной без ограничений с дифференцируемой целевой функцией является нахождение нулей производной целевой функции и проверка их на экстремум. Соответственно, для нахождения аналитических решений задач оптимизации с несколькими переменными без ограничений с дифференцируемой целевой функцией является нахождение нулей частных производных целевой функции по всем переменным и проверка их на экстремум.

В случае задач оптимизации с непрерывно дифференцируемой по всем переменным целевой функцией и ограничениями в форме равенств, для отыскания аналитического решения задач оптимизации может быть использован *метод множителей Лагранжа*, названный в честь известного французского математика и механика Жозефа Луи Лагранжа (1736 — 1813).

Однако, как правило, аналитическое решение задач нелинейной оптимизации возможно только для простейших задач с дополнительными предположениями о характере целевой функции и ограничений. Поэтому альтернативой аналитическому решению является алгоритмическое или вычислительное решение задачи оптимизации.

При этом под *алгоритмическим решением* задачи оптимизации понимают разработку или конструирование такой вычислительной процедуры, которая позволяет на основе известных исходных данных задачи находить ее решение. Соответствующая процедура может быть задана некоторым формальным образом и зафиксирована в форме алгоритма, т. е. формального предписания выполнить точно определенную последовательность действий, направленных на решение поставленной задачи. Как правило, современные методы алгоритмического решения задач оптимизации предполагают использование компьютеров и соответствующих программ.

Примечание

Кроме аналитического и алгоритмического подходов к решению задач оптимизации применительно к простейшим задачам рассматривают также способ *графического решения*, который основан на изображении графиков целевой функции и ограничений на плоскости или в трехмерном пространстве с последующим визуальным нахождением решения. Наиболее часто графический способ решения используется для иллюстрации особенностей тех или иных методов решения задач линейного программирования *(см. главу 4)*.

Как уже отмечалось ранее, методы алгоритмического решения задач оптимизации делятся на две категории: методы нахождения точного решения и методы нахождения приближенного решения. Методы и алгоритмы первой группы позволяют за конечное время найти точное решение задачи оптимизации, т. е. решение, удовлетворяющее условию (1.6.1) для задач максимизации или условию (1.6.2) для задач минимизации.

Однако само понятие *конечного времени* при решении практических задач оптимизации является весьма неопределенным и подразумевает получение результата решения задачи за приемлемое время. К сожалению, многие интересные задачи оптимизации не могут быть точно решены даже с использованием компьютеров за время, не превышающее, к примеру, нескольких часов.

Именно этот факт послужил основой для разработки специальных алгоритмов *приближенного решения* задач оптимизации, которые позволяют находить одно или несколько локально-оптимальных решений. В этом случае приближенное решение задачи оптимизации должно удовлетворять условию (1.6.3) для задачи максимизации или (1.6.4) для задачи минимизации.

Нахождение точного решения конкретной задачи оптимизации является наиболее предпочтительным с точки зрения практического результата. Если же точное решение по каким-либо причинам невозможно, то следует попытаться найти приближенное решение. При этом общей рекомендацией для нахождения приближенных решений является применение нескольких методов или использование одного метода с различными параметрами с целью получения нескольких приближенных решений и выбора из них наибольшего для задач максимизации или наименьшего для задач минимизации.

Что касается задач нелинейного программирования, в программе MS Excel реализованы приближенные методы их решения с достаточно высокой степенью точности. Оценить точность получаемых решений можно посредством сравнения аналитических и алгоритмических решений отдельных практических задач. Именно с этой целью в данной главе приводится описание двух способов решения задач о коробке максимального объема и задачи о пожарном ведре.

3.2. Задача о коробке максимального объема

Содержательная постановка задачи о коробке максимального объема приводится в *разд. 1.2.1*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи оптимизации.

3.2.1. Математическая постановка задачи о коробке максимального объема

Для математической постановки данной задачи необходимо ввести в рассмотрение некоторые параметры, характеризующие геометрические размеры коробки. С этой целью дополним содержательную постановку задачи соответствующими параметрами. С этой целью будем рассматривать квадратную заготовку из некоторого гибкого материала, которая имеет длину стороны L (рис. 3.1). Из этой заготовки следует вырезать четыре равных квадрата со стороной r по ее углам, а полученную фигуру согнуть, так чтобы получилась коробка без верхней крышки. Задача состоит в таком выборе размера вырезаемых квадратов, чтобы в результате получилась коробка максимального объема.

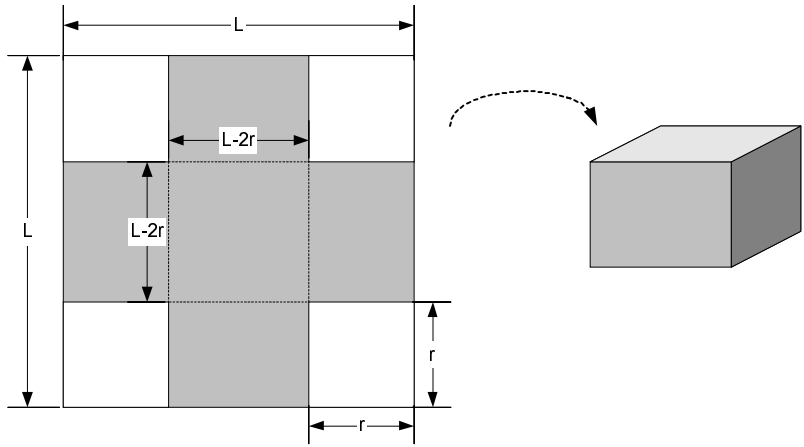


Рис. 3.1. Схема изготовления коробки из прямоугольной заготовки с указанием ее размеров

Для математической постановки данной задачи необходимо определить переменные соответствующей задачи оптимизации, задать целевую функцию и специфицировать ограничения. В качестве переменной следует взять длину стороны вырезаемого квадрата r, которая в общем случае, исходя из содержательной постановки задачи, принимает непрерывные действительные значения. Целевой функцией является объем полученной коробки. Поскольку длина стороны основания коробки равна: $L - 2r$, а высота коробки равна r, то ее объем находится по формуле: $V(r) = (L - 2r)^2 \cdot r$. Исходя из физических соображений, значения переменной r не могут быть отрицательными и превышать величину половины размера исходной заготовки L, т. е. $0{,}5 \cdot L$.

Примечание

При значениях $r = 0$ и $r = 0{,}5L$ соответствующие решения задачи о коробке являются вырожденными. Действительно, в первом случае заготовка остается без изменения, а во втором случае она разрезается на 4 одинаковых части. Поскольку эти решения имеют физическую интерпретацию, задачу о коробке для удобства ее постановки и анализа можно считать задачей оптимизации с ограничениями типа *нестрогих* неравенств.

С целью унификации, обозначим переменную через $x = r$, что не оказывает влияния на характер решаемой задачи оптимизации. Тогда математическая постановка задачи о коробке максимального объема может быть записана в следующем виде:

$$f(x) = x \cdot (L - 2x)^2 \to \max_{x \in \Delta_\beta}, \quad \text{где} \quad \Delta_\beta = \{x \in \boldsymbol{R}^1 \mid 0 \le x \le 0{,}5 \cdot L\}. \tag{3.2.1}$$

Целевая функция данной задачи является нелинейной, поэтому задача о коробке максимального размера относится к классу задач нелинейного программирования или нелинейной оптимизации.

3.2.2. Решение задачи о коробке максимального объема с помощью программы MS Excel

Не уменьшая общности математической постановки задачи (3.2.1), предположим: $L = 1$. Для решения данной задачи с помощью программы MS Excel создадим новую книгу с именем Нелинейная Оптимизация и изменим имя ее первого листа на Задача о коробке. Сделаем необходимые надписи в ячейках **A1:B2**. После этого введем в ячейку **C2** формулу: `=C1*(1-2*C1)^2`, которая представляет целевую функцию (3.2.1). Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о коробке максимального объема имеет следующий вид (рис. 3.2).

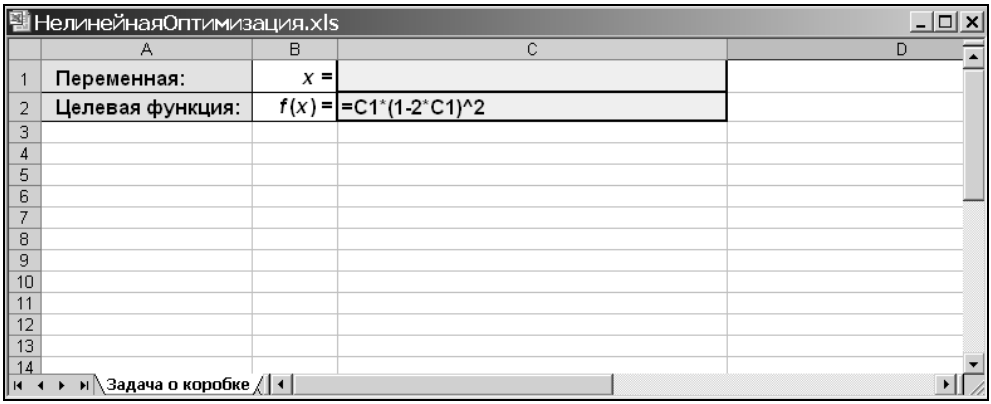


Рис. 3.2. Исходные данные для решения задачи о коробке максимального объема

При изображении исходных данных для решения задачи о коробке максимального объема (рис. 3.2) выбран режим отображения формул в ячейках рабочего листа, что оказывается весьма удобным для визуального контроля правильности задания выражений для соответствующих формул. Этот режим может быть установлен с помощью выполнения операции главного меню: **Сервис** | **Параметры** и выставления отметки в позиции выбора **Формулы** вкладки **Вид** диалогового окна **Параметры**. С помощью удаления этой отметки можно вернуться к обычному режиму изображения ячеек рабочего листа. Данный способ представления формул будет использоваться и далее при решении задачи оптимизации других классов.

Для дальнейшего решения задачи следует воспользоваться инструментом поиска решения программы MS Office Excel 2003. С этой целью необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**.

Примечание

Следует заметить, что при первоначальном обращении к инструменту поиска решения можно не обнаружить в главном меню **Сервис** операции **Поиск решения**. Это означает, что компонент поиска решения в программе MS Office Excel 2003 не установлен. Поэтому для продолжения работы необходимо предварительно установить данный компонент, для чего следует выполнить операцию главного меню: **Сервис** | **Надстройки** и установить отметку в строке с именем **Поиск решения**. При дополнительной установке компонента поиска решения потребуется установочный диск пакета MS Office System 2003.

После вызова инструмента поиска решения появится диалоговое окно мастера задания параметров для нахождения решения (рис. 3.3). Ввиду важности этого инструмента следует более подробно остановиться на элементах его диалоговых окон.

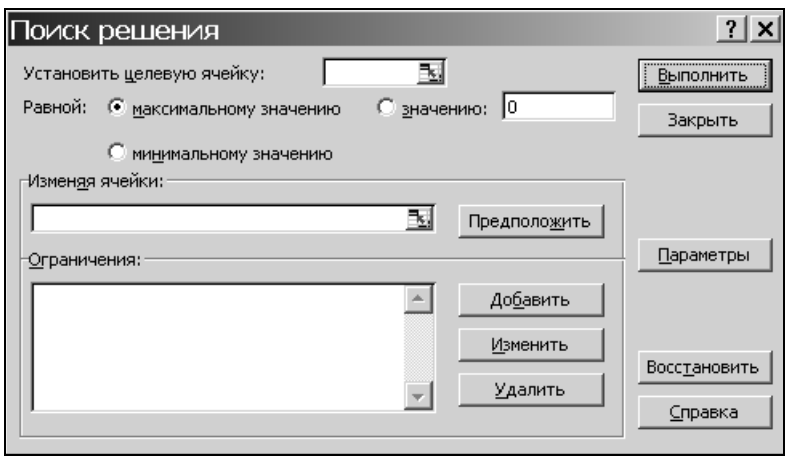


Рис. 3.3. Диалоговое окно **Поиск решения**

Первое диалоговое окно **Поиск решения** (рис. 3.3) имеет следующие элементы:

- поле **Установить целевую ячейку** — используется для задания ссылки на ячейку, в которой задана формула с выражением целевой функции решаемой задачи оптимизации;
- переключатели в группе **Равной** — определяют характер решаемой задачи оптимизации. Для нахождения решения с максимальным значением

целевой функции этот переключатель ставится в положение **максимальному значению**, для нахождения решения с минимальным значением — в положение **минимальному значению**. Наконец, для нахождения решения, при котором целевая функция принимает некоторое фиксированное значение, переключатель ставится в положение **Значению**, справа от которого можно ввести требуемое фиксированное значение;

- поле **Изменяя ячейки** — служит для указания ячеек, которые должны изменяться в процессе поиска решения задачи. Именно в этих ячейках должны находиться переменные решаемой задачи оптимизации;
- в многострочном поле **Ограничения** отображаются ограничения решаемой задачи оптимизации. Для задания ограничений предназначено дополнительное окно **Добавление ограничения**;
- кнопка **Предположить** — служит для автоматического заполнения поля **Изменяя ячейки** теми ячейками, которые указаны в формуле задания целевой функции;
- кнопка **Добавить** — служит для вызова дополнительного окна **Добавление ограничения** (рис. 3.4);

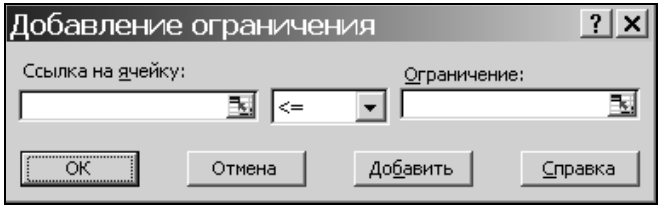


Рис. 3.4. Диалоговое окно **Добавление ограничения**

- кнопка **Изменить** — служит для вызова дополнительного окна **Добавление ограничения**, в котором будет отображено ограничение, выбранное в многострочном поле **Ограничения**;
- кнопка **Удалить** — служит для удаления ограничения, выбранного в многострочном поле **Ограничения**;
- кнопка **Выполнить** — служит для запуска процесса нахождения оптимального решения после спецификации всех параметров поиска решения;
- кнопка **Закрыть** — служит для закрытия диалогового окна **Поиск решения**;
- кнопка **Параметры** — служит для вызова дополнительного окна **Параметры поиска решения** (рис. 3.5) для спецификации дополнительных параметров поиска решения, часть которых уже задана по умолчанию;



Рис. 3.5. Диалоговое окно **Параметры поиска решения**

- кнопка **Восстановить** — служит для очистки полей данного диалогового окна и восстановления значений параметров поиска решения, используемых по умолчанию;
- кнопка **Справка** — служит для получения справочной информации об элементах диалогового окна **Поиск решения**.

Диалоговое окно **Добавление ограничения** (рис. 3.4) предназначено для задания одного ограничения и имеет следующие элементы:

- поле **Ссылка на ячейку** — служит для указания ячейки или диапазона ячеек, в которых содержится левая часть задаваемого ограничения;
- выпадающий список в центре окна — содержит перечень знаков неравенств ограничений, а также возможность спецификации требования целочисленности или двоичных значений переменных. Мастер поиска решений допускает спецификацию ограничений в виде равенств и неравенств;
- поле **Ограничение** — служит для указания ячейки, диапазона ячеек или конкретного числа, которое специфицирует правую часть задаваемого ограничения;
- кнопка **ОК** — служит для добавления ограничения к параметрам поиска решения и закрытия окна **Добавление ограничения**;
- кнопка **Отмена** — для закрытия окна **Добавление ограничения** без добавления ограничения к параметрам поиска решения;

- кнопка **Добавить** — для добавления ограничения к параметрам поиска решения без закрытия окна **Добавление ограничения**;
- кнопка **Справка** — служит для получения справочной информации об элементах диалогового окна **Добавление ограничения**.

Диалоговое окно **Параметры поиска решения** (рис. 3.5) предназначено для спецификации дополнительных параметров нахождения решения или изменения уже заданных параметров. Следует заметить, что значения большинства этих параметров, используемых по умолчанию, подходят для решения типовых задач оптимизации. Диалоговое окно **Параметры поиска решения** имеет следующие элементы.

- Поле **Максимальное время** служит для ограничения времени, отпускаемого на поиск решения задачи. В это поле можно ввести время (в секундах), не превышающее 32 767. Установленное по умолчанию значение 100 подходит для решения рассматриваемых типовых задач.
- Поле **Предельное число итераций** служит для ограничения времени решения задачи посредством задания некоторого предельного числа промежуточных вычислений. Установленное по умолчанию значение 100 подходит для решения рассматриваемых типовых задач.
- Поля **Относительная погрешность** и **Допустимое отклонение** служат для задания точности, с которой будет выполняться поиск решения. В отдельных случаях целесообразно после нахождения решения со значениями этих параметров, заданными по умолчанию, повторить вычисления с большей точностью и меньшим допустимым отклонением. После выполнения поиска решения сравнить полученные результаты.
- Флажок **Линейная модель** служит для ускорения поиска решения задачи оптимизации с линейной целевой функцией и линейными ограничениями. Для задач нелинейной оптимизации этот флажок должен быть сброшен, а для задач линейного программирования — установлен, поскольку в противном случае возможно получение неверного результата.
- Флажок **Показывать результаты итераций** служит для приостановки поиска решения для просмотра результатов отдельных итераций.
- Флажок **Автоматическое масштабирование** служит для включения автоматической нормализации входных и выходных значений, качественно различающихся по величине, — например, максимизация прибыли в процентах по отношению к вложениям, исчисляемым в миллионах рублей.
- Флажок **Неотрицательные значения** позволяет установить нулевую нижнюю границу для тех влияющих ячеек, для которых она не была указана в поле **Ограничение** диалогового окна **Добавить ограничение**.

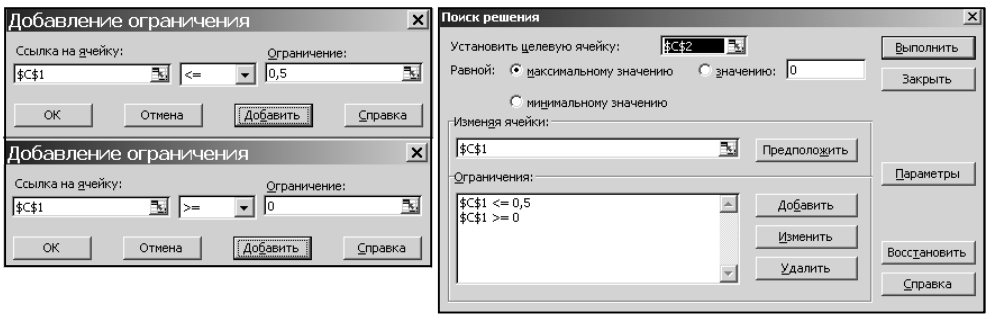
- Параметры группы **Оценки** служат для указания метода экстраполяции (линейная или квадратичная) используемого для получения исходных оценок значений переменных в каждом одномерном поиске. При этом:
  - переключатель **линейная** служит для использования линейной экстраполяции вдоль касательного вектора, которая дает лучшие результаты при решении линейных задач;
  - переключатель **квадратичная** служит для использования квадратичной экстраполяции, которая дает лучшие результаты при решении нелинейных задач.
- Параметры группы **Разности** служат для указания метода численного дифференцирования (прямые или центральные производные), который используется для вычисления частных производных целевых и ограничивающих функций. При этом:
  - переключатель **прямые** используется в большинстве задач, где скорость изменения ограничений относительно невысока;
  - переключатель **центральные** используется для функций, имеющих разрывную производную. Данный способ требует больше вычислений, однако его применение может быть оправданным, если выдается сообщение о том, что получить более точное решение не удается.
- Параметры группы **Метод поиска** служат для выбора алгоритма оптимизации (метод Ньютона или сопряженных градиентов) для указания направление поиска. При этом:
  - переключатель **Ньютона** обеспечивает выбор для нахождения решения квазиньютоновского метода, который требует больше памяти, но выполняется за меньшее число итераций, чем в методе сопряженных градиентов;
  - переключатель **сопряженных градиентов** обеспечивает выбор для нахождения решения метода сопряженных градиентов, который требует меньше памяти, но выполняется за большее число итераций, чем в методе Ньютона. Данный метод следует использовать, если задача достаточно велика, и необходимо экономить память, а также, если итерации дают слишком малое отличие в последовательных приближениях.
- Кнопка **ОК** служит для добавления дополнительных параметров и закрытия окна **Параметры поиска решения**.
- Кнопка **Отмена** для закрытия окна **Параметры поиска решения** без изменения дополнительных параметров поиска решения.
- Кнопка **Загрузить модель** служит для вызова диалогового окна **Загрузить модель**, в котором можно задать ссылку на область ячеек, содержащих загружаемую модель.

- Кнопка **Сохранить модель** служит для вызова диалогового окна **Сохранить модель**, в котором можно задать ссылку на область ячеек, предназначенную для хранения модели оптимизации. Данный вариант предусмотрен для хранения на листе более одной модели оптимизации, при этом первая модель сохраняется автоматически.
- Кнопка **Справка** служит для получения справочной информации об элементах диалогового окна **Параметры поиска решения**.

Продолжим решение задачи о коробке максимального объема. В первом окне мастера поиска решения следует в поле с именем **Установить целевую ячейку** указать ячейку **$C$2**, в которой содержится формула для расчета целевой функции задачи, а в поле с именем **Изменяя ячейки** ввести абсолютный адрес ячейки `$C$1`, в которую будет записано искомое решение задачи.

После этого следует добавить два ограничения на допустимые значения переменной. Для этого следует нажать кнопку **Добавить**, в результате чего появится дополнительное окно задания ограничений (рис. 3.6, *а*). Для ввода адресов ячеек в левую часть ограничения предназначено поле с именем **Ссылка на ячейку**. Форма неравенства ограничения выбирается из вложенного списка в средней части окна. Наконец, для задания правой части ограничений в поле ввода с именем **Ограничение** следует ввести с клавиатуры следующие числа: 0,5 — для первого ограничения и 0 — для второго. После спецификации каждого из ограничений их необходимо включить в модель расчета, для чего следует нажать кнопку **Добавить** или **OK**. Для того чтобы закрыть дополнительное окно задания ограничений и вернуться в исходное окно мастера поиска решения, следует нажать кнопку **Отмена** или **OK**.

Остальные параметры мастера поиска решения, для редактирования которых предназначено дополнительное второе окно этого мастера, можно оставить без изменения. Окончательный внешний вид диалогового окна мастера поиска решения после задания необходимой информации изображен на рис. 3.6, *б*.



а *б*

Рис. 3.6. Параметры мастера поиска решения для задачи о коробке

Для редактирования некоторого ограничения его следует выделить в многострочном поле с именем **Ограничения** и нажать кнопку с именем **Изменить**. В этом случае появится дополнительное окно, аналогичное изображенным на рис. 3.6, *а*, в котором можно выполнить необходимые действия по редактированию выбранного ограничения.

После задания необходимых параметров поиска решения можно приступить к выполнению численных расчетов, для чего следует нажать кнопку **Выполнить**. После выполнения численных расчетов программой MS Excel практически мгновенно будет получено количественное решение, которое имеет следующий вид (рис. 3.7).

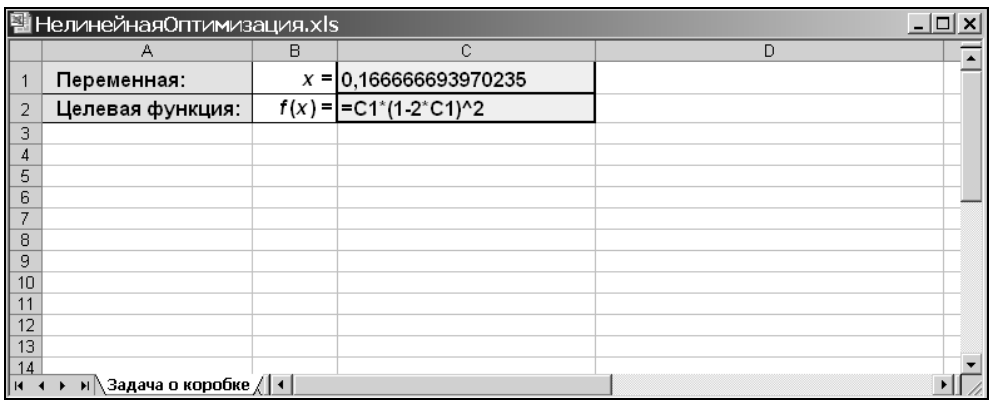


Рис. 3.7. Результат количественного решения задачи о коробке максимального объема

Интерпретируя полученные количественные значения, можно прийти к следующему заключению. Результатом решения является оптимальное значение стороны вырезаемого квадрата: r_{opt} = 0,166 666 693 970 235, при котором изготовленная коробка будет иметь максимальный объем: V_{max} = 0,074 074 074 074 073. Напомним, что это решение соответствует длине стороны исходной заготовки, равной 1.

Примечание

При выполнении расчетов для ячеек **C1:C2** был выбран числовой формат с 15 знаками после запятой. Это можно рассматривать в качестве предельного случая, поскольку реально подобная точность может потребоваться весьма редко. Тем не менее, анализ вычислительной точности методов нахождения решений данной задачи оптимизации программой MS Excel основывается на сравнении найденного решения и аналитического приближенного решения *(см. разд. 3.2.3)* с максимальным количеством значащих цифр.

Полученное решение имеет общий характер, поскольку в любом конкретном случае для получения решения рассматриваемой задачи оптимизации значение L следует умножить на найденное значение r_{opt}. Например, если $L = 2$ *м*, то для получения коробки максимального объема следует вырезать по углам исходной заготовки квадраты со стороной $0{,}166\,666\,693\,970\,235 \times 2 = 0{,}333\,333\,387\,940\,470$ *м*.

Таким образом, задача о коробке максимального объема в каждом конкретном случае может быть решена простой подстановкой в расчетную формулу: $r_{opt} = 0{,}166\,666\,693\,970\,235 \times L$ конкретного значения величины L. Допустимая точность решения задается в каждом конкретном случае, исходя из специфики задачи и соответствующей проблемной области.

Примечание

Получение решения задачи оптимизации в виде расчетной формулы, вообще говоря, делает излишним использование программы MS Excel для решения конкретных задач данного типа. Тем не менее, из методических соображений процесс получения данного решения с помощью программы MS Excel рассмотрен достаточно подробно, поскольку он может быть использован для решения аналогичных задач нелинейной оптимизации.

Для анализа найденного решения можно построить график целевой функции (3.2.1) и визуально оценить его корректность. С этой целью на отдельном рабочем листе с помощью операции автозаполнения ячеек зададим последовательный ряд чисел исходной переменной r в диапазоне от 0 до 1 с интервалом 0,01, которые запишем в ячейки с адресами **A2:A52**. Рядом в ячейки **B2:B52** запишем соответствующие значения целевой функции (3.2.1). Для этого можно записать формулу: `=A2*(1-2*A2)^2` в ячейку **B2** и с помощью операции автозаполнения "протащить" содержание этой ячейки на диапазон **B3:B52**.

После этого для построения графика целевой функции для задачи о коробке следует воспользоваться мастером диаграмм. Построенный график целевой функции будет иметь следующий вид (рис. 3.8).

Визуальный анализ этого графика показывает, что максимум целевой функции находится между значениями $x = 0{,}16$ и $x = 0{,}17$. Этот факт можно также проверить, сравнив значения в ячейках **B18** и **B19**. Тем самым, можно сделать вывод о корректности полученного результата решения данной задачи оптимизации.

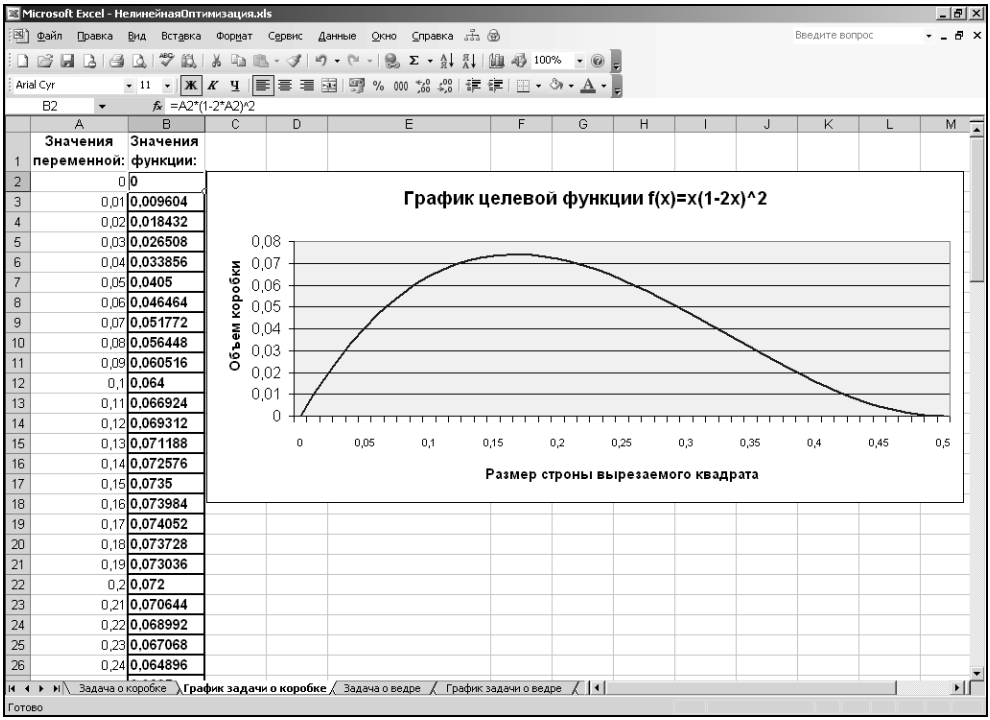


Рис. 3.8. График целевой функции
в задаче о коробке максимального объема

3.2.3. Аналитическое решение задачи о коробке максимального объема

Поскольку задача о коробке максимального объема имеет достаточно простое аналитическое решение, оно приводится для дополнительной проверки правильности численных расчетов программы MS Excel.

Прежде всего, следует заметить, что целевая функция (3.2.1) является непрерывной и дифференцируемой на всем интервале ее задания, включая множество допустимых альтернатив Δ_β, которое, в свою очередь, является замкнутым. Как следует из курса математического анализа, непрерывная функция на замкнутом множестве достигает своих экстремальных значений, из которых нас интересуют точки максимума данной функции. Для нахождения экстремума аналитическим способом следует определить первую производную целевой функции и отыскать ее корни.

Первая производная функции: $f(x) = x \cdot (1-2x)^2$ равна: $\frac{df(x)}{dx} = 12x^2 - 8x + 1$.

Соответствующее уравнение: $12x^2 - 8x + 1 = 0$ является квадратным, которое имеет два корня: $x_1 = 1/6$ и $x_2 = 1/2$. Анализ графика целевой функции (см. рис. 3.5) показывает, что максимальному значению этой функции соответствует первый корень. Данное значение является допустимым и служит точным аналитическим решением задачи оптимизации о коробке максимального размера. Таким образом, $r_{opt} = 1/6$. Поскольку число 1/6 является иррациональным и представляет собой периодическую дробь 0,16(6), при желании можно получить его любое приближение произвольной точности. Сравнивая найденное ранее значение 0,166 666 693 970 235 с соответствующим округлением точного значения 0,166 666 666 666 667, можно получить относительную погрешность программы MS Excel. Это значение равно: 0,000 016 382 137 877 % и свидетельствует о том, что решение задач оптимизации программой MS Excel с гладкой целевой функцией выполняется весьма эффективно.

3.3. Задача о пожарном ведре

Содержательная постановка задачи о пожарном ведре максимального объема приводится в *разд. 1.2.2*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи оптимизации.

3.3.1. Математическая постановка задачи о пожарном ведре

Дополним содержательную постановку этой задачи соответствующими параметрами, аналогично задаче о коробке. Имеется заготовка в форме круга из некоторого гибкого и влагостойкого материала радиуса R (рис. 3.9, *а*). Из заготовки следует вырезать некоторой сектор, имеющий угол α. После чего, согнув полученную фигуру и сварив шов, можно получить конус без своего основания или ведро, которое можно вполне использовать для тушения пожара (рис. 3.9, *б*). Необходимо так выбрать угол вырезаемого сектора, чтобы пожарное ведро оказалось максимального объема.

Для математической постановки данной задачи также необходимо определить переменные соответствующей задачи оптимизации, задать целевую функцию и специфицировать ограничения. Очевидно, в качестве переменной задачи следует взять угол вырезаемого сектора α, который, исходя из содержательной постановки задачи, может принимать непрерывные действительные

значения. Целевой функцией данной задачи является объем полученного ведра или конуса. Поскольку радиус основания конуса равен r, а высота конуса равна h, то ее объем находится по формуле: $V(\alpha) = (1/3) \cdot \pi \cdot r^2 \cdot h$. Сложность заключается в том, что как радиус основания, так и высота конуса являются некоторыми функциями от угла сектора, т. е. $r = r(\alpha)$ и $h = h(\alpha)$ соответственно.

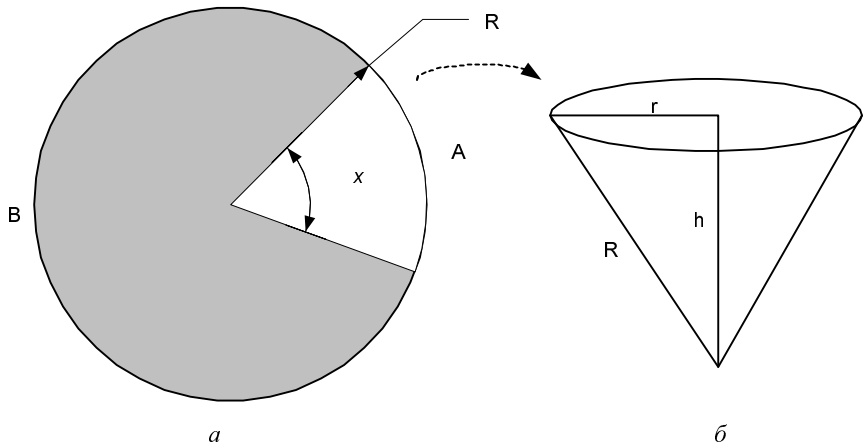


Рис. 3.9. Схема изготовления ведра из круглой заготовки с указанием ее размеров

Хотя из физических соображений значения переменной α могут быть как положительными, так и отрицательными, целесообразно ограничиться их рассмотрением в диапазоне от 0° до 360°.

Примечание

При значениях $\alpha = 0°$ и $\alpha = 360°$, что в сущности означает одно и то же, соответствующие решения задачи о ведре являются вырожденными. Как в первом случае, так и во втором — заготовка остается без изменения, поэтому данные предельные значения можно считать допустимыми. Тем самым задачу о пожарном ведре следует считать задачей оптимизации с ограничениями типа *нестрогих* неравенств. Для удобства градусы можно опустить, фиксируя тем самым физическую размерность переменной.

Для математической постановки данной задачи целевую функцию необходимо преобразовать к виду, в котором установлена явная зависимость объема конуса от угла вырезаемого сектора из исходной заготовки. Из рис. 3.9, *а* видно, что длина окружности исходной заготовки равна сумме длины дуги A и длины дуги B, причем длина дуги B равна длине окружности основания

конуса. С другой стороны, длина окружности исходной заготовки равна: $2\pi \cdot R$, где R — радиус исходной заготовки.

Используя несложные формулы пропорций, можно получить выражение для длины дуги A, которая равна: $2\pi \cdot R \cdot \alpha/360$. Откуда длина основания конуса или ведра равна: $2\pi \cdot R - 2\pi \cdot R \cdot \alpha/360$. Поскольку $2\pi \cdot r = 2\pi \cdot R - 2\pi \cdot R \cdot \alpha/360$, после несложных преобразований получим: $r(\alpha) = R(1 - \alpha/360)$.

Для нахождения зависимости высоты конуса от угла сектора заметим, что R, r и h образуют стороны прямоугольного треугольника (рис. 3.9, *б*). Откуда по теореме Пифагора непосредственно следует: $h(\alpha) = \sqrt{R^2 - r^2} = R \cdot \sqrt{1 - (1 - \frac{\alpha}{360})^2}$.

С целью унификации обозначений и удобства дальнейших рассуждений, обозначим переменную задачи оптимизации через $x = \alpha$. Тогда математическая постановка задачи о пожарном ведре максимального объема может быть записана в следующем виде:

$$f(x) = \frac{1}{3}\pi \cdot R^3 \cdot (1 - \frac{x}{360})^2 \cdot \sqrt{1 - (1 - \frac{x}{360})^2} \to \max_{x \in \Delta_\beta}, \qquad (3.3.1)$$

где $$\Delta_\beta = \{x \in R^1 \mid 0 \le x \le 360\}.$$

Целевая функция данной задачи также является нелинейной, поэтому задача о ведре максимального объема относится к классу задач нелинейного программирования или нелинейной оптимизации.

3.3.2. Решение задачи о пожарном ведре максимального объема с помощью программы MS Excel

Не уменьшая общности математической постановки задачи (3.3.1), предположим: $R = 1$. Из вида целевой функции можно заключить, что R выступает в качестве постоянного коэффициента и так же, как параметр π, не оказывает влияния на нахождение оптимального решения. Для решения данной задачи с помощью программы MS Excel перейдем на новый лист, который назовем Задача о ведре. Сделаем необходимые надписи в ячейках **A1:B2**. После этого введем в ячейку **C2** формулу: `=(1/3)*3,1415916*((1-C1/360)^2)*КОРЕНЬ(1-(1-C1/360)^2)`, которая представляет целевую функцию (3.3.1). Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о пожарном ведре максимального объема имеет следующий вид (рис. 3.10).

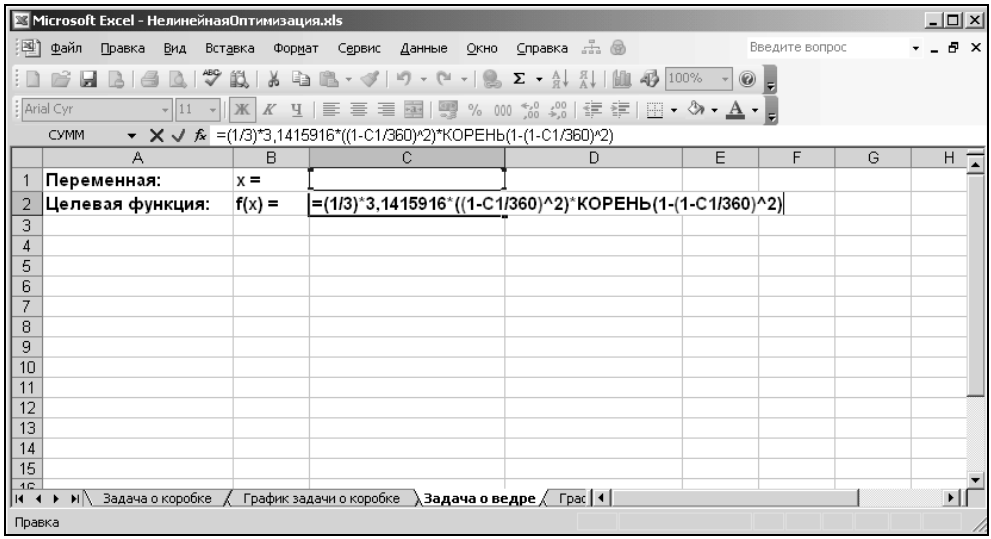
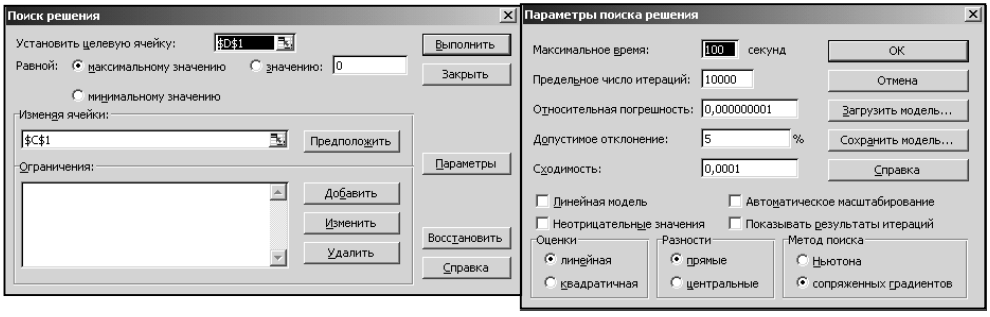


Рис. 3.10. Исходные данные для решения задачи о пожарном ведре

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**… После этого следует в поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячейки `$C$1` (рис. 3.11, *а*). Поля с ограничениями можно оставить пустыми, поскольку целевая функция является выпуклой на всем множестве допустимых значений.



а *б*

Рис. 3.11. Параметры мастера поиска решения задачи о пожарном ведре

Остальные параметры мастера поиска решения (рис. 3.11, *б*) можно оставить без изменения. После этого можно приступить к выполнению численных расчетов, для чего следует нажать кнопку **Выполнить**. После выполнения

численных расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 3.12).

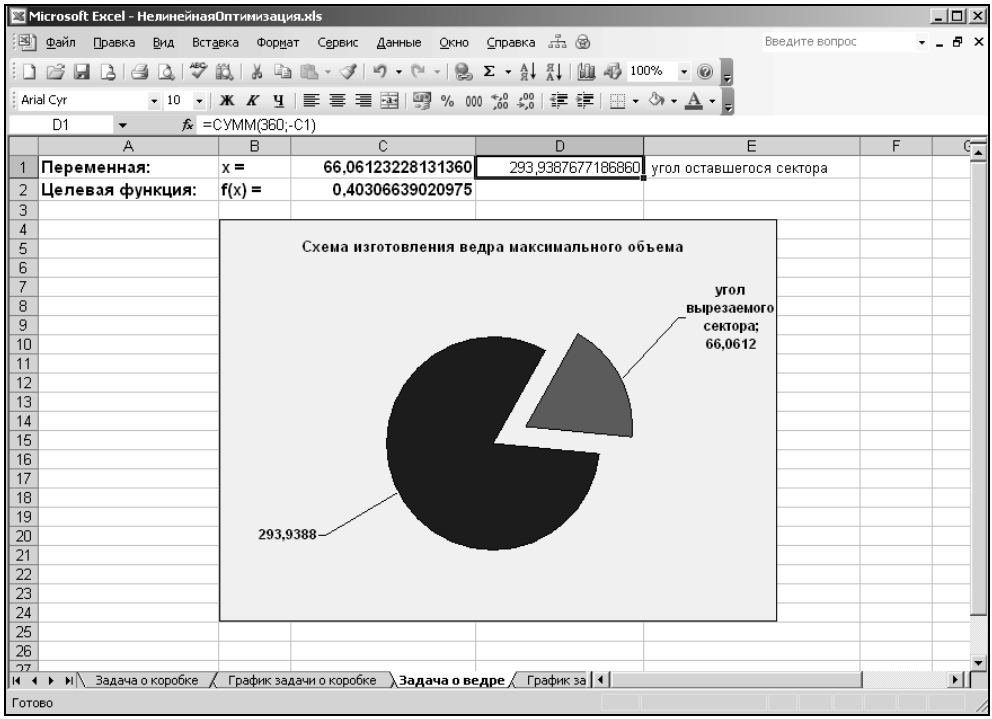


Рис. 3.12. Результат количественного решения задачи о пожарном ведре

Результатом решения является оптимальное значение угла вырезаемого сектора: $\alpha_{opt} = 66{,}061\ 232\ 281\ 3136°$, при котором изготовленное пожарное ведро будет иметь максимальный объем: $V_{max} = 0{,}403\ 066\ 390\ 2097$. Само оптимальное решение α_{opt} не зависит от радиуса исходной заготовки, в отличие от объема пожарного ведра V_{max}. Это означает, что во всех конкретных случаях для получения оптимального решения рассматриваемой задачи оптимизации следует принять $\alpha_{opt} = 66{,}061\ 232\ 281\ 3136°$. Для получения же значения V_{max} следует воспользоваться формулой для целевой функции (3.3.1). Допустимая точность решения задается в каждом конкретном случае отдельно, исходя из специфики задачи и соответствующей проблемной области.

Примечание

Получение решения данной задачи оптимизации в виде $\alpha_{opt} = 66{,}061\ 232\ 281\ 3136°$ также делает излишним использование программы MS Excel для решения конкретных задач данного типа. Данное решение было

получено при установлении числа десятичных знаков числового формата для ячейки **C1**, равным 13.

Для анализа найденного решения можно построить график целевой функции (3.3.1) и визуально оценить его корректность. С этой целью на отдельном рабочем листе с помощью операции автозаполнения ячеек зададим последовательный ряд чисел исходной переменной α в диапазоне от 0 до 91 с интервалом 1, которые запишем в ячейки с адресами **A3:A94**. Рядом в ячейки **B3:B94** запишем соответствующие значения целевой функции (3.3.1). Для этого можно скопировать формулу:

```
=(1/3)*3,1415916*((1-C1/360)^2)*КОРЕНЬ(1-(1-C1/360)^2)
```

в ячейку **B3** и с помощью операции автозаполнения "протащить" содержание этой ячейки на диапазон **B3:B94**.

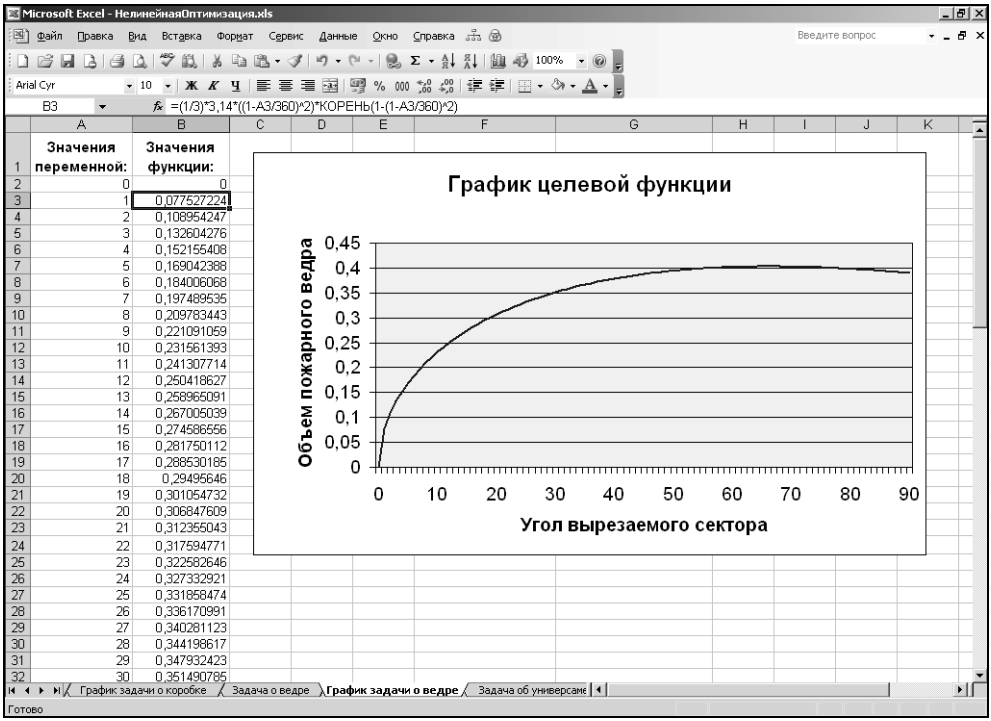


Рис. 3.13. График целевой функции в задаче о пожарном ведре максимального объема

После этого для построения графика целевой функции для задачи о пожарном ведре следует воспользоваться мастером диаграмм. Построенный график целевой функции будет иметь следующий вид (рис. 3.13). Для повышения

наглядности на этом же листе можно построить круговую диаграмму, которая иллюстрирует графически размер оптимального вырезаемого сектора, соответствующего пожарному ведру максимального объема.

Визуальный анализ этого графика показывает, что максимум целевой функции находится между значениями $x = 0{,}402\ 862\ 083$ и $x = 0{,}402\ 837\ 715$. Этот факт можно также проверить, сравнив значения в ячейках **B69** и **B70**. Тем самым, можно сделать вывод о корректности полученного результата решения данной задачи оптимизации.

3.3.3. Аналитическое решение задачи о пожарном ведре

Задача о пожарном ведре максимального объема имеет аналитическое решение, трудность нахождения которого связана с нахождением выражения для первой производной целевой функции (3.3.1).

Прежде всего следует заметить, что целевая функция (3.3.1) является непрерывной и дифференцируемой на всем интервале ее задания, включая множество допустимых альтернатив Δ_β, которое, в свою очередь, является замкнутым. Как следует из курса математического анализа, непрерывная функция на замкнутом множестве достигает своих экстремальных значений, из которых нас интересуют точки максимума данной функции. Для нахождения экстремума аналитическим способом следует определить первую производную целевой функции и отыскать ее корни.

Первая производная функции:

$$f(x) = \frac{1}{3}\pi \cdot R^3 \cdot (1 - \frac{x}{360})^2 \cdot \sqrt{1 - (1 - \frac{x}{360})^2}$$ равна:

$$\frac{df(x)}{dx} = -\frac{\pi \cdot R^3}{3 \cdot 360}\left[2 \cdot \left(1 - \frac{x}{360}\right) \cdot \left(1 - \left(1 - \frac{x}{360}\right)^2\right)^{\frac{1}{2}} - \left(1 - \frac{x}{360}\right)^3 \cdot \left(1 - \left(1 - \frac{x}{360}\right)^2\right)^{-\frac{1}{2}}\right].$$

Соответствующее данной производной уравнение после несложных преобразований примет следующий вид:

$$2 - \left(1 - \frac{x}{360}\right)^2 \cdot \left(1 - \left(1 - \frac{x}{360}\right)^2\right)^{-1} = 0 \text{ или } \frac{y^2}{1 - y^2} = 2, \text{ где } y = 1 - \frac{x}{360}.$$

Решая квадратное уравнение относительно переменной y, получим два корня: $y_1 = \sqrt{\frac{2}{3}}$ и $y_2 = -\sqrt{\frac{2}{3}}$. Откуда, подставляя найденные значения в формулу: $x = 360 \cdot (1 - y)$, можно получить точное аналитическое решение задачи о пожарном ведре. Поскольку второе значение корня y_2 выходит за пределы множества допустимых альтернатив, искомое аналитическое решение будет равно: $x = 360 \cdot \left(1 - \sqrt{\frac{2}{3}}\right)$ или $x = 360 \cdot \left(\frac{\sqrt{3} - \sqrt{2}}{\sqrt{3}}\right)$. Данное значение является допустимым и служит точным решением задачи оптимизации о пожарном ведре максимального размера. Таким образом, для удобства расчетов, можно воспользоваться выражением: $\alpha_{opt} = 360 - 120 \cdot \sqrt{6}$. Поскольку число $\sqrt{6}$ является иррациональным и не может быть представлено в виде некоторой периодической дроби, точность численных расчетов по данной формуле является ограниченной. Так, например, если взять в качестве приближенного значения: $\sqrt{6} \approx 2{,}449\,489\,742\,7832$, то получим также приближенное значение $\alpha_{opt} = 66{,}061\,230\,866\,0187°$. Сравнивая найденное ранее значение $\alpha_{opt} = 66{,}061\,232\,281\,3136°$ с соответствующим приближенным значением аналитического решения 66,061 230 866 0187, можно получить относительную погрешность программы MS Excel. Это значение равно: 0,000 002 142 398 671%. Это также свидетельствует о том, что решение данной задачи оптимизации программой MS Excel выполнено достаточно эффективно.

3.4. Задача о строительстве универсама

Обе рассмотренные ранее задачи относятся к классу задач *одномерной* нелинейной оптимизации, поскольку имеют единственную переменную. Далее описывается задача о строительстве универсама, которая является примером задач *многомерной* нелинейной оптимизации. В математической модели этой задачи используется две независимые переменные, каждая из которых представляет отдельную координату точки на плоскости. Поскольку данная задача не описывалась в *главе 1*, ее рассмотрение начинается с содержательной постановки задачи.

3.4.1. Содержательная постановка задачи о строительстве универсама

Вообще говоря, данная задача может иметь несколько возможных вариантов постановки, отличающихся друг от друга количеством исходных жилых домов. Для конкретности рассмотрим один из вариантов этой задачи.

Имеется 4 жилых дома, расположенных в некотором микрорайоне города. Требуется определить местоположение для строительства универсама, так чтобы общее расстояние от построенного универсама до всех жилых домов было минимальным (рис. 3.14).

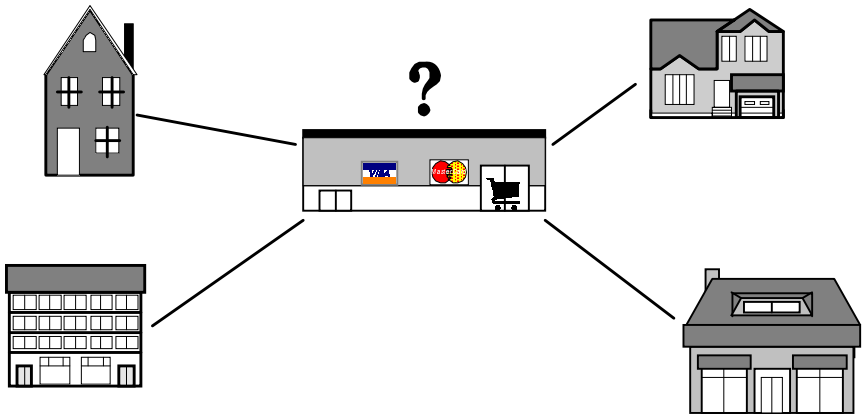


Рис. 3.14. Иллюстрация содержательной постановки задачи о строительстве универсама

Данная задача является обобщением задачи о трех точках, которая предлагается в качестве упражнения 3.6.19 в конце этой главы. Другие варианты задачи о строительстве универсама могут быть сформулированы как для различных значений количества домов, так и для различных видов целевой функции, например, такой, в которой используется взвешенная сумма расстояний до всех домов. Последняя задача может являться конкретизацией обобщенной задачи о точках, которая предлагается в качестве упражнения 3.6.20.

3.4.2. Математическая постановка задачи о строительстве универсама

Для математической постановки задачи следует ввести некоторые обозначения для географических координат 4-х исходных жилых домов. С этой целью, не уменьшая общности задачи, введем некоторую прямоугольную систему координат, в которой исходные дома и универсам будут представлять собой отдельные точки на плоскости (рис. 3.15). Координаты исходных домов могут быть записаны как координаты соответствующих точек в виде: (x_i, y_i), где $i \in \{1, 2, 3, 4\}$. Искомые координаты универсама, который предполагается построить, можно положить равными: (x, y). Очевидно, они служат переменными рассматриваемой задачи оптимизации, каждая из которых по своему характеру может принимать действительные значения из $\boldsymbol{R}^1$.

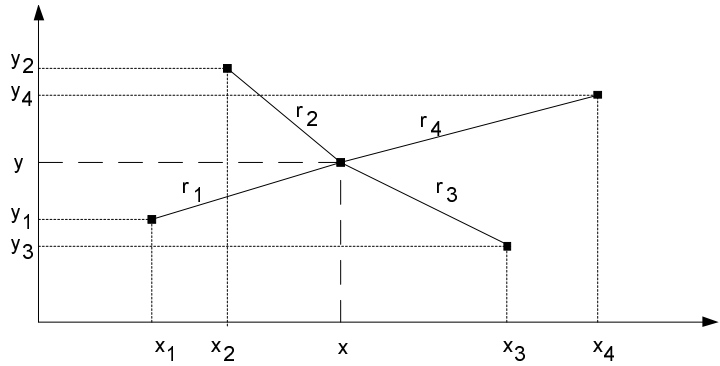


Рис. 3.15. Иллюстрация математической постановки задачи о строительстве универсама

В некоторой фиксированной прямоугольной системе координат значения переменных x и y могут быть как положительными, так и отрицательными. Тем самым задачу о строительстве универсама можно считать задачей оптимизации без ограничений.

В качестве целевой функции данной задачи будем рассматривать сумму расстояний от искомой точки (x, y) до каждой из заданных точек (x_i, y_i), $(\forall i \in \{1, 2, 3, 4\})$, рассчитанных по формуле Евклида. Каждое отдельное расстояние в этом случае равно: $r_i = \sqrt{(x - x_i)^2 - (y - y_i)^2}$, где $i \in \{1, 2, 3, 4\}$. Тогда общее расстояние будет определяться выражением: $r = r_1 + r_2 + r_3 + r_4$.

Таким образом, математическая постановка задачи о строительстве универсама может быть записана в следующем виде:

$$f(x, y) = \sum_{i=1}^{4} \sqrt{(x - x_i)^2 - (y - y_i)^2} \to \min_{x, y \in \Delta}, \qquad (3.4.1)$$

где $$\Delta = \{x, y \in \boldsymbol{R}^1\}.$$

Поскольку целевая функция данной задачи является нелинейной, задача о строительстве универсама относится к классу задач нелинейного программирования без ограничений.

3.4.3. Решение задачи о строительстве универсама с помощью программы MS Excel

Для решения данной задачи с помощью программы MS Excel создадим новый лист с именем Задача об универсаме. Сделаем необходимые надписи в ячейках **A1:A12**, **B1**, **C2:C7**. После этого введем в ячейку **B9** формулу: `=КОРЕНЬ(($B$2-B4)^2+($D$2-D4)^2)`, которую скопируем в ячейки **B10:B12**.

В ячейку **D3** введем формулу: `=СУММ(B9:B12)`, которая представляет целевую функцию (3.4.1). Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о строительстве универсама имеет следующий вид (рис. 3.16).

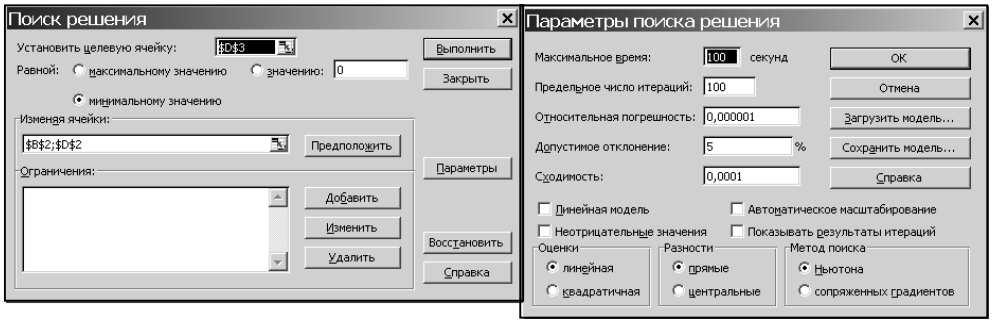
НелинейнаяОптимизация.xls

	A	B	C	D	E
1	**Переменные:**	**Оптимальное решение:**			
2	X=		y=		
3	**Исходные координаты домов:**		**Целевая функция:**	=СУММ(B9:B12)	
4	x1=		y1=		
5	x2=		y2=		
6	x3=		y3=		
7	x4=		x4=		
8	**Расстояния до домов:**				
9	r1=	=КОРЕНЬ(($B$2-B4)^2+($D$2-D4)^2)			
10	r2=	=КОРЕНЬ(($B$2-B5)^2+($D$2-D5)^2)			
11	r3=	=КОРЕНЬ(($B$2-B6)^2+($D$2-D6)^2)			
12	r4=	=КОРЕНЬ(($B$2-B7)^2+($D$2-D7)^2)			
13					
14					

Задача об универсаме

Рис. 3.16. Исходные данные для решения задачи о строительстве универсама

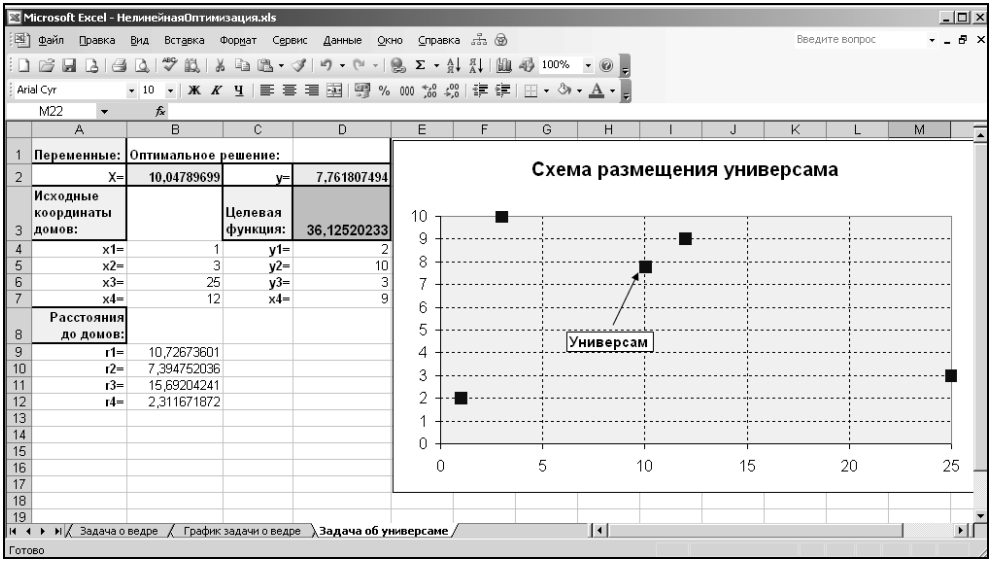
Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**. После этого следует в поле с именем **Установить целевую ячейку** ввести абсолютный адрес ячейки `$D$3` со значением целевой функции, а в поле с именем **Изменяя ячейки** ввести абсолютный адрес ячеек `$B$2; $D$2` (рис. 3.17, *а*). Поля с ограничениями можно оставить пустыми, поскольку целевая функция является выпуклой на всем множестве допустимых значений. Параметры поиска решения можно оставить без изменения (рис. 3.17, *б*).



а *б*

Рис. 3.17. Параметры мастера поиска решения задачи о строительстве универсама

После этого можно приступить к выполнению численных расчетов, для чего следует нажать кнопку **Выполнить**. После выполнения численных расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 3.18). На основе полученного численного решения можно построить диаграмму, которая наглядно иллюстрирует расположение универсама относительно жилых домов.



	A	B	C	D
1	Переменные:	Оптимальное решение:		
2	X=	10,04789699	y=	7,761807494
3	Исходные координаты домов:		Целевая функция:	36,12520233
4	x1=	1	y1=	2
5	x2=	3	y2=	10
6	x3=	25	y3=	3
7	x4=	12	x4=	9
8	Расстояния до домов:			
9	r1=	10,72673601		
10	r2=	7,394752036		
11	r3=	15,69204241		
12	r4=	2,311671872		

Рис. 3.18. Результат количественного решения задачи о строительстве универсама

Результатом решения задачи для четырех жилых домов с координатами (1, 2), (3, 10), (25, 3) и (12, 9) является оптимальное расположение универсама с координатами $x^* = 10{,}047\ 896\ 989\ 5119$ и $y^* = 7{,}761\ 807\ 493\ 743\ 75$, при которых общее расстояние от универсама до жилых домов будет минимальным и равно значению: $f(x^*, y^*) = 32{,}125\ 202\ 33$. Допустимая точность решения задается в каждом конкретном случае отдельно, исходя из специфики задачи.

Аналитическое решение задачи о строительстве универсама достаточно трудоемко, что наглядно демонстрирует достоинство использования программы MS Excel для решения задач оптимизации подобного класса.

Примечание

Заинтересованные читатели могут попытаться самостоятельно найти аналитическое решение и сравнить его с полученным расчетным оптимальным значением. В качестве упражнения предлагается также изменить координаты жилых

домов и выполнить соответствующий поиск решения для различных вариантов их расположения. Предлагается также выполнить поиск решения для различного количества жилых домов, изменив соответствующим образом структуру задания исходных данных на рабочем листе программы MS Excel (рис. 3.18).

3.5. Тестовые задачи нелинейного программирования

Настоящая глава содержит специальную информацию для более глубокого анализа вычислительных алгоритмов, реализованных в программе MS Excel. В общем случае для практической оценки вычислительных программ и алгоритмов решения задач нелинейной оптимизации служат некоторые специально подобранные задачи, имеющие не совсем "хорошие" целевые функции. Традиционно в качестве таких функций используются: функция Розенброка, функция Пауэлла и двумерная экспоненциальная функция. Задачи оптимизации с этими целевыми функциями являются предметом рассмотрения в данном разделе.

3.5.1. Задача оптимизации с целевой функцией Розенброка и ее решение с помощью программы MS Excel

Задача является чисто математической и формулируется следующим образом. Необходимо найти минимум следующей целевой функции двух переменных (без ограничений):

$$f(x_1,x_2)=100(x_2-x_1^2)^2+(1-x_1)^2 \rightarrow \min_{x_1,x_2\in\Delta_\beta}, \quad \text{где} \quad \Delta_\beta=\{x_1,x_2\in \boldsymbol{R}^1\}. \qquad (3.5.1)$$

Целевая функция данной задачи является нелинейной, поэтому соответствующая задача оптимизации относится к классу задач нелинейного программирования или нелинейной оптимизации. Особенностью этой функции является ее так называемый *двуовражный* характер, что затрудняет нахождение ее минимума некоторыми вычислительными алгоритмами.

Поскольку функция Розенброка принимает неотрицательные значения, точное решение соответствующей задачи оптимизации находится достаточно просто и равно: $x_1^* = 1$, $x_2^* = 1$.

Для вычислительного решения данной задачи с помощью программы MS Excel создадим новый лист с именем: Функция Розенброка. Сделаем необходимые надписи в ячейках **A1:A2**, **B1**, **C2:C3**. После этого введем в ячей-

ку **D3** формулу: `=100*(B2-D2^2)^2+(1-D2)^2`, которая представляет функцию Розенброка (3.5.1). После задания целевой ячейки и изменяемых ячеек в мастере можно выполнить поиск решения данной задачи.

Для повышения точности решения можно изменить значения параметров расчета и повторно выполнить поиск решения. Внешний вид рабочего листа MS Office Excel 2003 с решением задачи оптимизации с целевой функцией Розенброка имеет следующий вид (рис. 3.19).

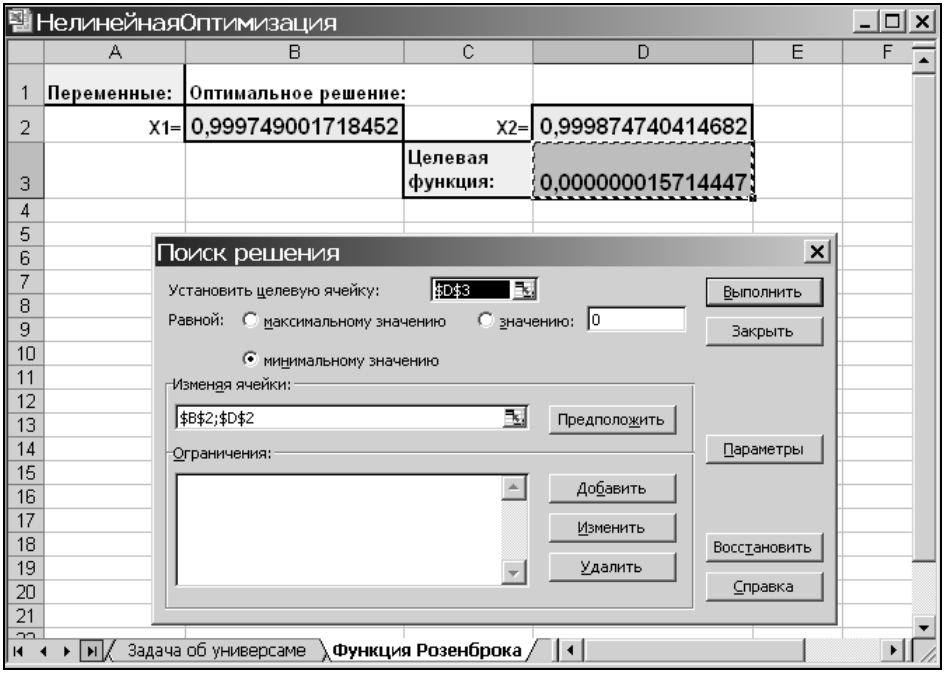


Рис. 3.19. Результат количественного решения задачи оптимизации с целевой функцией Розенброка

Для наглядного представления особенностей функции Розенброка построим диаграмму, содержащую трехмерный график данной функции. С этой целью на отдельном рабочем листе с именем График функции Розенброка зададим исходные данные со значениями этой функции в некотором интервале изменения переменных, например, x_1, $x_2 \in [-1{,}25,\ 1{,}25]$. Внешний вид рабочего листа MS Office Excel 2003 с исходными рядами данных для построения графика функции Розенброка имеет следующий вид (рис. 3.20).

Далее с использованием процедуры построения графика функции двух переменных, описанной в *разд. 2.4.2*, на этом же рабочем листе построим график поверхности функции Розенброка, который будет иметь следующий вид (рис. 3.21).

Microsoft Excel - НелинейнаяОптимизация

Файл Правка Вид Вставка Формат Сервис Данные Окно Справка

	A	B	C	D	E	F	G	H	I	J	K	L
1	Значения переменных x1 и x2->:	-1,25	-1	-0,75	-0,5	-0,25	0	0,25	0,5	0,75	1	1,25
2	-1,25	796,08	510,25	331,58	227,25	173,83	157,25	172,83	225,25	328,58	506,25	791,08
3	-1	661,70	404,00	247,20	158,50	114,45	101,00	113,45	156,50	244,20	400,00	656,70
4	-0,75	539,83	310,25	175,33	102,25	67,58	57,25	66,58	100,25	172,33	306,25	534,83
5	-0,5	430,45	229,00	115,95	58,50	33,20	26,00	32,20	56,50	112,95	225,00	425,45
6	-0,25	333,58	160,25	69,08	27,25	11,33	7,25	10,33	25,25	66,08	156,25	328,58
7	0	249,20	104,00	34,70	8,50	1,95	1,00	0,95	6,50	31,70	100,00	244,20
8	0,25	177,33	60,25	12,83	2,25	5,08	7,25	4,08	0,25	9,83	56,25	172,33
9	0,5	117,95	29,00	3,45	8,50	20,70	26,00	19,70	6,50	0,45	25,00	112,95
10	0,75	71,08	10,25	6,58	27,25	48,83	57,25	47,83	25,25	3,58	6,25	66,08
11	1	36,70	4,00	22,20	58,50	89,45	101,00	88,45	56,50	19,20	0,00	31,70
12	1,25	14,83	10,25	50,33	102,25	142,58	157,25	141,58	100,25	47,33	6,25	9,83

Функция Розенброка | График функции Розенброка

Готово

Рис. 3.20. Исходные ряды данных для построения графика функции Розенброка

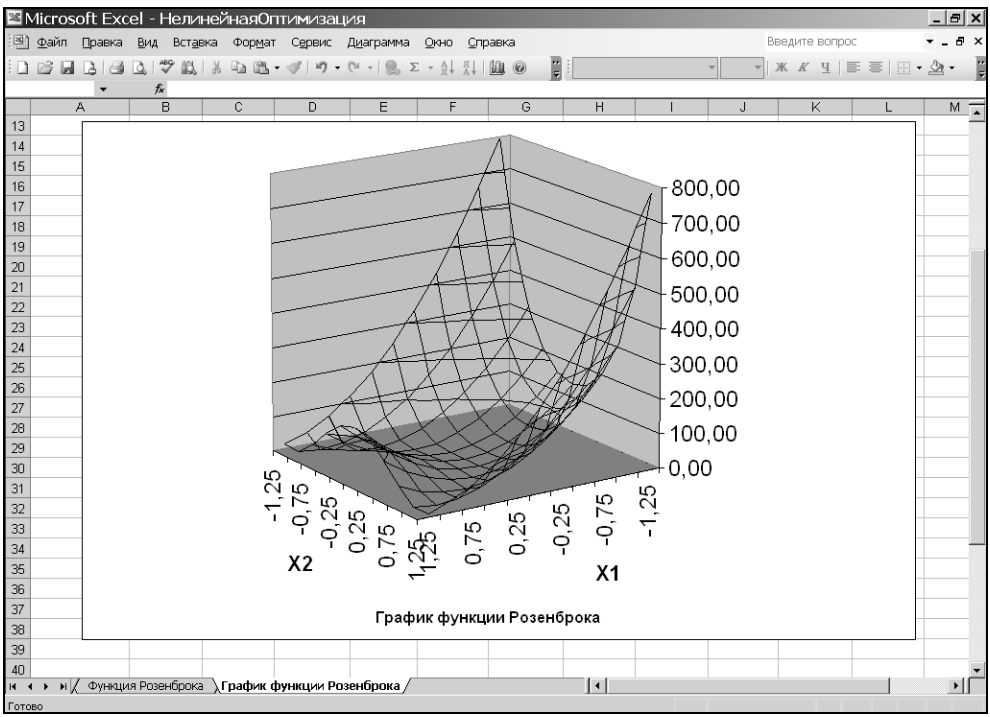


Рис. 3.21. График поверхности функции Розенброка в интервале изменения переменных $x_1, x_2 \in [-1{,}25,\ 1{,}25]$

Анализ графика показывает, что функция Розенброка при увеличении положительных значений x_1 приобретает два оврага, один из которых содержит искомый глобальный минимум функции $f^* = 0$ для значений: $x_1^* = 1$ и $x_2^* = 1$, а другой — всего лишь локальный минимум $f' = 0{,}25$ для значений: $x_1' = 0{,}25$ и $x_2' = 0{,}5$. Данное обстоятельство может привести к тому, что отдельные вычислительные алгоритмы вместо глобального экстремума могут определять локальный экстремум данной функции.

Однако сравнение найденных значений позволяет сделать вывод не только о достаточно высокой точности найденного с помощью программы MS Excel решения, но и о корректности реализованных в ней алгоритмов решения задач нелинейной оптимизации подобного класса.

3.5.2. Задача оптимизации с целевой функцией Пауэлла и ее решение с помощью программы MS Excel

Данная задача также является чисто математической и формулируется следующим образом. Необходимо найти минимум следующей целевой функции четырех переменных (без ограничений):

$$f(x_1, x_2, x_3, x_4) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + \\ + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 \to \min_{x_1, x_2, x_3, x_4 \in \Delta_\beta}, \qquad (3.5.2)$$

где $\Delta_\beta = \{x_1, x_2, x_3, x_4 \in \boldsymbol{R}^1\}$.

Целевая функция данной задачи является нелинейной, поэтому соответствующая задача оптимизации относится к классу задач нелинейного программирования или нелинейной оптимизации. Особенностью этой функции также является ее "овражный" характер, что затрудняет нахождение ее минимума некоторыми вычислительными алгоритмами.

Функция Пауэлла принимает неотрицательные значения, а точное решение соответствующей задачи оптимизации равно: $x_1^* = 0$, $x_2^* = 0$, $x_3^* = 0$, $x_4^* = 0$.

Для вычислительного решения данной задачи с помощью программы MS Excel создадим новый лист с именем: Функция Пауэлла. Сделаем необходимые надписи в ячейках **A1:A6**, **B1**. После этого введем в ячейку **B6** формулу: `=(B2+10*B3)^2+5*(B4-B5)^2+(B3-2*B4)^4+10*(B2-B5)^4`, которая представляет функцию Пауэлла (3.5.2). Далее вызывается мастер поиска решения, в котором задается целевая ячейка **B6** и изменяемые ячейки **B2:B5**. Остальные параметры поиска решения можно оставить без изменения.

Поиск решения данной задачи сразу приводит к нахождению точного решения. Внешний вид рабочего листа MS Office Excel 2003 с решением задачи оптимизации с целевой функцией Пауэлла имеет следующий вид (рис. 3.22).

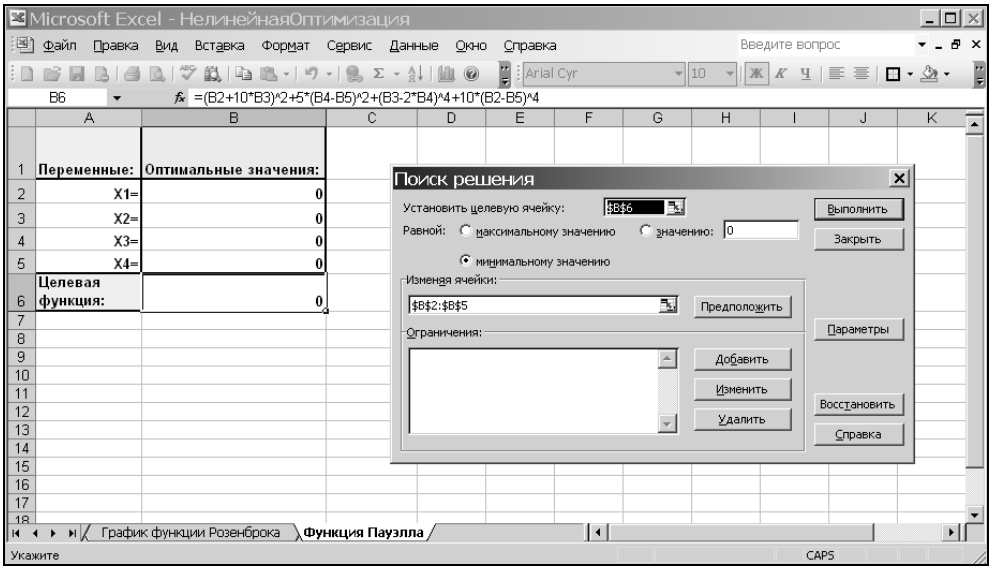


Рис. 3.22. Результат количественного решения задачи оптимизации с целевой функцией Пауэлла

Сравнение найденных значений также позволяет сделать вывод о достаточно высокой точности найденного с помощью программы MS Excel решения и о корректности реализованных в ней алгоритмов решения задач нелинейной оптимизации подобного класса.

3.5.3. Задача оптимизации с двумерной экспоненциальной целевой функцией и ее решение с помощью программы MS Excel

Данная задача также является чисто математической и формулируется следующим образом. Необходимо найти минимум целевой функции двух переменных без ограничений на их допустимые значения:

$$f(x_1, x_2) = \sum_{a=0.1(0.1)}^{1} [(e^{-ax_1} - e^{-ax_2}) - (e^{-a} - e^{-10a})]^2 \to \min_{x_1, x_2 \in \Delta_\beta}, \qquad (3.5.3)$$

где $\Delta_\beta = \{x_1, x_2 \in \boldsymbol{R}^1\}$.

Здесь суммирование осуществляется последовательно для возрастающих значений параметра a, начиная от значения 0,1 и заканчивая значением 1 с интервалом изменения данного параметра, равным 0,1.

Целевая функция данной задачи является нелинейной, поэтому соответствующая задача оптимизации относится к классу задач нелинейного программирования или нелинейной оптимизации. Особенностью этой функции является ее почти плоский характер вблизи минимума, что затрудняет нахождение оптимального значения вычислительными алгоритмами. При этом двумерная экспоненциальная функция принимает неотрицательные значения, а точное решение соответствующей задачи оптимизации равно: $x_1^* = 1$, $x_2^* = 10$.

Для вычислительного решения данной задачи с помощью программы MS Excel создадим новый лист с именем Экспоненциальная функция. Сделаем необходимые надписи в ячейках **A1:A4**, **A15**, **B1**. В ячейки **A5:A14** методом автозаполнения введем значения параметра суммирования a. После этого в ячейку **B5** введем формулу:

```
=((EXP(-A5*$B$2)-EXP(-A5*$B$3))-(EXP(-A5)-EXP(-10*A5)))^2,
```

которая представляет отдельный член суммы (3.5.3). Далее методом автозаполнения скопируем эту формулу в ячейки диапазона **B6:B14**. Здесь встроенная функция программы MS Excel с именем EXP( ) использована для получения значения основания натуральных логарифмов e.

Наконец, в целевую ячейку **B15** введем сумму ячеек **B5:B14**, что в итоге и является двумерной экспоненциальной функцией (3.5.3). Далее вызывается мастер поиска решения, в котором задается целевая ячейка **B15** и изменяемые ячейки **B2:B3**. Остальные параметры поиска решения можно оставить без изменения.

Поиск решения данной задачи приводит к нахождению решения, которое практически равно точному решению данной задачи. Внешний вид рабочего листа MS Office Excel 2003 с решением задачи оптимизации с двумерной экспоненциальной целевой функцией имеет следующий вид (рис. 3.23).

Построение графика данной функции непосредственным вводом формулы (3.5.3) в ячейки рядов исходных данных в настоящей главе не рассматривается. Именно в подобных случаях целесообразно использовать возможности программы MS Excel, основанные на предварительном задании функции пользователя либо с помощью разработки функции пользователя на языке программирования VBA, либо с помощью других средств на основе создания внешней библиотеки динамической компоновки. Эти способы являются предметом рассмотрения в *главах 11* и *12*, где описываются соответствующие способы подготовки данных для построения графика двумерной экспоненциальной функции.

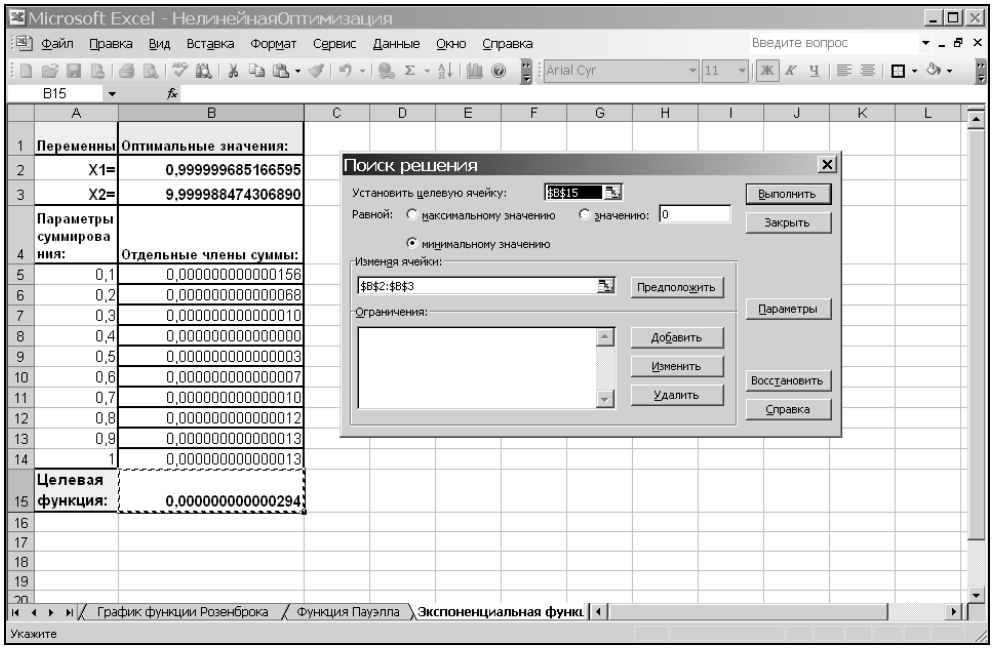


Рис. 3.23. Результат количественного решения задачи оптимизации с двумерной экспоненциальной целевой функцией

Сравнение найденных значений также подтверждает ранее сделанный вывод о достаточно высокой точности найденного с помощью программы MS Excel решения и о корректности реализованных в ней алгоритмов решения задач нелинейной оптимизации подобного класса.

3.6. Упражнения

В качестве упражнений для самостоятельного решения предлагаются задачи на нахождение экстремума, которые в различное время сыграли заметную роль в развитии математики. Поскольку большинство предлагаемых в качестве упражнений задач оптимизации сформулированы в общем виде, для их количественного решения с помощью программы MS Excel целесообразно для определенности задать некоторые количественные значения тех или иных параметров. Те из читателей, кто сочтет для себя возможным найти аналитическое решение данных задач, могут это сделать и для общего случая.

3.6.1. Задача Тартальи

Разделить число 8 на два числа так, чтобы произведение произведения этих чисел на разность этих чисел было максимально.

3.6.2. Задача Ферма

Найти прямоугольный треугольник наибольшей площади, если сумма длин его катетов равна заданному числу. Данной задачей в 1638 г. Ферма иллюстрировал свой метод нахождения минимумов, основанный на ставшей уже классической теореме Ферма.

3.6.3. Задача Кеплера

Среди всех цилиндров, вписанных в шар единичного радиуса, найти цилиндр с максимальным объемом. Данная задача была поставлена и решена Кеплером геометрическим способом в его фундаментальном труде "Стереометрия винных бочек" в 1615 г.

3.6.4. Обобщенная задача Кеплера

Необходимо вписать в единичный шар пространства R^n цилиндр наибольшего объема.

3.6.5. Задача Евклида

На стороне BC треугольника ABC найти точку E так, чтобы параллелограмм $ADEF$, у которого точки D и F лежат соответственно на сторонах AB и AC, имел наибольшую площадь. Данная задача является единственной задачей на экстремум в "Началах" Евклида.

3.6.6. Обобщенная задача Евклида

На некоторой фиксированной грани тетраэдра берется точка, через которую проводятся плоскости, параллельные трем оставшимся граням. Выбрать точку таким образом, чтобы объем полученного параллелепипеда был максимальным.

3.6.7. Задача Зенодора

Среди всех n-угольников, имеющих заданный периметр, найти n-угольник наибольшей площади.

3.6.8. Задача Архимеда

Среди сегментов шаров, имеющих заданную площадь боковой поверхности, найти сегмент наибольшего объема.

3.6.9. Задача Герона

На заданной прямой найти точку C, чтобы сумма расстояний от C до некоторых фиксированных точек A и B была минимальной.

3.6.10. Задача Аполлония для эллипса

Найти минимальное расстояние от некоторой точки на плоскости до заданного эллипса.

3.6.11. Задача Аполлония для параболы

Найти минимальное расстояние от некоторой точки на плоскости до заданной параболы.

3.6.12. Задача Аполлония для гиперболы

Найти минимальное расстояние от некоторой точки на плоскости до заданной гиперболы

3.6.13. Задача о вписанном прямоугольнике

Необходимо в круг заданного радиуса вписать прямоугольник максимальной площади.

3.6.14. Задача о вписанном треугольнике

Необходимо в круг заданного радиуса вписать треугольник максимальной площади.

3.6.15. Задача о вписанном конусе

Необходимо в трехмерный шар заданного радиуса вписать конус наибольшего объема.

3.6.16. Задача о вписанном тетраэдре

Необходимо в трехмерный шар заданного радиуса вписать тетраэдр наибольшего объема.

3.6.17. Задача о треугольнике

Среди треугольников данного периметра найти треугольник наибольшей площади.

3.6.18. Задача об угле и точке

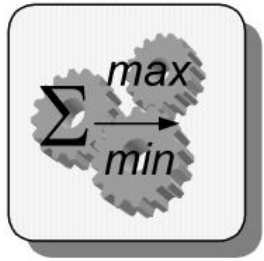
На плоскости заданы некоторый угол и точка внутри него. Через эту точку провести отрезок, имеющий концы на сторонах угла, так, чтобы полученный треугольник имел наименьшую площадь.

3.6.19. Задача о трех точках

На плоскости или в трехмерном пространстве заданы три различных точки: x_1, x_2 и x_3. Найти такую точку x_0, чтобы сумма квадратов расстояний от x_0 до точек x_1, x_2, x_3 была минимальной.

3.6.20. Обобщенная задача о точках

На плоскости или в трехмерном пространстве задано n точек: x_1, x_2, …, x_n и n положительных чисел: a_1, a_2, …, a_n. Найти такую точку x_0, чтобы взвешенная сумма с весами a_i квадратов расстояний от x_0 до x_1, x_2, …, x_n была наименьшей.



Глава 4

Задачи линейного программирования

К классу задач линейного программирования относятся такие задачи однокритериальной оптимизации, в которых переменные являются непрерывными и неотрицательными, целевая функция является линейной функцией своих аргументов, а ограничения могут быть представлены в форме линейных неравенств и равенств. При этом на значения переменных не накладываются никакие дополнительные ограничения, такие как, например, ограничения целочисленности или булевости.

На формирование линейного программирования в качестве самостоятельного направления научно-прикладных исследований наибольшее влияние оказали американские ученые Дж. Данциг, Т. Купманс, Дж. фон Нейман и ученые из России Л. В. Канторович, А. С. Немировский, Л. Г. Хачиян и Д. Б. Юдин. Хотя необходимость создания специальных методов решения неклассических оптимизационных задач осознавалась и раньше, в частности, экономистами и военными специалистами во время второй мировой войны, только в послевоенное время были разработаны теоретические основы линейного программирования и предложены специальные методы решения соответствующих практических задач.

Примечание

Собственно термин "линейное программирование" впервые появился в 1951 г. в работах Дж. Данцига и лауреата Нобелевской премии по экономике Т. Купманса. Однако общепризнано, что первые исследования по линейному программированию, связанные с формулировкой основной задачи, рассмотрением приложений, нахождением критерия оптимальности, экономической интерпретацией, были выполнены в конце 30-х годов XX в. в СССР лауреатом Нобелевской премии по экономике Л. В. Канторовичем. По этому поводу Дж. Данциг в одной из своих монографий отмечает, что "Канторовича Л. В. следует признать первым, кто обнаружил, что широкий класс важнейших производственных задач поддается четкой математической формулировке, которая, по его убеждению, дает возможность подходить к задачам с количественной стороны и решать их численными методами..."

Достижения в области линейного программирования содействовали прогрессу в разработке методов и алгоритмов решения задач оптимизации других классов, в том числе задач нелинейного, целочисленного и комбинаторного программирования. В настоящее время задачи линейного программирования широко используются в процессе подготовки специалистов самой различной квалификации. Чтобы понять особенности задач данного класса и методы их решения, необходимо рассмотреть математическую постановку задачи линейного программирования в общем случае.

4.1. Общая характеристика задачи линейного программирования

При рассмотрении задач линейного программирования, следует помнить, что, с одной стороны, они являются специальным случаем общей задачи оптимизации. Тем самым для задач линейного программирования оказываются справедливыми соответствующие результаты относительно общих свойств и способов их решения, разработанные в теории решения экстремальных задач. С другой стороны, специальная форма задания целевой функции и ограничений в форме линейных функций приводит к появлению у данного класса задач целого ряда специальных свойств, которые послужили основой разработки специализированных методов и алгоритмов их решения. Для детального анализа этих специальных свойств следует рассмотреть общую математическую постановку задачи линейного программирования.

4.1.1. Математическая постановка задачи линейного программирования

В общем случае математическая постановка задачи линейного программирования может быть сформулирована в следующем виде:

$$f(x_1, x_2, ..., x_n) \to \max_{x \in \Delta_\beta}, \text{ где} \tag{4.1.1}$$

$$\Delta_\beta = \{\Delta \mid g_k(x_1, x_2, ..., x_n) \le (=) b_k; \quad x_1, x_2, ..., x_n \ge 0\}, \quad (k \in \{1, 2, \ ..., m\}). \tag{4.1.2}$$

При этом следует принимать во внимание следующие принципиальные предположения о характере целевой функции и левых частей ограничений:

1. Целевая функция $f(x_1, x_2, \ldots, x_n)$ предполагается линейной относительно всех своих переменных, т. е. может быть представлена в форме: $f(x_1, x_2, \ldots, x_n) = = c_1x_1 + c_2x_2 + \ldots + c_nx_n$.

2. Левые части ограничений $g_k(x_1, x_2, \ldots, x_n)$ $(\forall k \in \{1, 2, \ldots, m\})$ также являются линейными функциями относительно своих переменных $x_1, x_2, \ldots, x_n$, т. е. могут быть представлены в форме: $g_k(x_1, x_2, \ldots, x_n) = a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n$.
3. Переменные $x_1, x_2, \ldots, x_n$ могут принимать свои значения только из множества неотрицательных действительных чисел $\boldsymbol{R}^1_+$, т. е. $x_i \in \boldsymbol{R}^1_+ (\forall i \in \{1, 2, \ldots, n\})$.

С учетом сделанных предположений *общая задача линейного программирования* может быть сформулирована следующим образом.

Необходимо найти максимум линейной целевой функции n переменных $x_1, x_2, \ldots, x_n \in \boldsymbol{R}^1_+$ следующего вида:

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max_{x \in \Delta_\beta}, \tag{4.1.3}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств и неравенств:

$$\begin{cases} a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i & (\forall i \in \{1, 2, \ldots, q\}) \quad (4.1.4) \\ a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n \le b_k & (\forall k \in \{q+1, \ldots, m\}) \quad (4.1.5) \end{cases}$$

В математической постановке общей задачи линейного программирования через c_i, a_{ki}, b_k $(\forall i \in \{1, 2, \ldots, n\}, \forall k \in \{1, 2, \ldots, m\})$ обозначены постоянные величины, которые могут принимать произвольные, не обязательно целочисленные значения, определяемые спецификой конкретной задачи линейного программирования.

Примечание

Задача минимизации линейной целевой функции при ограничениях типа равенств и неравенств может быть сведена к общей задаче линейного программирования вида (4.1.3) и (4.1.4), поскольку $f(x) \to min$ эквивалентно $-f(x) \to max$. Именно поэтому, не ограничивая общности, в последующем, говоря о задачах линейного программирования, будем иметь в виду задачи максимизации линейных функций типа (4.1.3) и (4.1.4).

В случае отсутствия ограничений типа равенств (4.1.4), т. е. при $q = 0$, задача линейного программирования называется *стандартной задачей линейного программирования*, которая, с учетом сделанных предположений, может быть записана в следующем виде:

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max_{x \in \Delta_\beta}, \tag{4.1.6}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \le b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \le b_2 \\ \qquad\qquad ... \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \le b_m \end{cases} \qquad (4.1.7)$$

$$\text{и} \quad x_1, x_2, ..., x_n \ge 0$$

С другой стороны, при отсутствии ограничений типа неравенств (4.1.5), т. е. при $q = m$, задача линейного программирования называется *канонической* или *основной задачей линейного программирования*, которая, с учетом сделанных предположений, может быть записана в следующем виде:

$$c_1x_1 + c_2x_2 + ... + c_nx_n \to \max_{x \in \Delta_\beta}, \qquad (4.1.8)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ \qquad\qquad ... \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n = b_m \end{cases} \qquad (4.1.9)$$

$$\text{и} \quad x_1, x_2, ..., x_n \ge 0$$

Примечание

Следует заметить, что стандартная задача линейного программирования (4.1.6) и (4.1.7) посредством введения дополнительных переменных может быть сведена к канонической задаче линейного программирования (4.1.8) и (4.1.9) и наоборот. Эти два типа задач отличаются только особенностью своей формулировки, которая, в свою очередь, определяется соображениями удобства формального анализа методов их решения.

При рассмотрении общих особенностей задачи линейного программирования удобной оказывается стандартная форма математической постановки задачи линейного программирования (4.1.6) и (4.1.7). Анализ множества допустимых альтернатив Δ_β стандартной задачи линейного программирова-

ния (4.1.6) и (4.1.7) позволяет прийти к выводу о справедливости только одной из трех возможных ситуаций:

1. Система ограничений (4.1.7) противоречива или несовместна, т. е. не существует ни одного набора значений $x_1, x_2, \ldots, x_n$, которые удовлетворяют ограничениям (4.1.7). В этом случае задача линейного программирования не имеет решения.
2. Система ограничений (4.1.7) не является противоречивой, однако соответствующая ей область пространства $\boldsymbol{R}^n$ является неограниченной. В этом случае задача линейного программирования не имеет решения, в случае, если линейная функция (4.1.6) не ограничена в неограниченной области, соответствующей множеству допустимых альтернатив Δ_β.
3. Система ограничений (4.1.7) не является противоречивой, и при этом соответствующая ей область пространства $\boldsymbol{R}^n$ является ограниченной. В этом случае задача линейного программирования имеет решение.

В последней ситуации задача линейного программирования может иметь либо единственное решение, либо континуум решений. Континуум решений имеет место в том случае, когда линейная целевая функция оказывается параллельной функции левой части одного из ограничений.

Примечание

Поскольку прикладной характер настоящей книги не позволяет детально рассмотреть теоретические аспекты анализа задач линейного программирования, заинтересованные в соответствующих вопросах читатели могут обратиться к специальной литературе, список которой приведен в конце книги.

4.1.2. Основные методы решения задач линейного программирования

Как уже отмечалось в *главе 1*, в общем случае существует два подхода к решению задач оптимизации. С одной стороны, для решения задачи линейного программирования теоретически может быть использован некоторый аналитический способ решения, применимый для решения задач оптимизации в общей постановке.

Однако использование для решения задач линейного программирования аналитического способа решения, основанного, например, на методе множителей Лагранжа, с учетом дифференцируемости целевой функции и ограничений, связано с преодолением серьезных трудностей вычислительного характера. В этом случае, даже для небольшого числа переменных и ограничений,

решение задачи линейного программирования сводится к нахождению частных производных функции Лагранжа с последующим решением системы уравнений с большим числом переменных. Именно по этой причине аналитический способ решения задач линейного программирования не используется на практике.

С другой стороны, для решения задачи линейного программирования могут быть использованы алгоритмические методы решения, применимые для решения задач оптимизации в общей постановке. Эти методы основываются на идее градиентного поиска для задач оптимизации с ограничениями.

Однако наибольшее применение для задач линейного программирования получили алгоритмические способы решения соответствующих задач, которые учитывают специфические особенности целевой функции и множества допустимых решений. Из алгоритмических способов следует отметить получивший широкую известность *симплекс-метод* для решения задач линейного программирования и *метод потенциалов* для решения транспортной задачи.

Примечание

Кроме аналитического и алгоритмического подходов к решению задач оптимизации, простейшие задачи линейного программирования могут быть решены *графическим способом*, который основан на изображении графиков целевой функции и ограничений на плоскости или в трехмерном пространстве с последующим визуальным нахождением решения. В настоящей главе графический способ решения используется для иллюстрации формы множества допустимых альтернатив и особенностей решения задач линейного программирования.

Для решения задач линейного программирования в программе MS Excel реализованы приближенные методы их решения с достаточно высокой степенью точности. Оценить точность получаемых решений можно посредством сравнения аналитических и алгоритмических решений отдельных практических задач. Именно с этой целью в данной главе приводится описание различных способов решения задачи о производстве красок. Однако рассмотрение задач линейного программирования начнем с уже сформулированной в *главе 1* задачи об оптимальной диете.

4.2. Задача об оптимальной диете

Содержательная постановка задачи об оптимальной диете приводится в *разд. 1.2.3*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи оптимизации.

4.2.1. Математическая постановка задачи об оптимальной диете

Для математической постановки данной задачи необходимо определить переменные соответствующей задачи оптимизации, задать целевую функцию и специфицировать ограничения, позволяющие представить исходную задачу как стандартную задачу линейного программирования. С этой целью дополним содержательную постановку задачи соответствующими параметрами. Тогда в общем случае задача об оптимальной диете может быть сформулирована следующим образом.

Имеется n видов продуктов питания, в которых содержится m типов питательных веществ (белки, жиры, углеводы). В одной весовой единице продукта i-го типа ($i \in \{1, 2, \ldots, n\}$) содержится a_{ij} единиц питательного вещества j-го вида ($j \in \{1, 2, \ldots, m\}$). Известна минимальная суточная потребность b_j ($j \in \{1, 2, \ldots, m\}$) человека в каждом из видов питательных веществ. Задана калорийность c_i одной весовой единицы i-го продукта ($i \in \{1, 2, \ldots, n\}$). Требуется определить оптимальный состав рациона продуктов, такой, чтобы каждое питательное вещество содержалось в нем в необходимом количестве, обеспечивающем суточную потребность человека, и при этом суммарная калорийность рациона была минимальной.

Ведем в рассмотрение следующие переменные: x_i — весовое количество продукта питания i-го типа в суточном рационе. Тогда в общем случае математическая постановка задачи об оптимальной диете может быть сформулирована следующим образом.

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \min_{x \in \Delta_\beta}, \tag{4.2.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \geq b_2 \\ \qquad\qquad \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \geq b_m \\ \text{и} \quad x_1, x_2, \ldots, x_n \geq 0. \end{cases} \tag{4.2.2}$$

Хотя математически задача об оптимальной диете формулируется как задача о минимизации целевой функции, это не имеет принципиального значения для ее последующего решения. В этой связи следует помнить, что с учетом сделанного ранее замечания всегда можно перейти от задачи максимизации

целевой функции к эквивалентной ей задаче минимизации целевой функции и наоборот.

4.2.2. Решение задачи об оптимальной диете с помощью программы MS Excel

Для решения задачи об оптимальной диете с помощью программы MS Excel необходимо задать конкретные значения параметрам исходной задачи. Для определенности предположим, что в качестве исходных типов продуктов рассматриваются: хлеб, мясо, сыр, бананы, огурцы, помидоры, виноград ($n = 7$), а в качестве питательных веществ рассматриваются белки, жиры, углеводы ($m = 3$). Калорийность одной весовой единицы каждого из продуктов следующая: $c_1 = 2060$, $c_2 = 2430$, $c_3 = 3600$, $c_4 = 890$, $c_5 = 140$, $c_6 = 230$, $c_7 = 650$. Содержание питательных веществ в каждом из продуктов может быть задано в форме следующей таблицы (см. табл. 4.1).

Таблица 4.1. *Содержание питательных веществ в продуктах питания*

Продукты / Питательные вещества	Хлеб ржаной	Мясо баранина	Сыр "Российский"	Банан	Огурцы	Помидоры	Виноград
Белки	61	220	230	15	8	11	6
Жиры	12	172	290	1	1	2	2
Углеводы	420	0	0	212	26	38	155

Минимальная суточная потребность в питательных веществах следующая: в белках $b_1 = 100$, в жирах $b_2 = 70$, в углеводах $b_3 = 400$.

Примечание

Не уменьшая общности решаемой задачи, можно считать, что калорийность продуктов измеряется в *ккал/кг*, суточная потребность в питательных веществах — в *граммах*, а содержание питательных веществ в продуктах — в *грамм/кг*. В этом случае оказывается возможным выполнить дополнительную проверку условий сформулированной задачи на основе рассмотрения физической размерности целевой функции и ограничений.

Для решения данной задачи с помощью программы MS Excel создадим новую книгу с именем Линейное программирование и изменим имя ее первого

рабочего листа на Задача о диете. Для решения поставленной задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:I1**, **A2:A7**, **B4, I4**, **J4**. Следует отметить, что конкретное содержание этих надписей не оказывает никакого влияния на решение рассматриваемой задачи линейного программирования.
2. В ячейки **B3:H3** введем значения коэффициентов целевой функции: $c_1 = 2060$, $c_2 = 2430$, $c_3 = 3600$, $c_4 = 890$, $c_5 = 140$, $c_6 = 230$, $c_7 = 650$.
3. В ячейку **I2** введем формулу: `=СУММПРОИЗВ(B2:H2;B3:H3)`, которая представляет целевую функцию (4.2.1).
4. В ячейки **B5:H7** введем значения коэффициентов ограничений, взятых из табл. 4.1.
5. В ячейки **J5:J7** введем значения правых частей ограничений, соответствующих минимальной суточной потребности в питательных веществах: в белках $b_1 = 100$, жирах $b_2 = 70$ и углеводах $b_3 = 400$.
6. В ячейку **I5** введем формулу: `=СУММПРОИЗВ($B$2:$H$2;B5:H5)`, которая представляет левую часть первого ограничения (4.2.2).
7. Скопируем формулу, введенную в ячейку **I5**, в ячейки **I6** и **I7**.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи об оптимальном рационе питания имеет следующий вид (рис. 4.1).

ЛинейноеПрограммирование.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ:**	
2	**Значения:**								**=СУММПРОИЗВ(B2:H2;B3:H3)**	
3	**Калорийность:**	2060	2430	3600	890	140	230	650		
4		**Коэффициенты ограничений:**							Значения огр-ний:	Сут. Потребности:
5	по белкам:	61	220	230	15	8	11	6	=СУММПРОИЗВ($B$2:$H$2;B5:H5)	100
6	по жирам:	12	172	290	1	1	2	2	=СУММПРОИЗВ($B$2:$H$2;B6:H6)	70
7	по углеводам:	420	0	0	212	26	38	155	=СУММПРОИЗВ($B$2:$H$2;B7:H7)	400
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										

Задача о диете

Рис. 4.1. Исходные данные для решения задачи об оптимальной диете

Следует напомнить, что для отображения формул в ячейках рабочего листа необходимо выполнить операцию главного меню: **Сервис** | **Параметры**

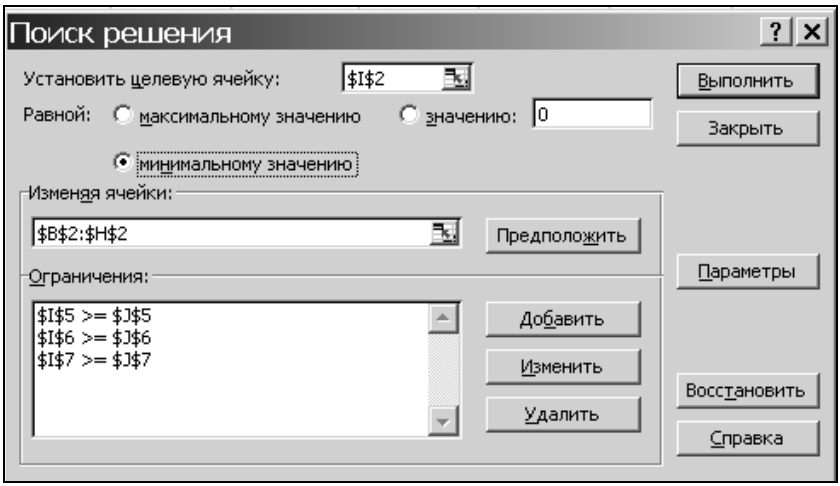
и в открывшемся диалоговом окне на вкладке **Вид** отметить флажком строку выбора **Формулы** в группе **Параметры окна**.

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**...

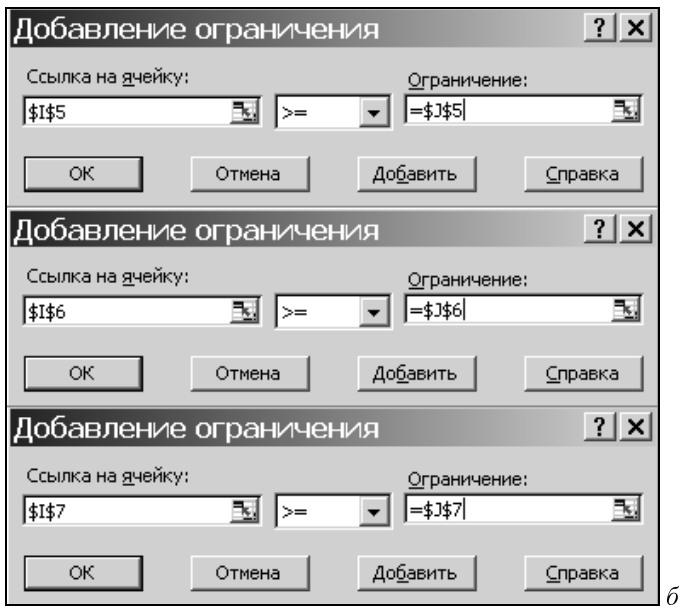
После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$I$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$H$2`.
4. Добавить 3 ограничения, представляющие минимальные суточные потребности в питательных веществах. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить** (рис. 4.2, *а*);
   - в появившемся дополнительном окне выбрать ячейку `$I$5`, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";
   - в качестве значения правой части ограничения выбрать ячейку **$J$5**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**;
   - аналогичным образом задать оставшиеся два ограничения (рис. 4.2, *б*).
5. Добавить ограничение на допустимые значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$H$2**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";

- в качестве значения правой части ограничения в поле с именем **Ограничение:** ввести значение 0;
- для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить** (рис. 4.3, *а*).

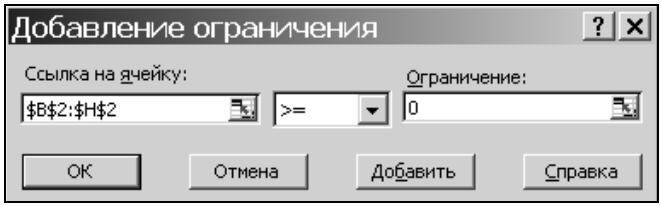


а

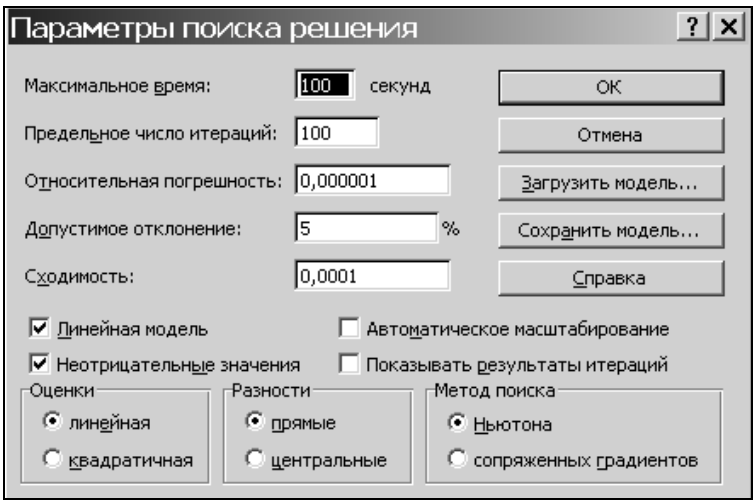


б

Рис. 4.2. Параметры мастера поиска решения и базовые ограничения для задачи об оптимальной диете



а



б

Рис. 4.3. Ограничения на значения переменных и параметры мастера поиска решения для задачи об оптимальной диете

6. В дополнительном окне параметров поиска решения следует выбрать отметки **Линейная модель** и **Неотрицательные значения** (рис. 4.3, *б*).

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет вид, представленный на рис. 4.4.

Результатом решения задачи об оптимальной диете являются найденные оптимальные значения переменных: $x_1 = 0$, $x_2 \cong 0{,}211$, $x_3 \cong 0{,}109$, $x_4 \cong 1{,}887$, $x_5 = 0$, $x_6 = 0$, $x_7 = 0$, которым соответствует значение целевой функции: $f_{\text{opt}} \cong 2587{,}140$. При выполнении расчетов для ячеек **B2:I2** был выбран числовой формат с 3 знаками после запятой.

Анализ найденного решения показывает, что для удовлетворения суточной потребности в питательных веществах (белки, жиры, углеводы) следует использовать 211 *г* мяса баранины, 109 *г* сыра и 1887 *г* бананов, совсем отказавшись от хлеба, огурцов, помидоров и винограда. При этом общая кало-

рийность найденной оптимальной диеты будет приближенно равна 2590 *ккал*, что вполне соответствует малоактивному образу жизни без серьезных физических нагрузок. Напомним, что согласно медицинским данным, энергетические затраты работников интеллектуального труда (юристы, бухгалтера, врачи, педагоги) лежат в пределах 3000 *ккал*.

ЛинейноеПрограммирование

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ:**	
2	**Значения:**	0,000	0,211	0,109	1,887	0,000	0,000	0,000	2587,140	
3	**Калорийность:**	2060	2430	3600	890	140	230	650		
4		**Коэффициенты ограничений:**							Значения огр-ний	Сут. Потребности:
5	по белкам:	61	220	230	15	8	11	6	100,000	100
6	по жирам:	12	172	290	1	1	2	2	70,000	70
7	по углеводам:	420	0	0	212	26	38	155	400,000	400
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										
20										
21										

Задача о диете

Рис. 4.4. Результат количественного решения задачи об оптимальной диете

Содержательный анализ результатов задачи об оптимальной диете явно выявляет недостатки рассмотренной математической модели. С одной стороны, для вкусной и питательной пищи не всегда приемлемым оказывается ограниченный рацион продуктов питания, который совершенно игнорирует индивидуальные предпочтения при выборе отдельных продуктов. С другой стороны, найденное оптимальное решение (мясо + сыр + бананы) для многих покажется однообразным и способным повергнуть в уныние даже пациентов больницы. Наконец, рассмотренная математическая модель задачи об оптимальной диете не учитывает суточную потребность в витаминах и микроэлементах, учет которых может существенно повлиять на выбор оптимального состава продуктов.

Примечание

С другими вариантами данной типовой задачи линейного программирования можно познакомиться в дополнительной литературе, список которой приведен в конце книги. В то же время заинтересованным читателям в качестве упражнения предлагается рассмотреть собственную постановку задачи об оптимальной диете, отражающую их индивидуальные предпочтения в выборе тех или иных продуктов. При этом условия задачи можно несколько изменить,

включив в качестве одного из ограничений общую калорийность диеты, а в качестве целевой функции рассмотреть общую массу наиболее предпочтительных продуктов.

Несмотря на выявленные недостатки рассмотренной математической модели задачи об оптимальной диете, найденное оптимальное решение в точности соответствует исходной постановке задачи. Это свидетельствует о достаточно высокой точности решения задач линейного программирования программой MS Excel.

4.3. Задача о производстве красок

Задача о производстве красок отличается простотой постановки и служит, главным образом, для иллюстрации графического способа решения задач линейного программирования. Данный способ позволяет не только наглядно представить форму множества допустимых альтернатив, но и собственно процесс нахождения оптимального решения. Говоря о простоте и наглядности графического способа, в то же время следует отметить его ограниченный характер, поскольку его использование возможно только для задач линейного программирования с двумя переменными, хотя и произвольным числом ограничений. Поскольку точность графического способа также оставляет желать лучшего, его применение служит в основном иллюстративным и методическим целям.

4.3.1. Общая постановка задачи производственного планирования

Задача о производстве красок является частным случаем типовой задачи линейного программирования, связанной с планированием производства или оптимальным выпуском продукции. Данная типовая задача была одной из первых, с которой началось развитие собственно моделей и методов линейного программирования. Цель рассмотрения модели планирования производства состоит в определении уровней или объемов производства отдельных видов производственной деятельности, при которых оптимизируется (максимизируется или минимизируется) общий результат производственной деятельности системы в целом без нарушения ограничений, накладываемых на использование ресурсов.

В общем случае для задачи о выпуске продукции вводятся в рассмотрение несколько видов выпускаемой предприятием продукции, на изготовление которой затрачивается несколько типов исходных материалов или ресурсов. При этом предполагается, что запасы этих материалов и ресурсов на пред-

приятии ограничены. Необходимо определить такой объем выпускаемой продукции, который максимизирует общую стоимость продукции, а использованные материалы и ресурсы не превосходят имеющихся на предприятии запасов.

Примечание

При формулировке общей задачи планирования производства следует учитывать тот факт, что рассматриваемая продукция не должна измеряться в штучных единицах. Если же это условие не выполняется, т. е. выпускаемая предприятием продукция измеряется в штучных единицах, то соответствующая задача, строго говоря, будет относиться к классу задач целочисленного линейного программирования, особенности постановки и методы решения которых рассматриваются в *главе 5*.

При построении математических моделей задач производственного планирования в форме задачи линейного программирования следует принимать во внимание допущения пропорциональности, аддитивности и неотрицательности.

Пропорциональность означает, что затраты ресурсов на некоторый вид производственной деятельности, а также вклад этого вида производственной деятельности в целевую функцию линейно пропорциональны его уровню. *Аддитивность* указывает на то, что общая величина ресурсов, потребляемых в системе всеми видами производственной деятельности, равна сумме затрат ресурсов на отдельные виды производственной деятельности. Это допущение находит отражение и при построении целевой функции.

Первые два допущения о пропорциональности и аддитивности обеспечивают строгую линейность соответствующих функций. В практических ситуациях действительный характер зависимостей редко бывает строго линейным. Однако допущение о линейности позволяет разработать специальные вычислительные методы, использование которых показало их практическую эффективность. Именно поэтому довольно часто задачи с нелинейными зависимостями формулируют в виде задач линейного программирования, что вполне оправдано лишь в тех случаях, когда линейная аппроксимация и соответствующий приближенный характер решений не оказывают заметного влияния на адекватность математической модели.

Допущение *неотрицательности* означает, что ни одному из видов производственной деятельности не может быть приписан отрицательный уровень. Для большинства производственных и экономических систем это допущение является естественным физическим требованием их функционирования. Его включение в структуру модели линейного программирования обусловлено удобствами логического анализа.

Применительно к производству красок соответствующая задача оптимизации может быть сформулирована следующим образом. Некоторое производственное предприятие выпускает несколько видов красок. Для производства этих видов красок используется несколько типов исходных красителей и других химических веществ. Известен расход этих красителей и веществ для получения каждого вида краски и запасы красителей и веществ на складе предприятия, а также стоимость каждого вида краски для оптовых покупателей. Требуется определить оптимальный объем выпуска красок каждого вида, обеспечивающий максимум общей стоимости готовой продукции.

4.3.2. Математическая постановка задачи о производстве красок

Поскольку графический способ решения задач линейного программирования на плоскости может быть применен только для задач с двумя переменными, рассмотрим математическую постановку задачи о производстве двух видов краски.

В этом случае производственное предприятие выпускает два вида краски ($n = 2$), одна из которых предназначена для внутренних работ, а другая — для наружных работ. Для производства этих видов краски используется три типа исходных красителей и химических веществ ($m = 3$) — индиго, железный купорос и свежегашеная известь. На производство одной весовой единицы краски i-го вида ($i \in \{1, 2\}$) требуется a_{ij} единиц исходного красителя j-го вида ($j \in \{1, 2, 3\}$). Расход этих красителей для получения каждого вида краски приводится в следующей таблице (табл. 4.2).

Таблица 4.2. *Расход красителей для производства краски*

Красители / Виды красок	Для внутренних работ	Для наружных работ
Индиго	0,1	0,2
Железный купорос	0,2	0,1
Свежегашеная известь	0,15	0,05

Запасы исходных красителей на складе предприятия ограничены следующими значениями: индиго $b_1 = 10$, железный купорос $b_2 = 7$, свежегашеная известь $b_3 = 5$. Стоимость каждого вида краски для оптовых покупателей равна: $c_1 = 250$ и $c_2 = 230$.

Примечание

Не уменьшая общности решаемой задачи, можно считать, что стоимость каждого вида краски измеряется в *руб/кг*, запасы исходных красителей — в *кг*, расход этих красителей для получения каждого вида краски — в *кг*. В этом случае оказывается возможным выполнить также дополнительную проверку условий сформулированной задачи на основе рассмотрения физической размерности целевой функции и ограничений.

Исходными переменными математической модели задачи о производстве красок являются: x_1 — объем выпуска краски для внутренних работ и x_2 — объем выпуска краски для наружных работ. Тогда математическая постановка рассматриваемой индивидуальной задачи о производстве красок может быть записана в следующем виде:

$$250x_1 + 230x_2 \to \max_{x \in \Delta_\beta}, \tag{4.3.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} 0{,}1x_1 + 0{,}2x_2 \le 10 \\ 0{,}2x_1 + 0{,}1x_2 \le 7 \\ 0{,}15x_1 + 0{,}05x_2 \le 5 \end{cases} \tag{4.3.2}$$

$$\text{и} \quad x_1, x_2 \ge 0.$$

4.3.3. Графическое решение задачи о производстве красок

Для графического решения задачи необходимо построить на плоскости множество допустимых альтернатив. Хотя для этой цели традиционно используются карандаш и лист бумаги, в настоящем случае воспользуемся программой MS Excel.

Для графического решения задачи о производстве красок с помощью программы MS Excel создадим новый рабочий лист с именем Задача о красках. Далее необходимо построить на плоскости диаграмму, содержащую изображение области допустимых альтернатив, соответствующей системе ограничений (4.3.2) и целевой функции (4.3.1). При этом следует заметить, что область допустимых альтернатив будет в точности ограничена прямыми линиями, которым соответствуют выполнение равенств для всех ограничений.

Для построения области допустимых альтернатив следует воспользоваться методикой построения графика функции одной переменной, рассмотренной

в *п. 2.4.1*. Применительно к задаче о красках (4.3.1) и (4.3.2) на одной диаграмме следует построить 4 графика линейных функций. При этом в качестве исходных функций для построения области допустимых альтернатив необходимо использовать 3 следующих функции: $y_1 = 10/0{,}2 - (0{,}1/0{,}2) \cdot x_1 = 50 - 0{,}5 \cdot x_1$; $y_2 = 7/0{,}1 - (0{,}2/0{,}1) \cdot x_1 = 70 - 2 \cdot x_1$ и $y_3 = 5/0{,}05 - (0{,}15/0{,}05) \cdot x_1 = 100 - 3 \cdot x_1$, где в качестве зависимой переменной y с соответствующим индексом используется вторая переменная данной задачи. В качестве четвертой функции следует использовать целевую линейную функцию (4.3.1).

Для подготовки исходных данных для последующего построения области допустимых альтернатив выполним следующие действия:

1. На рабочем листе в ячейку **A1** введем текст "Значения переменной x_1:", в ячейку **A2** введем текст "Значения функции f_1:", в ячейку **A3** введем текст "Значения функции f_2:", в ячейку **A4** введем текст "Значения функции f_3:" и, наконец, в ячейку **A5** введем текст "Значения целевой функции:".
2. Хотя независимая переменная x_1 принимает неотрицательные значения из непрерывного интервала действительных чисел $[0, \infty)$, для построения графиков линейных функций достаточно для этой независимой переменной задать два значения, например, 0 и 34. С этой целью в ячейку **B1** введем значение 0, а в ячейку **C1** — значение 34.
3. Далее в ячейку **B2** введем формулу: `=50-0,5*B1`, которая соответствует первому ограничению. Скопируем эту формулу в ячейку **C2**, которая должна иметь вид: `50-0,5*C1`.
4. Далее в ячейку **B3** введем формулу: `=70-2*B1`, которая соответствует второму ограничению. Скопируем эту формулу в ячейку **C3**, которая должна иметь вид: `70-2*C1`.
5. Далее в ячейку **B4** введем формулу: `=100-3*B1`, которая соответствует третьему ограничению. Скопируем эту формулу в ячейку **C4**, которая должна иметь вид: `100-3*C1`.
6. Наконец, в ячейку **B5** введем формулу: `=60-(25/23)*B1`, которая соответствует целевой функции. Скопируем эту формулу в ячейку **C5**, которая должна иметь вид: `=60-(25/23)*C1`. Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 4.5).

Для построения графиков необходимо воспользоваться мастером диаграмм, который может быть вызван с помощью кнопки стандартной панели инструментов или операции главного меню: **Вставка** | **Диаграмма**. Хотя процесс построения графика одной функции был рассмотрен ранее в *п. 2.4.1*, при

построении графиков нескольких функций на одной диаграмме следует учитывать дополнительные особенности, которые описываются далее.

ЛинейноеПрограммирование.xls

	A	B	C	D
1	Значения переменной x_1:	0	34	
2	Значения функции f_1:	=50-0,5*B1	=50-0,5*C1	
3	Значения функции $f2$:	=70-2*B1	=70-2*C1	
4	Значения функции f_3:	=100-3*B1	=100-3*C1	
5	Значения целевой функции:	=60-(25/23)*B1	=60-(25/23)*C1	
6				
7				
8				
9				
10				
11				

Задача о красках

Рис. 4.5. Исходные данные для построения графиков функций линейных ограничений и целевой функции

На первом шаге построения диаграммы необходимо выбрать тип диаграммы и ее вид. С этой целью следует выделить на вкладке **Стандартные** в левом списке тип диаграммы **График**, а в правом списке с графическими миниатюрами — первый вид графика, который отражает развитие процесса во времени или по категориям. После выбора типа и разновидности диаграммы следует нажать кнопку **Далее** и перейти ко второму шагу построения диаграммы с помощью мастера диаграмм.

На втором шаге построения диаграммы необходимо выбрать ячейки с данными, которые должны быть отображены на четырех графиках. Применительно к рассматриваемому примеру — это значения функций, которые содержатся в диапазоне ячеек **B2:C5**. Для указания этих значений следует установить переключатель **Ряды в:** в положение **строках**. После этого выполнить щелчок на кнопке, расположенной в правой части поля ввода с именем **Диапазон**, и выделить диапазон ячеек **B2:C5**. Далее на этом же шаге работы мастера диаграмм следует задать подписи по горизонтальной оси. С этой целью необходимо перейти на вкладку **Ряд** и выполнить щелчок на кнопке, расположенной в правой части поля ввода с именем **Подписи оси *X***. Для указания соответствующего источника данных следует на рабочем листе с помощью мыши или клавиатуры выделить диапазон значений функции **B1:C1**.

После редактирования свойств диаграммы на третьем и четвертом шагах работы мастера будет построена диаграмма, содержащая графики четырех линейных функций следующего вида (рис. 4.6).

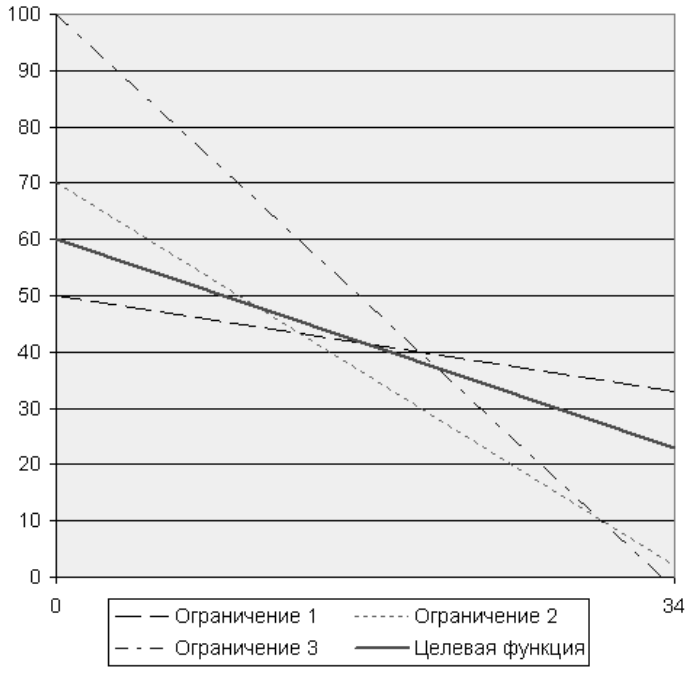


Рис. 4.6. Результат построения графиков функций ограничений и целевой функции для задачи о красках

Область допустимых альтернатив образуется пересечением пяти полуплоскостей, каждая из которых соответствует отдельному ограничению (4.3.2), включая и ограничения неотрицательности переменных задачи. Таким образом, область допустимых альтернатив задачи о красках представляет собой выпуклый многогранник на плоскости. Каждая точка из этой области является некоторым допустимым решением рассматриваемой задачи о красках. Чтобы это проверить, возьмем, например, точку с координатами $x_1 = 10$ и $x_2 = 10$. Подставляя эти значения в первое ограничение, получим: $1 + 2 \le 10$, во второе ограничение: $2 + 1 \le 7$, в третье ограничение: $1{,}5 + 0{,}5 \le 5$. Соответствующее значение целевой функции равно: $2500 + 2300 = 4800$. Тем самым точка с координатами $x_1 = 10$ и $x_2 = 10$ является допустимым решением исходной задачи со значением целевой функции 4800.

Если взять любую другую точку из построенной области допустимых альтернатив, которая лежит ближе к прямой, соответствующей целевой функции, то может быть получено допустимое решение исходной задачи с бóльшим значением целевой функции. Чтобы это проверить, возьмем, например, точку с координатами $x_1 = 10$ и $x_2 = 40$. Подставляя эти значения в первое ограничение, получим: $1 + 8 \le 10$, во второе ограничение: $2 + 4 \le 7$, в третье

ограничение: $1{,}5 + 2 \le 5$. Соответствующее значение целевой функции равно: $2500 + 9200 = 11\,700$. Тем самым точка с координатами $x_1 = 10$ и $x_2 = 40$ также является допустимым решением исходной задачи со значением целевой функции 11 700, которое превышает значение 4800, соответствующее точке с координатами $x_1 = 10$ и $x_2 = 10$.

Продолжая выбор точек из построенной области допустимых альтернатив, можно заметить, что оптимальное решение исходной задачи, соответствующее максимуму целевой функции (4.3.1), находится в точке пересечения первого и второго ограничений. Именно эта точка, являясь допустимой, располагается ближе остальных точек многогранника к построенной линии уровня целевой функции (рис. 4.7).

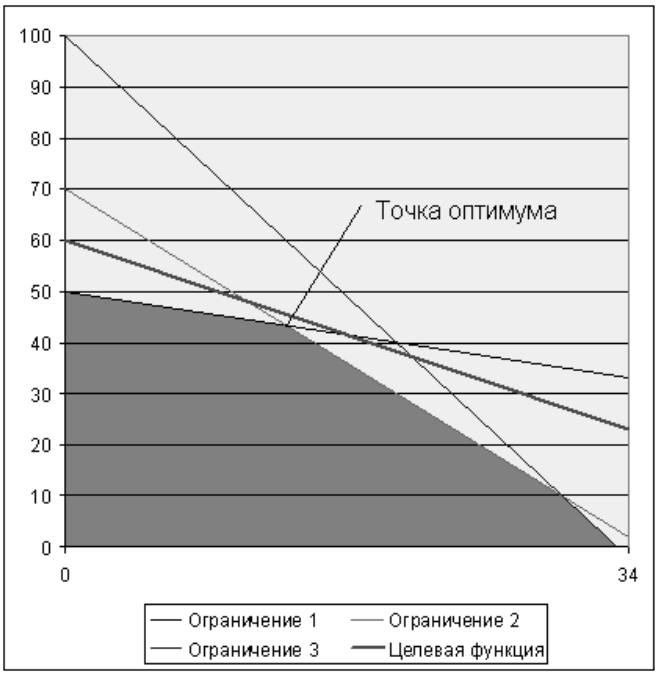


Рис. 4.7. Графическое решение задачи о красках

Для нахождения значений переменных оптимального решения необходимо найти координаты точки пересечения двух прямых, соответствующих линейным функциям первого и второго ограничений. Другими словами, необходимо решить систему линейных уравнений следующего вида:

$$\begin{cases} 0{,}1x_1 + 0{,}2x_2 = 10 \\ 0{,}2x_1 + 0{,}1x_2 = 7. \end{cases} \qquad (4.3.3)$$

Хотя эта система уравнений без труда решена с помощью карандаша и листа бумаги или калькулятора, в данном случае снова воспользуемся программой MS Excel. Это тем более оправдано, что практически вся необходимая для этого информация уже содержится на рабочем листе со значениями исходных линейных функций ограничений.

Для нахождения количественных значений переменных оптимального решения следует выполнить следующие действия:

1. На рабочем листе Задача о красках в ячейку **A6** введем текст "Значения переменных:", в ячейку **A7** введем текст "Значение разности функций:", а в ячейку **A8** введем текст "Значение целевой функции:".
2. Далее в ячейку **B7** введем формулу: `=50-0,5*B6-70+2*B6`, которая соответствует разности первых двух ограничений. Следует заметить, что значение первой переменной из ячейки **B6**, которое обеспечивает этой разности равенство 0, является искомым оптимальным значением для x_1.
3. В ячейку **B8** введем формулу: `=250*B6+230*C6`, которая необходима для расчета оптимального значения целевой функции.
4. В ячейку **C6** введем формулу: `=50-0,5*B6`, которая необходима для нахождения оптимального значения для x_1. Результат выполнения данной последовательности операций по подготовке данных для нахождения значений переменных оптимального решения будет иметь следующий вид (рис. 4.8).

ЛинейноеПрограммирование.xls

	A	B	C
1	Значения переменной x_1:	0	34
2	Значения функции f_1:	=50-0,5*B1	=50-0,5*C1
3	Значения функции f2:	=70-2*B1	=70-2*C1
4	Значения функции f_3:	=100-3*B1	=100-3*C1
5	Значения целевой функции:	=60-(25/23)*B1	=60-(25/23)*C1
6	Значения переменных:		=50-0,5*B6
7	Значение разности функций:	=50-0,5*B6-70+2*B6	
8	Значение целевой функции:	=250*B6+230*C6	
9			
10			
11			
12			

Задача о красках

Рис. 4.8. Дополнительные данные для нахождения оптимального решения задачи о красках

Для нахождения количественных значений переменных, обеспечивающих максимальное значение целевой функции исходной задачи о красках, воспользуемся встроенным инструментом программы MS Excel, который называется Подбор параметра. Этот инструмент может быть вызван с помощью

операции главного меню: **Сервис | Подбор параметра**. В результате выполнения этой операции появится диалоговое окно с тремя строками ввода.

В первую строку с именем **Установить в ячейке** следует ввести ячейку **B7**, в которой содержится формула разности первых двух ограничений. Во вторую строку с именем **Значение** следует ввести с клавиатуры значение 0, которое преобразует формулу ячейки **B7** в линейное уравнение одной переменной. Наконец, в третью строку с именем **Изменяя значение в ячейке** следует ввести ячейку **B6**, которая будет содержать искомое значение решения этого уравнения.

Внешний вид мастера подбора параметра с необходимыми значениями его свойств изображен на рис. 4.9.

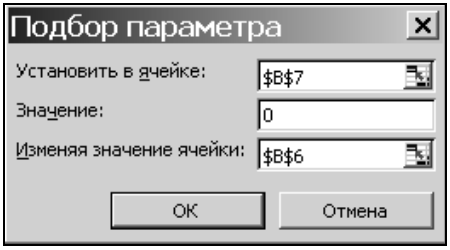


Рис. 4.9. Свойства мастера подбора параметра для задачи о красках

ЛинейноеПрограммирование

	A	B	C	D	E	F	G	H
1	Значения переменной x_1:	0	34					
2	Значения функции f_1:	50	33					
3	Значения функции f2:	70	2					
4	Значения функции f_3:	100	-2					
5	Значения целевой функции:	60	23,043					
6	Значения переменных:	13,333	43,333					
7	Значение разности функций:	0						
8	Значение целевой функции:	13300						
9								
10								
11								
12								
13								

Задача о диете | Задача о красках

Рис. 4.10. Результат количественного решения задачи о красках с помощью мастера подбора параметра

После задания необходимых значений свойств мастера подбора параметра, следует нажать кнопку **OK**. В результате этого будут найдены оптимальные значения переменных исходной задачи о красках, которые будут содержаться в ячейках **B6** и **C6**: $x_1 = 13{,}333$ и $x_2 = 43{,}333$. Соответствующее максимальное значение целевой функции для данной задачи содержится в ячейке **B8**: $f_{opt} = 13\ 300$ (рис. 4.10).

Таким образом, оптимальный объем выпуска красок каждого вида, обеспечивающий максимум общей стоимости готовой продукции, равен 13,333 *кг* для краски первого типа и 43,333 *кг* для краски второго типа. При этом будет обеспечено максимальное значение стоимости продукции: 13 300 *руб.* На этом графическое решение задачи о красках может быть закончено.

В качестве последнего штриха проверим найденное решение с помощью основного инструмента решения задач линейного программирования программы MS Excel — мастера поиска решения. Для этого воспользуемся рекомендациями, рассмотренными ранее в *разд. 4.2*. Таблицу с соответствующими исходными данными для удобства можно разместить на этом же рабочем листе. Результаты решения исходной задачи о красках с помощью инструмента поиска решения изображены в нижней области на рис. 4.11.

ЛинейноеПрограммирование

	A	B	C	D	E	F	G
1	Значения переменной x_1:	0	34				
2	Значения функции f_1:	50	33				
3	Значения функции f2:	70	2				
4	Значения функции f_3:	100	-2				
5	Значения целевой функции:	60	23,043				
6	Значения переменных:	13,333	43,333				
7	Значение разности функций:	0					
8	Значение целевой функции:	13300					
9							
10							
11	Переменные:	*x1*	*x2*	Значение ЦФ:			
12	Значения переменных:	13,333	43,333	13300			
13	Стоимость:	250	230				
14	Коэффициенты ограничений:			Значения огр-ний:	Запас на складе:		
15	по индиго:	0,1	0,2	10	10		
16	по железному купоросу:	0,2	0,1	7	7		
17	по свежегашеной извести:	0,15	0,05	4,167	5		
18							
19							

Задача о диете | Задача о красках

Рис. 4.11. Результаты количественного решения задачи о красках с помощью инструментов Подбор параметра и Поиск решения

Анализ полученных результатов решения задачи двумя различными методами показывает их полное совпадение. Дополнительно можно заметить, что запасы первых двух ингредиентов (индиго и железного купороса) используются полностью, а часть свежегашеной извести останется невостребованной. Все это может служить достаточно веским основанием для вывода о достоверности найденного оптимального решения задачи. Напомним, что сравнительный анализ результатов решения вычислительных задач в математике является одним из основных способов оценки качества не только соответствующих алгоритмов и расчетных программ, но зачастую единственным средством, позволяющим убедиться в достоверности или безошибочности искомого результата решения практических оптимизационных задач.

4.3.4. Решение задачи о производстве красок с помощью симплекс-метода

Чтобы оценить точность и правильность результатов решения задач линейного программирования, полученных с помощью программных средств, в общем случае можно воспользоваться каким-либо аналитическим или алгоритмическим методом нахождения решения задач оптимизации. Однако специально для решения задач линейного программирования еще в 1947 г. Дж. Данцигом был разработан *симплекс-метод*. С тех пор известность данного метода стала столь высока, что для многих само название метода превратилось в синоним линейного программирования. Опуская вопросы общей теории симплекс-метода, с которыми можно познакомиться по специальной литературе, рассмотрим его основные особенности и проиллюстрируем использование его на примере решения задачи о красках.

Общая идея симплекс-метода заключается в следующем. Если исходная задача линейного программирования имеет решение, то соответствующее множество допустимых альтернатив представляет собой некоторый многогранник или, более строго, *симплекс* в пространстве $\boldsymbol{R}_+^{\boldsymbol{n}}$, где n — общее количество переменных задачи. При этом, в случае единственности решения, оптимальное значение линейной целевой функции, которая соответствует некоторой гиперплоскости в $\boldsymbol{R}_+^{\boldsymbol{n}}$, достигается в одной из вершин этого многогранника (симплекса). Таким образом, процесс нахождения решения можно ограничить перебором только вершин этого многогранника (симплекса) и проверки соответствующих значений целевой функции на оптимальность. Для сокращения времени поиска решения целесообразно осуществлять переход к такой вершине симплекса, которая обеспечивает прирост оптимальности целевой функции по сравнению с предыдущей вершиной.

Очевидно, условием окончания поиска решения в этом случае является невозможность улучшить некоторое уже имеющееся решение задачи.

Для практической реализации общей идеи симплекс-метода необходимо ответить на следующие вопросы:

1. Как получить исходное допустимое решение задачи линейного программирования, которое будет соответствовать некоторой вершине симплекса?
2. Как установить связь между значениями переменных задачи и координатами вершин симплекса?
3. Каким образом осуществить переход от одной вершины симплекса к следующей, чтобы обеспечить прирост оптимальности целевой функции?
4. Каким образом осуществлять пересчет координат вершин симплекса при переходе от одной его вершины к следующей?

Для ответа на все эти вопросы общая или стандартная задача линейного программирования должна быть преобразована в каноническую задачу линейного программирования, которая, в свою очередь, записывается в форме так называемой симплекс-таблицы. В последующем все расчеты выполняются над элементами этой симплекс-таблицы, которая и дала название симплекс-методу.

В общем случае процедура симплекс-метода имеет итеративный характер и заключается в выполнении следующих действий:

1. Исходная стандартная задача линейного программирования (4.1.5) и (4.1.6) преобразуется в каноническую форму (4.1.7)—(4.1.9). С этой целью вводятся m дополнительных переменных x_{n+1}, x_{n+2}, …, x_{n+m}, которым соответствуют фиктивные нулевые коэффициенты целевой функции: $c_{n+1} = c_{n+2} = \ldots = c_{n+m} = 0$. Тогда соответствующая задача линейного программирования может быть записана в следующем виде:

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n + 0x_{n+1} + 0x_{n+2} + \ldots + 0x_{n+m} \to \max_{x \in \Delta_\beta}, \qquad (4.3.4)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + x_{n+1} = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + x_{n+2} = b_2 \\ \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n + x_{n+m} = b_m \end{cases} \qquad (4.3.5)$$

$$\text{и } x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m} \geq 0.$$

2. Задача в канонической форме (4.3.4) и (4.3.5) записывается в исходную симплекс-таблицу, которая в общем случае имеет следующий вид (см. табл. 4.3).

Таблица 4.3. *Общий вид исходной симплекс-таблицы*

	X_0	X_1	X_2	…	X_n	X_{n+1}	X_{n+2}	…	X_{n+m}
X_{n+1}	b_1	a_{11}	a_{12}	…	a_{1n}	**1**	**0**	…	**0**
X_{n+2}	b_2	a_{21}	a_{22}	…	a_{2n}	**0**	**1**	…	**0**
…				…		…			…
X_{n+m}	b_m	a_{m1}	a_{m2}	…	a_{mn}	**0**	**0**	…	**1**
$F(x)$	c_0	c_1	c_2	…	c_n	**0**	**0**	…	**0**

Рассмотрим особенности построения исходной симплекс-таблицы. Ее верхняя строка содержит обозначения переменных решаемой задачи линейного программирования. При этом в первом столбце X_0 записываются значения правых частей ограничений задачи (4.3.5). В последующем элементы этого столбца будут соответствовать значениям переменных промежуточных решений задачи линейного программирования. Первые m элементов столбца X_1 соответствуют значениям коэффициентов ограничений задачи для переменной x_1, столбца X_2 — значениям коэффициентов ограничений задачи для переменной x_2 и т. д.

Самая нижняя строка исходной симплекс-таблицы, кроме первых двух элементов, содержит значения коэффициентов критериальной функции (4.3.4): $c_1, c_2, \ldots, c_n, \ldots, c_{n+m}$. Эта строка получила свое особое название — *индексная строка*. При этом значение элемента c_0, который не входит в индексную строку, всегда равно значению целевой функции для рассматриваемого допустимого решения задачи.

Левый столбец симплекс-таблицы, за исключением самого нижнего элемента, содержит обозначения *базисных переменных*, т. е. таких, которые образуют *базис* некоторого допустимого решения задачи линейного программирования в канонической форме. Самый нижний элемент левого столбца просто обозначает имя целевой функции.

Примечание

Строгое понятие базиса решения задачи линейного программирования основывается на рассмотрении теории линейных пространств. В общем случае под

базисом понимается минимальное линейно независимое множество векторов, с помощью которых любой вектор линейного пространства может быть представлен как линейная комбинация этих базисных векторов. Более подробно познакомиться с соответствующим материалом можно по дополнительной литературе, список которой приведен в конце книги.

В исходной симплекс-таблице в качестве переменных базиса удобно рассматривать дополнительные переменные, значения которых равны правым частям ограничений: $x_{n+1} = b_1$, $x_{n+2} = b_2$, …, $x_{n+m} = b_m$. В этом случае первоначальные переменные принимают нулевые значения, т. е. $x_1 = x_2 = \ldots = x_n = 0$, что позволяет в общем случае ответить на первый поставленный ранее вопрос — задать некоторое первоначальное допустимое решение. Что касается ответа на второй вопрос, то сама структура симплекс-таблицы устанавливает необходимую связь. При этом для исходной симплекс-таблицы значение критериальной функции рассчитывается по формуле: $c_0 = c_{n+1}x_{n+1} + c_{n+2}x_{n+2} + \ldots + c_{n+m}x_{n+m} = 0$, где x_{n+1}, x_{n+2}, …, x_{n+m} — базисные переменные, а c_{n+1}, c_{n+2}, …, c_{n+m} — коэффициенты целевой функции (4.3.4).

Следует заметить, что первые два этапа симплекс-метода являются подготовительными. Все последующие действия имеют итеративный повторяющийся характер и выполняются в рамках построенной исходной симплекс-таблицы.

1. В построенной симплекс-таблице выполняется анализ знаков элементов ее индексной строки, исключая элемент c_0. При этом возможны 3 случая:

   - Все элементы индексной строки неотрицательны. В этом случае базисное решение является оптимальным, а значение элемента c_0 является *оптимальным* значением целевой функции. На этом вычисления симплекс-метода заканчиваются.

     Примечание

     Строгое доказательство оптимальности содержащегося в симплекс-таблице базисного решения задачи линейного программирования является основным результатом теоретического обоснования симплекс-метода. Заинтересованные читатели могут познакомиться с соответствующим материалом по дополнительной литературе, список которой приведен в конце книги.

   - Среди элементов индексной строки есть хотя бы один отрицательный, а над ним в симплекс-таблице нет ни одного положительного. В этом случае целевая функция не ограничена сверху на множестве допустимых альтернатив и, следовательно, оптимального решения не существует. На этом вычисления симплекс-метода также заканчиваются.

- Среди элементов индексной строки есть хотя бы один отрицательный, а над ним в симплекс-таблице есть хотя бы один положительный элемент. В этом случае текущее решение можно улучшить, для чего необходимо построить новую симплекс-таблицу, которой будет соответствовать новое допустимое решение с наименьшим значением целевой функции. В этом случае следует перейти к выполнению следующего этапа процедуры симплекс-метода.

2. Среди отрицательных элементов индексной строки, над каждым из которых в симплекс-таблице есть хотя бы один положительный элемент, выбираем наибольший по абсолютной величине. Столбец, в основании которого оказался выбранный элемент, получил название *ключевого столбца*. Ключевой столбец указывает на переменную, которая *вводится* в базис.

3. Далее находится *ключевое отношение* — наименьшее из отношений значений элементов столбца X_0 (кроме c_0) к соответствующим *положительным* значениям элементов ключевого столбца. В ключевом столбце выбираем и отмечаем *ключевой элемент* — знаменатель ключевого отношения. Если ключевых отношений, равных по величине, несколько, то можно выбрать произвольное из них. Строка симплекс-таблицы, которой соответствует ключевое отношение, указывает на переменную, *выводимую* из базиса.

Примечание

Вообще говоря, при выборе ключевого столбца не обязательно среди отрицательных элементов выбирать наибольший по абсолютной величине. Можно выбрать любой из них. Это связано с тем, что существует лишь вероятностные оценки минимального количества симплекс-таблиц, необходимых для решения задачи. Заметим также, что ключевой элемент всегда положительный.

4. Строится новая симплекс-таблица, в которой, прежде всего, записываются в левый столбец новые базисные переменные. При этом одна из них выводится из базиса, а одна — вводится в базис. Далее рассчитываются значения ключевой строки новой симплекс-таблицы, которая получается делением всех элементов соответствующей строки исходной симплекс-таблицы на ключевой элемент.

5. Остальные элементы новой симплекс-таблицы подсчитываются по правилу "двух перпендикуляров" — каждый элемент новой симплекс-таблицы, за исключением элементов ключевой строки, равен разности между соответствующим элементом предыдущей симплекс-таблицы и произведением элементов, на которые указывают основания двух перпендикуляров, опу-

щенных из данного элемента на ключевой столбец предыдущей симплекс-таблицы и ключевую строку в новой симплекс-таблице. После построения новой симплекс-таблицы следует перейти к выполнению действий этапа 3.

Для получения значений элементов новой симплекс-таблицы следует использовать следующие расчетные формулы:

$$x'_{ij} = x_{ij} - \frac{x_{kj}}{x_{kl}} x_{il}, \tag{4.3.6}$$

$$(\forall i \in \{1, 2, ..., l-1, l+1, ..., m+1\}, \forall j \in \{0, 1, 2, ..., n+m\})$$

$$x'_{kj} = \frac{x_{kj}}{x_{kl}}, (\forall j \in \{0, 1, 2, ..., n+m\}), \tag{4.3.7}$$

где x'_{ij} — элементы новой симплекс-таблицы, x_{ij} — элементы предыдущей симплекс-таблицы, l — номер ключевого столбца, k — номер ключевой строки. При этом формула (4.3.6) используется для расчета значений всех элементов новой симплекс-таблицы за исключением элементов ключевой строки; для расчета последних используется формула (4.3.7). Следует заметить, что для принятых обозначений значение x'_{kl} равно ключевому отношению.

Рассмотренная процедура симплекс-метода может быть изображена графически в форме диаграммы деятельности языка UML (рис. 4.12).

Проиллюстрируем использование рассмотренного алгоритма симплекс-метода для решения индивидуальной задачи (4.3.1) и (4.3.2). Для этого предварительно выполним простейшие алгебраические преобразования, после чего преобразуем ее к канонической форме, которая примет следующий вид:

$$250x_1 + 230x_2 + 0x_3 + 0x_4 + 0x_5 \to \max_{x \in \Delta_\beta}, \tag{4.3.8}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств:

$$\begin{cases} x_1 + 2x_2 + x_3 = 100 \\ 2x_1 + x_2 + x_4 = 70 \\ 3x_1 + x_2 + x_5 = 100 \end{cases} \tag{4.3.9}$$

$$\text{и} \quad x_1, x_2, x_3, x_4, x_5 \geq 0.$$

Для задачи линейного программирования (4.3.8) и (4.3.9) в канонической форме исходная симплекс-таблица будет иметь следующий вид (табл. 4.4).

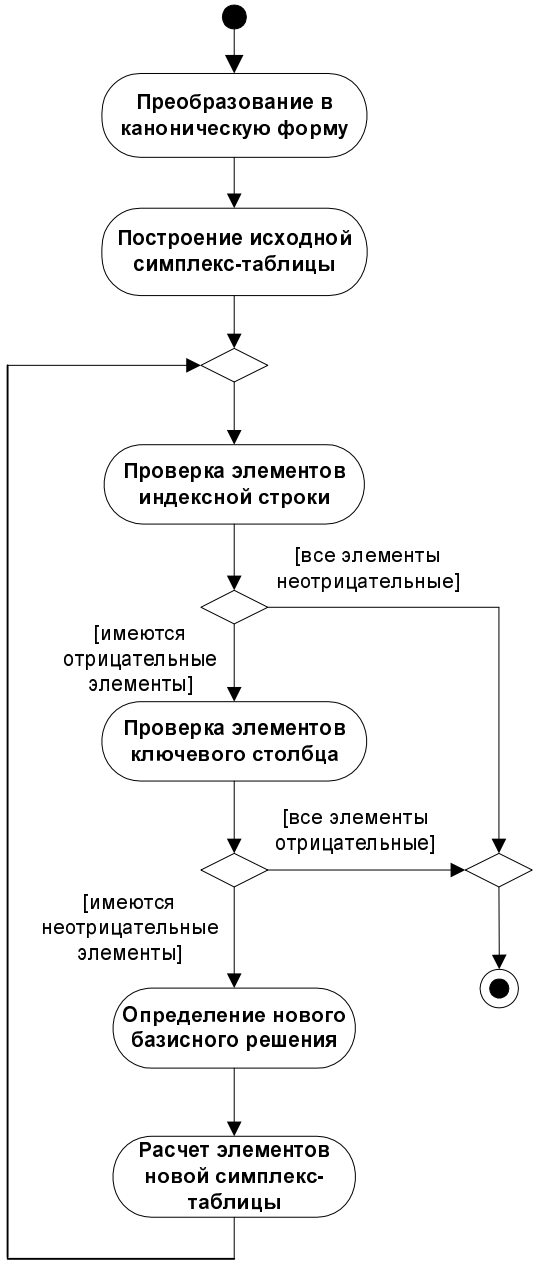


Рис. 4.12. Диаграмма деятельности алгоритма симплекс-метода

Таблица 4.4. *Общий вид исходной симплекс-таблицы для задачи о красках*

	X_0	X_1	X_2	X_3	X_4	X_5
X_3	100	**1**	2	1	0	0
X_4	70	**2**	1	0	1	0
X_5	**100**	**3**	**1**	**0**	**0**	**1**
$F(x)$	0	**–250**	–230	0	0	0

После выполнения подготовительных этапов 1 и 2 симплекс-метода можно приступить к проверке условия получения оптимального решения (этап 3). Для этого проверяются элементы индексной строки. Поскольку среди них имеются 2 отрицательных: –250 и –230, далее проверяем условие наличия выше этих элементов в симплекс-таблице неотрицательных элементов (этап 4). Поскольку соответствующее условие выполняется для обоих отрицательных элементов индексной строки, то выбираем из них максимальный по абсолютной величине: –250. Этот выбор указывает на ключевой столбец X_1 и на переменную x_1, которую следует ввести в базис на первой итерации.

Для нахождения ключевого отношения (этап 5) рассмотрим три отношения: 100/1, 70/2, 100/3 и выберем минимальное из них: 100/3. Этот выбор указывает на ключевую строку X_5 и на переменную x_5, которую следует вывести из базиса на первой итерации. Для наглядности элементы ключевого столбца и ключевой строки выделены полужирным шрифтом.

Далее строится новая симплекс-таблица (этап 6), в которой в левый столбец записываются новые базисные переменные: x_3, x_4, x_1. После этого рассчитываются значения всех элементов новой симплекс-таблицы, которая после выполнения расчетов по формулам (4.3.6) и (4.3.7) примет следующий вид (табл. 4.5).

Таблица 4.5. *Общий вид симплекс-таблицы после первой итерации*

	X_0	X_1	X_2	X_3	X_4	X_5
X_3	200/3	0	**5/3**	1	0	–1/3
X_4	**10/3**	**0**	**1/3**	**0**	**1**	**–2/3**
X_1	100/3	1	**1/3**	0	0	1/3
$F(x)$	25000/3	0	**–440/3**	0	0	250/3

Для получения значений элементов этой симплекс-таблицы следует выполнить следующие действия: $x_{30} = 100 - (100/3) \cdot 1 = 200/3$, $x_{40} = 70 - (100/3) \cdot 2 = 10/3$, $x_{10} = 100/3$, $x_{32} = 2 - (1/3) \cdot 1 = 5/3$, $x_{12} = 1/3$, $x_{42} = 1 - (1/3) \cdot 2 = 1/3$, $x_{35} = 0 - (1/3) \cdot 1 = -1/3$, $x_{45} = 0 - (1/3) \cdot 2 = -2/3$, $x_{15} = 1/3$, $x_{f2} = -230 + (250/3) \cdot 1 = -440/3$, $x_{f5} = 0 + (250/3) \cdot 1 = 250/3$. Остальные элементы новой симплекс-таблицы имеют нулевые или единичные значения.

Примечание

Заметим, что для удобства расчетов элементы симплекс-таблицы нумеруются двумя индексами, первый из которых равен номеру базисной переменной, а второй — номеру переменных верхней строки. При этом элементы индексной строки в качестве первого индекса имеют имя целевой функции *f*.

Полученной симплекс-таблице соответствует базисное решение: $x_1 = 100/3$, $x_3 = 200/3$, $x_4 = 10/3$, которому соответствует значение целевой функции: $F(x) = 25\,000/3$.

После расчета элементов симплекс-таблицы (табл.4.5) выполняется проверка условия оптимальности найденного решения, для чего следует проверить элементы индексной строки. Среди них имеется отрицательный: –440/3. Далее проверяем условие наличия неотрицательных элементов выше этого элемента в симплекс-таблице (этап 4). Поскольку для этого отрицательного элемента индексной строки соответствующее условие выполняется, то найденное решение можно улучшить, для чего следует перейти ко второй итерации алгоритма. При этом в качестве ключевого столбца на второй итерации выбирается X_2 и, следовательно, переменную x_2 следует ввести в базис.

Для нахождения ключевого отношения (этап 5) рассмотрим три отношения: 40, 10, 100 и выберем минимальное из них: 10. Этот выбор указывает на ключевую строку X_4 и на переменную x_4, которую следует вывести из базиса на второй итерации. Для наглядности элементы ключевого столбца и ключевой строки также выделены полужирным шрифтом.

Далее на второй итерации строится новая симплекс-таблица (этап 6), в которой в левый столбец записываются новые базисные переменные: x_3, x_2, x_1. После этого рассчитываются значения всех элементов новой симплекс-таблицы, которая после выполнения расчетов по формулам (4.3.6) и (4.3.7) примет следующий вид (табл. 4.6).

Таблица 4.6. *Общий вид симплекс-таблицы после второй итерации*

	X_0	X_1	X_2	X_3	X_4	X_5
X_3	50	0	0	1	–5	3
X_2	10	0	1	0	3	–2
X_1	30	1	0	0	0	1/3
$F(x)$	9800	0	0	0	440	–210

Для получения значений элементов этой симплекс-таблицы следует выполнить следующие действия: $x_{30} = 200/3 - 10 \cdot (5/3) = 50$, $x_{20} = 10$, $x_{10} = 100/3 - 10 \cdot (1/3) = 30$, $x_{34} = 0 - 3 \cdot (5/3) = -5$, $x_{24} = 3$, $x_{14} = 0 - 3 \cdot (1/3) = -1$, $x_{35} = -1/3 + 2 \cdot (5/3) = 3$, $x_{25} = -2$, $x_{15} = 1/3 + 2 \cdot (1/3) = 1$, $x_{f4} = 0 + 440 = 440$, $x_{f5} = 250/3 - (440/3) \cdot 2 = -210$. Остальные элементы новой симплекс-таблицы имеют нулевые или единичные значения.

Полученной симплекс-таблице соответствует базисное решение: $x_1 = 30$, $x_2 = 10$, $x_3 = 50$, которому соответствует значение целевой функции: $F(x) = 9800$.

После расчета элементов симплекс-таблицы (табл.4.6) выполняется проверка условия оптимальности найденного решения, для чего следует проверить элементы индексной строки. Среди них имеется отрицательный элемент: –210. Далее проверяем условие наличия выше этого элемента в симплекс-таблице неотрицательных элементов (этап 4). Поскольку для этого отрицательного элемента индексной строки соответствующее условие выполняется, то найденное решение можно улучшить, для чего следует перейти к третьей итерации алгоритма. При этом в качестве ключевого столбца выбирается X_5 и, следовательно, переменную x_5 следует ввести в базис.

Для нахождения ключевого отношения (этап 5) рассмотрим два отношения: 50/3, 30 и выберем минимальное из них: 50/3. Этот выбор указывает на ключевую строку X_3 и на переменную x_3, которую следует вывести из базиса на третьей итерации. Для наглядности элементы ключевого столбца и ключевой строки также выделены полужирным шрифтом.

Далее на третьей итерации строится новая симплекс-таблица (этап 6), в которой в левый столбец записываются новые базисные переменные: x_5, x_2, x_1. После этого рассчитываются значения всех элементов новой симплекс-таблицы, которая после выполнения расчетов по формулам (4.3.6) и (4.3.7) примет следующий вид (табл. 4.7).

Таблица 4.7. *Общий вид симплекс-таблицы после третьей итерации*

	X_0	X_1	X_2	X_3	X_4	X_5
X_5	50/3	0	0	1/3	–5/3	1
X_2	130/3	0	1	2/3	–1/3	0
X_1	40/3	1	0	–1/3	2/3	0
$F(x)$	13 300	0	0	70	90	0

Для получения значений элементов этой симплекс-таблицы следует выполнить следующие действия: $x_{50} = 50/3$, $x_{20} = 10 + (50/3) \cdot 2 = 130/3$, $x_{10} = 30 - (50/3) \cdot 1 = 40/3$, $x_{53} = 1/3$, $x_{23} = 0 - (1/3) \cdot (-2) = 2/3$, $x_{24} = 3 - (-5/3) \cdot (-2) = -1/3$, $x_{13} = 0 - (1/3) \cdot 1 = -1/3$, $x_{14} = -1 + (-5/3) \cdot (-1) = 2/3$, $x_{54} = -5/3$, $x_{f3} = 0 + 210/3 = 70$, $x_{f4} = 440 - (210/3) \cdot 5 = 90$. Остальные элементы новой симплекс-таблицы имеют нулевые или единичные значения.

Полученной симплекс-таблице соответствует базисное решение: $x_1 = 40/3$, $x_2 = 130/3$, $x_3 = 50/3$, которому соответствует значение целевой функции: $F(x) = 13\ 300$.

После расчета элементов симплекс-таблицы (табл. 4.7) снова выполняется проверка условия оптимальности найденного решения, для чего следует проверить элементы индексной строки. Поскольку среди них нет отрицательных, то найденное на третьей итерации решение является оптимальным.

Сравнение найденных оптимальных решений задачи о красках с помощью графического способа (см. рис. 4.10) и симплекс-метода (табл. 4.7) показывают их полное совпадение, что свидетельствует о достоверности соответствующих результатов.

4.4. Двойственная задача линейного программирования

Значение разработанного Дж. Данцигом симплекс-метода стало еще более важным после того, как основоположником теории игр Дж. фон Нейманом была развита концепция двойственности в задачах линейного программирования. Умение формулировать и решать двойственные задачи расширяет сферу применения методов линейного программирования, а взаимосвязь оптимальных решений прямой и двойственных задач позволяет, решив одну из них, с минимальными усилиями определить решение другой.

4.4.1. Математическая формулировка двойственной задачи линейного программирования

В общем случае *двойственной* по отношению к стандартной задаче линейного программирования (4.1.6) и (4.1.7) называется такая задача линейного программирования, которая может быть записана в следующем виде:

$$b_1 y_1 + b_2 y_2 + ... + b_m y_m \to \min_{y \in \Delta'_\beta}, \tag{4.4.1}$$

где множество допустимых альтернатив Δ'_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} a_{11} y_1 + a_{21} y_2 + ... + a_{m1} y_m \geq c_1 \\ a_{12} y_1 + a_{22} y_2 + ... + a_{m2} y_m \geq c_2 \\ \qquad\qquad ... \\ a_{1n} y_1 + a_{2n} x_2 + ... + a_{mn} y_m \geq c_n \end{cases} \tag{4.4.2}$$

$$\text{и} \quad y_1, y_2, ..., y_m \geq 0.$$

Как нетрудно заметить, количество переменных двойственной задачи линейного программирования равно количеству ограничений стандартной задачи, а количество ограничений двойственной задачи равно количеству переменных стандартной задачи линейного программирования. При этом, если исходная задача формулируется как задача максимизации целевой функции, то двойственная — как задача минимизации и наоборот. Аналогично изменяются и знаки ограничений двойственной задачи по отношению к исходной задаче линейного программирования.

Коэффициенты целевой функции двойственной задачи линейного программирования равны правым частям ограничений стандартной задачи, а правые части ограничений двойственной задачи равны коэффициентам целевой функции стандартной задачи линейного программирования.

Примечание

Если по определению задача линейного программирования (4.4.1) и (4.4.2) является двойственной по отношению к стандартной задаче (4.1.6) и (4.1.7), то имеет место и обратное. А именно, задача (4.1.6) и (4.1.7) является двойственной к (4.4.1) и (4.4.2), поскольку с учетом сделанного ранее замечания стандартная задача линейного программирования может быть сформулирована в форме задачи на минимум, а знаки ограничений легко могут быть изменены. Тем самым установлено свойство симметричности отношения двойственности задач линейного программирования.

Существует важная взаимосвязь между двойственной и стандартной задачами линейного программирования. А именно, если одна из задач (4.1.6) и (4.1.7) или (4.4.1) и (4.4.2) имеет оптимальное решение, то и двойственная ей задача линейного программирования имеет оптимальное решение, при этом оптимальные значения соответствующих целевых функций двойственных задач имеют равные значения, т. е. $f'_{op} = f_{opt}$, где $f'(y)$ — целевая функция в выражении (4.4.1), а $f(x)$ — целевая функция в выражении (4.1.6). Если же для одной из задач (4.1.6) и (4.1.7) или (4.4.1) и (4.4.2) целевая функция не ограничена на допустимом множестве альтернатив, то соответствующая ей двойственная задача линейного программирования не имеет решения, т. е. имеет пустое множество допустимых альтернатив. Наконец, если одна из задач (4.1.6) и (4.1.7) или (4.4.1) и (4.4.2) имеет пустое множество допустимых альтернатив, то соответствующая ей двойственная задача линейного программирования либо имеет неограниченную целевую функцию, либо пустое множество допустимых альтернатив.

В общем случае совместное рассмотрение пары двойственных задач линейного программирования позволяет не только выполнить качественный анализ их решения, но и практически использовать найденное решение одной из них для более простого решения другой задачи. Хотя данное свойство оказывается полезным, главным образом, при выполнении ручных расчетов, далее рассмотрим процесс решения двойственных задач линейного программирования с помощью программы MS Excel применительно к задаче о красках.

4.4.2. Математическая постановка двойственной задачи о красках

Напомним, что исходная математическая постановка задачи о красках сформулирована в форме (4.3.1) и (4.3.2). Двойственная к ней задача линейного программирования после выполнения простейших преобразований с целью получения целочисленных коэффициентов целевой функции и ограничений может быть записана в следующем виде:

$$100y_1 + 70y_2 + 100y_3 \to \min_{y \in \Delta'_\beta}, \tag{4.4.3}$$

где множество допустимых альтернатив Δ'_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} y_1 + 2y_2 + 3y_3 \geq 250 \\ 2y_1 + y_2 + y_3 \geq 230 \end{cases} \quad \text{и} \quad y_1, y_2, y_3 \geq 0. \tag{4.4.4}$$

Как нетрудно заметить, задача линейного программирования (4.4.3) и (4.4.4) по своей форме аналогична задаче об оптимальной диете, которая была рассмотрена в *разд. 4.2*.

4.4.3. Решение двойственной задачи о красках с помощью программы MS Excel

Для решения двойственной задачи о производстве красок с помощью программы MS Excel создадим новый рабочий лист с именем Двойственная задача. Далее необходимо выполнить следующие действия:

1. Внесем необходимые надписи в ячейки **A1:E1**, **A2:A6**, **E4**, **F4**. Следует отметить, что конкретное содержание этих надписей не оказывает никакого влияния на решение рассматриваемой двойственной задачи линейного программирования.
2. В ячейки **B3:D3** введем значения коэффициентов целевой функции (4.4.3): $b_1 = 10$, $b_2 = 7$, $b_3 = 5$.
3. В ячейку **E2** введем формулу: `=СУММПРОИЗВ(B2:D2; B3:D3)`, которая представляет целевую функцию (4.4.3).
4. В ячейки **B5:D6** введем значения коэффициентов ограничений (4.4.4).
5. В ячейки **F5:F6** введем значения правых частей ограничений (4.4.4).
6. В ячейку **E5** введем формулу: `=СУММПРОИЗВ($B$2:$D$2; B5:D5)`, которая представляет левую часть первого ограничения (4.4.4).
7. Скопируем формулу, введенную в ячейку **E5**, в ячейку **E6**.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи об оптимальном рационе питания имеет следующий вид (рис. 4.13).

ЛинейноеПрограммирование.xls

	A	B	C	D	E	F	G
1	**Переменные:**	*y1*	*y2*	*y3*	**Значение ЦФ:**		
2	**Значения:**				=СУММПРОИЗВ(B2:D2;B3:D3)		
3	**Коэфф-ты ЦФ:**	100	70	100			
4	**Коэффициенты ограничений:**				Значения огр-ний:	Правая часть:	
5	первое неравенство:	1	2	3	=СУММПРОИЗВ($B$2:$D$2;B5:D5)	250	
6	второе неравенство:	2	1	1	=СУММПРОИЗВ($B$2:$D$2;B6:D6)	230	
7							
8							
9							
10							
11							

Двойственная задача

Рис. 4.13. Исходные данные для решения двойственной задачи о производстве красок

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$E$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$D$2`.
4. Добавить три ограничения, соответствующие (4.4.5), и одно ограничение на допустимые значения переменных.
5. В дополнительном окне параметров поиска решения следует выбрать отметки **Линейная модель** и **Неотрицательные значения** (рис. 4.14).

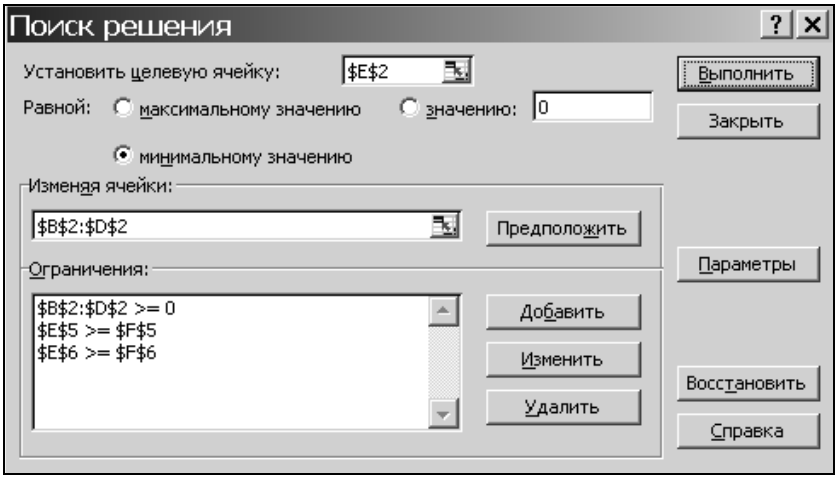


Рис. 4.14. Ограничения на значения переменных и параметры мастера поиска решения для двойственной задачи о красках

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 4.15).

Результатом решения двойственной задачи о производстве красок являются найденные оптимальные значения двойственных переменных: $y_1 = 70$, $y_2 = 90$,

$y_3 = 0$, которым соответствует значение целевой функции: $f'_{opt} = 13\,300$. При выполнении расчетов для ячеек был выбран числовой формат с тремя знаками после запятой.

ЛинейноеПрограммирование

	A	B	C	D	E	F	G
1	**Переменные:**	***y1***	***y2***	***y3***	**Значение ЦФ:**		
2	**Значения:**	**70,000**	**90,000**	**0,000**	**13300,000**		
3	**Коэфф-ты ЦФ:**	100	70	100			
4	**Коэффициенты ограничений:**				Значения огр-ний:	Правая часть:	
5	первое нер-во:	1	2	3	250,000	250	
6	второе нер-во	2	1	1	230,000	230	
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							

Задача о красках | **Двойственная задача**

Рис. 4.15. Результат количественного решения двойственной задачи о красках

Одним из наиболее важных свойств двойственных задач является наличие в симплекс-таблице, соответствующей оптимальному решению одной из них, значений оптимального решения двойственной задачи. Применительно к задаче о красках, значения оптимального решения двойственной задачи о красках (4.4.3) и (4.4.4) можно сразу получить из последней симплекс-таблицы (табл. 4.7). А именно, оптимальное решение двойственной задачи содержится в индексной строке в столбцах, соответствующих дополнительным переменным x_3, x_4, x_5. Поскольку переменная x_3 вводится в первое ограничение прямой задачи, которому, в свою очередь, соответствует первая переменная y_1 двойственной задачи, то из табл. 4.7 непосредственно следует оптимальное значение для $y_1 = x_{f3} = 70$. Аналогично могут быть получены и значения $y_2 = x_{f4} = 90$ и $y_3 = x_{f5} = 0$. При этом нет никакой необходимости в непосредственном решении двойственной задачи.

Экономическая интерпретация полученных решений прямой и двойственной задач заключается в следующем. Решение прямой задачи о красках (4.3.1) и (4.3.2) дает оптимальный план производства красок первого и второго вида. Решение двойственной задачи о красках (4.4.3) и (4.4.4) — оптимальную *систему оценок* типов сырья, используемого для производства этих красок. При этом выполняются следующие условия. Если некоторый тип сырья используется полностью, то соответствующая этому типу двойственная переменная в оптимальном решении двойственной задачи будет иметь положи-

тельное значение. Если же некоторый тип сырья используется не полностью, то соответствующая этому типу двойственная переменная в оптимальном решении двойственной задачи будет равна нулю.

Применительно к паре решенных двойственных задач (4.3.1) и (4.3.2) и (4.4.3) и (4.4.4) первые два неравенства прямой задачи (4.3.2) превращаются в равенства, откуда следует, что запасы индиго и железного купороса используются полностью. Об этом свидетельствуют и оптимальные значения двойственных переменных: $y_1 = 70$, $y_2 = 90$. Напротив, запасы свежегашеной извести используются не полностью, что согласуется со значением третьей двойственной переменной найденного оптимального решения $y_3 = 0$.

Для целей экономического анализа модели задачи линейного программирования удобно предположить, что двойственные переменные могут выступать в роли оценок типов сырья, используемого в производстве красок. Более того, величина данной двойственной оценки показывает, на сколько возрастет максимальное значение целевой функции прямой задачи при увеличении количества сырья соответствующего типа на 1 *кг*.

Таким образом, двойственные оценки могут быть использованы для определения степени дефицитности типов сырья для производства продукции. В связи с этим анализ оптимальных решений прямой и двойственных задач линейного программирования становится необходимым этапом экономического анализа эффективного планирования производства продукции.

4.5. Транспортная задача линейного программирования

Содержательная постановка транспортной задачи линейного программирования приводится в *разд. 1.2.4*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи оптимизации.

4.5.1. Математическая постановка транспортной задачи

В общем случае математическая постановка транспортной задачи может быть сформулирована в следующем виде. Имеется m пунктов производства или хранения и n пунктов потребления некоторого однородного продукта (например, нефть, уголь, песок, цемент и т. п.). Для каждого из пунктов задан a_i — объем производства или запаса продукта в i-том пункте ($i \in \{1, 2, \ldots, m\}$), а для каждого пункта потребления задана b_j — потребность в продукте

в j-том пункте потребления ($j \in \{1, 2, \ldots, n\}$). Известна c_{ij} — стоимость перевозки или транспортировки одной единицы продукта из i-го пункта производства в j-й пункт потребления. Требуется определить оптимальный план перевозок продукта, так чтобы потребность во всех пунктах потребления были удовлетворены, а суммарные затраты на транспортировку всей продукции были минимальными.

Ведем в рассмотрение следующие переменные: x_{ij} — количество транспортируемого продукта или объем перевозок из i-го пункта производства в j-й пункт потребления. Тогда в общем случае математическая постановка транспортной задачи может быть сформулирована следующим образом.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min_{x\in\Delta_\beta}, \tag{4.5.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i, \ \forall i \in \{1, 2, ..., m\} & (4.5.2) \\ \sum_{i=1}^{m} x_{ij} = b_j, \quad \forall j \in \{1, 2, ..., n\} & (4.5.3) \\ \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j, & (4.5.4) \\ x_{ij} \ge 0, \forall i \in \{1, 2, ..., m\}, \forall j \in \{1, 2, ..., n\}. & (4.5.5) \end{cases}$$

Следует заметить, что, в отличие от стандартной задачи линейного программирования, в математической постановке транспортной задачи в виде (4.5.1)—(4.5.5) для удобства используются переменные с двумя индексами. При этом общее число переменных транспортной задачи равно: $m{\cdot}n$, что делает возможным сформулировать эквивалентную математическую постановку транспортной задачи с одноиндексными переменными.

Классическая транспортная задача линейного программирования является *сбалансированной* или *закрытой*, т. е. формулируется в форме, когда имеет место равенство общего объема производства рассматриваемого продукта общему объему его потребления. Этому условию соответствует отдельное ограничение (4.5.5). В противном случае, если равенство (4.5.5) не имеет места, то транспортная задача называется *несбалансированной* или *открытой*.

На практике встречаются различные модификации транспортной задачи. Наиболее известные из них используют дополнительную структуру типа графа для задания структуры транспортной сети, соединяющей пункты про-

изводства и потребления. Соответствующая транспортная задача может быть сформулирована в сетевой постановке применительно к конкретному графу и поэтому относится к классу задач оптимизации на графах.

В то же время классическая транспортная задача может быть дополнена условиями на ограничение сверху возможных значений некоторых или всех переменных: $x_{ij} \le h_{ij}$, где h_{ij} — пропускная способность транспорта между i-м пунктом производства и j-м пунктом потребления. Как нетрудно заметить, подобная модификация приведет к включению в модель (4.5.1)—(4.5.5) дополнительных ограничений. Однако эти дополнительные ограничения не оказывают существенного влияния на процесс их решения с помощью программы MS Excel.

4.5.2. Решение транспортной задачи с помощью программы MS Excel

Для решения классической транспортной задачи с помощью программы MS Excel необходимо задать конкретные значения параметрам исходной задачи. Для определенности рассмотрим задачу оптимального планирования перевозок бензина некоторой марки между нефтеперерабатывающими заводами (НПЗ) и автозаправочными станциями (АЗС). В этом случае в качестве транспортируемого продукта рассматривается бензин, в качестве пунктов производства — 3 нефтеперерабатывающих завода ($m = 3$), а в качестве пунктов потребления — 4 автозаправочные станции ($n = 4$). Объемы производства бензина следующие: НПЗ №1 — 10 m, НПЗ №2 — 14 m, НПЗ №3 — 17 m. Объемы потребления бензина следующие: АЗС №1 — 15 m, АЗС №2 — 12 m, АЗС №3 — 8,5 m, АЗС №4 — 5,5 m. Стоимость транспортировки одной тонны бензина между НПЗ и АЗС задана в форме следующей таблицы (табл. 4.8).

Таблица 4.8. *Стоимость транспортировки бензина между НПЗ и АЗС (в тыс. рублей)*

Пункты потребления /Пункты производства	**АЗС №1**	**АЗС №2**	**АЗС №3**	**АЗС №4**
НПЗ №1	3	5	7	11
НПЗ №2	1	4	6	3
НПЗ №3	5	8	12	7

Соответствующая математическая постановка рассматриваемой индивидуальной транспортной задачи может быть записана в следующем виде:

$$3x_{11}+5x_{12}+7x_{13}+11x_{14}+x_{21}+4x_{22}+6x_{23}+3x_{24}+ \\ +5x_{31}+8x_{32}+12x_{33}+7x_{34}\to\min\limits_{x\in\Delta_\beta}, \tag{4.5.6}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств:

$$\begin{cases} x_{11}+x_{12}+x_{13}+x_{14}=10; \\ x_{21}+x_{22}+x_{23}+x_{24}=14; \\ x_{31}+x_{32}+x_{33}+x_{34}=17; \\ x_{11}+x_{21}+x_{31}=15; \\ x_{12}+x_{22}+x_{32}=12; \\ x_{13}+x_{23}+x_{33}=8{,}5; \\ x_{14}+x_{24}+x_{34}=5{,}5; \\ x_{ij}\ge 0, \forall i\in\{1,\,2,\,3\}, \forall j\in\{1,\,2,\,3,\,4\}. \end{cases} \tag{4.5.7}$$

Заметим, что первых 3 ограничения данной задачи соответствуют общему ограничению (4.5.2), следующие 4 ограничения — общему ограничению (4.5.3), а последнее ограничение — общему ограничению (4.5.5). При этом общее ограничение (4.5.4), соответствующее требованию сбалансированности транспортной задачи, не входит в математическую модель рассматриваемой индивидуальной задачи. Это вполне допустимо, поскольку непосредственная проверка позволяет установить выполнение общего ограничения (4.5.4), а значит, исходная транспортная задача (4.5.6) и (4.5.7) является сбалансированной.

Примечание

Хотя в общем случае методы решения закрытых и открытых транспортных задач имеют некоторые отличия, с точки зрения решения этих задач с помощью программы MS Excel это не имеет принципиального значения. Поэтому в дальнейшем проверку условия сбалансированности можно вообще не принимать во внимание.

Для решения сформулированной индивидуальной транспортной задачи с помощью программы MS Excel создадим в книге Линейное программирование новый лист и изменим его имя на Транспортная задача. Для решения задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A5:A10**, **B1**, **F1**, **B5:G5**, как это изображено на рис. 4.15. Следует отметить, что конкретное содержание

этих надписей не оказывает никакого влияния на решение рассматриваемой транспортной задачи.

2. В ячейки **B2:E4** введем значения коэффициентов целевой функции (табл. 4.8).
3. В ячейку **F2** введем формулу: `=СУММПРОИЗВ(B2:E4; B6:E8)`, которая представляет целевую функцию (4.5.6).
4. В ячейки **G6:G8** и **B10:E10** введем значения, соответствующие правым частям ограничений (4.5.7).
5. В ячейку **F6** введем формулу: `=СУММ(B6:E6)`, которая представляет первое ограничение (4.5.7).
6. Скопируем формулу, введенную в ячейку **F6**, в ячейки **F7** и **F8**.
7. В ячейку **B9** введем формулу: `=СУММ(B6:B8)`, которая представляет четвертое ограничение (4.5.7).
8. Скопируем формулу, введенную в ячейку **B9**, в ячейки **C9**, **D9** и **E9**.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения транспортной задачи показан на рис. 4.16.

ЛинейноеПрограммирование.xls

	A	B	C	D	E	F	G
1		Коэффициенты целевой функции:				**Значение ЦФ:**	
2		3	5	7	11	**=СУММПРОИЗВ(B2:E4;B6:E8)**	
3		1	4	6	3		
4		5	8	12	7		
5	**Переменные:**	***Xi1***	***Xi2***	***Xi3***	***Xi4***	Значения огр-ний:	Производство НПЗ:
6	***X1j***					=СУММ(B6:E6)	10
7	***X2j***					=СУММ(B7:E7)	14
8	***X3j***					=СУММ(B8:E8)	17
9	Значения огр-ний:	=СУММ(B6:B8)	=СУММ(C6:C8)	=СУММ(D6:D8)	=СУММ(E6:E8)		
10	Потребности АЗС:	15	12	8,5	5,5		
11							
12							
13							
14							
15							
16							

Транспортная задача

Рис. 4.16. Исходные данные для решения транспортной задачи

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**…

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$F$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.

3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес диапазона ячеек `$B$2:$E$4`.
4. Добавить 7 ограничений, соответствующих базовым ограничениям исходной постановки решаемой транспортной задачи. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$F$6**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строгое неравенство "=";
   - в качестве значения правой части ограничения выбрать ячейку **$G$6**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**;
   - аналогичным образом задать оставшиеся 6 ограничений.
5. Добавить последнее ограничение на неотрицательность значений переменных задачи. Необходимые для этого действия были рассмотрены ранее в *разд. 4.1*. Внешний вид диалогового окна мастера поиска решения с ограничениями для транспортной задачи изображен на рис. 4.17.
6. В дополнительном окне параметров поиска решения следует выбрать отметки **Линейная модель** и **Неотрицательные значения**.

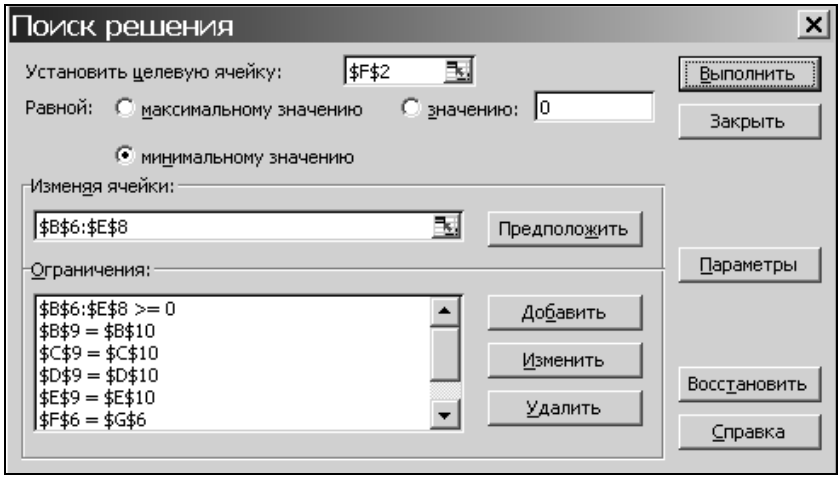
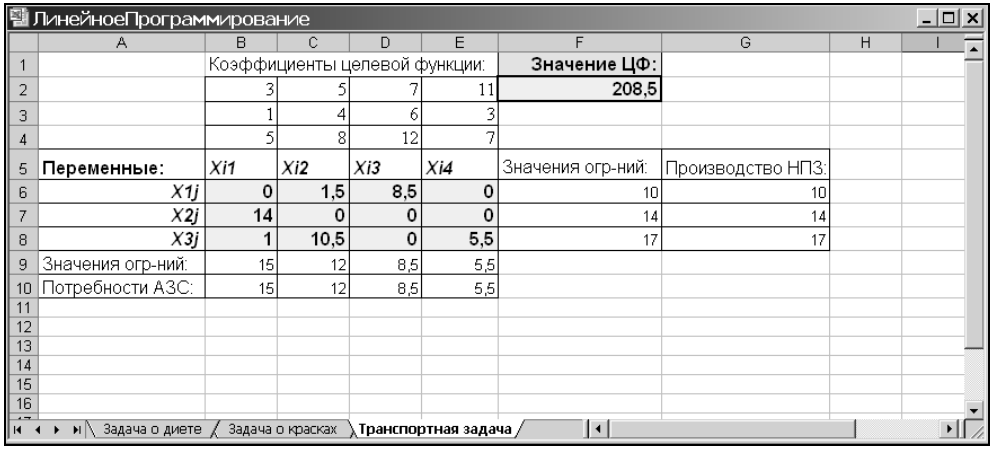


Рис. 4.17. Параметры мастера поиска решения и базовые ограничения для транспортной задачи

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 4.18).



	A	B	C	D	E	F	G	H	I
1		Коэффициенты целевой функции:				**Значение ЦФ:**			
2		3	5	7	11	**208,5**			
3		1	4	6	3				
4		5	8	12	7				
5	**Переменные:**	*Xi1*	*Xi2*	*Xi3*	*Xi4*	Значения огр-ний:	Производство НПЗ:		
6	*X1j*	**0**	**1,5**	**8,5**	**0**	10	10		
7	*X2j*	**14**	**0**	**0**	**0**	14	14		
8	*X3j*	**1**	**10,5**	**0**	**5,5**	17	17		
9	Значения огр-ний:	15	12	8,5	5,5				
10	Потребности АЗС:	15	12	8,5	5,5				
11									
12									
13									
14									
15									
16									

Рис. 4.18. Результат количественного решения транспортной задачи

Результатом решения транспортной задачи являются найденные оптимальные значения переменных: $x_{11} = 0$, $x_{12} = 1{,}5$, $x_{13} = 8{,}5$, $x_{14} = 0$, $x_{21} = 14$, $x_{22} = 0$, $x_{23} = 0$, $x_{24} = 0$, $x_{31} = 1$, $x_{32} = 10{,}5$, $x_{33} = 0$, $x_{34} = 5{,}5$, которым соответствует значение целевой функции: $f_{opt} = 208{,}5$. При выполнении расчетов для ячеек **B6:E8** был выбран числовой формат с тремя знаками после запятой.

Анализ найденного решения показывает, что для удовлетворения потребностей АЗС №1 следует транспортировать 14 т бензина из НПЗ №2 и 1 т — из НПЗ №3, для удовлетворения потребностей АЗС №2 следует транспортировать 1,5 т бензина из НПЗ №1 и 10,5 т — из НПЗ №3, для удовлетворения потребностей АЗС №3 следует транспортировать 8,5 т бензина из НПЗ №1 и, наконец, для удовлетворения потребностей АЗС №4 следует транспортировать 5,5 т бензина из НПЗ №3. При этом общая стоимость найденного плана перевозок составит 208,5 тыс. рублей.

Примечание

Найденное оптимальное решение в точности соответствует исходной постановке задачи. Для проверки правильности найденного решения можно воспользоваться либо ручным просчетом, используя для этого один из специальных алгоритмов решения транспортных задач, например, алгоритм метода потенциалов, либо другую программу, например, MATLAB или Mathcad. Заин-

тересованные читатели могут выполнить это в качестве упражнения. С другими вариантами транспортной задачи линейного программирования можно познакомиться в дополнительной литературе, список которой приведен в конце книги.

4.5.3. Решение транспортной задачи с помощью метода потенциалов

Для оценки точности и правильности результатов решения транспортных задач линейного программирования, полученных с помощью программных средств, в общем случае можно воспользоваться рассмотренным ранее симплекс-методом. Однако специально для решения транспортной задачи линейного программирования был разработан метод потенциалов. Основная идея этого метода основывается на *критерии оптимальности*, который может быть сформулирован следующим образом.

Для того чтобы исходная закрытая транспортная задача линейного программирования (4.5.1)—(5.5.5) имела оптимальное решение, необходимо и достаточно существование таких неотрицательных чисел $\{v_1, v_2, \ldots, v_n, u_1, u_2, \ldots, u_m\}$, которые обеспечивают выполнение двух групп условий:

$$u_i + v_j \le c_{ij}, \ (\forall i \in \{1, 2, \ldots, m\}, \ \forall j \in \{1, 2, \ldots, n\}) \text{ и} \qquad (4.5.8)$$

$$\text{если некоторое } x_{ij} > 0, \text{ то } u_i + v_j = c_{ij}. \qquad (4.5.9)$$

Соответствующие данным условиям числа $\{v_1, v_2, \ldots, v_n, u_1, u_2, \ldots, u_m\}$ получили название *потенциалов*. Очевидно, данные условия могут служить признаком окончания поиска решения транспортной задачи.

Общая идея определения оптимального решения транспортной задачи методом потенциалов аналогична идее решения задачи линейного программирования симплекс-методом, а именно: сначала находят некоторое начальное допустимое решение транспортной задачи, а затем его последовательно улучшают до получения оптимального решения. В общем случае алгоритм метода потенциалов имеет итеративный характер и заключается в выполнении следующих действий:

1. Если исходная транспортная задача линейного программирования является открытой, то она преобразуется к замкнутому виду (4.5.1)—(4.5.5). С этой целью могут быть введены дополнительные переменные $\{x_{m+1,j}\}$ $(\forall j \in \{1, 2, \ldots, n\})$ для фиктивного пункта производства a_{m+1}, если выполняется неравенство: $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$, или дополнительные переменные $\{x_{i,n+1}\}$ $(\forall i \in \{1, 2, \ldots, m\})$ для фиктивного пункта потребления b_{n+1}, если выполня-

ется неравенство: $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$. При этом дополнительным переменным должны соответствовать нулевые коэффициенты целевой функции: $c_{m+1,1} = c_{m+1,2} = \ldots = c_{m+1,n} = 0$ или $c_{1,n+1} = c_{2,n+1} = \ldots = c_{m,n+1} = 0$. Тем самым, с точностью до обозначений индексов переменных, в качестве исходной транспортной задачи будем рассматривать ее математическую модель в замкнутой форме (4.5.1)—(4.5.5).

2. Для транспортной задачи в замкнутой форме (4.5.1)—(4.5.5) находится некоторое начальное допустимое решение, которое записывается в специальную таблицу следующего вида (табл. 4.9).

Таблица 4.9. *Общий вид таблицы метода потенциалов*

$F(x)$	v_1 b_1	v_2 b_2	…	v_n b_n
u_1 a_1	c_{11} x_{11}	c_{12} x_{12}	…	c_{1n} x_{1n}
u_2 a_2	c_{12} x_{12}	c_{22} x_{22}	…	c_{2n} x_{2n}
…	…	…	…	…
u_m a_m	c_{m1} x_{m1}	c_{m2} x_{m2}	…	c_{mn} x_{mn}

Рассмотрим особенности построения данной таблицы. Верхняя строка и левый столбец содержат искомые значения потенциалов, которые требуется отыскать на последующих этапах алгоритма, и значения правых частей ограничений (4.5.2) и (4.5.3). В каждой ячейке таблицы содержится два значения: c_{ij} — стоимость транспортировки единицы продукта из i-го пункта производства в j-й пункт потребления и x_{ij} — значения переменных начального допустимого решения. При этом значения c_{ij} соответствуют коэффициентам целевой функции исходной замкнутой транспортной задачи (4.5.1) и в последующем не изменяются. Элементы x_{ij} соответствуют значениям переменных промежуточных решений транспортной задачи линейного программирования и изменяются на каждой итерации алгоритма.

Если в некоторой ячейке $x_{ij} = 0$, то такая ячейка называется *свободной*, если же $x_{ij} > 0$, то такая ячейка называется *занятой*. Самая верхняя слева ячейка исходной таблицы содержит значение целевой функции (4.5.1) для содержащегося в таблице промежуточного решения. При этом значение целевой функции рассчитывается по формуле: $F(x) = c_{11}x_{11} + c_{12}x_{12} + \ldots + c_{nm}x_{nm}$, где x_{ij} — ненулевые элементы табл. 4.9, соответствующие переменным решаемой задачи.

Следует заметить, что первые два этапа метода потенциалов являются подготовительными. Все последующие действия имеют итеративный повторяющийся характер и выполняются в рамках построенной исходной таблицы.

3. Для построенной таблицы (табл. 4.9) находятся значения потенциалов пунктов производства и потребления: $v_1, v_2, \ldots, v_n, u_1, u_2, \ldots, u_m$. С этой целью составляется и решается следующая система линейных уравнений:

$$v_j + u_i = c_{ij} \ (\forall i \in \{1, 2, \ldots, m\}, \forall j \in \{1, 2, \ldots, n\}), \qquad (4.5.10)$$

где индексы i и j соответствуют только ненулевым значениям переменных x_{ij} или занятым ячейкам табл. 4.9. Как нетрудно заметить, существование решения системы уравнений (4.5.10) обеспечивает выполнение второй группы условий критерия оптимальности (4.5.9). Для удобства расчетов найденные значения потенциалов также записываются в табл. 4.9.

4. Для найденного решения системы уравнений (4.5.1) проверяется первая группа условий (4.5.8) критерия оптимальности. С этой целью вначале рассчитываются оценки свободных ячеек табл. 4.9 по следующей формуле:

$$\Delta_{ij} = c_{ij} - v_j - u_i \ (\forall i \in \{1, 2, \ldots, m\}, \forall j \in \{1, 2, \ldots, n\}), \qquad (4.5.11)$$

где индексы i и j соответствуют только нулевым значениям переменных x_{ij} или свободным ячейкам табл. 4.9. В этом случае проверка первой группы условий критерия оптимальности найденного решения сводится к проверке следующего условия только для свободных ячеек:

$$\Delta_{ij} \geq 0 \ (\forall i \in \{1, 2, \ldots, m\}, \ \forall j \in \{1, 2, \ldots, n\}). \qquad (4.5.12)$$

Если условие (4.5.12) выполняется, то найденное решение является оптимальным, и на этом дальнейшие расчеты могут быть завершены. Если же условие (4.5.12) не выполняется, то следует перейти к выполнению следующего этапа алгоритма метода потенциалов.

Из всех $\Delta_{ij} < 0$ выбирается наименьшее значение (если их несколько — то любое из них). Соответствующая свободная ячейка помечается знаком (+), и для нее в таблице метода потенциалов строится цикл. При этом *циклом* в таблице метода потенциалов называется ломаная линия, вершины кото-

рой расположены в занятых ячейках таблицы, а звенья — вдоль строк и столбцов, причем в каждой вершине цикла встречается ровно два звена, одно из которых находится в строке, а другое — в столбце. Если ломаная линия, образующая цикл, пересекается, то точки самопересечения не являются вершинами. При правильном построении таблицы допустимого решения для любой свободной ячейки можно построить лишь один цикл.

5. После того как построен цикл для выбранной свободной ячейки, следует рассчитать значения переменных нового допустимого решения. Для этого необходимо изменить значения переменных предыдущего допустимого решения в пределах ячеек, связанных с данной свободной ячейкой. Это изменение производят по следующим правилам:

   - каждой ячейке, принадлежащей построенному циклу от выбранной свободной ячейки, приписывают определенный знак, причем свободной клетке — знак (+), а всем остальным клеткам — поочередно знаки (–) и (+). Соответствующие ячейки называют также *минусовыми* и *плюсовыми*;
   - в выбранную свободную ячейку записывают меньшее из чисел x_{ij}, стоящих в минусовых ячейках. Одновременно это число прибавляют к соответствующим числам, стоящим в плюсовых ячейках, и вычитают из чисел, стоящих в минусовых ячейках таблицы. При этом ячейка, которая ранее была свободной, становится занятой, а минусовая ячейка, в которой стояло минимальное из чисел x_{ij}, считается свободной.

   В результате указанного изменения значений переменных в пределах ячеек, связанных циклом с данной свободной ячейкой, находится новое допустимое решение транспортной задачи, которому соответствует меньшее по сравнению с предыдущим решением значение целевой функции. После получения новой таблицы метода потенциалов следует перейти к выполнению действий этапа 3 настоящего алгоритма.

Примечание

Следует заметить, что при пересчете значений переменных по циклу, число занятых ячеек в таблице остается неизменным. При этом если в минусовых ячейках имеется два или более одинаковых числа x_{ij}, то освобождают лишь одну из таких ячеек, а остальные оставляют занятыми с нулевыми значениями переменных нового допустимого решения.

Рассмотренный алгоритм метода потенциалов может быть изображен графически в форме диаграммы деятельности языка UML (рис. 4.19).

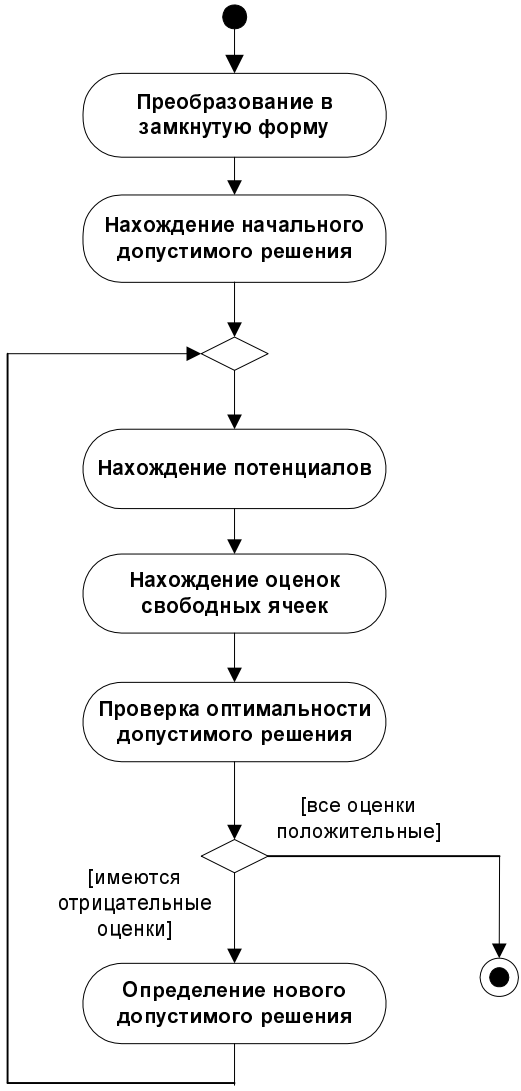


Рис. 4.19. Диаграмма деятельности алгоритма метода потенциалов

В заключение следует отметить, что при определении начального допустимого решения или в процессе решения задачи может быть получено вырожденное решение. Чтобы избежать в этом случае зацикливания алгоритма, следует соответствующие нулевые элементы допустимого решения заменить сколь угодно малым положительным числом ε, после чего решать задачу как невырожденную. В оптимальном решении такой задачи необходимо считать ε равным нулю.

Для нахождения исходного допустимого решения транспортной задачи на этапе 2 алгоритма может быть использован так называемый *метод минимального элемента.* Сущность этого метода состоит в том, что начальное допустимое решение находится за $n + m - 1$ шагов. При этом на каждом шаге находится значение только одной переменной x_{ij}, которая записывается в соответствующую ячейку. После чего данная ячейка становится занятой. Первоначально все ячейки таблицы свободные и среди них отыскивается такая ячейка, которой соответствует минимальное значение из коэффициентов целевой функции c_{ij}. Если таких ячеек несколько, то следует выбрать любую из них. Для найденной свободной ячейки определяется значение соответствующей переменной: $x_{ij} = min\{a_i, b_j\}$. Заполнение выбранной ячейки обеспечивает полностью либо удовлетворение потребности в пункте потребления, если $x_{ij} = b_j = min\{a_i, b_j\}$, либо вывоз всех запасов из пункта производства, если $x_{ij} = a_i = min\{a_i, b_j\}$.

В первом случае исключают из дальнейшего рассмотрения столбец таблицы, соответствующий b_j, а для i-й строки полагают новое значение $a_i' = a_i - b_j$. Во втором случае исключают из дальнейшего рассмотрения строку, соответствующую a_i, а для j-го столбца полагают новое значение $b_j' = b_j - a_i$.

После исключения строки или столбца из дальнейшего рассмотрения происходит нахождение среди свободных ячеек следующего минимального значения c_{ij} и заполнение найденной ячейки очередным значением переменной: $x_{ij} = min\{a_i, b_j\}$ с соответствующим исключением строки или столбца. В итоге после $n + m - 1$ шагов метод минимального элемента позволяет получить начальное допустимое решение закрытой транспортной задачи линейного программирования.

Примечание

Следует отметить, что метод минимального элемента, как правило, позволяет найти начальное допустимое решение транспортной задачи, при котором общая стоимость перевозок груза меньше, чем общая стоимость перевозок при допустимом решении, найденном для данной задачи с помощью популярного метода северо-западного угла. Именно поэтому целесообразно начальное решение транспортной задачи находить методом минимального элемента.

Проиллюстрируем использование рассмотренного алгоритма метода потенциалов для решения индивидуальной транспортной задачи (4.5.6) и (4.5.7). Поскольку исходная задача является закрытой, то выполнение действий этапа 1 рассмотренного алгоритма метода потенциалов не требуется.

Исходная таблица метода потенциалов, необходимая для нахождения начального допустимого решения задачи (4.5.6) и (4.5.7), будет иметь следующий вид (табл. 4.10).

Таблица 4.10. *Исходная таблица для нахождения начального допустимого решения*

$F(x)$	v_1 **15**	v_2 **12**	v_3 **8.5**	v_4 **5.5**
u_1 **10**	3	5	7	11
u_2 **14**	1	4	6	3
u_3 **17**	5	8	12	7

Для нахождения начального допустимого решения воспользуемся методом минимального элемента. Для этого в табл. 4.10 следует найти минимальное значение c_{ij}, которое равно 1. Этому значению соответствует второй пункт производства и первый пункт потребления, при этом $x_{21} = min\{a_2,\ b_1\} = 14$. Из дальнейшего рассмотрения следует исключить второй пункт производства, а для первого пункта потребления определить новое значение $b_1' = b_1 - a_2 = 15 - 14 = 1$.

На следующем шаге метода минимального элемента в сокращенной таблице (табл. 4.10) найдем минимальное значение c_{ij}, которое равно 3. Этому значению соответствует первый пункт производства и первый пункт потребления, при этом $x_{11} = min\{a_1,\ b_1\} = 1$. Из дальнейшего рассмотрения следует исключить первый пункт потребления, а для первого пункта производства определить новое значение $a'_1 = a_1 - b_1 = 10 - 1 = 9$.

Поступая аналогичным образом, в результате будет получено начальное допустимое решение транспортной задачи (4.5.6) и (4.5.7), исходная таблица метода потенциалов которой будет иметь следующий вид (табл. 4.11).

Таблица 4.11. *Исходная таблица метода потенциалов с начальным допустимым решением*

$F(x)$= **226,5**	v_1 **15**	v_2 **12**	v_3 **8,5**	v_4 **5,5**
u_1 **10**	3 1	5 9	7	11

Таблица 4.11 *(окончание)*

$F(x)$= 226,5	v_1	v_2	v_3	v_4
	15	**12**	**8,5**	**5,5**
u_2	1	4	6	3
14	14			
u_3	5	8	12	7
17		3	8,5	5,5

Непосредственной проверкой можно убедиться, что найденное начальное решение действительно является допустимым. Этому начальному решению соответствует значение целевой функции: $F(x) = 3 \cdot 1 + 1 \cdot 14 + 5 \cdot 9 + 8 \cdot 3 + 12 \cdot 8{,}5 + 7 \cdot 5{,}5 = 226{,}5$.

После выполнения подготовительных этапов 1 и 2 метода потенциалов можно приступить к проверке условия получения оптимального решения (этап 3). Для этого необходимо найти потенциалы пунктов производства и потребления. Поскольку число заполненных ячеек исходной таблицы равно $n + m - 1 = 6$, то искомая система должна содержать $n + m = 7$ неизвестных для 6 уравнений. А именно, для определения значений потенциалов следует решить следующую систему уравнений: $\{v_1 + u_1 = 3,\ v_1 + u_2 = 1,\ v_2 + u_1 = 5,\ v_2 + u_3 = 8,\ v_3 + u_3 = 12,\ v_4 + u_3 = 7\}$, содержащую шесть уравнений с семью неизвестными. Поскольку число неизвестных превышает на единицу число уравнений, то одно из неизвестных можно положить равным произвольному числу, например $v_1 = 0$. Далее можно найти последовательно из данной системы уравнений значения остальных неизвестных: $v_2 = 2$, $v_3 = 6$, $v_4 = 1$, $u_1 = 3$, $u_2 = 1$, $u_3 = 6$. На этом действия этапа 3 заканчиваются, а найденные значения потенциалов записываются в исходную таблицу, которая на первой итерации алгоритма будет иметь следующий вид (табл. 4.12).

Для выполнения этапа 4 алгоритма по формуле (4.5.11) необходимо последовательно рассчитать значения оценок для свободных ячеек: $\Delta_{13} = 7 - 3 - 6 = -2$, $\Delta_{14} = 11 - 3 - 1 = 7$, $\Delta_{22} = 4 - 1 - 2 = 1$, $\Delta_{23} = 6 - 1 - 6 = -1$, $\Delta_{24} = 3 - 1 - 1 = 1$, $\Delta_{31} = 5 - 6 - 0 = -1$. Поскольку среди оценок свободных ячеек имеются отрицательные, то условие (4.5.12) не выполняется, и найденное решение не является оптимальным, т. е. его можно улучшить. Из всех $\Delta_{ij} < 0$ выбирается наименьшее значение $\Delta_{13} = -2$. Соответствующая свободная ячейка для x_{13} помечается знаком (+), и для нее в таблице метода потенциалов строится цикл, содержащий занятые ячейки: x_{12}, x_{32}, x_{33}. После этого следует перейти к выполнению действий этапа 5.

Таблица 4.12. *Таблица метода потенциалов на первой итерации*

$F(x) = 226{,}5$	**0**	**2**	**6**	**1**
	15	**12**	**8,5**	**5,5**
3	3	5	7	11
10	1	9		
1	1	4	6	3
14	14			
6	5	8	12	7
17		3	8,5	5,5

На этапе 5 необходимо определить плюсовые и минусовые ячейки. Поскольку ячейка для x_{13} имеет знак (+), то соседние с ней в цикле занятые ячейки x_{12} и x_{33} будут иметь знак (–). Следуя правилу чередования знаков, оставшаяся ячейка x_{32} будет иметь знак (+). Наименьшее из чисел в минусовых ячейках равно 8,5. Ячейка x_{33}, в которой находится это число, становится свободной в новой таблице метода потенциалов. Другие значения ячеек цикла в новой таблице получаются следующим образом: новое значение в минусовой ячейке равно: $x'_{12} = x_{12} - 8{,}5 = 0{,}5$, новое значение в плюсовой ячейке равно: $x'_{32} = x_{32} + 8{,}5 = 11{,}5$. В полученной таким образом новой таблице ячейка $x'_{33} = 0$ становится свободной. После выполненных на этапе 5 преобразований получаем новое допустимое решение транспортной задачи с лучшим значением целевой функции $F(x) = 209{,}5$. Этому допустимому решению соответствует новая таблица метода потенциалов, которая имеет следующий вид (табл. 4.13).

После получения табл. 4.13 следует приступить к проверке условия получения оптимального решения (вторая итерация, этап 3). Для этого предварительно необходимо найти новые потенциалы пунктов производства и потребления. Для определения значений потенциалов следует решить следующую систему уравнений: $\{v_1 + u_1 = 3,\ v_1 + u_2 = 1,\ v_2 + u_1 = 5,\ v_2 + u_3 = 8,\ v_3 + u_1 = 7,\ v_4 + u_3 = 7\}$. Полагая $v_1 = 0$, находятся значения остальных неизвестных: $v_2 = 2$, $v_3 = 4$, $v_4 = 1$, $u_1 = 3$, $u_2 = 1$, $u_3 = 6$. На этом действия этапа 3 заканчиваются, а найденные значения потенциалов записываются в таблицу, которая на второй итерации алгоритма будет иметь следующий вид (табл. 4.14).

Таблица 4.13. *Таблица метода потенциалов после выполнения первой итерации*

$F(x) = 209{,}5$	v_1 **15**	v_2 **12**	v_3 **8,5**	v_4 **5,5**
u_1 **10**	3 1	5 0,5 (–)	7 8,5 (+)	11
u_2 **14**	1 14	4	6	3
u_3 **17**	5	8 11,5 (+)	12 (–)	7 5,5

Таблица 4.14. *Таблица метода потенциалов на второй итерации*

$F(x) = 209.5$	**0** **15**	**2** **12**	**4** **8,5**	**1** **5,5**
3 **10**	3 1	5 0,5	7 8,5	11
1 **14**	1 14	4	6	3
6 **17**	5	8 11,5	12	7 5,5

Для выполнения этапа 4 на второй итерации алгоритма по формуле (4.5.11) необходимо последовательно рассчитать значения оценок для свободных ячеек: $\Delta_{14} = 11 - 3 - 1 = -7$, $\Delta_{22} = 4 - 1 - 2 = 1$, $\Delta_{23} = 6 - 1 - 4 = 1$, $\Delta_{24} = 3 - 1 - 1 = 1$, $\Delta_{31} = 5 - 6 - 0 = -1$, $\Delta_{33} = 12 - 6 - 4 = 2$. Поскольку среди оценок свободных ячеек имеется единственная отрицательная, то условие (4.5.12) не выполняется, и найденное решение не является оптимальным, т. е. его можно улучшить. Для единственного значения $\Delta_{31} = -1$ соответствующая свободная ячейка для x_{31} помечается знаком (+), и для нее в таблице метода потенциалов строится цикл, содержащий занятые ячейки: x_{11}, x_{12}, x_{32}. После этого следует перейти к выполнению действий этапа 5 второй итерации.

На этапе 5 необходимо определить плюсовые и минусовые ячейки. Поскольку ячейка для x_{31} имеет знак (+), то соседние с ней в цикле занятые ячейки x_{11} и x_{32} будут иметь знак (–). Следуя правилу чередования знаков, оставшаяся ячейка x_{12} будет иметь знак (+). Наименьшее из чисел в минусовых ячейках равно 1. Ячейка x_{11}, в которой находится это число, становится свободной в новой таблице метода потенциалов. Другие значения ячеек цикла в новой таблице получаются следующим образом: новое значение в минусовой ячейке равно: $x'_{32} = x_{32} - 1 = 10{,}5$, а новое значение в плюсовой ячейке равно: $x'_{12} = x_{12} + 1 = 1{,}5$. В полученной таким образом новой таблице ячейка $x'_{31} = 0$ становится свободной. После выполненных на этапе 5 преобразований получаем новое допустимое решение транспортной задачи с лучшим значением целевой функции $F(x) = 208{,}5$. Этому допустимому решению соответствует новая таблица метода потенциалов, которая имеет следующий вид (табл. 4.15).

Таблица 4.15. *Таблица метода потенциалов после выполнения второй итерации*

$F(x)$= 208,5	v_1 **15**	v_2 **12**	v_3 **8,5**	v_4 **5,5**
u_1 **10**	3 (–)	5 1,5 (–)	7 8,5	11
u_2 **14**	1 14	4	6	3
u_3 **17**	5 1 (+)	8 10,5 (–)	12	7 5,5

После получения табл. 4.15 следует снова проверить условия получения оптимального решения (третья итерация, этап 3). Для этого необходимо найти новые потенциалы пунктов производства и потребления, т. е. решить следующую систему уравнений: $\{v_1 + u_2 = 1,\ v_1 + u_3 = 5,\ v_2 + u_1 = 5,\ v_2 + u_3 = 8,\ v_3 + u_1 = 7,\ v_4 + u_3 = 7\}$. Полагая $v_1 = 0$, находятся значения остальных неизвестных: $v_2 = 3$, $v_3 = 5$, $v_4 = 2$, $u_1 = 2$, $u_2 = 1$, $u_3 = 5$. На этом действия этапа 3 заканчиваются, а найденные значения потенциалов записываются в таблицу, которая на третьей итерации алгоритма будет иметь следующий вид (табл. 4.16).

Таблица 4.16. *Таблица метода потенциалов на третьей итерации*

$F(x) = 208{,}5$	**0** **15**	**3** **12**	**5** **8,5**	**2** **5,5**
2 **10**	3	5 1,5	7 8,5	11
1 **14**	1 14	4	6	3
5 **17**	5 1	8 10,5	12	7 5,5

Для выполнения этапа 4 на третьей итерации алгоритма по формуле (4.5.11) необходимо последовательно рассчитать значения оценок для свободных ячеек: $\Delta_{11} = 3 - 2 - 0 = 1$, $\Delta_{14} = 11 - 2 - 2 = 7$, $\Delta_{22} = 4 - 1 - 3 = 0$, $\Delta_{23} = 6 - 1 - 5 = 0$, $\Delta_{24} = 3 - 1 - 2 = 0$, $\Delta_{33} = 12 - 5 - 5 = 2$. Поскольку среди оценок свободных ячеек отсутствуют отрицательные значения, то условие (4.5.12) выполняется, и найденное решение является оптимальным. Таким образом, искомое оптимальное решение исходной транспортной задачи, полученное с использованием описанного алгоритма метода потенциалов, содержится в табл. 4.16 и равно: $x_{12} = 1{,}5$, $x_{13} = 8{,}5$, $x_{21} = 14$, $x_{31} = 1$, $x_{32} = 10{,}5$, $x_{34} = 5{,}5$, значения остальных переменных равны 0. Оптимальное значение целевой функции при этом равно: $F(x) = 208{,}5$.

Сравнение найденных оптимальных решений транспортной задачи с помощью программы MS Excel (см. рис. 4.18) и метода потенциалов (табл. 4.16) показывает их полное совпадение, что может свидетельствовать о достоверности соответствующих результатов.

4.6. Упражнения

В качестве упражнений для самостоятельного решения с помощью программы MS Excel предлагаются несколько типовых задач линейного программирования. Заинтересованные читатели могут найти оптимальные решения представленных задач всеми рассмотренными способами, а полученные результаты сравнить.

4.6.1. Задача о производстве клея

Некоторое производственное предприятие выпускает три вида клея. Для производства клея используется 4 типа химических веществ: крахмал, желатин,

квасцы и мел. Расход этих веществ в *кг* для получения 1 *кг* каждого вида клея и их запас на складе предприятия представлены в табл. 4.17.

Таблица 4.17. *Расход химических веществ на изготовления клея, их запас на складе*

Вид клея/ Химические вещества	Клей №1	Клей №2	Клей №3	Запас на складе
Крахмал	0,4	0,3	0,2	20
Желатин	0,2	0,3	0,4	35
Квасцы	0,05	0,07	0,1	7
Мел	0,01	0,05	0,15	10

Стоимость каждого вида клея для оптовых покупателей следующая: $c_1 = 380$ *руб/кг*, $c_2 = 430$ *руб/кг*, $c_3 = 460$ *руб/кг*. Требуется определить оптимальный объем выпуска клея каждого вида, обеспечивающий максимум общей стоимости готовой продукции.

4.6.2. Задача об оптимальной диете

Имеется конечное число видов продуктов питания: ананас, арбуз, грейпфрут, язык говяжий, сардельки говяжьи, хлеб "Бородинский", картофель ($n = 7$), а в качестве питательных веществ рассматриваются белки, жиры, углеводы ($m = 3$). Калорийность 1 *кг* каждого из продуктов следующая: $c_1 = 470$, $c_2 = 380$, $c_3 = 350$, $c_4 = 1460$, $c_5 = 2150$, $c_6 = 2070$, $c_7 = 800$. Минимальная суточная потребность в питательных веществах следующая: в белках $b_1 = 100$, в жирах $b_2 = 70$, в углеводах $b_3 = 400$. Содержание питательных веществ в каждом из продуктов может быть задано в форме следующей таблицы (табл. 4.18).

Таблица 4.18. *Содержание питательных веществ в продуктах питания*

Продукты/ Питательные вещества	Ананас	Арбуз	Грейпфрут	Язык говяжий	Сардельки говяжьи	Хлеб "Бородинский"	Картофель
Белки	4	7	9	122	114	68	20
Жиры	2	2	2	109	182	13	4
Углеводы	115	88	65	0	15	407	163

Требуется определить такой рацион питания, чтобы каждое питательное вещество содержалось в нем в необходимом количестве, обеспечивающем

суточную потребность человека, и при этом суммарная калорийность рациона была минимальной.

Примечание

В случае получения оптимального решения, соответствующего диете, страдающей однообразием продуктов питания, можно самостоятельно составить аналогичную задачу. Для этого можно использовать все упомянутые в настоящей главе продукты, а также ввести дополнительные ограничения на значения отдельных переменных, например, ограничив сверху потребление хлеба или ананасов. Подобные дополнительные ограничения не оказывают принципиального влияния на порядок решения полученной таким образом задачи линейного программирования данного типа.

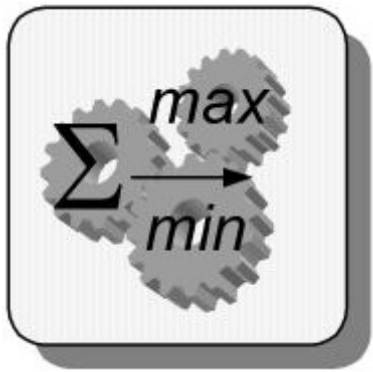
4.6.3. Транспортная задача

Сельскохозяйственные кооперативы ($m = 4$) выращивают и зерно. В качестве пунктов потребления выступают зерновые склады или элеваторы для хранения зерна ($n = 3$). Объемы производства зерна следующие: кооператив № 1 — 155,5 *т*, кооператив № 2 — 250 *т*, кооператив № 3 — 170 *т*, кооператив № 4 — 134,5 *т*. Требуемые объемы хранения зерна следующие: склад № 1 — 350 *т*, склад № 2 — 250 *т*, склад № 3 — 400 *т* (открытая задача). Стоимость транспортировки 1 *т* зерна между кооперативами и складами задана в следующей таблице (табл. 4.19).

Таблица 4.19. *Стоимость транспортировки зерна между кооперативами и складами*

Пункты потребления/ Пункты производства	**Склад №1**	**Склад №3**	**Склад №3**
Кооператив №1	100	90	110
Кооператив №2	75,5	80	95
Кооператив №3	80	85	100
Кооператив №4	70	95	105

Требуется определить оптимальный план перевозок зерна, так чтобы все выращенное зерно было отправлено на склады, а суммарные затраты на транспортировку всего зерна были минимальными.



Часть III

Задачи дискретной и комбинаторной оптимизации

Глава 5. Задачи целочисленного линейного программирования

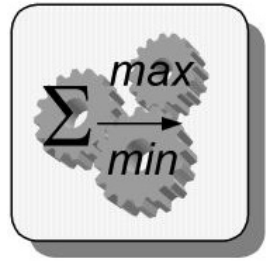
Глава 6. Задачи оптимизации с булевыми переменными

Глава 7. Задачи оптимизации на графах

Глава 8. Задачи комбинаторной оптимизации

Задачи дискретной и комбинаторной оптимизации отличаются от классических задач нелинейной оптимизации и линейного программирования исключением из их постановки условия непрерывности допустимых значений исходных переменных. В общем случае все экстремальные задачи подобного рода принято относить к классу задач дискретной оптимизации. При этом целевая функция и ограничения задач дискретной оптимизации могут быть как линейными, так и нелинейными. В случае линейности целевой функции и ограничений соответствующие задачи образуют класс задач целочисленного линейного программирования.

В то же время имеется целый ряд практических задач дискретной оптимизации, которые более наглядно и просто формулируются в виде нахождения некоторого комбинаторного объекта (перестановки, сочетания, выборки), удовлетворяющего дополнительным требованиям и доставляющего экстремум некоторой целевой функции. Это обстоятельство послужило основой для введения в рассмотрение специального класса задач оптимизации, который получил название задач комбинаторной оптимизации. Задачи комбинаторной оптимизации не только требуют знания специальных понятий комбинаторики, но и служат основой для разработки специальных методов своего решения. Именно по этим причинам особенности постановки и решения задач комбинаторной оптимизации рассматриваются в отдельной главе.



Глава 5

Задачи целочисленного линейного программирования

В общем случае целочисленное программирование ориентировано на решение задач оптимизации, в которых все или некоторые переменные должны принимать целочисленные значения. Задача оптимизации называется *полностью целочисленной*, если условие целочисленности наложено на все ее переменные. Если же требование целочисленности относится лишь к некоторым переменным, то такая задача оптимизации называется *частично целочисленной*. Если при этом целевая функция и функции, входящие в ограничения, линейные, то задача является линейной целочисленной. В настоящей главе будут рассматриваться только задачи полностью целочисленного линейного программирования.

Класс задач целочисленного линейного программирования является подклассом задач дискретной оптимизации, который от последнего наследует свойство целочисленности переменных. С другой стороны, класс задач целочисленного линейного программирования можно рассматривать как специальный подкласс задач линейного программирования, который от последнего наследует свойство линейности целевой функции и ограничений. Такое множественное наследование является вполне корректным и служит основой как для анализа новых свойств задач соответствующего класса, так и для разработки специальных методов их решения.

5.1. Общая постановка задачи целочисленного линейного программирования

К классу задач целочисленного линейного программирования относятся такие задачи однокритериальной оптимизации, в которых переменные могут принимать только целочисленные неотрицательные значения, целевая функция

является линейной функцией своих аргументов, а левые части ограничений могут быть представлены в форме линейных неравенств и равенств. При этом на значения переменных не накладываются никакие дополнительные ограничения, такие, например, как ограничения булевости. Более строгое определение данного класса задач можно получить на основе рассмотрения общей математической постановки задачи целочисленного линейного программирования.

5.1.1. Математическая постановка задачи целочисленного линейного программирования

В общем случае математическая постановка задачи целочисленного линейного программирования может быть сформулирована в следующем виде:

$$f(x_1, x_2, ..., x_n) \to \max_{x \in \Delta_\beta}, \text{ где} \tag{5.1.1}$$

$$\begin{gathered} \Delta_\beta = \{\Delta \mid g_k(x_1, x_2, ..., x_n) \le (=) b_k; \\ x_1, x_2, ..., x_n \ge 0; \\ x_1, x_2, ..., x_n - \text{целые числа}, \\ k \in \{1, 2, ..., m\}\} \end{gathered} \tag{5.1.2}$$

При этом имеют место следующие принципиальные предположения о характере целевой функции и левых частей ограничений:

1. Целевая функция $f(x_1, x_2, ..., x_n)$ предполагается линейной, т. е. может быть представлена в форме: $f(x_1, x_2, ..., x_n) = c_1 x_1 + c_2 x_2 + ... + c_n x_n$.
2. Левые части ограничений $g_k(x_1, x_2, ..., x_n) \cdot (\forall k \in \{1, 2, ..., m\})$ также являются линейными функциями относительно своих переменных $x_1, x_2, ..., x_n$, т. е. могут быть представлены в форме: $g_k(x_1, x_2, ..., x_n) = a_{k1} x_1 + a_{k2} x_2 + ... + a_{kn} x_n$.
3. Переменные $x_1, x_2, ..., x_n$ принимают свои значения из множества неотрицательных целых чисел $\boldsymbol{Z}_+^1$, т. е. $x_j \in \boldsymbol{Z}_+^1 \cdot (\forall i \in \{1, 2, ..., n\})$.

С учетом сделанных предположений *общая задача целочисленного линейного программирования* может быть сформулирована следующим образом.

Необходимо найти максимум линейной целевой функции n переменных $x_1, x_2, ..., x_n \in \boldsymbol{Z}_+^1$ следующего вида:

$$c_1 x_1 + c_2 x_2 + ... + c_n x_n \to \max_{x \in \Delta_\beta}, \tag{5.1.3}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств и неравенств:

$$\begin{cases} a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i & (\forall i \in \{1, 2, \ldots, q\}) \quad (5.1.4) \\ a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n \le b_k & (\forall k \in \{q+1, \ldots, m\}). \quad (5.1.5) \end{cases}$$

В математической постановке общей задачи целочисленного линейного программирования через $c_i, a_{ki}, b_k \cdot (\forall i \in \{1,2,\ldots,n\}, \forall k \in \{1, 2,\ldots, m\})$ обозначены постоянные величины, которые могут принимать произвольные, как правило, целочисленные значения, предопределяемые спецификой конкретной задачи линейного программирования.

Примечание

Задача минимизации линейной целевой функции при ограничениях типа равенств и неравенств может быть сведена к общей задаче целочисленного линейного программирования вида (5.1.3) и (5.1.4), поскольку, как уже отмечалось ранее, $f(x) \to min$ эквивалентно $-f(x) \to max$. Именно поэтому, не ограничивая общности, в последующем говоря о целочисленных задачах линейного программирования, будем иметь в виду задачи максимизации линейных функций типа (5.1.3) на множестве неотрицательных целочисленных переменных $\boldsymbol{Z}_+^1$.

В случае отсутствия ограничений типа равенств (5.1.4), т. е. при $q = 0$, задача целочисленного линейного программирования называется *стандартной задачей целочисленного линейного программирования*, которая, с учетом сделанных предположений, может быть записана в следующем виде:

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max_{x \in \Delta_\beta}, \quad (5.1.6)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \le b_1; \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \le b_2; \\ \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \le b_m \end{cases} \quad (5.1.7)$$

и $x_1, x_2, \ldots, x_n \ge 0$; $x_1, x_2, \ldots, x_n$ – целые числа.

Примечание

В случае отсутствия ограничений типа неравенств (5.1.5), т. е. при $q = m$, задача целочисленного линейного программирования называется *канонической* или *основной задачей целочисленного линейного программирования*. При этом стандартная задача целочисленного линейного программирования (5.1.6) и (5.1.7) аналогично задачам линейного программирования посредством введения дополнительных переменных может быть сведена к канонической задаче целочисленного линейного программирования и наоборот. Однако поскольку при рассмотрении общих особенностей постановки и решения задач целочисленного линейного программирования, как правило, используется стандартная форма математической постановки задач линейного программирования (5.1.6) и (5.1.7), каноническая форма задачи целочисленного линейного программирования в дальнейшем рассматриваться не будет.

При анализе множества допустимых альтернатив Δ_β стандартной задачи целочисленного линейного программирования (5.1.6) и (5.1.7) оказывается справедливой одна из трех возможных ситуаций:

1. Система ограничений (5.1.7) противоречива или несовместна, т. е. не существует такого набора значений $x_1, x_2, ..., x_n$, которые удовлетворяют ограничениям (5.1.7). В этом случае задача целочисленного линейного программирования не имеет решения.
2. Система ограничений (5.1.7) не является противоречивой, однако соответствующая ей область пространства $\boldsymbol{Z^n}$ является неограниченной. Если при этом линейная функция (5.1.6) неограниченна в неограниченной области, соответствующей множеству допустимых альтернатив Δ_β, то задача целочисленного линейного программирования не имеет решения.
3. Система ограничений (5.1.7) не является противоречивой, и соответствующая ей область пространства $\boldsymbol{Z^n}$ является ограниченной. В этом случае задача целочисленного линейного программирования имеет решение, которое, в общем случае, может быть не единственным.

При постановке и решении задач целочисленного линейного программирования следует иметь в виду следующее важное свойство: в случае ограниченности множества допустимых альтернатив Δ_β (ситуация 3) оно имеет *конечную мощность*. Другими словами, если множество Δ_β задачи целочисленного линейного программирования ограничено в пространстве $\boldsymbol{Z^n}$, то оно имеет конечное множество допустимых альтернатив.

5.1.2. Основные методы решения задач целочисленного линейного программирования

Рассмотрение в исходной математической постановке задачи целочисленного линейного программирования внешне мало заметного дополнительного требования целочисленности значений переменных приводит к принципиальному изменению не только характера соответствующих задач, но возможных методов их решения.

Так, применительно к задачам целочисленного линейного программирования оказывается неприменимым аналитический способ решения, основанный, например, на методе множителей Лагранжа, поскольку понятие дифференцируемости целевой функции и ограничений для целочисленных значений переменных теряет свой исходный смысл. Естественным выходом из данной ситуации представляется предварительное сведение исходной задачи целочисленного линейного программирования (5.1.6) и (5.1.7) посредством отказа от требования целочисленности переменных к соответствующей задаче линейного программирования (4.1.6) и (4.1.7) с последующим нахождением ее оптимального решения и его округлением до ближайших целочисленных допустимых значений переменных в случае их нецелочисленности. Однако, как показывают многочисленные примеры, округление до ближайших целочисленных значений переменных вовсе не гарантирует получение оптимального решения задачи (5.1.6) и (5.1.7).

Речь идет о том, что если получено оптимальное решение непрерывной задачи линейного программирования, значение одной из переменных которого равно, например, 10,25, то кажется вполне очевидным заменить это значение на ближайшее целое 10 и считать его оптимальным решением соответствующей задачи целочисленного линейного программирования. Однако в общем случае нет никакой гарантии, что округленное решение будет удовлетворять системе ограничений исходной задачи.

Эффект округления может быть не слишком заметен, когда переменные задачи подчинены нестрогим ограничениям. Однако между переменными задач целочисленного программирования с ограничениями типа равенств имеются достаточно жесткие связи. Более того, для задач линейного программирования с ограничениями в виде равенств округленное решение всегда не удовлетворяет этим ограничениям. Как следует из теории линейного программирования, округленное решение не может быть допустимым, поскольку это означало бы, что один и тот же базис при условии равенства нулю небазисных переменных определяет два различных решения задачи.

Несостоятельность округления подчеркивается также содержательными особенностями задач целочисленного программирования, поскольку целочис-

ленные переменные обычно выражают количество неделимых предметов, например, коробок, ящиков, контейнеров, машин, людей, кораблей.

Одна из основных трудностей решения задач целочисленного программирования с использованием программных средств и компьютеров связана с эффектом ошибки округления при выполнении численных расчетов. Даже наличие алгоритмов, применимых для решения задач с целочисленными коэффициентами и позволяющих обойтись без оперирования дробями и, следовательно, избежать влияния ошибок округления, не упрощает ситуации, поскольку такие алгоритмы в отдельных случаях сходятся чрезвычайно медленно.

С другой стороны, отмеченное ранее свойство конечности множества допустимых альтернатив для стандартной задачи целочисленного линейного программирования служит основой невольного оптимизма начинающих аналитиков и программистов, основанного на потенциальной возможности полного перебора всех допустимых значений переменных с целью выбора из них оптимального решения исходной задачи (5.1.6) и (5.1.7). Однако данный способ, являясь наиболее простым по своей идее и реализации, вовсе не гарантирует нахождения оптимального решения практических задач за приемлемое время даже на компьютерах с высоким быстродействием.

Таким образом, задачи целочисленного линейного программирования требуют разработки специальных алгоритмических методов для своего решения, которые учитывают специфические особенности и дискретный характер множества допустимых решений. Из алгоритмических методов, обеспечивающих нахождение точного решения задачи (5.1.6) и (5.1.7), наибольшее распространение получили метод ветвей и границ, метод динамического программирования и метод отсечений Гомори. При этом отдельные задачи целочисленного линейного программирования с двумя или тремя переменными могут быть решены графическим способом, аналогично общему случаю задач линейного программирования *(см. главу 4)*.

Применительно к решению практических задач целочисленного линейного программирования с большим числом переменных и ограничений были разработаны также специальные алгоритмы нахождения приближенного целочисленного решения, которые позволяют находить одно или несколько локально-оптимальных решений. Однако использование приближенных методов оправдано лишь тогда, когда по тем или иным причинам нахождение точного решения соответствующих задач оказывается невозможным.

Несмотря на то, что к настоящему времени разработано несколько специальных методов решения целочисленных задач, ни один из них не обеспечивает желаемой эффективности соответствующих вычислительных процедур при

увеличении размерности задачи. Таким образом, в отличие от задач линейного программирования, время решения которых относительно невелико, реализация методов и алгоритмов решения задач целочисленного линейного программирования в ряде случаев весьма затруднительна.

Программа MS Excel позволяет находить точное решение задач целочисленного линейного программирования. При этом общий порядок подготовки и решения полностью аналогичен рассмотренному ранее процессу решения задач линейного программирования. Для оценки точности получаемых решений можно сравнить полученные различными способами оптимальные решения отдельных практических задач. Именно с этой целью в данной главе приводится описание двух способов решения задач о рюкзаке и задачи о производстве часов.

5.2. Задача о рюкзаке

Содержательная постановка задачи о рюкзаке приводится в *разд. 1.2.7*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи оптимизации в форме математической модели целочисленного линейного программирования.

5.2.1. Математическая постановка одномерной задачи о рюкзаке

В общем случае переносимый в рюкзаке груз может включать n видов предметов, при этом каждый предмет вида $i \in \{1, 2, ..., n\}$ обладает массой a_i, например, в *кг*. Для каждого вида предмета турист, исходя из своих субъективных предпочтений, определяет его индивидуальную ценность c_i во время перехода. При этом общая масса всех предметов ограничена емкостью рюкзака и составляет b *кг*.

Требуется определить количество предметов каждого вида, которые следует положить в рюкзак, чтобы суммарная ценность предметов была максимальной, а их общая масса не превысила допустимого фиксированного значения.

Введем в рассмотрение следующие целочисленные переменные: x_i — количество предметов i-го вида, которые следует положить в рюкзак. Тогда в общем случае математическая постановка задачи о рюкзаке может быть сформулирована следующим образом.

$$c_1 x_1 + c_2 x_2 + ... + c_n x_n \to \max_{x \in \Delta_\beta}, \qquad (5.2.1)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} a_1x_1 + a_2x_2 + ... + a_nx_n \le b; \\ x_1, x_2, ..., x_n \ge 0; \\ x_1, x_2, ..., x_n - \text{целые числа.} \end{cases} \quad (5.2.2)$$

Задача (5.2.1) и (5.2.2) называется также *одномерной* задачей о рюкзаке. В случае введения в рассмотрение второй характеристики видов продуктов, например, их геометрического объема, в постановку задачи (5.2.1) и (5.2.2) следует ввести дополнительное ограничение на общий объем продуктов, которые следует положить в рюкзак. Соответствующая задача получила название *двумерной* задачи о рюкзаке. В общем случае аналогичным образом может быть получено ее обобщение на m типов характеристик продуктов. Соответствующая задача называется *многомерной задачей о рюкзаке*.

Примечание

Если по условиям задачи каждый из видов продуктов рассматривается в единственном экземпляре, а проблема туриста заключается в ответе на вопрос: "Брать или не брать…", то соответствующая задача о рюкзаке относится к классу задач булева программирования. Особенности постановки и решения последних рассматриваются в *главе 6*. Известны и другие разновидности задачи о рюкзаке, например, с ограниченными сверху значениями переменных, когда для задачи (5.2.1) и (5.2.2) вводится одно или несколько дополнительных ограничений вида: $x_i \le k_i$, где k_i — некоторое положительное целое число для всех или части исходных переменных задачи x_i. Как будет видно из дальнейшего изложения, подобные дополнительные ограничения не оказывает принципиального влияния на процесс решения соответствующих задач с помощью программы MS Excel.

5.2.2. Решение одномерной задачи о рюкзаке с помощью программы MS Excel

Для решения задачи о рюкзаке с помощью программы MS Excel необходимо задать конкретные значения параметрам исходной задачи. Для определенности предположим, что в качестве исходных видов предметов рассматриваются продукты питания: хлеб, сухари, тушеная говядина, лосось в масле, сыр, сушеные бананы, сахар рафинад ($n = 7$), а в качестве их характеристики — масса продукта в *кг* ($m = 1$). Субъективная ценность для туриста 1 *экземпляра* каждого из продуктов следующая: $c_1 = 20$, $c_2 = 30$, $c_3 = 60$, $c_4 = 30$, $c_5 = 40$,

$c_6 = 20$, $c_7 = 50$. Масса 1 *экземпляра* каждого из продуктов следующая: $a_1 = 0{,}6$ *кг*, $a_2 = 0{,}5$ *кг*, $a_3 = 0{,}3$ *кг*, $a_4 = 0{,}2$ *кг*, $a_5 = 0{,}5$ *кг*, $a_6 = 0{,}1$ *кг*, $a_7 = 0{,}4$ *кг*. Предполагается, что общая масса продуктов в рюкзаке не должна превышать $b = 7$ *кг*.

Исходными переменными математической модели одномерной задачи о рюкзаке являются: x_i — количество экземпляров продуктов i-го вида, которые следует положить в рюкзак. Тогда математическая постановка рассматриваемой индивидуальной задачи о рюкзаке может быть записана в следующем виде.

$$20x_1 + 30x_2 + 60x_3 + 30x_4 + 40x_5 + 20x_6 + 50x_7 \to \max_{x \in \Delta_\beta}, \tag{5.2.3}$$

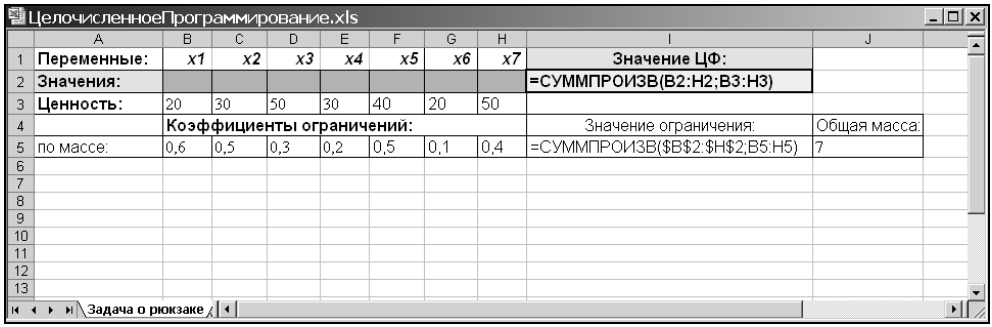
где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} 0{,}6x_1 + 0{,}5x_2 + 0{,}3x_3 + 0{,}2x_4 + 0{,}5x_5 + 0{,}1x_6 + 0{,}4x_7 \le 7; \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \ge 0; \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7 - \text{целые числа}. \end{cases} \tag{5.2.4}$$

Для решения данной задачи с помощью программы MS Excel создадим новую книгу с именем Целочисленное Программирование и изменим имя ее первого рабочего листа на Задача о рюкзаке. Для решения поставленной задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:I1**, **A2:A5**, **B4**, **I4**, **J4**. Следует отметить, что конкретное содержание этих надписей не оказывает никакого влияния на решение рассматриваемой задачи линейного программирования.
2. В ячейки **B3:H3** введем значения коэффициентов целевой функции: $c_1 = 20$, $c_2 = 30$, $c_3 = 60$, $c_4 = 30$, $c_5 = 40$, $c_6 = 20$, $c_7 = 50$.
3. В ячейку **I2** введем формулу: `=СУММПРОИЗВ(B2:H2; B3:H3)`, которая представляет собой целевую функцию (5.2.3).
4. В ячейки **B5:H5** введем значения коэффициентов первого ограничения (5.2.4): $a_1 = 0{,}6$, $a_2 = 0{,}5$, $a_3 = 0{,}3$, $a_4 = 0{,}2$, $a_5 = 0{,}5$, $a_6 = 0{,}1$, $a_7 = 0{,}4$.
5. В ячейку **J5** введем значение правой части первого ограничения, соответствующего максимальной общей массе продуктов в рюкзаке, $b = 7$.
6. В ячейку **I5** введем формулу: `=СУММПРОИЗВ($B$2:$H$2; B5:H5)`, которая представляет собой левую часть первого ограничения (5.2.4).

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения одномерной задачи о рюкзаке имеет следующий вид (рис. 5.1).



ЦелочисленноеПрограммирование.xls

	A	B	C	D	E	F	G	H	I	J
1	Переменные:	x1	x2	x3	x4	x5	x6	x7	Значение ЦФ:	
2	Значения:								=СУММПРОИЗВ(B2:H2;B3:H3)	
3	Ценность:	20	30	50	30	40	20	50		
4		Коэффициенты ограничений:							Значение ограничения:	Общая масса:
5	по массе:	0,6	0,5	0,3	0,2	0,5	0,1	0,4	=СУММПРОИЗВ($B$2:$H$2;B5:H5)	7
6										
7										
8										
9										
10										
11										
12										
13										

Задача о рюкзаке

Рис. 5.1. Исходные данные для решения одномерной задачи о рюкзаке

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$I$2`.
2. Для группы **Равной:** выбрать вариант поиска решения **— максимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$H$2`.
4. Добавить первое ограничение для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$I$5**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения выбрать ячейку **$J$5**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Добавить ограничение на неотрицательность значений переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;

- в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$H$2**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
- в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";
- в качестве значения правой части ограничения в поле с именем **Ограничение:** ввести значение 0;
- для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

6. Добавить ограничение на целочисленность значений переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$H$2**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**цел**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**целое**";
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
7. Дополнительные параметры мастера поиска решения можно оставить без изменения либо выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения представлен на рис. 5.2.

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 5.3).

Результатом решения задачи о рюкзаке являются найденные оптимальные значения переменных: $x_1 = 0$, $x_2 = 0$, $x_3 = 23$, $x_4 = 0$, $x_5 = 0$, $x_6 = 1$, $x_7 = 0$, которым соответствует значение целевой функции: $f_{opt} = 1400$. При этом общая масса продуктов будет равна заданному максимальному значению продуктов в рюкзаке.

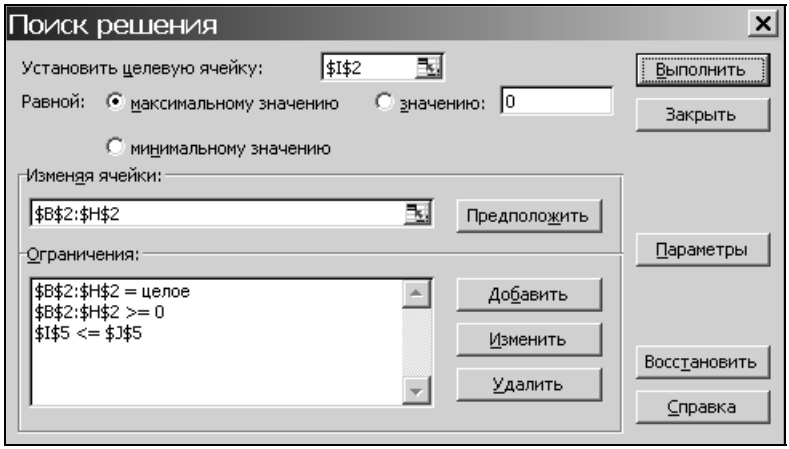


Рис. 5.2. Ограничения на значения переменных и параметры мастера поиска решения для одномерной задачи о рюкзаке

ЦелочисленноеПрограммирование

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ:**	
2	**Значения:**	0,000	0,000	23,000	0,000	0,000	1,000	0,000	1400,000	
3	**Ценность:**	20	30	60	30	40	20	50		
4		**Коэффициенты ограничений:**							Значения огр-ний:	Общая масса:
5	по массе:	0,6	0,5	0,3	0,2	0,5	0,1	0,4	7,000	7

Задача о рюкзаке / Лист2 / Лист3

Рис. 5.3. Результат количественного решения одномерной задачи о рюкзаке

Примечание

При выполнении расчетов для ячеек **B2:I2** был выбран числовой формат с тремя знаками после запятой, что вовсе не является обязательным. Это сделано для дополнительного контроля появления возможных недоразумений в работе программы MS Excel, связанных с округлением искомых значений переменных задачи, что было бы, по меньшей мере, странным.

Анализ найденного решения показывает, что с точки зрения заданных значений субъективной ценности продуктов туристу в поход следует взять 23 *банки* тушеной говядины и 1 *пачку* сушеных бананов. В этом случае содержательный анализ результатов задачи о рюкзаке выявляет недостаток

рассмотренной математической модели, связанный с формированием слишком однообразного состава продуктов, которые следует взять в поход.

Для преодоления этого недостатка выполним коррекцию исходной математической модели (5.2.3) и (5.2.4). А именно ограничим сверху допустимые значения каждой из переменных некоторыми разумными величинами, например, $k_1 = 3$, $k_2 = 5$, $k_3 = 10$, $k_4 = 10$, $k_5 = 7$, $k_6 = 10$, $k_7 = 5$. Соответствующая математическая модель индивидуальной одномерной задачи о рюкзаке с ограниченными сверху значениями переменных будет иметь следующий вид:

$$20x_1 + 30x_2 + 60x_3 + 30x_4 + 40x_5 + 20x_6 + 50x_7 \to \max_{x \in \Delta_\beta}, \tag{5.2.5}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} 0{,}6x_1 + 0{,}5x_2 + 0{,}3x_3 + 0{,}2x_4 + 0{,}5x_5 + 0{,}1x_6 + 0{,}4x_7 \le 7; \\ 0 \le x_1 \le 3,\ 0 \le x_2 \le 5,\ 0 \le x_3 \le 10,\ 0 \le x_4 \le 10; \\ 0 \le x_5 \le 7,\ 0 \le x_6 \le 10,\ 0 \le x_7 \le 5; \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7 - \text{целые числа} \end{cases} \tag{5.2.6}$$

Поскольку целевая функция и часть ограничений модифицированной задачи о рюкзаке (5.2.5) и (5.2.6) совпадает с соответствующей целевой функцией и ограничениями исходной задачи о рюкзаке (5.2.3) и (5.2.4), для решения модифицированной задачи можно воспользоваться исходными данными, созданными для решения предыдущей задачи. Для этого достаточно на листе Задача о рюкзаке вызвать мастер поиска решения и, не изменяя сохраненных параметров поиска решения, добавить 7 дополнительных ограничений на отдельные значения каждой из переменных.

Общий вид диалогового окна спецификации параметров мастера поиска решения модифицированной одномерной задачи о рюкзаке после добавления дополнительных ограничений будет иметь следующий вид (рис. 5.4).

После задания дополнительных ограничений можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет вид, представленный на рис. 5.5.

Результатом решения задачи о рюкзаке являются новые оптимальные значения переменных: $x_1 = 0$, $x_2 = 0$, $x_3 = 10$, $x_4 = 9$, $x_5 = 0$, $x_6 = 10$, $x_7 = 3$, которым соответствует значение целевой функции: $f_{opt} = 1220$. При этом общая масса продуктов также будет равна заданному максимальному значению продуктов в рюкзаке.

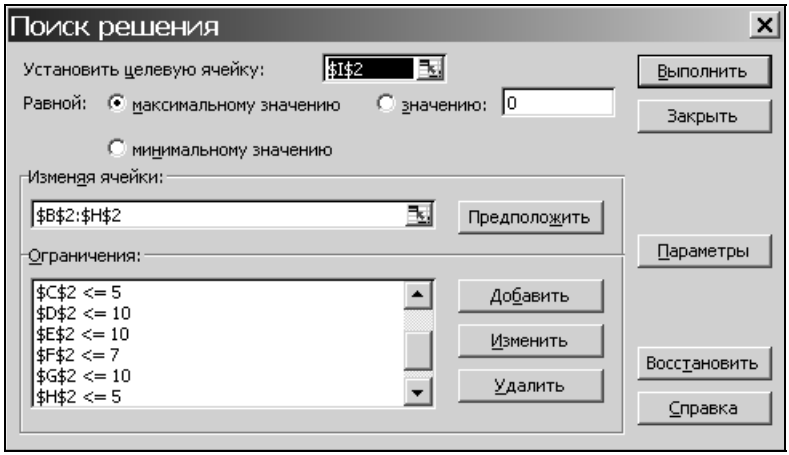


Рис. 5.4. Дополнительные ограничения на значения переменных мастера поиска решения для модифицированной одномерной задачи о рюкзаке

ЦелочисленноеПрограммирование

	A	B	C	D	E	F	G	H	I	J	K
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ:**		
2	**Значения:**	0,000	0,000	10,000	9,000	0,000	10,000	3,000	1220,000		
3	**Ценность:**	20	30	60	30	40	20	50			
4		**Коэффициенты ограничений:**							Значения огр-ний:	Общая масса:	
5	по массе:	0,6	0,5	0,3	0,2	0,5	0,1	0,4	7,000	7	
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											

Задача о рюкзаке / Лист2 / Лист3

Рис. 5.5. Результат количественного решения модифицированной одномерной задачи о рюкзаке

Анализ найденного решения показывает, что с точки зрения заданных значений субъективной ценности продуктов и дополнительных ограничений на их количество туристу в поход следует взять 10 *банок* тушеной говядины, 9 *банок* лосося в масле, 10 *пачек* сушеных бананов и 3 *пачки* сахара рафинада. Легко заметить, что в этом случае сформирован менее однообразный состав продуктов, чем в результате решения исходной задачи о рюкзаке.

Примечание

Внимательный читатель может заметить, что среди продуктов отсутствует необходимый для отечественных туристов хлеб, который вряд ли смогут заменить даже сушеные бананы, особенно при употреблении тушеной говядины

и лосося в масле. Для учета дополнительного требования на то, чтобы взять в поход хотя бы 1 *шт* хлеба, в модифицированную задачу о рюкзаке (5.2.5) и (5.2.6) следует включить дополнительное ограничение вида: $x_1 \geq 1$ или модифицировать существующее ограничение: $0 \leq x_1 \leq 3$ на $1 \leq x_1 \leq 3$. Выполнить поиск решения соответствующей задачи с помощью программы MS Excel предлагается внимательному читателю самостоятельно в качестве упражнения.

5.2.3. Аналитическое решение одномерной задачи о рюкзаке

Хотя исходная одномерная задача о рюкзаке (5.2.1) и (5.2.2) содержит всего одно ограничение и представляется несложной по своей структуре, она не может быть решена аналитически в общем случае. В то же время существует очевидная идея нахождения некоторого допустимого решения, достаточно близкого к оптимальному, а в отдельных случаях, — и совпадающего с оптимальным решением одномерной задачи о рюкзаке. Поскольку данный способ в общем случае не гарантирует получения глобально оптимального решения, его можно отнести к приближенным методам решения задач оптимизации соответствующего класса.

Сущность данного метода заключается в следующем. Для всех переменных $x_i (\forall i \in \{1, 2, ..., n\})$ одномерной задачи о рюкзаке (5.2.1) и (5.2.2) рассмотрим арифметические отношения вида c_i / a_i и выберем из них отношение с максимальным значением, пусть это будет c_r / a_r. После этого можно сразу определить допустимое значение одной из переменных x_r, а именно той, которой соответствует данное максимальное отношение. При этом допустимое значение этой переменной x_r равно целой части от деления b на c_r / a_r. Или в математической символике: $x_r = c_r \cdot \lfloor b / a_r \rfloor$, где через $\lfloor b / a_r \rfloor$ обозначена целая часть отношения b / a_r. После этого как переменная x_r, так и соответствующее ей отношение c_r / a_r исключаются из дальнейшего рассмотрения.

Очевидно, если b / a_r — целое, то на этом дальнейшие вычисления могут быть закончены. Если же это не так, то расчеты следует продолжить. При этом разность $b - \lfloor b / a_r \rfloor$ принимается за новое значение правой части первого ограничения (5.2.2): $b' = b - \lfloor b / a_r \rfloor \cdot a_r$, которое используется для дальнейшего решения задачи. Далее из оставшихся отношений $c_i / a_i \cdot (\forall i \in \{1, 2, ..., n\}), i \neq r$ снова выбирается отношение с максимальным значением, и вся процедура расчета допустимого значения следующей переменной повторяется. Поскольку общее число переменных исходной одно-

мерной задачи о рюкзаке является конечным, данная итеративная процедура в самой худшем случае потребует n итераций, где n — количество переменных задачи.

Проиллюстрируем применение описанной процедуры для решения индивидуальной одномерной задачи о рюкзаке (5.2.3) и (5.2.4). Для этого рассмотрим отношения:

$$c_1 / a_1 = 20/0{,}6;\ c_2 / a_2 = 30/0{,}5;\ c_3 / a_3 = 60/0{,}3;\ c_4 / a_4 = 30/0{,}2;$$

$$c_5 / a_5 = 40/0{,}5;\ c_6 / a_6 = 20/0{,}1;\ c_7 / a_7 = 50/0{,}4.$$

Из них максимальное значение имеют сразу 2 отношения: $c_3 / a_3 = 60/0{,}3$ и $c_6 / a_6 = 20/0{,}1$, равные 200. Выберем первое из них, которому соответствует переменная x_3. Ее допустимое значение будет равно целому от деления 7 на 0,3, т. е. $x_3 = 23$. При этом остаток от деления равен 1/3, а значит, процедуру расчета допустимых значений переменных можно продолжить.

Новое значение правой части первого ограничения будет равно: $b' = 7 - 23 \cdot 0{,}3 = 0{,}1$. Поскольку оно не равно 0, дальнейшие вычисления следует продолжить. Исключая из рассмотрения отношение c_3 / a_3, найдем из оставшихся отношение с максимальным значением — это $c_6 / a_6 = 200$. Этому отношению соответствует переменная x_6. Ее допустимое значение будет равно целому от деления 0,1 на 0,1, т. е. $x_3 = 1$. Поскольку в этом случае остаток от деления равен 0, то процедуру расчета значений оптимальных переменных следует закончить.

В результате выполненных действий аналитически получено допустимое решение одномерной задачи о рюкзаке (5.2.3) и (5.2.4), равное: $x_1 = 0$, $x_2 = 0$, $x_3 = 23$, $x_4 = 0$, $x_5 = 0$, $x_6 = 1$, $x_7 = 0$, которому соответствует значение целевой функции: $f_{opt} = 1400$. Это решение полностью совпадает с найденным решением с помощью программы MS Excel.

Заметим, что при аналитическом решении задачи о рюкзаке на первой итерации была выбрана переменная x_3. Проверим, изменится ли решение исходной задачи, если будет выбрана переменная x_6, которой также соответствует максимальное отношение со значением 200. Ее оптимальное значение будет равно целому от деления 7 на 0,1, т. е. $x_6 = 70$. При этом остаток от деления равен 0, а значит, процедуру расчета значений оптимальных переменных следует завершить. При этом будет получено второе оптимальное решение: $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 70$, $x_7 = 0$, которому также соответствует значение целевой функции: $f_{opt} = 1400$. Это решение программой MS Excel не найдено.

Примечание

В действительности индивидуальная задача о рюкзаке (5.2.3) и (5.2.4) имеет несколько оптимальных решений, которые могут быть получены из решения уравнения: $0{,}3x_3 + 0{,}1x_6 = 7$ в целых неотрицательных числах. После несложных преобразований, данное уравнение можно записать в виде: $x_3 = (70 - x_6)/3$, которое на множестве неотрицательных целых чисел имеет 24 решения. Для нахождения этих решений необходимо рассмотреть ряд целых чисел: 0, 3, 6,…, 69, которые являются допустимыми значениями числителя, кратные 3 и не превосходят 70. При этом первое оптимальное решение будет соответствовать значению числителя 0 : $x_3 = 0$, $x_6 = 70$, которому соответствует значение целевой функции: $f_{opt} = 1400$. Это решение было аналитически получено ранее. Второе оптимальное решение будет соответствовать значению числителя 3 : $x_3 = 1$, $x_6 = 67$, которому также соответствует значение целевой функции: $f_{opt} = 60 + 67 \cdot 20 = 1400$. Это решение не было получено ни аналитическим способом, ни с помощью программы MS Excel. Аналогично могут быть получены остальные оптимальные решения рассматриваемой задачи, что предлагается читателям сделать самостоятельно в качестве упражнения.

Таким образом, при решении задачи о рюкзаке программа MS Excel позволяет найти только одно из оптимальных решений. Если же их несколько, то этот факт может остаться неизвестным для аналитика, использующего программу MS Excel. В этом случае может потребоваться выполнение дополнительных расчетов с использованием других программных инструментов или вычислительных методов, специально ориентированных на решение задач целочисленного линейного программирования.

5.3. Задача об изготовлении часов

Задача об изготовлении часов является частным случаем задачи производственного планирования, которая отличается простотой постановки и служит, главным образом, для иллюстрации графического способа решения задач целочисленного линейного программирования.

5.3.1. Математическая постановка задачи об изготовлении часов

Поскольку графический способ решения задач целочисленного линейного программирования на плоскости может быть применен только для задач с двумя переменными, рассмотрим математическую постановку индивидуальной задачи об изготовлении двух видов часов.

Некоторое предприятие выпускает две марки элитных часов ($n = 2$), которые имеют условные названия: "Банкир" и "Президент". Для изготовления часов

"Банкир" требуется 50 *г* золота и 30 *г* платины, а для изготовления часов "Президент" требуется 40 *г* золота и 50 *г* платины ($m = 2$). Предприятие продает часы "Банкир" по цене \$1500 за 1 *шт*, а часы "Президент" — по цене \$2100 за 1 *шт*. Запасы драгоценных металлов на складе предприятия ограничены следующими значениями: золота — 1,5 *кг* и платины — 1,8 *кг*. Требуется определить оптимальную партию часов "Банкир" и "Президент", которую может изготовить предприятие с целью получения максимальной прибыли.

Исходными переменными математической модели задачи об изготовлении часов являются: x_1 — количество часов марки "Банкир" и x_2 — количество часов марки "Президент". Тогда математическая постановка рассматриваемой индивидуальной задачи об изготовлении часов может быть записана в следующем виде.

$$1500x_1 + 2100x_2 \to \max_{x \in \Delta_\beta}, \tag{5.3.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} 50x_1 + 40x_2 \le 1500; \\ 30x_1 + 50x_2 \le 1800; \\ x_1, x_2 \ge 0,\ x_1, x_2 - \text{целые числа.} \end{cases} \tag{5.3.2}$$

С учетом введенных обозначений левые части ограничений равны: $b_1 = 1500$ и $b_2 = 1800$, коэффициенты целевой функции равны: $c_1 = 1500$ и $c_2 = 2100$.

5.3.2. Графическое решение задачи об изготовлении часов

Для графического решения данной задачи предварительно необходимо построить на плоскости множество допустимых альтернатив, для чего воспользуемся программой MS Excel.

Для графического решения задачи о производстве красок с помощью программы MS Excel создадим новый рабочий лист с именем Задача о часах. Далее необходимо построить на плоскости диаграмму, содержащую изображение области допустимых альтернатив, соответствующей системе ограничений (5.3.2) и целевой функции (5.3.1). При этом следует заметить, что область допустимых альтернатив будет состоять из дискретных точек с целочисленными координатами внутри области, ограниченной прямыми линиями, которым соответствуют выполнение равенств для всех ограничений.

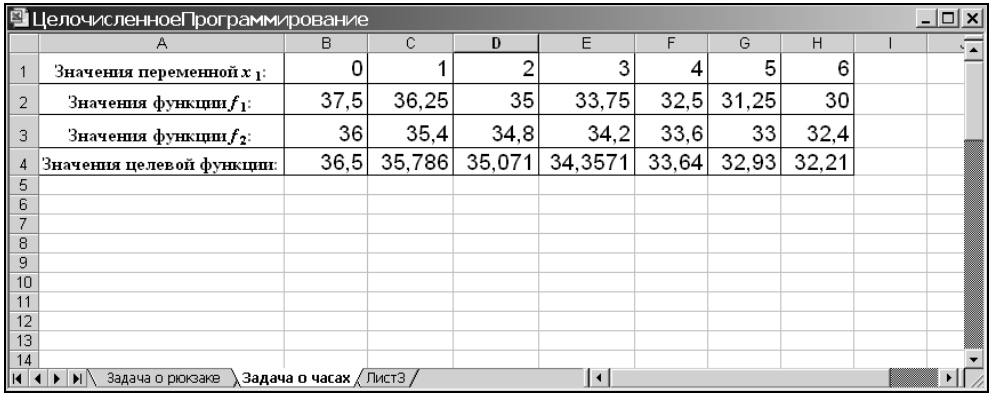
Для построения области допустимых альтернатив следует воспользоваться методикой построения области допустимых решений задачи линейного программирования, рассмотренной в *разд. 4.3.2*. Применительно к задаче об изготовлении часов (5.3.1) и (5.3.2) на одной диаграмме следует построить 3 графика линейных функций. При этом в качестве исходных функций для построения области допустимых альтернатив необходимо использовать 3 следующих функции: $y_1 = 1500/40 - (50/40) \cdot x_1 = 150/4 - (5/4) \cdot x_1$ и $y_2 = 1800/50 - (30/50) \cdot x_1 = 36 - (3/5) \cdot x_1$, где в качестве зависимой переменной y с соответствующим индексом используется вторая переменная данной задачи. В качестве третьей функции следует использовать целевую линейную функцию (5.3.1).

Для подготовки исходных данных для последующего построения области допустимых альтернатив вблизи точки экстремума выполним следующие действия:

1. На рабочем листе в ячейку **A1** введем текст `Значения переменной x1:` , в ячейку **A2** введем текст `Значения функции f1:` , в ячейку **A3** введем текст `Значения функции f2:` и, наконец, в ячейку **A4** введем текст `Значения целевой функции:`.
2. В диапазон ячеек **B1:H1** методом автозаполнения ввести значения от 0 до 6 для независимой переменной, которые принимает x_1 вблизи точки экстремума.
3. Далее в ячейку **B2** введем формулу: `=150/4-(5/4)*B1`, которая соответствует первому ограничению. Скопируем эту формулу в ячейки **C2:H2**.
4. Далее в ячейку **B3** введем формулу: `=36-(3/5)*B1`, которая соответствует второму ограничению. Скопируем эту формулу в ячейки **C3:H3**.
5. Наконец, в ячейку **B4** введем формулу: `=36,5-(5/7)*B1`, которая соответствует некоторому значению целевой функции. Скопируем эту формулу в ячейки **C4:H4**. Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 5.6).

Для построения графиков необходимо воспользоваться мастером диаграмм, который может быть вызван с помощью кнопки стандартной панели инструментов или операции главного меню: **Вставка** | **Диаграмма**. На первом шаге построения диаграммы необходимо выбрать тип диаграммы и ее вид. С этой целью следует выделить на вкладке **Стандартные** в левом списке тип диаграммы **График**, а в правом списке с графическими миниатюрами — первый вид графика, который отражает развитие процесса во времени или по категориям. После выбора типа и разновидности диаграммы следует нажать

кнопку **Далее** и перейти ко второму шагу построения диаграммы с помощью мастера диаграмм.



	A	B	C	D	E	F	G	H	I
1	Значения переменной x_1:	0	1	2	3	4	5	6	
2	Значения функции f_1:	37,5	36,25	35	33,75	32,5	31,25	30	
3	Значения функции f_2:	36	35,4	34,8	34,2	33,6	33	32,4	
4	Значения целевой функции:	36,5	35,786	35,071	34,3571	33,64	32,93	32,21	
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									

Рис. 5.6. Исходные данные для построения графиков функций линейных ограничений и целевой функции задачи о часах

На втором шаге построения диаграммы необходимо выбрать ячейки с данными, которые должны быть отображены на четырех графиках. Применительно к рассматриваемому примеру — это значения функций, которые содержатся в диапазоне ячеек **B2:H4**. Для указания этих значений следует установить переключатель **Ряды в:** в положение **строках**. После этого выполнить щелчок на кнопке, расположенной в правой части поля ввода с именем **Диапазон**, и выделить диапазон ячеек **B2:H4**. Далее на этом же шаге работы мастера диаграмм следует задать подписи по горизонтальной оси. С этой целью необходимо перейти на вкладку **Ряд** и выполнить щелчок на кнопке, расположенной в правой части поля ввода с именем **Подписи оси *X***. Для указания соответствующего источника данных следует на рабочем листе с помощью мыши или клавиатуры выделить диапазон значений функции **B1:H1**.

После редактирования свойств диаграммы на третьем и четвертом шагах работы мастера будет построена диаграмма, содержащая графики трех линейных функций вблизи точки экстремума (рис. 5.7).

Множество допустимых альтернатив задачи о часах будет состоять из точек с целочисленными координатами, содержащимися внутри области, образуемой пересечением четырех полуплоскостей, каждая из которых соответствует отдельному ограничению (5.3.2). На построенной диаграмме этим допустимым решениям соответствуют точки пересечения сетки графика. В то же время область допустимых альтернатив задачи о часах без условия целочисленности представляет собой выпуклый многогранник на плоскости. При этом оп-

тимальное решение задачи о часах без учета ограничения на целочисленность переменных, как видно из рис. 5.7, достигается в точке пересечения двух линий, соответствующих первым двум ограничениям исходной задачи. Именно эта точка, не являясь целочисленной, располагается ближе остальных точек многогранника к построенной линии уровня целевой функции.

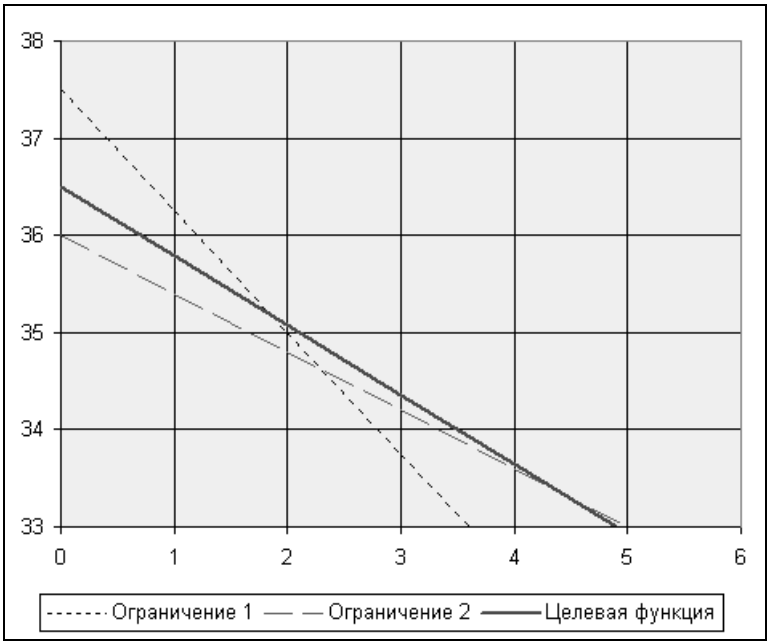


Рис. 5.7. Результат построения графиков функций ограничений и целевой функции для задачи о часах

Для более наглядного представления линии целевой функции изменим содержащуюся в ячейке **B4** формулу на формулу: `=36-(5/7)*B1`, которую скопируем в ячейки **C4:H4**. Как нетрудно заметить, новая формула означает параллельный сдвиг прямой линии, соответствующей целевой функции задачи о часах. В результате этого изменится построенная диаграмма, которая будет иметь следующий вид (рис. 5.8).

Из данной диаграммы видно, что оптимальное решение задачи о часах достигается в точке A, координаты которой равны: $x_1 = 0$ и $x_2 = 36$. При этом оптимальное значение целевой функции равно: $f_{opt} = \$75\ 600$.

Для сравнения заметим, что оптимальное решение задачи о часах без ограничения целочисленности достигается в точке B, координаты которой могут быть найдены из решения двух линейных уравнений *(см. разд.4.3.2)* и равны: $x_1 = 30/13 \cong 2{,}308$ и $x_2 = 450/13 \cong 34{,}615$. Этому решению соответствует зна-

чение целевой функции: $f_B \cong \$76\,154$. Округление данного решения до ближайших целых чисел, соответствующих допустимому решению исходной задачи о часах, приводит к точке C, координаты которой равны: $x_1 = 2$ и $x_2 = 34$. При этом значение целевой функции в точке C равно: $f_C = \$74\,400$, что меньше f_{op} на \$1200.

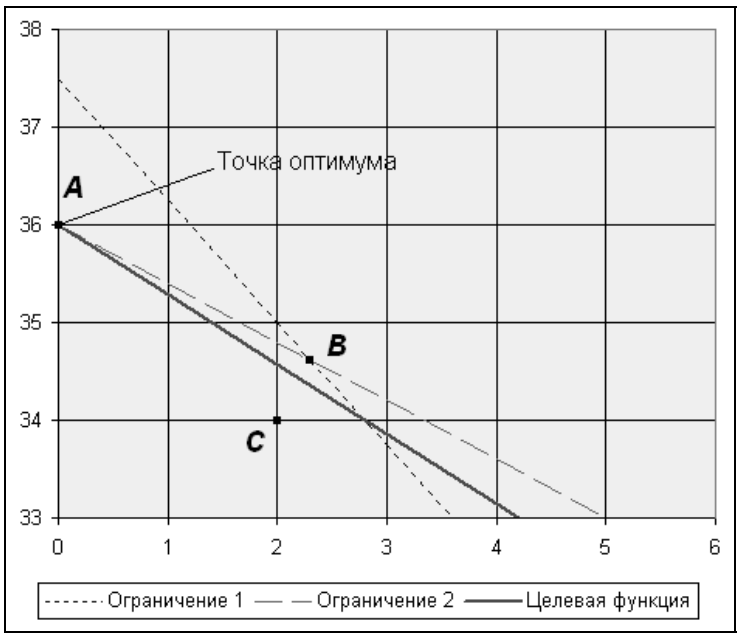


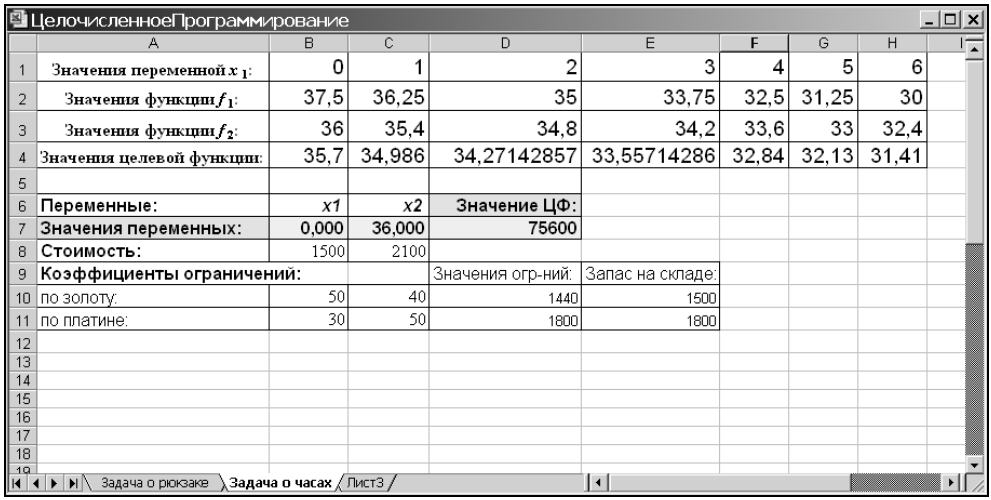
Рис. 5.8. Графическое решение задачи о часах

Данный пример служит наглядным подтверждением того факта, что округление оптимального решения задачи линейного программирования без ограничения целочисленности до ближайшего допустимого целочисленного решения вовсе не гарантирует получение оптимального решения исходной задачи целочисленного линейного программирования и может привести к далеко не оптимальным рекомендациям.

Таким образом, оптимальная партия изготовления часов, обеспечивающая максимум общей стоимости готовой продукции, должна состоять только из 36 *шт* часов марки "Президент". От изготовления часов марки "Банкир" предприятию следует вовсе отказаться. При этом будет обеспечено максимальное значение стоимости продукции \$75 600. На этом графическое решение задачи о часах может быть закончено.

Проверим найденное решение с помощью программы MS Excel. Для этого воспользуемся рекомендациями, рассмотренными ранее в *разд. 5.2*. Таблицу

с соответствующими исходными данными для удобства можно разместить на этом же рабочем листе. Результаты решения исходной задачи о часах с помощью инструмента поиска решения имеют следующий вид (рис. 5.9).


ЦелочисленноеПрограммирование

	A	B	C	D	E	F	G	H
1	Значения переменной x_1:	0	1	2	3	4	5	6
2	Значения функции f_1:	37,5	36,25	35	33,75	32,5	31,25	30
3	Значения функции f_2:	36	35,4	34,8	34,2	33,6	33	32,4
4	Значения целевой функции:	35,7	34,986	34,27142857	33,55714286	32,84	32,13	31,41
5								
6	Переменные:	x1	x2	Значение ЦФ:				
7	Значения переменных:	0,000	36,000	75600				
8	Стоимость:	1500	2100					
9	Коэффициенты ограничений:			Значения огр-ний:	Запас на складе:			
10	по золоту:	50	40	1440	1500			
11	по платине:	30	50	1800	1800			
12								
13								
14								
15								
16								
17								
18								
19								

Задача о рюкзаке | Задача о часах | Лист3

Рис. 5.9. Результаты количественного решения задачи о часах с помощью программы MS Excel

Анализ полученных результатов решения задачи двумя различными методами показывает их полное совпадение. Дополнительно можно заметить, что запасы платины используются полностью, а 60 *г* золота останутся неиспользованными. Эти результаты служат достаточно веским основанием для вывода о достоверности найденного оптимального решения задачи.

5.4. Задача о планировании перевозок пассажиров

Данная задача является разновидностью типовой задачи планирования производства, которая может быть сформулирована и эффективно решена с помощью модели целочисленного линейного программирования. Сущность задачи о планировании перевозок пассажиров железнодорожным транспортом заключается в следующем.

Между двумя городами установлено железнодорожное сообщение, при этом перевозка пассажиров осуществляется пассажирскими и скорыми поездами. Каждый из видов поездов имеет в своем составе определенное число вагонов различного типа: багажный, почтовый, плацкартный, купейный, мягкий. Известно количество пассажиров, перевозимых в каждом из типов вагонов, а также общее число вагонов каждого типа на станции формирования поездов (табл. 5.1).

Таблица 5.1. *Исходные данные задачи о планировании перевозок пассажиров*

Поезда	Вагоны				
	багаж-ный	почто-вый	плац-картный	купей-ный	мягкий
Скорый	1	1	5	6	3
Пассажирский	1	—	8	4	1
Число пассажиров	—	—	54	36	32
Парк вагонов	12	8	81	70	26

Требуется определить оптимальное число скорых и пассажирских поездов, при которых общее число перевозимых пассажиров будет максимальным.

5.4.1. Математическая постановка задачи о планировании перевозок пассажиров

Исходными переменными математической модели задачи о планировании перевозок пассажиров являются: x_1 — количество сформированных скорых поездов и x_2 — количество сформированных пассажирских поездов. Предварительно следует заметить, что в одном скором поезде может быть перевезено $c_1 = 5 \cdot 54 + 6 \cdot 36 + 3 \cdot 32 = 582$ пассажира, соответственно, в одном пассажирском поезде может быть перевезено $c_2 = 8 \cdot 54 + 4 \cdot 36 + 1 \cdot 32 = 608$ пассажиров. Тогда математическая постановка рассматриваемой индивидуальной задачи о планировании перевозок пассажиров может быть записана в следующем виде:

$$582x_1 + 608x_2 \to \max_{x \in \Delta_\beta}, \tag{5.4.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} x_1 + x_2 \le 12; \\ x_1 \le 8; \\ 5x_1 + 8x_2 \le 81; \\ 3x_1 + x_2 \le 26; \\ 6x_1 + 4x_2 \le 70; \\ x_1, x_2 \ge 0,\ x_1, x_2 - \text{целые числа}. \end{cases} \tag{5.4.2}$$

С учетом введенных обозначений левые части ограничений равны: $b_1 = 12$, $b_2 = 8$, $b_3 = 81$, $b_4 = 26$, $b_5 = 70$, при этом каждое из этих ограничений соответствует отдельным типам вагонов в составах скорых и пассажирских поездов.

5.4.2. Решение задачи о планировании перевозок пассажиров с помощью программы MS Excel

Для решения задачи о планировании перевозок пассажиров с помощью программы MS Excel создадим новый рабочий лист Задача о поездах в книге с именем Целочисленное программирование. Далее необходимо выполнить следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:D1**, **A2:A9**, **B4**, **D4**, **E4**. Следует отметить, что конкретное содержание этих надписей не оказывает никакого влияния на решение рассматриваемой задачи целочисленного линейного программирования.
2. В ячейки **B3:C3** введем значения коэффициентов целевой функции: $c_1 = 582$, $c_2 = 608$.
3. В ячейку **D2** введем формулу: `=СУММПРОИЗВ(B2:C2; B3:C3)`, которая представляет собой целевую функцию (5.4.1).
4. В ячейки **B5:C9** введем значения коэффициентов левых частей первых пяти ограничений (5.4.2), взятых из исходной таблицы (табл. 5.1).
5. В ячейки **E5:E9** введем значения правых частей первых пяти ограничений, (5.4.2), взятых из исходной таблицы (табл. 5.1).
6. В ячейку **D5** введем формулу: `=СУММПРОИЗВ($B$2:$C$2; B5:C5)`, которая представляет собой левую часть первого ограничения (5.4.2).
7. Скопируем формулу из ячейки **D5** в ячейки **D6:D9**.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о планировании перевозок пассажиров имеет следующий вид (рис. 5.10).

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$D$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **максимальному значению**.

ЦелочисленноеПрограммирование.xls

	A	B	C	D	E	F
1	**Переменные:**	***x1***	***x2***	**Значение ЦФ:**		
2	**Значения переменных:**			**=СУММПРОИЗВ(B2:C2;B3:C3)**		
3	**Стоимость:**	582	608			
4	**Коэффициенты ограничений:**			Значения ограничений:	Наличие в парке:	
5	по багажным вагонам:	1	1	=СУММПРОИЗВ(B5:C5;$B$2:$C$2)	12	
6	по почтовым вагонам:	1	0	=СУММПРОИЗВ(B6:C6;$B$2:$C$2)	8	
7	по плацкартным вагонам:	5	8	=СУММПРОИЗВ(B7:C7;$B$2:$C$2)	81	
8	по купейным вагонам:	6	4	=СУММПРОИЗВ(B8:C8;$B$2:$C$2)	70	
9	по мягким вагонам:	3	1	=СУММПРОИЗВ(B9:C9;$B$2:$C$2)	26	
10						
11						
12						
13						

Задача о поездах

Рис. 5.10. Исходные данные для решения задачи о планировании перевозок пассажиров

3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$C$2`.
4. Добавить 5 ограничений для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$D$5**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения выбрать ячейку **$E$5**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Аналогичным образом добавить оставшиеся 4 ограничения, соответствующие ячейкам **D6:E9**.
6. Добавить ограничение на целочисленность значений переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$C$2**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**цел**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**целое**";

- для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

7. В дополнительных параметрах мастера поиска решения выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения представлен на рис. 5.11.

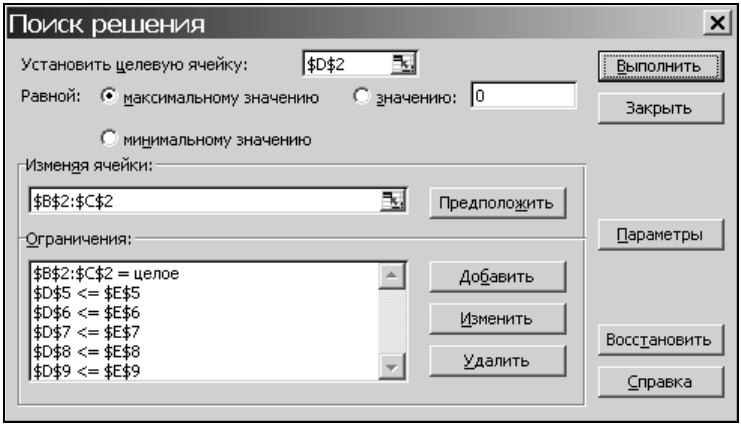


Рис. 5.11. Ограничения на значения переменных и параметры мастера поиска решения для задачи о планировании перевозок пассажиров

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 5.12).

ЦелочисленноеПрограммирование

	A	B	C	D	E	F	G	H
1	**Переменные:**	***x1***	***x2***	**Значение ЦФ:**				
2	**Значения переменных:**	**5,000**	**7,000**	**7166**				
3	**Стоимость:**	582	608					
4	**Коэффициенты ограничений:**			Значения огр-ний:	Наличие в парке:			
5	по багажным вагонам:	1	1	12	12			
6	по почтовым вагонам:	1	0	5	8			
7	по плацкартным вагонам:	5	8	81	81			
8	по купейным вагонам:	6	4	58	70			
9	по мягким вагонам:	3	1	22	26			
10								
11								
12								
13								
14								

Задача о рюкзаке | Задача о часах | **Задача о поездах**

Рис. 5.12. Результат количественного решения задачи о планировании перевозок пассажиров

Результатом решения задачи о планировании перевозок пассажиров являются найденные оптимальные значения переменных: $x_1 = 5$, $x_2 = 7$, которым соответствует значение целевой функции: $f_{opt} = 7166$.

Анализ найденного решения показывает, что из имеющегося парка вагонов для обеспечения максимального количества перевезенных пассажиров следует составить 5 скорых и 7 пассажирских поездов. При этом имеющиеся в парке багажные и плацкартные вагоны будут полностью использованы для формирования поездов, а 3 почтовых, 12 купейных и 4 мягких вагона из парка останутся неиспользованными в этих поездах.

Примечание

Заинтересованным читателям в качестве упражнения предлагается самостоятельно решить рассмотренную задачу о планировании перевозок пассажиров графическим способом, а полученные результаты сравнить. Сделать выводы о точности результатов и времени, затраченном на решение задачи различными способами.

5.5. Задача об изготовлении стержней

Данная задача, с одной стороны, также является разновидностью типовой задачи планирования производства, а с другой стороны, по своему характеру относится к классу задач *геометрического программирования*. Задачи этого класса после некоторых предварительных преобразований могут быть сформулированы и эффективно решены с помощью модели целочисленного линейного программирования.

5.5.1. Содержательная постановка задачи

Сущность задачи об изготовлении стержней заключается в следующем. Производственное предприятие изготавливает металлические стержни трех видов фиксированной длины: 2,9 *м*, 2,1 *м* и 1,5 *м* соответственно. Для изготовления этих стержней поступает партия исходного материала, который также представляет собой металлические стержни длиной 7,4 *м*. Способ изготовления стержней заключается в разрезании исходной заготовки на отрезки заданной длины (рис. 5.13).

Задача состоит в том, чтобы из имеющихся исходных заготовок изготовить 100 комплектов стержней требуемой длины наиболее эффективным способом разрезания исходного материала, при котором на изготовление необходимого количества комплектов стержней потребуется наименьшее количество исходных заготовок.

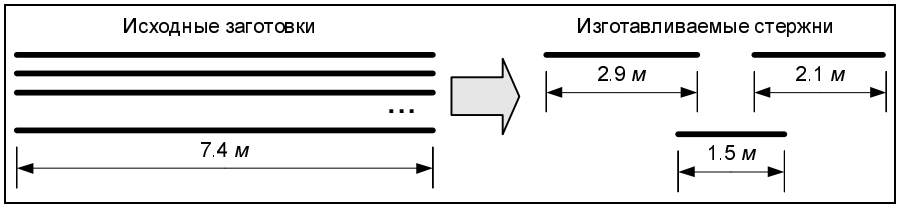


Рис. 5.13. Геометрическая интерпретация задачи об изготовлении стержней

Примечание

Задачи, аналогичные задаче об изготовлении стержней, достаточно часто встречаются на практике. При этом в качестве исходных заготовок могут выступать самые различные материалы, применяемые в промышленном производстве и поступающие от поставщиков в виде целых единиц (например, листы металла, стекла, фанеры, рулоны бумаги, трубы, доски, бревна). При их использовании в качестве заготовок для изготовления других изделий приходится разрезать эти единицы исходных материалов на нужные части требуемых размеров.

Особенность сформулированной задачи заключается в том, что она не может быть непосредственно преобразована в математическую модель. Для этого необходимо выполнить некоторые предварительные действия по рассмотрению способов разрезания исходного материала.

Действительно, из исходного стержня можно получить детали нужной длины различными способами его разрезания. Так, например, из одного исходного стержня можно изготовить один комплект деталей требуемой длины. В этом случае получается остаток длиной: $l_1 = 7{,}4 - (2{,}9 + 2{,}1 + 1{,}5) = 0{,}9$ *м*, который идет в отходы. При этом на изготовление 100 комплектов деталей потребуется 100 исходных стержней, а суммарная длина остатка составит: $L_1 = 0{,}9 \cdot 100 = 90$ *м*.

Можно предложить и другие способы разрезания исходных стержней, например, изготовить из одной заготовки 1 стержень длиной 2,9 *м* и 3 стержня длиной 1,5 *м*. В этом случае остаток вообще будет отсутствовать: $l_2 = 7{,}4 - (2{,}9 + 1{,}5 + 1{,}5 + 1{,}5) = 0$ *м*. Однако данный способ разрезания не может быть применен для всей партии заготовок, поскольку изготовление деталей длиной 2,1 *м* данным способом вообще не предусмотрено.

Таким образом, необходимо рассмотреть все возможные способы разрезания исходных заготовок и уже на этой основе попытаться разработать соответствующую математическую модель задачи об изготовлении стержней.

В общем случае исходный стержень длиной 7,4 *м* может быть разрезан для получения отдельных деталей требуемой длины 6 возможными способами (табл. 5.2).

Таблица 5.2. *Способы разрезания исходной заготовки для изготовления отдельных деталей требуемой длины*

Способы разрезания	**1**	**2**	**3**	**4**	**5**	**6**
Длина получаемых деталей	2,9 + 1,5 + 1,5 + 1,5	2,9 + 2,9 + 1,5	2,1 + 2,1 + 1,5 + 1,5	2,9 + 2,1 + 2,1	2,1 + 1,5 + 1,5 + 1,5	2,9 + 2,1 + 1,5
Расход материала	7.4	7,3	7,2	7,1	6,6	6,5
Остаток	0	0,1	0,2	0,3	0,8	0,9

Тем самым исходная задача преобразуется к следующему виду: требуется определить оптимальное число различных способов разрезания исходных заготовок, при котором будет изготовлено заданное число стержней требуемой длины, а общее количество разрезанных исходных заготовок будет минимальным.

5.5.2. Математическая постановка задачи об изготовлении стержней

Исходными переменными математической модели задачи об изготовлении стержней являются: x_i — количество исходных заготовок, разрезанных i-м способом для изготовления отдельных деталей ($i \in \{1, 2, \ldots, 6\}$).

Тогда математическая постановка рассматриваемой индивидуальной задачи об изготовлении стержней может быть записана в следующем виде:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \to \min_{x \in \Delta_\beta}, \qquad (5.5.1)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} x_1 + 2x_2 + x_4 + x_6 \geq 100; \\ 2x_3 + 2x_4 + x_5 + x_6 \geq 100; \\ 3x_1 + x_2 + 2x_3 + 3x_5 + x_6 \geq 100; \\ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0; \\ x_1, x_2, x_3, x_4, x_5, x_6 - \text{целые числа} \end{cases} \quad (5.5.2)$$

Заметим, что первые три ограничения (5.5.2) соответствуют требованию изготовить 100 деталей каждого из трех установленных размеров. При этом левые части этих ограничений равны: $b_1 = 100$, $b_2 = 100$, $b_3 = 100$. Математическая модель (5.5.1) и (5.5.2) относится к классу задач целочисленного линейного программирования, которая может быть решена с помощью программы MS Excel.

5.5.3. Решение задачи об изготовлении стержней с помощью программы MS Excel

Для решения задачи об изготовлении стержней с помощью программы MS Excel создадим новый рабочий лист с именем Задача о стержнях в книге Целочисленное программирование. Далее необходимо выполнить следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:H1**, **A2:A7**, **B4**, **H4**, **I4**. Следует отметить, что конкретное содержание этих надписей не оказывает никакого влияния на решение рассматриваемой задачи целочисленного линейного программирования.
2. В ячейки **B3:G3** введем значения коэффициентов целевой функции: $c_i = 1$ ($\forall i \in \{1, 2, \ldots, 6\}$).
3. В ячейку **D2** введем формулу: `=СУММПРОИЗВ(B2:G2; B3:G3)`, которая представляет собой целевую функцию (5.5.1).
4. В ячейки **B5:G7** введем значения коэффициентов левых частей первых трех ограничений (5.5.2).
5. В ячейки **I5:I7** введем значения правых частей первых трех ограничений (5.5.2).
6. В ячейку **H5** введем формулу: `=СУММПРОИЗВ($B$2:$G$2; B5:G5)`, которая представляет левую часть первого ограничения (5.5.2).
7. Скопируем формулу из ячейки **H5** в ячейки **H6:H7**.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи об изготовлении стержней представлен на рис. 5.14.



ЦелочисленноеПрограммирование.xls

	A	B	C	D	E	F	G	H	I
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	**Значение ЦФ:**	
2	**Значения:**							=СУММПРОИЗВ(B2:G2;B3:G3)	
3	**Коэффициенты в ЦФ:**	1	1	1	1	1	1		
4		**Коэффициенты ограничений:**						Значения ограничений:	Общее кол-во:
5	по деталям длины 2.9 м:	1	2	0	1	0	1	=СУММПРОИЗВ($B$2:$G$2;B5:G5)	100
6	по деталям длины 2.1 м:	0	0	2	2	1	1	=СУММПРОИЗВ($B$2:$G$2;B6:G6)	100
7	по деталям длины 1.5 м:	3	1	2	0	3	1	=СУММПРОИЗВ($B$2:$G$2;B7:G7)	100
8									
9									
10									
11									
12									

Задача о стержнях

Рис. 5.14. Исходные данные для решения задачи об изготовлении стержней

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$H$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$G$2`.
4. Добавить 3 первых ограничения для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$H$5**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";
   - в качестве значения правой части ограничения выбрать ячейку **$I$5**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

5. Аналогичным образом добавить оставшиеся 2 ограничения, соответствующие ячейкам **H6:H7**.
6. Добавить ограничение на целочисленность значений переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$G$2**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**цел**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**целое**";
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
7. В дополнительных параметрах мастера поиска решения выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения представлен на рис. 5.15.

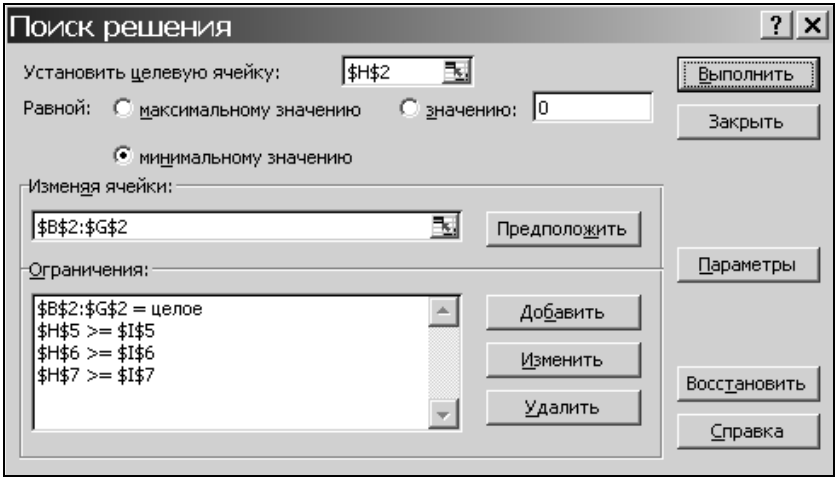
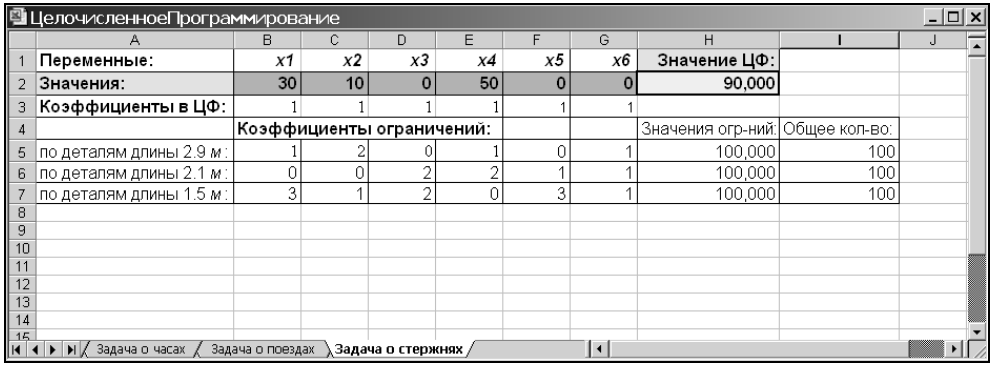


Рис. 5.15. Ограничения на значения переменных и параметры мастера поиска решения для задачи об изготовлении стержней

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 5.16).



ЦелочисленноеПрограммирование

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	**Значение ЦФ:**		
2	**Значения:**	30	10	0	50	0	0	90,000		
3	**Коэффициенты в ЦФ:**	1	1	1	1	1	1			
4		**Коэффициенты ограничений:**						Значения огр-ний:	Общее кол-во:	
5	по деталям длины 2.9 *м*:	1	2	0	1	0	1	100,000	100	
6	по деталям длины 2.1 *м*:	0	0	2	2	1	1	100,000	100	
7	по деталям длины 1.5 *м*:	3	1	2	0	3	1	100,000	100	
8										
9										
10										
11										
12										
13										
14										
15										

Задача о часах | Задача о поездах | **Задача о стержнях**

Рис. 5.16. Результат количественного решения задачи об изготовлении стержней

Результатом решения задачи об изготовлении стержней являются найденные оптимальные значения переменных: $x_1 = 30$, $x_2 = 10$, $x_3 = 0$, $x_4 = 50$, $x_5 = 0$, $x_6 = 0$, которым соответствует значение целевой функции: $f_{opt} = 90$.

Анализ найденного решения показывает, что из имеющихся заготовок для изготовления 100 комплектов деталей требуемой длины следует первым способом разрезать 30 стержней, вторым способом — 10 стержней и четвертым способом — 50 стержней. При этом общее число израсходованных заготовок будет равно 90, что является минимальным из всех возможных вариантов разрезания исходного материала. Для найденного оптимального решения задачи об изготовлении стержней будут отсутствовать лишние изготовленные детали.

5.6. Транспортная задача целочисленного линейного программирования

Содержательная постановка транспортной задачи линейного программирования приводится в *разд. 1.2.4*. В настоящей главе рассматривается ее уточнение и модификация, необходимое для формальной записи условий соответствующей задачи оптимизации в форме модели целочисленного линейного программирования.

5.6.1. Математическая постановка транспортной задачи

Математическая постановка транспортной задачи в форме модели линейного программирования была сформулирована в *разд. 4.5.1*. Если в дополнение к условиям транспортной задачи линейного программирования (4.5.1)—(4.5.5) вводится требование целочисленности всех переменных x_{ij} ($\forall i \in \{1, 2,\ldots, m\}$, $\forall j \in \{1, 2,\ldots, n\}$), то соответствующую транспортную задачу в общем случае следует отнести к классу задач целочисленного линейного программирования. Содержательно целочисленность переменных означает следующее требование: количество транспортируемого продукта или объем перевозок из i-го пункта производства в j-й пункт потребления должны принимать целочисленные значения. Это может иметь место, например, при организации перевозок автомобилями, контейнерами, коробками, ящиками, пачками, упаковками и пр., когда по условиям задачи невозможно или нецелесообразно деление продукта на более мелкие части.

Как и в транспортной задаче линейного программирования, требуется определить оптимальный план перевозок продукта, так чтобы потребность во всех пунктах потребления были удовлетворена, а суммарные затраты на транспортировку всей продукции были минимальными. Тогда, с учетом ранее введенных условных обозначений, в общем случае математическая постановка транспортной задачи в форме модели целочисленного линейного программирования может быть сформулирована следующим образом:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min_{x\in\Delta_\beta}, \tag{5.6.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i, \ \forall i \in \{1, 2,..., m\}; & (5.6.2) \\ \sum_{i=1}^{m} x_{ij} = b_j, \quad \forall j \in \{1, 2,..., n\}; & (5.6.3) \\ \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j; & (5.6.4) \\ x_{ij} \geq 0 \ (\forall i \in \{1, 2,..., m\}, \forall j \in \{1, 2,..., n\}); & (5.6.5) \\ x_{ij} - \text{целые числа } (\forall i \in \{1, 2,..., m\}, \forall j \in \{1, 2,..., n\}). & (5.6.6) \end{cases}$$

Следует заметить, что в отличие от транспортной задачи линейного программирования (4.5.1)—(4.5.5) в математической постановке целочисленной транспортной задачи в виде (5.6.1)—(5.6.6) используется дополнительное ограничение вида (5.6.6). При этом общее число переменных целочисленной транспортной задачи также равно: $m \cdot n$.

Классическая транспортная задача целочисленного линейного программирования также является *сбалансированной* или *закрытой*, т. е. формулируется в виде, когда имеет место равенство общего объема производства рассматриваемого продукта общему объему его потребления. Этому условию соответствует отдельное ограничение (5.6.4). В противном случае, если равенство (5.6.4) не имеет места, то транспортная задача называется *несбалансированной* или *открытой*.

На практике могут встречаться различные модификации целочисленной транспортной задачи. Для иллюстрации особенностей ее решения с помощью программы MS Excel рассмотрим многопродуктовую транспортную задачу целочисленного линейного программирования.

5.6.2. Решение многопродуктовой целочисленной транспортной задачи с помощью программы MS Excel

Рассмотрим индивидуальную транспортную задачу, которая является модификацией задачи оптимального планирования перевозок бензина *(см. разд. 4.5.2)*. Содержательно многопродуктовая целочисленная транспортная задача может быть сформулирована следующим образом.

В качестве транспортируемого продукта рассматривается бензин трех марок: 76, 92 и 95, в качестве пунктов производства — 2 нефтеперерабатывающих завода ($m = 2$), а в качестве пунктов потребления — 4 автозаправочные станции ($n = 4$). Объемы производства бензина следующие: для НПЗ №1: 76 бензин — 100 m, 92 бензин — 60 m, 95 бензин — 60 m; для НПЗ №2: 76 бензин — 80 m, 92 бензин — 100 m, 95 бензин — 60 m. Объемы потребления бензина следующие: на АЗС №1: 76 бензин — 40 m, 92 бензин — 60 m, 95 бензин — 20 m; на АЗС №2: 76 бензин — 60 m, 92 бензин — 40 m, 95 бензин — 40 m; на АЗС №3: 76 бензин — 44 m, 92 бензин — 24 m, 95 бензин — 28 m; на АЗС №4: 76 бензин — 36 m, 92 бензин — 36 m, 95 бензин — 32 m. При этом транспортировка бензина всех марок осуществляется бензовозами грузоподъемностью 4 m. Стоимость транспортировки одного бензовоза с бензином разных марок между НПЗ и АЗС задана в форме следующей таблицы (табл. 5.3).

Таблица 5.3. *Стоимость транспортировки бензина различных марок между НПЗ и АЗС (в тыс. рублей)*

Пункты потребления/ Пункты производства	**Марка бензина**	**АЗС №1**	**АЗС №2**	**АЗС №3**	**АЗС №4**
НПЗ №1	**76**	2	4	5	7
	92	2	6	5	8
	95	3	5	7	11
НПЗ №2	**76**	3	4	6	5
	92	3	5	9	10
	95	5	8	12	7

Для математической постановки рассматриваемой трехпродуктовой транспортной задачи в форме модели целочисленного линейного программирования необходимо выполнить некоторые предварительные преобразования. Во-первых, привести в соответствие объемы производства и потребления бензина с грузоподъемностью транспортных средств. Для этого следует разделить все объемы на 4 *m*. Во-вторых, ввести в рассмотрение 3-индексные переменные: $x_{ijk} \cdot (\forall i \in \{1, 2\}, \forall j \in \{1, 2, 3, 4\}, \forall j \in \{1, 2, 3\})$, каждая из которых будет соответствовать количеству бензовозов, перевозящих бензин *k*-й марки от *i*-го НПЗ к *j*-й АЗС.

Тогда соответствующая математическая постановка рассматриваемой индивидуальной целочисленной транспортной задачи с учетом выполненных предварительных преобразований может быть записана в следующем виде:

$$\begin{aligned} & 2x_{111} + 2x_{112} + 3x_{113} + 4x_{121} + 6x_{122} + 5x_{123} + \\ & + 5x_{131} + 5x_{132} + 7x_{133} + 7x_{141} + 8x_{142} + 11x_{143} + \\ & + 3x_{211} + 3x_{212} + 5x_{213} + 4x_{221} + 5x_{222} + 8x_{223} + \\ & + 6x_{231} + 9x_{232} + 12x_{233} + 5x_{241} + 10x_{242} + 7x_{243} \to \min\limits_{x \in \Delta_\beta}, \end{aligned} \tag{5.6.7}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$
\begin{cases}
x_{111} + x_{121} + x_{131} + x_{141} = 25; \\
x_{112} + x_{122} + x_{132} + x_{142} = 15; \\
x_{113} + x_{123} + x_{133} + x_{143} = 15; \\
x_{211} + x_{221} + x_{231} + x_{241} = 20; \\
x_{212} + x_{222} + x_{232} + x_{242} = 25; \\
x_{213} + x_{223} + x_{233} + x_{243} = 15; \\
x_{111} + x_{211} = 10; \\
x_{112} + x_{212} = 15; \\
x_{113} + x_{213} = 5; \\
x_{121} + x_{221} = 15; \\
x_{122} + x_{222} = 10; \\
x_{123} + x_{223} = 10; \\
x_{131} + x_{231} = 11; \\
x_{132} + x_{232} = 6; \\
x_{133} + x_{233} = 7; \\
x_{141} + x_{241} = 9; \\
x_{142} + x_{242} = 9; \\
x_{143} + x_{243} = 8; \\
x_{ijk} \geq 0 \ (\forall i \in \{1, 2\}, \forall j \in \{1, 2, 3, 4\}, \forall k \in \{1, 2, 3\}); \\
x_{ijk} - \text{целые числа } (\forall i \in \{1, 2\}, \forall j \in \{1, 2, 3, 4\}, \forall k \in \{1, 2, 3\}).
\end{cases}
\tag{5.6.8}
$$

Заметим, что первых 6 ограничений данной задачи соответствуют общему ограничению (5.6.2), следующие 12 ограничений — общему ограничению (5.6.3). При этом общее ограничение (5.6.4), соответствующее требованию сбалансированности транспортной задачи, не входит в математическую модель рассматриваемой индивидуальной задачи. Это вполне допустимо, поскольку непосредственная проверка позволяет установить выполнение общего ограничения (5.6.4), а значит индивидуальная 3-продуктовая транспортная задача (5.6.7) и (5.6.8) является сбалансированной.

Примечание

Как нетрудно заметить, исходная трехпродуктовая транспортная задача (5.6.7) и (5.6.8) распадается на три независимых транспортных задачи, соответствующие трем рассматриваемым маркам бензина. Поскольку задачи транспортировки различных марок бензина не связаны между собой, они могут быть

решены независимо друг от друга, что и предлагается выполнить читателям самостоятельно в качестве упражнения.

Для решения сформулированной индивидуальной транспортной задачи с трехиндексными переменными с помощью программы MS Excel создадим в книге Целочисленное программирование новый лист с именем Транспортная задача. Для решения задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A8:A20**, **B1**, **F1**, **B8:G8**, как это изображено на рис. 5.17. Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой транспортной задачи.
2. В ячейки **B2:E7** введем значения коэффициентов целевой функции (табл. 5.3).
3. В ячейку **F2** введем формулу: `=СУММПРОИЗВ(B2:E7; B9:E14)`, которая представляет собой целевую функцию (5.6.7).
4. В ячейки **G9:G14** и **B18:E20** введем значения, соответствующие правым частям ограничений (5.6.8).
5. В ячейку **F9** введем формулу: `=СУММ(B9:E9)`, которая представляет собой первое ограничение (5.6.8).
6. Скопируем формулу, введенную в ячейку **F9**, в ячейки **F10:F14**.
7. В ячейку **B15** введем формулу: `=СУММ(B9; B12)`, которая представляет собой седьмое ограничение (5.7.8).
8. Скопируем формулу, введенную в ячейку **B15**, в ячейки **C15:E15** и **B16:E17**.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения трехпродуктовой транспортной задачи с целочисленными переменными представлен на рис. 5.17.

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$F$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.

ЦелочисленноеПрограммирование.xls

	A	B	C	D	E	F	G
1		Коэффициенты целевой функции:				Значение ЦФ:	
2		2	4	5	7	=СУММПРОИЗВ(B2:E7;B9:E14)	
3		2	6	5	8		
4		3	5	7	11		
5		3	4	6	5		
6		3	5	9	10		
7		5	8	12	7		
8	Переменные:	*Xi1k*	*Xi2k*	*Xi3k*	*Xi4k*	Значения ограничений:	Производство НПЗ:
9	*X1j1*					=СУММ(B9:E9)	25
10	*X1j2*					=СУММ(B10:E10)	15
11	*X1j3*					=СУММ(B11:E11)	15
12	*X2j1*					=СУММ(B12:E12)	20
13	*X2j2*					=СУММ(B13:E13)	25
14	*X2j3*					=СУММ(B14:E14)	15
15	Значения огр-ний по 76:	=СУММ(B9;B12)	=СУММ(C9;C12)	=СУММ(D9;D12)	=СУММ(E9;E12)		
16	Значения огр-ний по 92:	=СУММ(B10;B13)	=СУММ(C10;C13)	=СУММ(D10;D13)	=СУММ(E10;E13)		
17	Значения огр-ний по 95:	=СУММ(B11;B14)	=СУММ(C11;C14)	=СУММ(D11;D14)	=СУММ(E11;E14)		
18	Потребности АЗС по 76:	10	15	11	9		
19	Потребности АЗС по 92:	15	10	6	9		
20	Потребности АЗС по 95:	5	10	7	8		
21							
22							

Транспортная задача

Рис. 5.17. Исходные данные для решения трехпродуктовой транспортной задачи

3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес диапазона ячеек `$B$9:$E$14`.
4. Добавить первых 6 ограничений, соответствующих базовым ограничениям на производство бензина различных марок НПЗ №1 и №2. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$F$9**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения выбрать ячейку **$G$9**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**;
   - аналогичным образом задать оставшиеся 5 ограничений первой группы.
5. Добавить следующие 12 ограничений, соответствующих базовым ограничениям на потребности в бензине различных марок АЗС №1 — №4. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$B$15**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;

- в качестве знака ограничения из выпадающего списка выбрать строгое неравенство "=";
- в качестве значения правой части ограничения выбрать ячейку **$B$18**;
- для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**;
- аналогичным образом задать оставшиеся 11 ограничений второй группы.

6. Добавить ограничение на целочисленность значений переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$9:$E$14**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**цел**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**целое**";
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
7. В дополнительных параметрах мастера поиска решения выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Внешний вид диалогового окна мастера поиска решения с ограничениями для транспортной задачи изображен на рис. 5.18.

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 5.19).

Результатом решения транспортной задачи являются найденные оптимальные ненулевые значения переменных: $x_{111} = 10$, $x_{121} = 4$, $x_{131} = 11$, $x_{132} = 6$, $x_{142} = 9$, $x_{123} = 8$, $x_{133} = 7$, $x_{221} = 11$, $x_{241} = 9$, $x_{212} = 15$, $x_{222} = 10$, $x_{113} = 5$, $x_{223} = 2$, $x_{243} = 8$, которым соответствует значение целевой функции: $f_{opt} = 563$.

Анализ найденного решения показывает, что для удовлетворения потребностей АЗС №1 в 76 бензине следует направить 10 бензовозов с бензином этой марки из НПЗ №1, в 92 бензине — 15 бензовозов с бензином этой марки из НПЗ №2, в 95 бензине — 5 бензовозов с бензином этой марки из НПЗ №2.

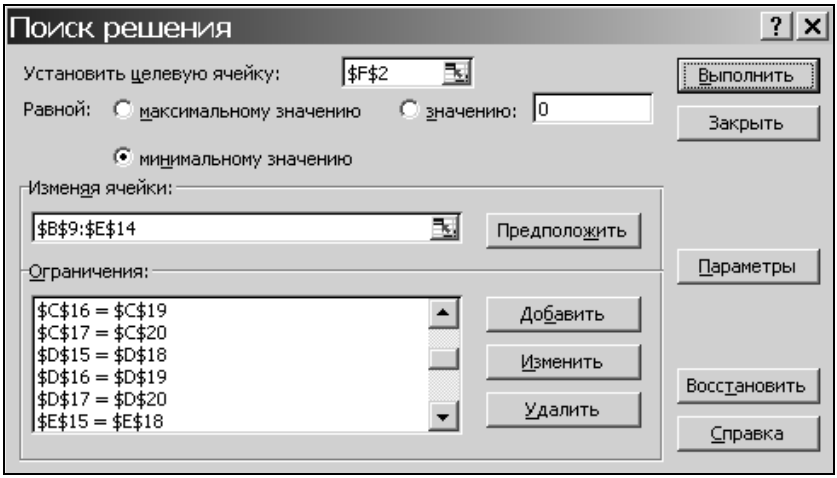


Рис. 5.18. Параметры мастера поиска решения и базовые ограничения для транспортной задачи

ЦелочисленноеПрограммирование

	A	B	C	D	E	F	G	H	I
1		Коэффициенты целевой функции:				Значение ЦФ:			
2		2	4	5	7	563			
3		2	6	5	8				
4		3	5	7	11				
5		3	4	6	5				
6		3	5	9	10				
7		5	8	12	7				
8	Переменные:	Xi1k	Xi2k	Xi3k	Xi4k	Значения огр-ний:	Производство НПЗ:		
9	X1j1	10	4	11	0	25	25		
10	X1j2	0	0	6	9	15	15		
11	X1j3	0	8	7	0	15	15		
12	X2j1	0	11	0	9	20	20		
13	X2j2	15	10	0	0	25	25		
14	X2j3	5	2	0	8	15	15		
15	Значения огр-ний по 76:	10	15	11	9				
16	Значения огр-ний по 92:	15	10	6	9				
17	Значения огр-ний по 95:	5	10	7	8				
18	Потребности АЗС по 76:	10	15	11	9				
19	Потребности АЗС по 92:	15	10	6	9				
20	Потребности АЗС по 95:	5	10	7	8				
21									
22									
23									
24									

Задача о поездах | Задача о стержнях | Транспортная задача

Рис. 5.19. Результат количественного решения 3-индексной транспортной задачи

Для удовлетворения потребностей АЗС №2 в 76 бензине следует направить 4 бензовоза с бензином этой марки из НПЗ №1 и 11 бензовозов с бензином этой марки из НПЗ №2, в 92 бензине — 10 бензовозов с бензином этой марки из НПЗ №2, в 95 бензине — 8 бензовозов с бензином этой марки из НПЗ №1 и 2 бензовоза с бензином этой марки из НПЗ №2.

Для удовлетворения потребностей АЗС №3 в 76 бензине следует направить 11 бензовозов с бензином этой марки из НПЗ №1, в 92 бензине — 6 бензовозов с бензином этой марки из НПЗ №1, в 95 бензине — 7 бензовозов с бензином этой марки из НПЗ №1.

Для удовлетворения потребностей АЗС №4 в 76 бензине следует направить 9 бензовозов с бензином этой марки из НПЗ №2, в 92 бензине — 9 бензовозов с бензином этой марки из НПЗ №1, в 95 бензине — 8 бензовозов с бензином этой марки из НПЗ №2. При этом общая стоимость найденного плана перевозок составит 563 *тыс. рублей*.

Для проверки правильности найденного решения можно воспользоваться специальным алгоритмом решения транспортных задач, например, рассмотренным в *главе 4* алгоритмом метода потенциалов. Для этого можно рассмотреть 3 независимых однопродуктовых транспортных задачи с целочисленными переменными.

Примечание

Для решения сформулированной индивидуальной транспортной задачи с помощью программы MS Excel можно также преобразовать исходную трехпродуктовую задачу в однопродуктовую. Для этого вместо 2-х НПЗ следует ввести в рассмотрение 6 пунктов производства, первые 3 из которых будут соответствовать НПЗ №1 для 3-х рассматриваемых марок бензина, а оставшиеся 3 — НПЗ №2 также для 3-х рассматриваемых марок бензина. Соответственно, вместо 4-х АЗС следует рассмотреть 12 пунктов потребления. В этом случае вместо исходной постановки (5.6.7) и (5.6.8) с трехиндексными переменными может быть сформулирована эквивалентная однопродуктовая транспортная задача с двухиндексными переменными в форме (5.6.1)—(5.6.6). Заинтересованным читателям предлагается выполнить это самостоятельно в качестве упражнения.

5.7. Упражнения

В качестве упражнений для самостоятельного решения с помощью программы MS Excel предлагаются несколько типовых задач целочисленного линейного программирования.

5.7.1. Задача о погрузке автомобиля

Для перевозки штучного товара используется грузовой автомобиль с ограниченной грузоподъемностью и объемом кузова. Груз представляет собой коробки с продуктами питания следующих видов: тушеная говядина, лосось

в масле, шпроты, сгущенное молоко, томатный соус и зеленый горошек. Масса одной коробки и ее объем, а также стоимость коробки для каждого из видов продуктов питания представлены в следующей таблице (табл. 5.4).

Таблица 5.4. *Исходные данные задачи о погрузке автомобиля*

Характеристика груза	**Виды продуктов питания**					
	тушеная говядина	**лосось в масле**	**шпроты**	**сгущенное молоко**	**томатный соус**	**зеленый горошек**
Масса (*кг*)	8	8	6	5	5	4
Объем ($м^2$)	0,3	0,25	0,2	0,15	0,2	0,15
Стоимость (*руб.*)	500	400	350	380	320	300

При этом общая грузоподъемность автомобиля равна 5 *т*, а объем кузова — 120 $м^2$. Требуется определить оптимальную загрузку автомобиля коробками с различными продуктами питания, так чтобы обеспечить максимальную стоимость общего груза, но при этом не допустить перегрузки автомобиля и превышения допустимого объема его кузова.

Примечание

Нетрудно заметить, что по своей структуре данная задача является разновидностью типовой задачи о рюкзаке, в которой для загружаемых предметов используется две характеристики — масса и объем. Соответствующая задача получила специальное название — *двумерная задача о рюкзаке*.

5.7.2. Задача об изготовлении обуви

Обувная фабрика производит три вида обуви: мужскую, женскую и детскую. На каждую пару мужской, женской и детской обуви, соответственно, требуется кожи 4, 2 и 1 $дм^2$, клея — 20, 20 и 10 *г*. Стоимость мужской, женской и детской обуви с учетом всех работ, соответственно, равна 1000, 1500, 500 *руб*. Запасы кожи на складе фабрики составляют 4000 $м^2$, а клея — 3 *т*. Требуется определить оптимальный план изготовления мужской, женской и детской обуви, при котором стоимость выпущенной продукции является максимальной.

5.7.3. Задача об изготовлении мебели

Мебельная фабрика изготавливает пять видов продукции: столы, шкафы, диваны, кресла и кровати. Нормы затрат труда, а также древесины и ткани на производство единицы продукции каждого их видов приведены в следующей таблице (табл. 5.5).

Таблица 5.5. *Исходные данные задачи об изготовлении мебели*

Ресурсы	**Норма расхода ресурса на единицу продукции**					**Общее количество ресурсов**
	Стол	**Шкаф**	**Диван**	**Кресло**	**Кровать**	
Трудозатраты (*человеко-час*)	4	8	12	9	10	3 456
Древесина (*м3*)	0,4	0,6	0,3	0,2	0,3	432
Ткань (*м*)	—	—	6	4	5	2 400
Стоимость реализации одного изделия (*руб.*)	2 000	2 500	4 000	3 500	3 800	—
Выпуск (*шт*): минимальный	120	90	20	40	30	—
максимальный	480	560	180	160	120	—

В этой же таблице указана стоимость реализации одного изделия каждого вида, приведено общее количество ресурсов отдельных типов, имеющихся в распоряжении фабрики. Дополнительно на основе проведенного маркетинга указаны пределы объемов изготовления каждого вида мебели. Требуется определить оптимальный план производства мебельной продукции, согласно которому общая стоимость ее реализации для фабрики будет максимальной.

5.7.4. Многопродуктовая транспортная задача

Три автомобильных завода выпускают три марки автомобилей: легковые, грузовые и спортивные, которые поставляются для продажи пяти автосалонам. Стоимость транспортировки одного автомобиля отдельных марок задана в табл. 5.6, а объемы производства автомобилей заводами и заказа их автосалонами представлены в табл. 5.7.

Таблица 5.6. *Стоимость транспортировки автомобилей разных марок (в тыс. рублей)*

Пункты потребления/ Пункты производства	Марка автомобиля	Автосалон №1	Автосалон №2	Автосалон №3	Автосалон №4	Автосалон №5
Автозавод №1	**Легковой**	2	4	5	7	6
	Грузовой	2	6	5	8	10
	Спортивный	3	5	7	4	5
Автозавод №2	**Легковой**	3	4	6	5	4
	Грузовой	3	5	9	10	12
	Спортивный	5	8	12	7	4
Автозавод №3	**Легковой**	2	3	6	3	6
	Грузовой	7	8	6	5	13
	Спортивный	5	6	11	8	9

Таблица 5.7. *Объемы выпуска автомобилей разных марок и спрос на них (в шт)*

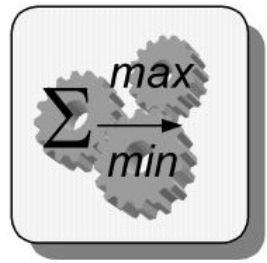
Пункты производства и потребления	Марка автомобиля		
	легковой	грузовой	спортивный
Автозавод №1	100	150	20
Автозавод № 2	150	80	50
Автозавод №3	200	50	30
Автосалон №1	80	30	21
Автосалон №2	50	20	15
Автосалон №3	40	25	16
Автосалон №4	60	50	18
Автосалон №5	110	70	12

Требуется определить оптимальный план поставок автомобилей в автосалоны, так чтобы удовлетворить имеющиеся заказы, а суммарные затраты на транспортировку всех автомобилей были минимальными.

Примечание

По своей структуре данная транспортная задача не является сбалансированной. В общем случае подобные транспортные задачи посредством введения фиктивных пунктов производства или фиктивных пунктов потребления предварительно требуется преобразовать в сбалансированную форму. Однако при решении транспортных задач с помощью программы MS Excel такого преобразования может и не потребоваться. Чтобы убедиться в этом, читателям рекомендуется решить описанную задачу двумя способами, а полученные результаты сравнить.

В качестве дополнительного упражнения можно рассмотреть модификацию данной транспортной задачи, в которой автозавод № 2 отказался от выпуска спортивных автомобилей, а автозавод №3 отказался от выпуска грузовых автомобилей, при этом заказы автосалонов и транспортные издержки остались без изменения.



Глава 6

Задачи оптимизации с булевыми переменными

Класс задач оптимизации с булевыми переменными является подклассом задач дискретной оптимизации, для которых вводится дополнительное требование — переменные должны принимать булевы значения. В рамках задач этого класса важное место занимает подкласс задач булева линейного программирования, к которому относятся задачи оптимизации с булевыми переменными с дополнительным свойством линейности целевой функции и ограничений. Именно эти основные классы задач образуют предмет рассмотрения настоящей главы.

6.1. Общая постановка задачи оптимизации с булевыми переменными

К классу задач оптимизации с булевыми переменными относятся такие задачи однокритериальной оптимизации, в которых переменные могут принимать только значения 0 или 1. При этом на характер целевой функции и ограничений в общем случае не накладывается никаких требований. Более строгое определение данного класса задач можно получить на основе рассмотрения общей математической постановки задачи оптимизации с булевыми переменными. Важность задач оптимизации с булевыми переменными в общем контексте задач оптимизации обуславливается тем обстоятельством, что к задачам этого класса сводятся многие другие задачи оптимизации, в частности, рассматриваемые далее задачи оптимизации на графах и задачи комбинаторной оптимизации.

6.1.1. Математическая постановка задачи оптимизации с булевыми переменными

В общем случае математическая постановка задачи оптимизации с булевыми переменными может быть сформулирована в следующем виде:

$$f(x_1, x_2, ..., x_n) \to \max_{x \in \Delta_\beta}, \text{ где} \tag{6.1.1}$$

$$\Delta_\beta = \{\Delta \mid g_k(x_1, x_2, ..., x_n) \le (=) b_k; \quad x_1, x_2, ..., x_n \in \{0, 1\}, (k \in \{1, 2, ..., m\})\} \tag{6.1.2}$$

Математическую модель (6.1.1) и (6.1.2) называют также задачей *булева программирования*. Если дополнительно в данную математическую модель вводятся предположения о линейном характере целевой функции и левых частей ограничений, то соответствующую задачу часто называют задачей *булева линейного программирования*.

В этом случае общая задача булева линейного программирования может быть сформулирована следующим образом. Необходимо найти максимум линейной целевой функции n переменных $x_1, x_2, ..., x_n \in \{0, 1\}$ следующего вида:

$$c_1 x_1 + c_2 x_2 + ... + c_n x_n \to \max_{x \in \Delta_\beta}, \tag{6.1.3}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств и неравенств:

$$\begin{cases} a_{i1} x_1 + a_{i2} x_2 + ... + a_{in} x_n = b_i & (\forall i \in \{1, 2, ..., q\}); \quad (6.1.4) \\ a_{k1} x_1 + a_{k2} x_2 + ... + a_{kn} x_n \le b_k & (\forall k \in \{q+1, ..., m\}). \quad (6.1.5) \end{cases}$$

В математической постановке общей задачи булева линейного программирования через $c_i, a_{ki}, b_k (\forall i \in \{1, 2, ..., n\}, \forall k \in \{1, 2, ..., m\}$ обозначены постоянные величины, которые могут принимать произвольные, как правило, целочисленные значения, предопределяемые спецификой конкретной задачи линейного программирования.

В случае отсутствия ограничений типа равенств (6.1.4), т. е. при $q = 0$, задачу булева линейного программирования называют также *задачей булева линейного программирования с ограничениями типа неравенств*. Напротив, в случае отсутствия ограничений типа неравенств (6.1.5), т. е. при $q = m$, задачу булева линейного программирования называют также *задачей булева линейного программирования с ограничениями типа равенств*. Нетрудно заметить,

что посредством введения дополнительных булевых переменных одна из этих задач может быть преобразована в другую.

При анализе множества допустимых альтернатив Δ_β общей задачи булева программирования (6.1.1) и (6.1.2) оказывается справедливой одна из двух возможных ситуаций:

1. Система ограничений (6.1.2) противоречива или несовместна, т. е. не существует такого набора значений x_1, x_2, …, x_n, которые удовлетворяют ограничениям (6.1.2). В этом случае задача булева программирования не имеет решения.
2. Система ограничений (6.1.2) не является противоречивой. В этом случае задача булева программирования имеет решение, которое, в общем случае, может быть не единственным.

При постановке и решении задач булева программирования следует иметь в виду следующее важное свойство: множество допустимых альтернатив Δ_β любой задачи булева программирования всегда имеет *конечную мощность*. При этом предполагается, что мощность пустого множества равна 0.

Примечание

Поскольку задачи булева линейного программирования образуют подкласс задач булева программирования, то отмеченное свойство конечности множества допустимых альтернатив наследуется всеми задачами данного подкласса.

6.1.2. Основные методы решения задач оптимизации с булевыми переменными

Рассмотрение в исходной математической постановке задачи целочисленного линейного программирования внешне мало заметного дополнительного требования целочисленности значений переменных приводит к принципиальному изменению не только характера соответствующих задач, но возможных методов ее решения.

Так, применительно к задачам булева линейного программирования, также как и для задач целочисленного линейного программирования, оказывается неприменимым аналитический способ решения, основанный, например, на методе множителей Лагранжа, поскольку понятие дифференцируемости целевой функции и ограничений для булевых значений переменных теряет свой исходный смысл. Как нетрудно заметить, собственно понятие округления до ближайших булевых значений переменных оптимального решения задачи (6.1.1) и (6.1.2), найденное без учета требования булевых значений перемен-

ных, теряет свой исходный смысл и не гарантирует получения даже локально оптимального решения задачи булева программирования.

Так, например, при выполнении ограничения: $x_1 + x_2 = 1$, где x_1, $x_2 \in \{0, 1\}$, и найденных оптимальных значениях переменных: $x_1 = 0{,}5$ и $x_2 = 0{,}5$, процедура округления не имеет смысла. Несостоятельность округления в задачах булева программирования подчеркивается также следующими соображениями. Пусть, например, решение о финансировании некоторого проекта представлено булевой переменной x. При этом переменная x принимает значение 0, если проект отклоняется, и значение 1, если проект принимается. В этом случае любое дробное значение переменной x вообще теряет смысл, и процедура округления оказывается логически неприемлемой.

С другой стороны, отмеченное ранее свойство конечности множества допустимых альтернатив Δ_β общей задачи булева программирования служит основой невольного оптимизма, основанного на потенциальной возможности полного перебора всех допустимых значений переменных с целью выбора из них оптимального решения исходной задачи (6.1.1) и (6.1.2). Однако данный способ, являясь наиболее простым по своей реализации, вовсе не гарантирует нахождения оптимального решения практических задач за приемлемое время даже на компьютерах с высоким быстродействием.

Таким образом, задачи булева программирования требуют разработки специальных алгоритмических методов для своего решения, которые учитывают специфические особенности и конечную мощность множества допустимых решений. Из алгоритмических методов, обеспечивающих нахождение точного решения общих задач (6.1.1) и (6.1.2), наибольшее распространение получили метод ветвей и границ и метод динамического программирования. Что касается графического способа, то его применение теряет всякий смысл, поскольку в случае двух или трех булевых переменных оптимальное решение соответствующей задачи может быть найдено простым перебором значений переменных.

Применительно к решению практических задач булева программирования с большим числом переменных и ограничений были разработаны также специальные алгоритмы нахождения приближенного целочисленного решения, которые позволяют находить одно или несколько локально-оптимальных решений. Однако использование приближенных методов оправдано лишь тогда, когда по тем или иным причинам нахождение точного решения соответствующих задач оказывается невозможным.

Программа MS Excel позволяет находить точное решение задач булева линейного программирования. При этом общий порядок подготовки и решения полностью аналогичен рассмотренному ранее процессу решения задач

линейного программирования. Для оценки точности получаемых решений можно сравнить полученные различными способами оптимальные решения отдельных практических задач. Именно с этой целью в данной главе приводится описание двух способов решения задач о рюкзаке и задачи о производстве часов.

6.2. Задача о рюкзаке с булевыми переменными

Содержательная постановка задачи о рюкзаке приводится в *разд. 1.2.7*. В данной главе рассматривается ее модификация, используемая для формальной записи условий и решения соответствующей индивидуальной задачи оптимизации.

6.2.1. Математическая постановка одномерной задачи о рюкзаке с булевыми переменными

Так же как и в одномерной целочисленной задаче о рюкзаке *(см. разд. 5.2)*, переносимый в рюкзаке груз может включать n видов предметов, при этом каждый предмет вида $i \in \{1, 2, ..., n\}$ обладает массой a_i, например, a_i *кг*. Для каждого вида предмета турист, исходя из своих субъективных предпочтений, определяет его индивидуальную ценность c_i во время перехода. При этом общая масса всех предметов ограничена емкостью рюкзака и составляет b *кг*. Дополнительно присутствует следующее требование: каждый из предметов имеется в *единственном экземпляре*.

Требуется определить такой набор из исходных предметов, которые следует положить в рюкзак, чтобы суммарная ценность предметов была максимальной, а их общая масса не превысила допустимого фиксированного значения.

Введем в рассмотрение следующие булевы переменные: x_i — количество предметов i-го вида, которые следует положить в рюкзак. Очевидно, в этом случае: $x_i = 1$, если предмет i-го вида следует положить в рюкзак, и $x_i = 0$, если предмет i-го вида не следует укладывать в рюкзак. Тогда в общем случае математическая постановка *одномерной задачи о рюкзаке с булевыми переменными* может быть сформулирована следующим образом.

$$c_1 x_1 + c_2 x_2 + ... + c_n x_n \to \max_{x \in \Delta_\beta}, \qquad (6.2.1)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} a_1x_1 + a_2x_2 + ... + a_nx_n \le b; \\ x_1, x_2, ..., x_n \in \{0, 1\}. \end{cases} \quad (6.2.2)$$

В случае введения в рассмотрение второй характеристики видов продуктов, например, их геометрического объема, в постановку задачи (6.2.1) и (6.2.2) следует ввести дополнительное ограничение на общий объем продуктов, которые следует положить в рюкзак. Соответствующая задача получила название *двумерной* задачи о рюкзаке с булевыми переменными. В общем случае аналогичным образом может быть получено ее обобщение на m типов характеристик продуктов. Соответствующая задача называется *многомерной задачей о рюкзаке с булевыми переменными*.

6.2.2. Решение одномерной задачи о рюкзаке с булевыми переменными с помощью программы MS Excel

Для решения задачи о рюкзаке с помощью программы MS Excel необходимо задать конкретные значения параметрам исходной задачи. С этой целью рассмотрим задачу оптимальной загрузки некоторого грузового транспортного средства, например, автомобиля, контейнерами фиксированной массы. Хотя содержательно в данной задаче речь идет вовсе не о загрузке рюкзака, но по своей структуре данная задача относится к типовой задаче о рюкзаке.

Для определенности предположим, что имеются 6 контейнеров: красный, оранжевый, желтый, зеленый, голубой и синий ($n = 6$), а в качестве их характеристики — масса контейнера в *тоннах*. При этом масса каждого из контейнеров имеет следующие значения: $a_1 = 3$, $a_2 = 4$, $a_3 = 4$, $a_4 = 5$, $a_5 = 2$, $a_6 = 3$. Общая грузоподъемность автомобиля равна: $b = 15$. Дополнительно известна стоимость каждого из контейнеров (например, в *тыс. рублей*): $c_1 = 50$, $c_2 = 60$, $c_3 = 40$, $c_4 = 50$, $c_5 = 30$, $c_6 = 40$. Требуется определить те контейнеры, которые следует погрузить на автомобиль, так чтобы обеспечить максимальную стоимость общего груза и при этом не допустить перегрузки.

Исходными переменными математической модели данной булевой задачи о рюкзаке являются переменные $x_i (\forall i \in \{1, 2, ..., 6\})$, каждая из которых принимает значение 1, если i-й контейнер загружается в автомобиль, и 0 — в противном случае. Тогда математическая постановка рассматриваемой индивидуальной задачи о рюкзаке может быть записана в следующем виде.

$$50x_1 + 60x_2 + 40x_3 + 50x_4 + 30x_5 + 40x_6 \to \max_{x \in \Delta_\beta}, \quad (6.2.3)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} 3x_1 + 4x_2 + 4x_3 + 5x_4 + 2x_5 + 3x_6 \le 15; \\ x_1, x_2, x_3, x_4, x_5, x_6 \in \{0, 1\}. \end{cases} \quad (6.2.4)$$

Для решения данной задачи с помощью программы MS Excel создадим новую книгу с именем Булево программирование и изменим имя ее первого рабочего листа на Задача о рюкзаке. Для решения поставленной задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:H1**, **A2:A5**, **B4**, **H4**, **I4**. Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи линейного программирования.
2. В ячейки **B3:G3** введем значения коэффициентов целевой функции: $c_1 = 50$, $c_2 = 60$, $c_3 = 40$, $c_4 = 50$, $c_5 = 30$, $c_6 = 40$.
3. В ячейку **H2** введем формулу: `=СУММПРОИЗВ(B2:G2; B3:G3)`, которая представляет собой целевую функцию (6.2.3).
4. В ячейки **B5:G5** введем значения коэффициентов первого ограничения (6.2.4): $a_1 = 3$, $a_2 = 4$, $a_3 = 4$, $a_4 = 5$, $a_5 = 2$, $a_6 = 3$.
5. В ячейку **I5** введем значение правой части первого ограничения, соответствующего максимальной грузоподъемности транспортного средства: $b = 15$.
6. В ячейку **H5** введем формулу: `=СУММПРОИЗВ($B$2:$G$2; B5:G5)`, которая представляет собой левую часть первого ограничения (6.2.4).

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения булевой задачи о рюкзаке имеет следующий вид (рис. 6.1).

БулевоПрограммирование.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	***x1***	***x2***	***x3***	***x4***	***x5***	***x6***	**Значение ЦФ:**		
2	**Значения:**							=СУММПРОИЗВ(B2:G2;B3:G3)		
3	**Стоимость:**	50	60	40	50	30	40			
4		**Коэффициенты ограничений:**						Значения ограничений:	Общая масса:	
5	по массе:	3	4	4	5	2	3	=СУММПРОИЗВ($B$2:$G$2;B5:G5)	15	
6										
7										
8										
9										
10										
11										

Задача о рюкзаке

Рис. 6.1. Исходные данные для решения булевой задачи о рюкзаке

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$H$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **максимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$G$2`.
4. Добавить первое ограничение для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$H$5**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения выбрать ячейку **$I$5**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Добавить ограничение на булевы значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$G$2**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**двоичное**";
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

6. В окне **Параметры** поиска решения следует выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения имеет следующий вид (рис. 6.2).

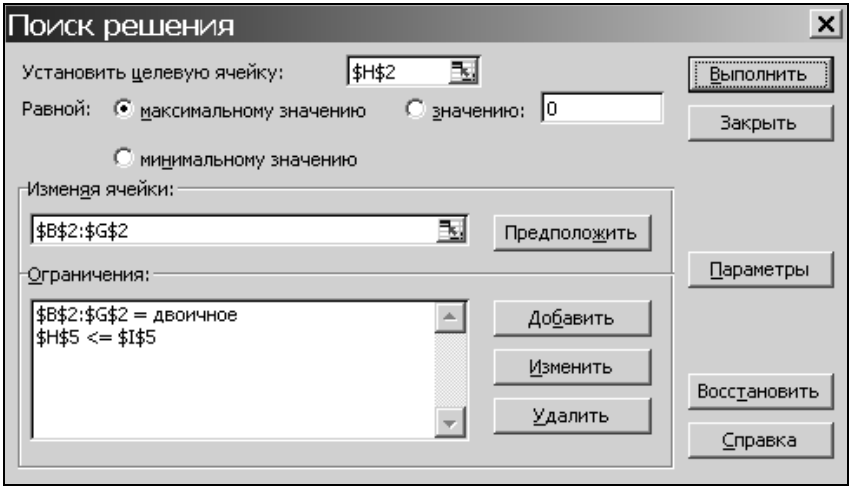


Рис. 6.2. Ограничения на значения переменных и параметры мастера поиска решения для булевой задачи о рюкзаке

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 6.3).

БулевоПрограммирование

	A	B	C	D	E	F	G	H	I	J	K
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	**Значение ЦФ:**			
2	**Значения:**	1	1	0	1	0	1	200,000			
3	**Стоимость:**	50	60	40	50	30	40				
4		**Коэффициенты ограничений:**						Значения огр-ний	Общая масса:		
5	по массе:	3	4	4	5	2	3	15,000	15		
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											

Задача о рюкзаке / Лист2 / Лист3

Рис. 6.3. Результат количественного решения булевой задачи о рюкзаке

Результатом решения булевой задачи о рюкзаке являются найденные оптимальные значения переменных: $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, $x_5 = 0$, $x_6 = 1$, которым соответствует значение целевой функции: $f_{opt} = 200$ *(тыс. рублей)*. При этом общая масса контейнеров будет равна грузоподъемности транспортного средства.

Анализ найденного решения показывает, что оптимальной загрузкой автомобиля с точки зрения максимума общей стоимости груза является погрузка красного, оранжевого, зеленого и синего контейнеров. Этому грузу будет соответствовать общая стоимость в 200 000 *рублей*.

Следует заметить, что оптимальное решение рассмотренной булевой задачи о рюкзаке может быть найдено методом *полного перебора*. В этом случае необходимо проверить на допустимость первому ограничению (6.2.4) $2^6 = 64$ возможных комбинаций значений переменных x_1, x_2, x_3, x_4, x_5, x_6, вычислить для допустимых решений значение целевой функции (6.2.3) и выбрать в качестве оптимального решения то из них, которому соответствует максимальное значение целевой функции.

На первый взгляд метод полного перебора представляется простым и универсальным средством решения задач булева программирования. Это неоспоримое достоинство метода склонны преувеличивать пользователи современных компьютеров, полагая, что подобным способом действительно можно решить любую задачу подобного класса. Однако это впечатление глубоко обманчиво и способно ввести в заблуждение неискушенных программистов. Действительно, объем вычислений в методе полного перебора зависит от количества переменных задачи даже не в геометрической, а в экспоненциальной прогрессии.

Так, например, для $n = 10$ необходимо проверить $2^{10} = 1024$ двоичных комбинаций переменных; для $n = 20$ — уже $2^{20} = 1048\,576$ комбинаций; для $n = 40$ — совсем не малое число: $2^{40} = 1099\,511\,627\,776$ комбинаций (это больше триллиона). Именно последнее число характеризует некоторый разумный предел количества переменных данной типовой задачи или, как принято говорить, предел размерности задачи булева программирования, которая в общем случае может быть решена методом полного перебора с использованием компьютера. В то же время среди практических задач булева программирования нередко встречаются задачи с числом переменных порядка 100—200.

Отмеченные особенности задач булева программирования существенно ограничивают возможности метода прямого перебора при решении практических задач. Именно эта причина послужила причиной разработки специальных методов точного и приближенного решения задач булева программирования, часть из которых рассматриваются далее в настоящей главе.

6.2.3. Решение задачи о рюкзаке с помощью метода динамического программирования

Динамическое программирование представляет собой специальный раздел математической теории управления, посвященный исследованию многошаговых задач принятия оптимальных решений. Основные идеи динамического программирования предложены американским математиком Ричардом Э. Беллманом в конце 50-х годов XX в., который в последующем разработал и соответствующую методологию решения задач оптимального управления, основанную на так называемом принципе оптимальности. Ключевым аспектом этой методологии является *многошаговость* решаемой задачи оптимизации, которая либо отражает естественное протекание некоторого процесса управления во времени, либо вводится в задачу оптимизации искусственно.

Общая схема динамического программирования сформулирована для многошаговых процессов принятия решений по управлению некоторой системой, описываемой дифференциальными, конечно-разностными или функциональными уравнениями. В то же время применительно к решению задач целочисленного и булева программирования рассматривается конкретизация общей схемы, которая учитывает специфику математической постановки соответствующих задач. При этом конкретизацию общей схемы динамического программирования, предназначенную для решения отдельных задач оптимизации, называют *методом динамического программирования*.

Применительно к задачам булева программирования с линейной целевой функцией (6.2.1), которая может быть интерпретирована в качестве многошаговой схемы последовательного принятия решения относительно каждой из исходных переменных, *принцип оптимальности* Р. Беллмана может быть сформулирован следующим образом. Оптимальные значения переменных $(x_1^*, x_2^*, \ldots, x_n^*)$ обладают тем свойством, что при фиксации любого подмножества этих переменных остальные переменные должны определяться как оптимальное решение по отношению к частному результату, полученному при фиксации переменных первого подмножества.

Применительно к решению задачи о рюкзаке с помощью метода динамического программирования необходимо конкретизировать данный принцип оптимальности в форме так называемого функционального уравнения, отражающего многошаговый характер схемы динамического программирования.

Основное функциональное уравнение, которое отражает принцип оптимальности применительно к решению задачи о рюкзаке (6.2.1) и (6.2.2.), имеет следующий вид:

$$h_i(z) = \max_{\substack{x_i \in \Delta_\beta \\ g_i(x_i) \le z}} (h_{i-1}(z - g_i(x_i)) + f_i(x_i)), (\forall i \in \{1, 2, \ldots, n\}). \qquad (6.2.5)$$

Здесь функция $h_0(z) = 0$ по определению, функции $f_i(x_i) = c_i x_i$, $g_i(x_i) = a_i x_i$, $(\forall i \in \{1, 2,..., n\})$, а переменная z характеризует остаток имеющегося ресурса и равна: $z = b - g_1(x_1) + g_2(x_2)... + g_n(x_n)$.

При решении основного функционального уравнения могут быть вычислены условно-оптимальные значения переменных. Для этого может быть использована следующая формула:

$$x_i(z) = \arg\max_{\substack{x_i \in \Delta_\beta \\ g_i(x_i) \le z}} (h_{i-1}(z - g_i(x_i)) + f_i(x_i)), (\forall i \in \{1, 2,..., n\}). \qquad (6.2.6)$$

Метод динамического программирования, ориентированный на решение одномерной задачи о рюкзаке, основывается на рекуррентном характере соотношений (6.2.5) и (6.2.6). Алгоритм метода динамического программирования имеет итеративный характер и заключается в выполнении следующих действий:

1. *Прямая последовательность расчета.* С использованием рекуррентных соотношений (6.2.5) последовательно рассчитываются значения функций $h_i(z_j)$ для значений z_j от 0 до b, где b — заданное значение правой части ограничения (6.2.2). При этом интервал изменения значений переменных z_j равен 1, т. е. $z_j \in \{0, 1, 2,..., b\}$. Одновременно с использованием рекуррентных соотношений (6.2.6) находятся условно оптимальные значения переменных $x_i(z_j)$ для значений i от 1 до n.

2. *Нахождение оптимального значения целевой функции.* Среди всех найденных на последнем шаге значений функций $h_n(z_j)$ находится максимальное. Это значение равно оптимальному значению целевой функции (6.2.1): $f_{opt} = \max\{h_n(z_j)\}$ для $\forall z_j \in \{0, 1, 2,..., b\}$.

3. *Обратная последовательность расчета.* С использованием найденных значений функций $h_i(z_j)$ и условно оптимальных значений переменных последовательно находятся оптимальные значения переменных x_i для значений i от n до 1.

Рассмотренный алгоритм метода динамического программирования может быть изображен графически в форме диаграммы деятельности языка UML (рис. 6.4).

Нетрудно заметить, что в силу конечности общего количества переменных задачи о рюкзаке, рассмотренный алгоритм метода динамического программирования является конечным.

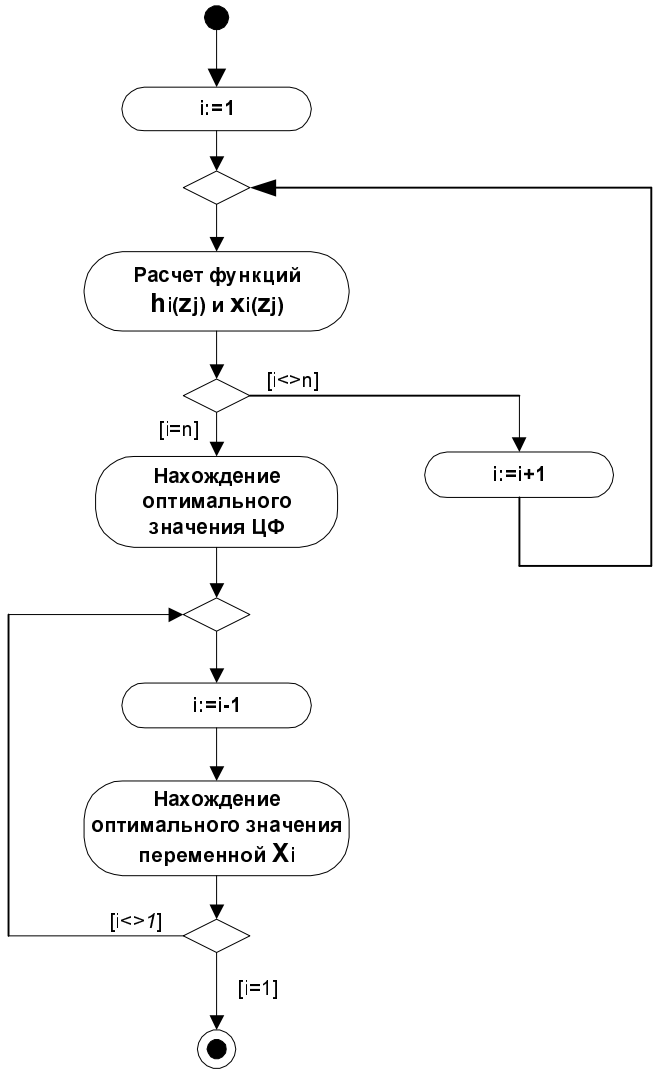


Рис. 6.4. Диаграмма деятельности алгоритма метода динамического программирования

Проиллюстрируем использование рассмотренного алгоритма метода динамического программирования для решения индивидуальной булевой задачи о рюкзаке (6.2.3) и (6.2.4). С этой целью в книге Булево программирование

создадим новый рабочий лист с именем Динамическое программирование. Далее следует выполнить следующие действия:

1. Внесем необходимые надписи в ячейки **A1:A14**, **B1:B14** и **C1:R1**, как это изображено на рис. 6.5. Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи о назначении венгерским методом.
2. С использованием рекуррентных соотношений (6.2.5) последовательно заполним ячейки **C9:R9**, в которых будут находиться значения функций $h_1(z_j)$ для значений переменной z_j от 0 до 15.
3. С использованием рекуррентных соотношений (6.2.6) последовательно заполним ячейки **C2:R2**, в которых будут находиться условно оптимальные значения переменных $x_1(z_j)$ для значений z_j от 0 до 15.
4. Аналогичным образом заполняются ячейки **C10:R14** и **C3:R7**.

Внешний вид рабочего листа MS Office Excel 2003 с результатами выполнения прямой последовательности расчетов при решении задачи о рюкзаке методом динамического программирования представлен на рис. 6.5.

БулевоПрограммирование

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	*Переменные:*	*gi*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	
2	*x1*	*3*	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
3	*x2*	*4*	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	
4	*x3*	*4*	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	
5	*x4*	*5*	0	0	0	0	0	0	0	0	0	1/0	1/0	0	1	1	1	1	
6	*x5*	*2*	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	1	
7	*x6*	*2*	0	0	0	0	0	0	1/0	0	1	0	1	1/0	1	1/0	1	1	
8	**Стоимости *hi*:**	*ci*																	
9	*x1*	**50**	0	0	0	50	50	50	50	50	50	50	50	50	50	50	50	50	
10	*x2*	**60**	0	0	0	50	60	60	60	110	110	110	110	110	110	110	110	110	
11	*x3*	**40**	0	0	0	50	60	60	60	110	110	110	110	150	150	150	150	150	
12	*x4*	**50**	0	0	0	50	60	60	60	110	110	110	110	150	160	160	160	160	
13	*x5*	**30**	0	0	30	50	60	80	90	110	110	140	140	150	160	180	180	180	
14	*x6*	**40**	0	0	30	50	60	80	90	110	120	140	150	150	180	180	190	200	
15																			
16																			
17																			
18																			
19																			
20																			

Задача о рюкзаке | Дианамическое программирование | 3

Рис. 6.5. Результаты выполнения прямой последовательности расчетов при решении задачи о рюкзаке методом динамического программирования

После выполнения прямой последовательности расчетов в ячейках **C14:R14** находится ячейка с максимальным значением. Это значение 200, находящееся в ячейке **R14**. Тем самым найдено максимальное значение целевой функции

исходной задачи о рюкзаке и оно равно: $f_{opt} = 200$. Ячейке **R14** соответствует ячейка **R7** с оптимальным значением переменной $x_6 = 1$.

Следующее оптимальное значение переменной $x_5 = 0$ находится в ячейке **O6**, адрес которой получается из выражения: $15 - a_6 x_6 = 15 - 3 = 12$. Оптимальное значение переменной $x_4 = 1$ находится в ячейке **O5**, адрес которой получается из выражения: $12 - a_5 x_5 = 12 - 0 = 12$. Оптимальное значение переменной $x_3 = 0$ находится в ячейке **J4**, адрес которой получается из выражения: $12 - a_4 x_4 = 12 - 5 = 7$. Оптимальное значение переменной $x_2 = 1$ находится в ячейке **J3**, адрес которой получается из выражения: $7 - a_3 x_3 = 7 - 0 = 7$. И, наконец, оптимальное значение переменной $x_1 = 1$ находится в ячейке **F2**, адрес которой получается из выражения: $7 - a_2 x_2 = 7 - 4 = 3$.

Таким образом, результатом решения булевой задачи о рюкзаке методом динамического программирования являются найденные оптимальные значения переменных: $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 0, x_6 = 1$, которым соответствует значение целевой функции: $f_{opt} = 200$.

Анализ результатов решения задачи о рюкзаке с помощью программы MS Excel и методом динамического программирования показывает их полное совпадение, что служит веским аргументом в пользу их достоверности.

6.3. Задача водопроводчика

Содержательная постановка задачи водопроводчика приводится в *разд. 1.2.11*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи оптимизации в форме математической модели булева линейного программирования.

6.3.1. Математическая постановка задачи водопроводчика

Для разработки математической модели индивидуальной задачи водопроводчика с конкретной схемой расположения труб (см. рис. 1.9) необходимо пронумеровать все имеющиеся плиты от 1 до 16, а трубы от 1 до 5. Тогда исходная схема расположения труб примет следующий вид (рис. 6.6).

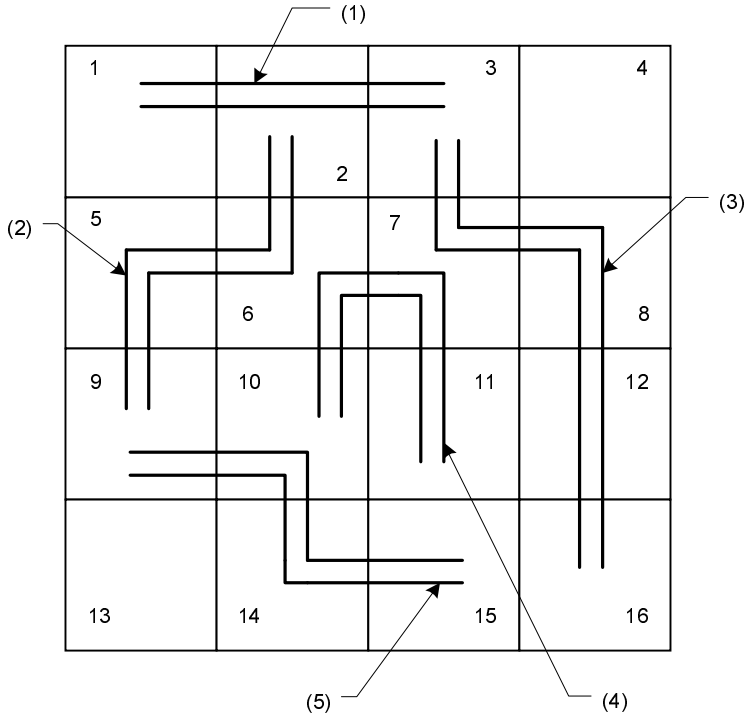


Рис. 6.6. Геометрическая интерпретация задачи водопроводчика

В качестве переменных математической модели задачи водопроводчика рассмотрим переменные: x_i $(i \in \{1, 2, ..., 16\})$, которые интерпретируются следующим образом. Переменная $x_i = 1$, если для установки вентильных перекрытий принято решение приподнять i-ю плиту, $x_i = 0$ в противном случае, т. е. когда при выполнении работ i-я плита не приподнимается $(\forall i \in \{1, 2, ..., 16\})$.

Тогда математическая постановка рассматриваемой индивидуальной задачи водопроводчика может быть записана в следующем виде:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + \\ + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} \to \min\limits_{x \in \Delta_\beta}, \qquad (6.3.1)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} x_1 + x_2 + x_3 \geq 1; \\ x_2 + x_5 + x_6 + x_9 \geq 1; \\ x_3 + x_7 + x_8 + x_{12} + x_{16} \geq 1; \\ x_6 + x_7 + x_{10} + x_{11} \geq 1; \\ x_9 + x_{10} + x_{14} + x_{15} \geq 1; \\ x_i \in \{0, 1\} \ (\forall i \in \{1, 2, ..., 16\}). \end{cases} \quad (6.3.2)$$

Заметим, что каждой трубе на схеме соответствует ровно одно ограничение (6.3.2), которое означает, что для получения доступа к той или иной трубе необходимо поднять, по крайней мере, одну из плит, под которыми проходит данная труба. Так, например, для установки перекрытия на первой трубе необходимо поднять плиту 1 или 2, или 3, или любую их комбинацию, чему соответствует первое ограничение (6.3.2). Аналогичным образом интерпретируются и остальные 4 из первых 5 ограничений данной математической модели. Целевая функция (6.3.1) соответствует требованию минимизации общего количества приподнимаемых плит при производстве данных работ.

Примечание

Может показаться, что данная задача может быть решена методом простого выбора для каждой трубы одной плиты, которую следует поднять для установления на ней вентильного перекрытия. В этом случае количество плит будет равно количеству труб, проложенных под плитами. Для рассматриваемой задачи это число равно 5. Однако, как нетрудно проверить, это количество плит не является оптимальным решением задачи.

Математическая модель (6.3.1) и (6.3.2) относится к классу задач булева линейного программирования, которая может быть решена с помощью программы MS Excel.

6.3.2. Решение задачи водопроводчика с помощью программы MS Excel

Для решения задачи водопроводчика с помощью программы MS Excel создадим новый рабочий лист с именем Задача водопроводчика в книге Булево

программирование. Далее необходимо выполнить следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:A17**, **C1**, **C3**. Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи целочисленного линейного программирования.
2. В ячейку **C2** введем формулу: `=СУММ(B2:B17)`, которая представляет собой целевую функцию (6.3.1).
3. В ячейку **C4** введем формулу: `=B2+B3+B4`, выражающую левую часть первого ограничения (6.3.2).
4. В ячейку **C5** введем формулу: `=B3+B6+B7+B10`, выражающую левую часть второго ограничения (6.3.2).
5. В ячейку **C6** введем формулу: `=B4+B8+B9+B13+B17`, выражающую левую часть третьего ограничения (6.3.2).
6. В ячейку **C7** введем формулу: `=B7+B8+B11+B12`, выражающую левую часть четвертого ограничения (6.3.2).
7. В ячейку **C8** введем формулу: `=B10+B11+B15+B16`, выражающую левую часть пятого ограничения (6.3.2).

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи водопроводчика представлен на рис. 6.7.

БулевоПрограммирование.xls

	A	B	C	D
1	**Переменные:**		**Значение ЦФ:**	
2	*x1*		**=СУММ(B2:B17)**	
3	*x2*		**Ограничения:**	
4	*x3*		**=B2+B3+B4**	
5	*x4*		**=B3+B6+B7+B10**	
6	*x5*		**=B4+B8+B9+B13+B17**	
7	*x6*		**=B7+B8+B11+B12**	
8	*x7*		**=B10+B11+B15+B16**	
9	*x8*			
10	*x9*			
11	*x10*			
12	*x11*			
13	*x12*			
14	*x13*			
15	*x14*			
16	*x15*			
17	*x16*			
18				

Задача водопроводчика

Рис. 6.7. Исходные данные для решения задачи водопроводчика

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$C$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$B$17`.
4. Добавить 5 первых ограничений для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - нажать кнопку с надписью **Добавить** в исходном диалоговом окне **Поиск решения**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$C$4:$C$8**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";
   - в качестве значения правой части ограничения выбрать ячейку и ввести с клавиатуры число 1;
   - для добавления этой группы ограничений в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Добавить ограничение на булевы значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$B$17**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**двоичное**";

- для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

6. В окне **Параметры** поиска решения выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения представлен на рис. 6.8.

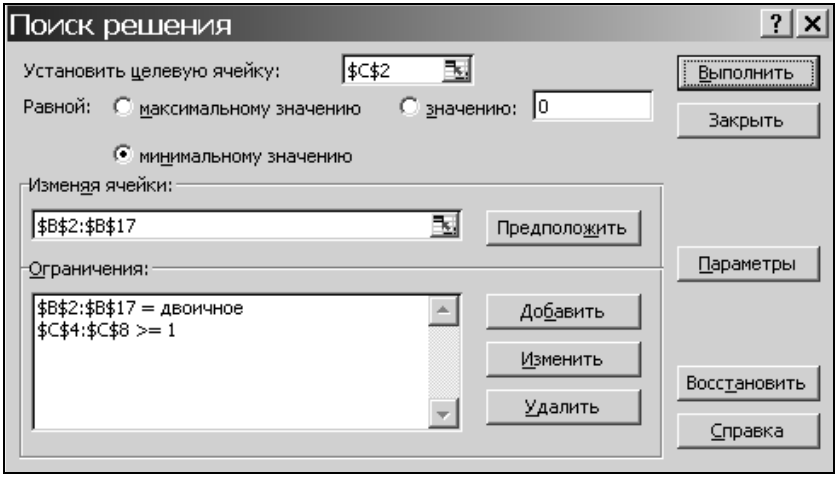


Рис. 6.8. Ограничения на значения переменных и параметры мастера поиска решения для задачи водопроводчика

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 6.9).

Результатом решения рассматриваемой задачи водопроводчика являются найденные оптимальные значения переменных: $x_3 = 1, x_7 = 1, x_9 = 1,$ остальные переменные равны 0, которым соответствует значение целевой функции: $f_{opt} = 3.$

Анализ найденного решения показывает, что при выполнении работ по установлению перекрытий на водопроводные трубы для заданной схемы их расположения (рис. 6.10) следует приподнять три плиты, а именно плиты с номерами 3, 7 и 9. При этом общее количество плит является минимальным.

Из схемы установки вентилей нетрудно заметить, что вентиль на трубу 3 можно установить как под поднятой плитой 3, так и под поднятой плитой 7.

БулевоПрограммирование.xls

	A	B	C	D	E	F	G	H
1	Переменные:		Значение ЦФ:					
2	x1	0	3					
3	x2	0	Ограничения:					
4	x3	1	1					
5	x4	0	1					
6	x5	0	2					
7	x6	0	1					
8	x7	1	1					
9	x8	0						
10	x9	1						
11	x10	0						
12	x11	0						
13	x12	0						
14	x13	0						
15	x14	0						
16	x15	0						
17	x16	0						
18								

Задача водопроводчика

Рис. 6.9. Результат количественного решения задачи водопроводчика

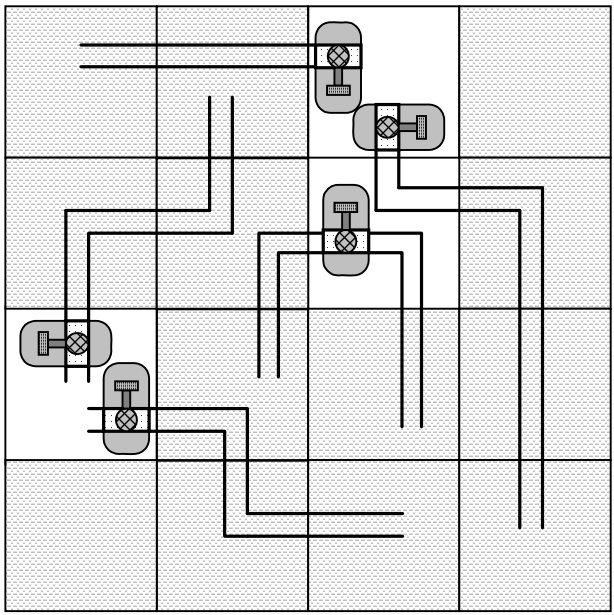


Рис. 6.10. Иллюстрация результата решения задачи водопроводчика

Тем не менее, оба эти варианта вентилей относятся к найденному оптимальному решению поднятия плит. Более сложно увидеть, что найденное решение не является единственным. Действительно, можно выбрать другой вариант поднятия плит, например, поднять 3 плиты с номерами: 3, 9 и 11, которые также удовлетворяют условиям рассматриваемой задачи.

Особенности математической модели задачи водопроводчика (6.3.1) и (6.3.2) позволяют найти аналитическое ее решение, которое может быть использовано для проверки правильности решения, найденного с помощью программы MS Excel.

6.3.3. Аналитическое решение задачи водопроводчика

Для аналитического решения сформулированной ранее задачи водопроводчика можно воспользоваться методом упрощения булевых уравнений, основанным на использовании основных свойств булевых операций сложения и умножения. В этом контексте следует заметить, что задача минимизации целевой функции (6.3.1) и (6.3.2) эквивалентна решению следующего булева уравнения с минимальным количеством ненулевых булевых переменных:

$$(x_1 \oplus x_2 \oplus x_3) \otimes (x_2 \oplus x_5 \oplus x_6 \oplus x_9) \otimes (x_3 \oplus x_7 \oplus x_8 \oplus x_{12} \oplus x_{16}) \otimes$$
$$\otimes (x_6 \oplus x_7 \oplus x_{10} \oplus x_{11}) \otimes (x_9 \oplus x_{10} \oplus x_{14} \oplus x_{15}) = 1. \qquad (6.3.3)$$

В выражении операции $\oplus$ и $\otimes$ обозначают операции булева сложения и умножения соответственно. Булево уравнение (6.3.3) эквивалентно требованию, чтобы каждое из выражений, заключенных в скобки, равнялось 1, т. е. на каждой трубе должно быть установлено вентильное перекрытие. При этом следует минимизировать общее количество переменных x_i, принимающих значение 1, поскольку каждое равенство $x_i = 1$ означает, что i-ю плиту требуется приподнять.

Аналитическое решение задачи (6.3.1) и (6.3.2) основывается на упрощении выражения левой части уравнения (6.3.3) с использованием аксиом булевой алгебры *(см. приложение 1)*.

Во-первых, следует заметить, что при раскрытии скобок в выражении (6.3.3) левая часть рассматриваемого булева уравнения будет представлять собой булеву сумму произведений переменных вида: $x_i \oplus x_j \oplus x_k \oplus x_l \oplus x_m$. Общее количество таких произведений переменных равно $3 \cdot 4 \cdot 5 \cdot 4 \cdot 4 = 960$.

Во-вторых, любое из произведений переменных вида: $x_i \oplus x_j \oplus x_k \oplus x_l \oplus x_m = 1$ является допустимым решением исходного урав-

нения, при котором значения переменных булевых равны: $x_i = 1, x_j = 1, x_k = 1, x_l = 1, x_m = 1$. Тем самым минимизация целевой функции (6.3.1) сводится к поиску произведения переменных вида: $x_i \oplus x_j \oplus x_k \oplus x_l \oplus x_m$, которое содержит минимальное количество различных булевых переменных.

Для минимизации количества этих переменных следует воспользоваться следующим тождеством: $x_i \otimes x_i = x_i$. Очевидно, это тождество можно применить только к тем произведениям, которые содержат одинаковые переменные в различных скобках левой части уравнения (6.3.3). Таких переменных сравнительно немного. Это переменная x_2, которая входит в сомножители 1 и 2; переменная x_3, которая входит в сомножители 1 и 3; переменная x_6, которая входит в сомножители 2 и 4; переменная x_7, которая входит в сомножители 3 и 4; переменная x_9, которая входит в сомножители 2 и 5, переменная x_{10}, которая входит в сомножители 4 и 5. При этом отсутствуют переменные, которые одновременно входили бы в 3 и большее число сомножителей.

Таким образом, из всех произведений вида $x_i \oplus x_j \oplus x_k \oplus x_l \oplus x_m$, можно исключить из рассмотрения те, которые содержат 5 различных переменных. Для получения оптимального решения следует выбрать переменные, которые бы входили в различные пары сомножителей. В качестве таких переменных можно взять, например, переменные x_2 и x_7. Тем самым может быть получено 4 оптимальных решения, основанных на решении уравнения: $x_2 \otimes x_7 \otimes (x_9 \oplus x_{10} \oplus x_{14} \oplus x_{15}) = 1$. Первое из этих оптимальных решений равно: $x_2 = 1, x_7 = 1, x_9 = 1$, второе — $x_2 = 1, x_7 = 1, x_{10} = 1, x_2 = 1$, третье — $x_2 = 1, x_7 = 1, x_{14} = 1$ и четвертое — $x_2 = 1, x_7 = 1, x_{15} = 1$.

Вторая группа оптимальных решений может быть получена, если в качестве искомых переменных взять переменные x_2 и x_{10}. Данным способом может быть получено еще 5 оптимальных решений, основанных на решении уравнения: $x_2 \otimes x_{10} \otimes (x_3 \oplus x_7 \oplus x_8 \oplus x_{12} \oplus x_{16}) = 1$. Первое из оптимальных решений второй группы равно: $x_2 = 1, x_{10} = 1, x_3 = 1$, второе — $x_2 = 1, x_{10} = 1, x_7 = 1$, третье $x_2 = 1, x_{10} = 1, x_8 = 1$ и т. д.

Аналогичным образом могут быть получены оптимальные решения третьей группы, если в качестве искомых переменных взять переменные x_3 и x_9, четвертой группы, если в качестве искомых переменных взять переменные x_3 и x_6, и пятой группы, если в качестве искомых переменных взять переменные x_7 и x_9.

Примечание

Определить оставшиеся оптимальные решения рассматриваемой задачи водопроводчика читателям предлагается выполнить самостоятельно в качестве упражнения.

Таким образом, рассматриваемая задача водопроводчика имеет несколько оптимальных решений, каждое из которых потребует для установки вентильных перекрытий поднятия 3-х плит. Программа MS Excel позволяет найти только одно из них, в чем проявляется некоторая ограниченность ее возможностей. Чтобы найти все оптимальные решения той или иной задачи оптимизации, можно составить для этой задачи специальную программу на языке VBA с последующим ее использованием в MS Excel. Методы написания и использования в MS Excel программ на языке VBA для решения отдельных задач оптимизации рассматриваются далее в *главах 11* и *12*.

6.4. Задача о назначении

Содержательная постановка задачи о назначении приводится в *разд. 1.2.8*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи оптимизации в форме математической модели булева линейного программирования.

6.4.1. Математическая постановка задачи о назначении

Имеется n видов работ, которые могут быть выполнены n кандидатами, где n — некоторое натуральное число. При этом каждого кандидата следует назначить на выполнение только одной работы, а каждая работа может выполняться только одним кандидатом. Заданы эффективности c_{ij} $(\forall i,\ j \in \{1, 2,..., n\})$ индивидуального выполнения каждым из потенциальных кандидатов всех рассматриваемых работ. Требуется распределить всех кандидатов по работам, так чтобы общая эффективность выполнения всех работ была наибольшей.

Введем в рассмотрение следующие булевы переменные: x_{ij}, которые будут соответствовать назначению кандидатов на выполнение работ. В этом случае резонно предположить, что $x_{ij} = 1$, если i-й кандидат назначается на выполнение j-й работы, и $x_{ij} = 0$, если i-й кандидат не назначается на выполнение

j-й работы. Тогда математическая постановка *задачи о назначении* может быть сформулирована следующим образом.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to \max_{x\in\Delta_\beta}, \tag{6.4.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = 1, \ \forall i \in \{1, 2,..., n\}; & (6.4.2) \\ \sum_{i=1}^{n} x_{ij} = 1, \ \forall j \in \{1, 2,..., n\}; & (6.4.3) \\ x_{ij} \in \{0, 1\}, (\forall i, j \in \{1, 2,..., n\}). & (6.4.4) \end{cases}$$

В математической постановке задачи о назначении (6.4.1)—(6.4.4) ограничение (6.4.2) соответствует требованию назначения каждого кандидата на выполнение только одной работы, а ограничение (6.4.3) — требованию выполнения каждой работы только одним кандидатом. Нетрудно заметить, что общее число булевых переменных задачи о назначении равно: n^2.

Классическая задача о назначении является *симметричной*, т. е. формулируется в форме, когда имеет место равенство общего числа работ и общего числа кандидатов. Если это условие не выполняется, то задача о назначении называется *несимметричной*. Однако при решении практических задач о назначении с помощью программы MS Excel данную особенность можно не принимать во внимание.

Хотя классическая задача о назначении формулируется как задача максимизации целевой функции (6.4.1), она достаточно просто может быть преобразована в эквивалентную задачу с минимизацией целевой функции. Для этого необходимо все значения c_{ij} умножить на –1, после чего полученные значения сложить с достаточно большим положительным числом, так чтобы все c_{ij} были положительными. Полученную подобным образом задачу о назначении с минимизацией целевой функции можно рассматривать как специальный случай транспортной задачи, когда объемы пунктов производства и потребности пунктов потребления равны 1.

Если вместо эффективности выполнения работ кандидатами рассматриваются затраты, связанные с назначением кандидатов на выполнение работ,

то исходная задача о назначении будет соответствовать задаче минимизации целевой функции (6.4.1). В итоге задача о назначении формулируется, как задача поиска оптимального назначения, которому соответствует минимум общих затрат на выполнение работ кандидатами. При этом ограничения задачи (6.4.2)—(6.4.4) остаются без изменения.

Нетрудно заметить, что любое допустимое решение задачи о назначении удобно представить в форме так называемой *булевой матрицы*, представляющей собой матрицу размерности $(n \times n)$, элементы которой принимают только двоичные значения 0 и 1. При этом булева матрица допустимого решения должна удовлетворять следующему условию: в каждой строке и каждом столбце этой матрицы должно находиться в точности по одной 1, а все остальные элементы матрицы равны 0. Очевидно, общее число единиц в матрице допустимого решения равно n.

С другой стороны, любое допустимое решение симметричной задачи о назначении можно также представить в форме специального объекта комбинаторики — так называемой *перестановки (см. приложение 1)*. Действительно, если исходному множеству работ поставить в соответствие подмножество натуральных чисел: $I = \{1, 2,..., n\}$, то любую перестановку этих чисел $(a_1, a_2,..., a_n)$ можно интерпретировать как назначение i-го кандидата на выполнение a_i-й работы или, что эквивалентно, назначение a_i-го кандидата на выполнение i-й работы. Отсюда следует важный вывод: общее число возможных назначений в задаче о назначении конечно и равно $n!$

6.4.2. Решение задачи о назначении с помощью программы MS Excel

Для решения задачи о назначении с помощью программы MS Excel необходимо задать конкретные значения параметрам. Для определенности рассмотрим вариант задачи о назначении в форме минимизации общих затрат на выполнение работ. В этом случае в качестве кандидатов рассмотрим сотрудников некоторой фирмы: Андреев, Бубнов, Васильев, Григорьев и Дмитриев, а в качестве работ — вакантные должности в этой фирме: менеджер, программист, бизнес-аналитик, маркетолог и руководитель проектов. Затраты $\tilde{n}_{ij}$ $(\forall i,\ j \in \{1, 2, 3, 4, 5\})$ на замещение должностей кандидатами, связанные с необходимостью их предварительного обучения и стажировки, заданы в форме следующей таблицы (табл. 6.1).

Таблица 6.1. *Затраты на замещение должностей кандидатами (в тыс. рублей)*

Кандидаты/ Должности	**Андреев**	**Бубнов**	**Васильев**	**Григорьев**	**Дмитриев**
Менеджер	5	10	9	14	6
Программист	13	15	11	19	17
Бизнес-аналитик	7	14	12	8	10
Маркетолог	8	11	6	7	9
Руководитель проектов	15	12	17	13	16

Соответствующая математическая постановка рассматриваемой индивидуальной задачи о назначении может быть записана в следующем виде:

$$\begin{aligned} &5x_{11}+10x_{12}+9x_{13}+14x_{14}+6x_{15}+\\ &+13x_{21}+15x_{22}+11x_{23}+19x_{24}+17x_{25}+\\ &+7x_{31}+14x_{32}+12x_{33}+8x_{34}+10x_{35}+\\ &+8x_{41}+11x_{42}+6x_{43}+7x_{44}+9x_{45}+\\ &15x_{51}+12x_{52}+17x_{53}+13x_{54}+16x_{55}\to\min\limits_{x\in\Delta_\beta}, \end{aligned} \tag{6.4.5}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств:

$$\begin{cases} x_{11}+x_{12}+x_{13}+x_{14}+x_{15}=1;\\ x_{21}+x_{22}+x_{23}+x_{24}+x_{25}=1;\\ x_{31}+x_{32}+x_{33}+x_{34}+x_{35}=1;\\ x_{41}+x_{42}+x_{43}+x_{44}+x_{45}=1;\\ x_{51}+x_{52}+x_{53}+x_{54}+x_{55}=1;\\ x_{11}+x_{21}+x_{31}+x_{41}+x_{51}=1;\\ x_{12}+x_{22}+x_{32}+x_{42}+x_{52}=1;\\ x_{13}+x_{23}+x_{33}+x_{43}+x_{53}=1;\\ x_{14}+x_{24}+x_{34}+x_{44}+x_{54}=1;\\ x_{15}+x_{25}+x_{35}+x_{45}+x_{55}=1;\\ x_{ij}\in\{0,1\}\ (\forall i,j\in\{1,2,3,4,5\}). \end{cases} \tag{6.4.6}$$

Заметим, что первые 5 ограничений данной задачи соответствуют общему ограничению (6.4.2), следующие 5 ограничений — общему ограничению (6.4.3), а последнее ограничение — общему ограничению (6.4.4).

Для решения сформулированной индивидуальной задачи о назначении с помощью программы MS Excel создадим в книге Булево программирование новый лист и изменим его имя на Задача о назначении. Для решения задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A7:A13**, **B1**, **G1**, **B7:G7**, как это изображено на рис. 6. 11. Следует отметить, что конкретное содержание этих надписей не оказывает никакого влияния на решение рассматриваемой задачи о назначении.
2. В ячейки **B2:F6** введем значения коэффициентов целевой функции (табл. 6.1).
3. В ячейку **G2** введем формулу: `=СУММПРОИЗВ(B2:F6; B8:F12)`, которая представляет целевую функцию (6.4.5).
4. В ячейку **G8** введем формулу: `=СУММ(B8:F8)`, которая представляет первое ограничение (6.4.6).
5. Скопируем формулу, введенную в ячейку **G8**, в ячейки **G9:G12**.
6. В ячейку **B13** введем формулу: `=СУММ(B8:B12)`, которая представляет шестое ограничение (6.4.6).
7. Скопируем формулу, введенную в ячейку **B13**, в ячейки **C13:F13**.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о назначении представлен на рис. 6.11.

БулевоПрограммирование.xls

	A	B	C	D	E	F	G
1		Коэффициенты целевой функции:					Значение ЦФ:
2		5	10	9	14	6	=СУММПРОИЗВ(B2:F6;B8:F12)
3		13	15	11	19	17	
4		7	14	12	8	10	
5		8	11	6	7	9	
6		15	12	17	13	16	
7	**Переменные:**	*Xi1*	*Xi2*	*Xi3*	*Xi4*	*Xi5*	Значения ограничений:
8	*X1j*						=СУММ(B8:F8)
9	*X2j*						=СУММ(B9:F9)
10	*X3j*						=СУММ(B10:F10)
11	*X4j*						=СУММ(B11:F11)
12	*X5j*						=СУММ(B12:F12)
13	Значения	=СУММ(B8:B12)	=СУММ(C8:C12)	=СУММ(D8:D12)	=СУММ(E8:E12)	=СУММ(F8:F12)	
14							
15							
16							
17							
18							
19							

Задача о назначении

Рис. 6.11. Исходные данные для решения задачи о назначении

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$G$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес диапазона ячеек `$B$8:$F$12`.
4. Задать первую группу ограничений, соответствующих первым 5 базовым ограничениям исходной постановки решаемой задачи о назначении (6.4.6). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$G$8:$G$12**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры число 1;
   - для добавления первой группы ограничений в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Задать вторую группу ограничений, соответствующих оставшимся 5 базовым ограничениям исходной постановки решаемой задачи о назначении (6.4.6). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$13:$F$13**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры число 1;

- для добавления второй группы ограничений в дополнительном окне нажать кнопку с надписью **Добавить**.

6. Добавить последнее ограничение на булевы значения переменных задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$8:$F$12**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**двоичное**";
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
7. В окне дополнительных параметров поиска решения выбрать отметку **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения представлен на рис. 6.12.

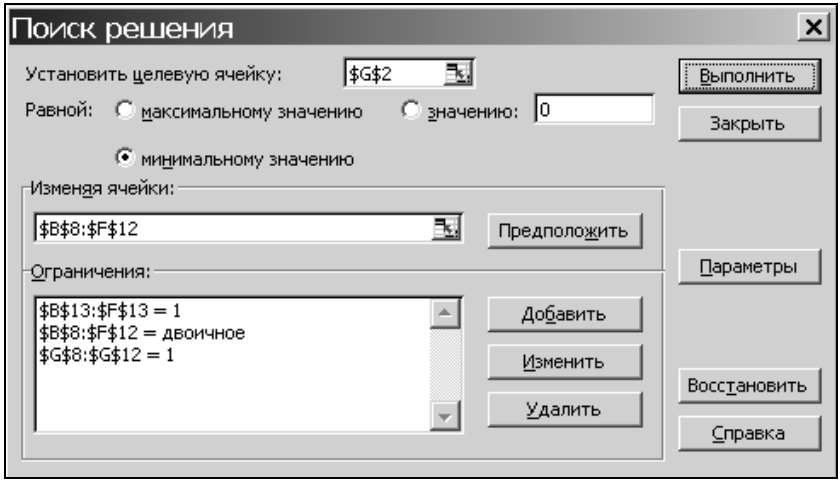
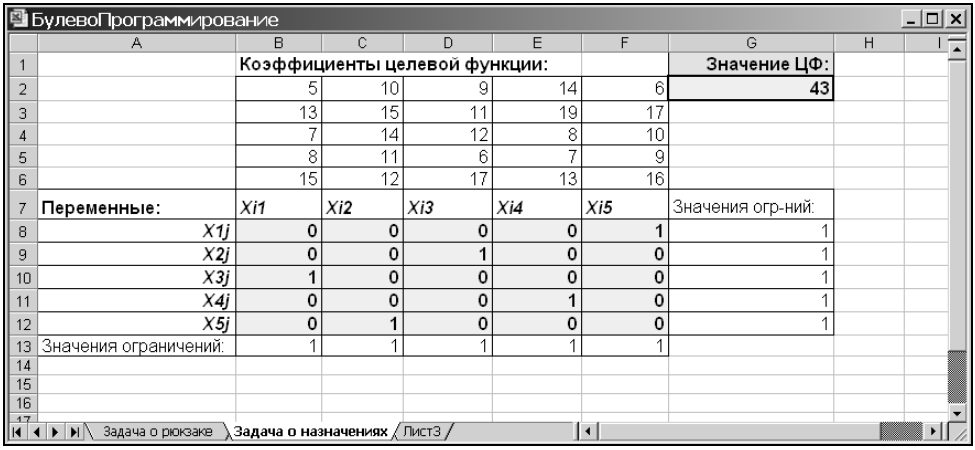


Рис. 6.12. Параметры мастера поиска решения и базовые ограничения для задачи о назначении

Примечание

Следует заметить, что некоторые специалисты при решении задач оптимизации с помощью программы MS Excel рекомендуют значения правых частей ограничений задавать явно и вводить из ячеек рабочего листа. Однако желание сократить количество операций по заданию ограничений за счет ввода целого интервала ячеек в качестве левой части ограничений позволяет обойти вниманием эту рекомендацию и вводить значение с клавиатуры.

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 6.13).



БулевоПрограммирование

	A	B	C	D	E	F	G	H	I
1		**Коэффициенты целевой функции:**					**Значение ЦФ:**		
2		5	10	9	14	6	**43**		
3		13	15	11	19	17			
4		7	14	12	8	10			
5		8	11	6	7	9			
6		15	12	17	13	16			
7	**Переменные:**	*Xi1*	*Xi2*	*Xi3*	*Xi4*	*Xi5*	Значения огр-ний:		
8	***X1j***	**0**	**0**	**0**	**0**	**1**	1		
9	***X2j***	**0**	**0**	**1**	**0**	**0**	1		
10	***X3j***	**1**	**0**	**0**	**0**	**0**	1		
11	***X4j***	**0**	**0**	**0**	**1**	**0**	1		
12	***X5j***	**0**	**1**	**0**	**0**	**0**	1		
13	Значения ограничений:	1	1	1	1	1			
14									
15									
16									

Задача о рюкзаке | **Задача о назначениях** | Лист3

Рис. 6.13. Результат количественного решения задачи о назначении

Результатом решения рассматриваемой задачи о назначении являются найденные оптимальные значения переменных: $x_{15} = 1$, $x_{23} = 1$, $x_{31} = 1$, $x_{44} = 1$, $x_{52} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствует минимальное значение целевой функции: $f_{opt} = 43$.

Анализ найденного решения показывает, что при замещении вакантных должностей в рассматриваемой фирме следует на должность менеджера назначить сотрудника Дмитриева, на должность программиста — Васильева, на должность бизнес-аналитика — Андреева, на должность маркетолога — Григорьева и, наконец, на должность руководителя проектов — Бубнова. При этом общие затраты на обучение и стажировку сотрудников составят 43 *тыс. рублей.*

Примечание

Найденное оптимальное решение в точности соответствует исходной постановке задачи. Для проверки правильности найденного решения можно воспользоваться либо ручным просчетом, используя для этого один из специальных алгоритмов решения задачи о назначении, например, алгоритм венгерского метода, либо другую программу, например, MATLAB или Mathcad. Заинтересованные читатели могут выполнить это в качестве упражнения. С другими вариантами задачи о назначении можно познакомиться в дополнительной литературе, список которой приводится в конце книги.

6.4.3. Решение задачи о назначении с помощью венгерского метода

Для оценки точности и правильности результатов решения задачи о назначении, полученного с помощью программных средств, можно воспользоваться специально разработанным для решения задач данного класса *венгерским* методом. Основная идея этого метода основывается на следующем свойстве матрицы затрат (табл. 6.1). Если ко всем элементам c_{ij} некоторой i-й строки прибавить произвольное постоянное число u_i, и ко всем элементам j-го столбца также прибавить произвольное постоянное число v_j, то в этом случае будет получена новая матрица затрат с элементами: $d_{ij} = c_{ij} + u_i + v_j$ $(\forall i,\ j \in \{1, 2,..., n\})$. При этом элементы прежней матрицы будут равны: $c_{ij} = d_{ij} - u_i - v_j$ $(\forall i,\ j \in \{1, 2,..., n\})$. Умножив обе части последнего выражения на x_{ij} и учитывая ограничения (6.4.2) и (6.4.3), можно получить следующее выражение:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} - \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j. \qquad (6.4.7)$$

Отсюда с очевидностью следует, что задача минимизации целевой функции (6.4.1) при ограничениях (6.4.2)—(6.4.4) оказывается эквивалентной задаче минимизации некоторой модифицированной целевой функции с этой же системой ограничений. При этом модифицированная целевая функция определяется следующим образом:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \to \min_{x\in\Delta_\beta}. \qquad (6.4.8)$$

Тем самым идея нахождения оптимального решения задачи о назначении венгерским методом заключается в том, чтобы последовательно вычитать из

элементов каждой строки и каждого столбца исходной матрицы затрат их наименьшие элементы. После чего анализируется получение допустимого решения. Если получено допустимое решение, которому соответствуют нули в модифицированной матрице затрат, то оно является оптимальным назначением. Если же решение недопустимое, то выполняется дальнейшая модификация матрицы затрат.

В общем случае алгоритм венгерского метода имеет итеративный характер и заключается в выполнении следующих шагов:

1. *Редукция строк и столбцов*. Для задачи о назначении в симметрической форме (6.4.1)—(6.4.4) из всех элементов каждой строки исходной матрицы затрат вычитается наименьший элемент этой строки, после чего из всех элементов каждого столбца полученной матрицы затрат вычитается наименьший элемент этого столбца. Тем самым получается некоторая редуцированная матрица затрат задачи о назначении с элементами d_{ij}, нулевые значения которых будут соответствовать возможным назначениям кандидатов на выполнение работ. Другими словами, если $d_{ij} = 0$, то это будет соответствовать $x_{ij} = 1$ и наоборот, если $d_{ij} ? 0$, то это будет соответствовать $x_{ij} = 0$.

2. *Анализ допустимости полученного решения*. Он заключается в проверке условий ограничений (6.4.2) и (6.4.3) для соответствующих значений переменных x_{ij}. Для редуцированной матрицы затрат анализ допустимости полученного решения заключается в проверке условий ограничений наличия в каждой ее строке и каждом столбце соответствующих нулевых элементов. При этом нулевые элементы должны быть расположены специальным образом. Чтобы убедиться в этом, следует выполнить отдельную **процедуру вычеркивания строк и столбцов**, которая описывается далее, после описания основного алгоритма. Если после выполнения данной процедуры будет получено полное допустимое назначение, то оно является оптимальным и принимается за искомое решение задачи о назначении. В этом случае выполнение алгоритма заканчивается. Если же после выполнения данной процедуры будет получено неполное назначение, а значит, — и недопустимое решение, то следует перейти к выполнению шага 3.

3. *Модификация редуцированной матрицы затрат*. Если полученное решение не является допустимым, т. е. в редуцированной матрице затрат недостаточно нулевых элементов для получения полного назначения, то необходимо модифицировать эту матрицу с целью получения новых нулевых элементов. Для этого можно использовать описанную далее отдельную

процедуру редукции ненулевых элементов, которая позволяет провести минимальное число прямых через строки и столбцы редуцированной матрицы, так чтобы все ее нулевые элементы оказались вычеркнутыми. После выполнения данной процедуры будет получена новая редуцированная матрица затрат с бóльшим количеством нулей, которой будет соответствовать новое возможное назначение. Для проверки допустимости этого решения следует перейти к выполнению шага 2 данного алгоритма.

Процедура вычеркивания строк и столбцов, которая используется на шаге 2 алгоритма венгерского метода, заключается в итеративном выполнении следующей последовательности шагов:

1. Найти в редуцированной матрице строку, содержащую ровно один невычеркнутый нулевой элемент (первоначально предполагается, что все строки и столбцы редуцированной матрицы затрат является невычеркнутыми). Для найденной строки производится назначение, соответствующее данному невычеркнутому нулевому элементу, в результате чего вся строка и соответствующий нулевому элементу столбец вычеркиваются. При этом строки удобно рассматривать в порядке возрастания, что, впрочем, не оказывает принципиального влияния на работу алгоритма.
2. Найти в редуцированной матрице столбец, содержащий ровно один невычеркнутый нулевой элемент. Для найденного столбца производится назначение, соответствующее данному невычеркнутому нулевому элементу, в результате чего весь столбец и соответствующая нулевому элементу строка вычеркиваются. При этом столбцы также удобно рассматривать в порядке возрастания, что, впрочем, не оказывает принципиального влияния на работу алгоритма.
3. Шаги 1 и 2 данной процедуры следует выполнять до тех пор, пока в редуцированной матрице затрат не останется строк или столбцов, содержащих невычеркнутые нулевые элементы. Если все строки и столбцы оказались вычеркнутыми, то назначение следует признать полным, а соответствующее решение задачи о назначении — допустимым и, следовательно, оптимальным. В противном случае назначение следует признать неполным, а соответствующее решение задачи о назначении — недопустимым и, следовательно, неоптимальным. На этом данная процедура заканчивается, в результате чего может быть получен ответ на вопрос о допустимости полученного решения задачи о назначении, которому соответствует редуцированная матрица затрат.

Процедура редукции ненулевых элементов, которая используется на шаге 3 алгоритма венгерского метода, заключается в итеративном выполнении следующей последовательности шагов:

1. Вычислить количество нулей в каждой невычеркнутой строке и каждом невычеркнутом столбце редуцированной матрицы затрат. Следует заме-

тить, что все элементы исходной редуцированной матрицы затрат предполагаются невычеркнутыми.

2. Вычеркнуть строку или столбец с максимальным количеством нулей. В случае равенства максимального количества нулей для нескольких строк или столбцов, можно вычеркнуть любую из них строку или столбец, поскольку это не оказывает принципиального влияния на работу алгоритма.
3. Шаги 1 и 2 данной процедуры следует выполнять до тех пор, пока в редуцированной матрице затрат не будут вычеркнуты все нулевые элементы.
4. Из всех невычеркнутых ненулевых элементов вычесть минимальный невычеркнутый элемент и прибавить его к каждому элементу, расположенному на пересечении каждой пары линий вычеркивания. На этом данная процедура заканчивается, в результате чего будет получена необходимая модификация редуцированной матрицы затрат.

Нетрудно заметить, что в силу конечности исходной матрицы затрат, рассмотренные процедуры имеют конечный характер, и, следовательно, алгоритм венгерского метода в целом также является конечным.

Примечание

Если исходная задача о назначении не является симметричной, то она преобразуется к симметричному виду (6.4.1)—(6.4.4) посредством последовательного введения дополнительных переменных $\{x_{m+1,j}\}$ и достаточно больших значений затрат $\{c_{m+1,j}\}$ $(\forall j \in \{1, 2, ..., n\})$, соответствующих дополнительной фиктивной $(m+1)$-работе, если количество работ m меньше количества кандидатов n. Или посредством введения дополнительных переменных $\{x_{m+1,j}\}$ и достаточно больших значений затрат $\{c_{i,m+1}\}$ $(\forall i \in \{1, 2, ..., n\})$, соответствующих дополнительному фиктивному $(m+1)$-кандидату, если количество кандидатов m меньше количества работ n.

Рассмотренный алгоритм венгерского метода может быть изображен графически в форме диаграммы деятельности языка UML (рис. 6.14).

Проиллюстрируем использование рассмотренного алгоритма венгерского метода для решения индивидуальной задачи о назначении (6.4.5) и (6.4.6). Поскольку исходная задача является симметричной, то введения каких-либо дополнительных переменных не требуется. Для решения данной задачи о назначении воспользуемся программой MS Excel.

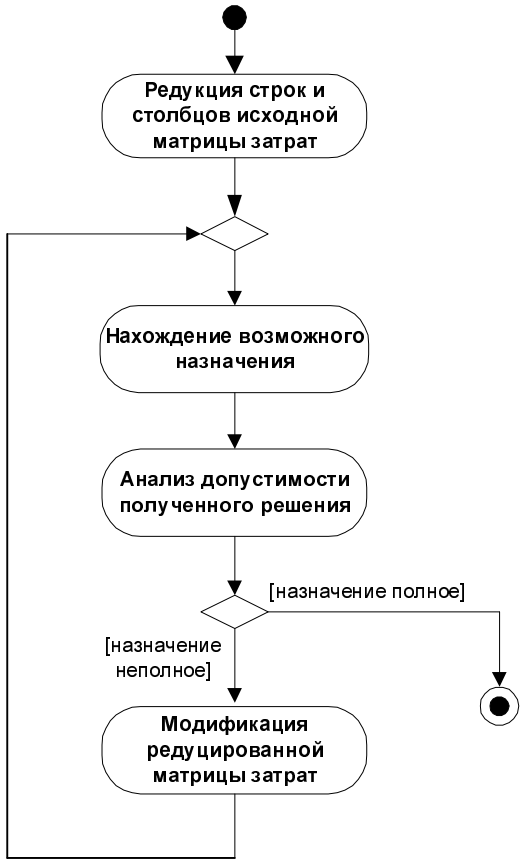
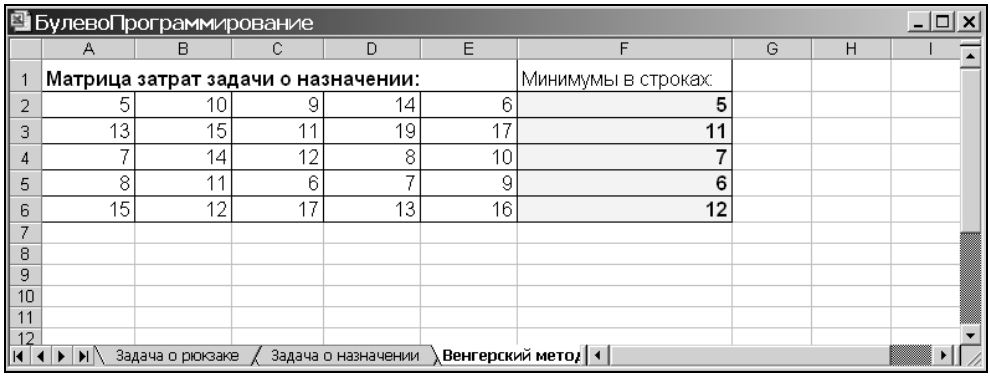


Рис. 6.14. Диаграмма деятельности алгоритма венгерского метода

С этой целью в книге Булево программирование создадим новый рабочий лист с именем Венгерский метод. Далее следует выполнить следующие действия:

1. Внесем необходимые надписи в ячейки **A1** и **F1**, как это изображено на рис. 6.15. Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи о назначении венгерским методом.
2. В ячейки **A2:E6** введем значения коэффициентов целевой функции (табл. 6.1), которые образуют исходную матрицу затрат, необходимую для выполнения операций алгоритма венгерского метода.
3. В ячейку **F2** введем формулу: `=МИН(A2:E2)`, которая позволяет найти минимальный элемент в первой строке матрицы затрат.
4. Скопируем формулу, введенную в ячейку **F2**, в ячейки **F3:F6**.

После выполнения этих действий в ячейках **F2:F6** будут записаны минимальные значения элементов каждой из строк. Внешний вид рабочего листа MS Office Excel 2003 с исходными данными, необходимыми для решения задачи о назначении венгерским методом на первой итерации, представлен на рис. 6.15.



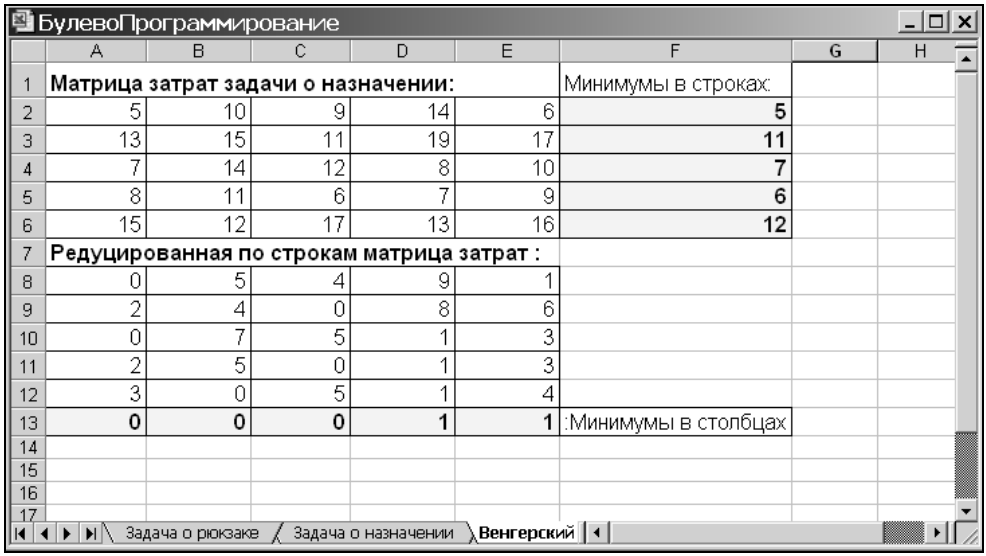
	A	B	C	D	E	F
1	**Матрица затрат задачи о назначении:**					Минимумы в строках:
2	5	10	9	14	6	**5**
3	13	15	11	19	17	**11**
4	7	14	12	8	10	**7**
5	8	11	6	7	9	**6**
6	15	12	17	13	16	**12**

Рис. 6.15. Исходные данные для решения задачи о назначении венгерским методом

Далее в соответствии с описанным алгоритмом венгерского метода найденные минимальные элементы строк следует вычесть из каждой строки. Для получения редуцированной по строкам матрицы затрат на первой итерации алгоритма венгерского метода следует выполнить следующие действия:

1. Внесем необходимые надписи в ячейки **A7** и **F13**, как это изображено на рис. 6.9.
2. В ячейку **A8** введем формулу: `=A2-$F2`, которая позволяет вычесть минимальный элемент первой строки матрицы затрат из первого элемента этой же строки.
3. Скопируем формулу, введенную в ячейку **A8**, в ячейки **B8:E8** и **A9:E12**.
4. В ячейку **A13** введем формулу: `=МИН(A8:A12)`, которая позволяет найти минимальный элемент в первом столбце редуцированной по строкам матрицы затрат.
5. Скопируем формулу, введенную в ячейку **A13**, в ячейки **B13:E13**.

После выполнения этих действий в ячейках **A13:E13** будут записаны минимальные значения элементов каждой из строк. Внешний вид рабочего листа MS Office Excel 2003 с промежуточными результатами решения задачи о назначении венгерским методом на первой итерации представлен на рис. 6.16.



	A	B	C	D	E	F	G	H
1	**Матрица затрат задачи о назначении:**					Минимумы в строках:		
2	5	10	9	14	6	**5**		
3	13	15	11	19	17	**11**		
4	7	14	12	8	10	**7**		
5	8	11	6	7	9	**6**		
6	15	12	17	13	16	**12**		
7	**Редуцированная по строкам матрица затрат :**							
8	0	5	4	9	1			
9	2	4	0	8	6			
10	0	7	5	1	3			
11	2	5	0	1	3			
12	3	0	5	1	4			
13	**0**	**0**	**0**	**1**	**1**	:Минимумы в столбцах		
14								
15								
16								
17								

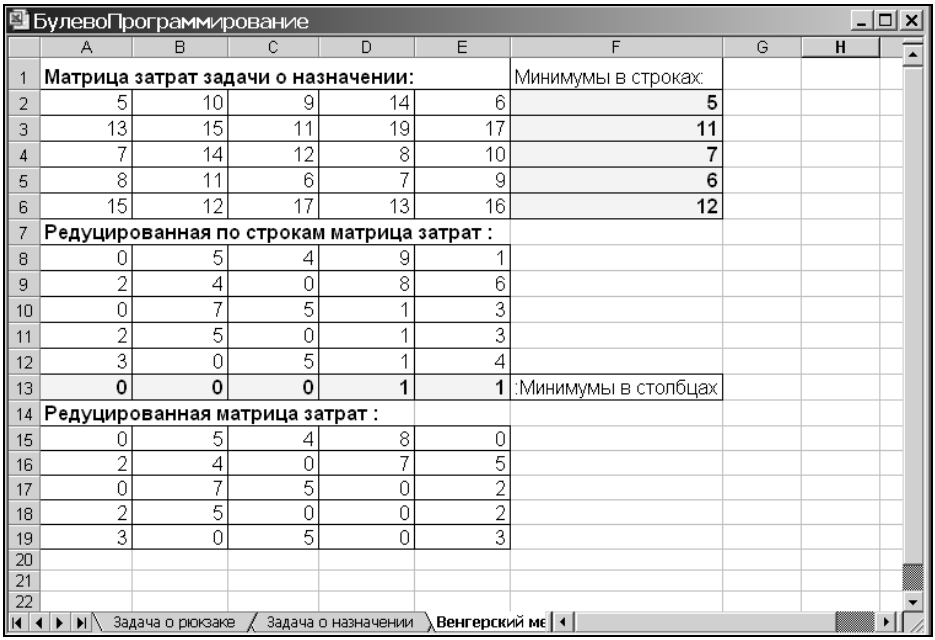
Рис. 6.16. Промежуточные результаты решения задачи о назначении венгерским методом на первой итерации

Для получения окончательной редуцированной матрицы затрат на первой итерации алгоритма венгерского метода следует выполнить следующие действия:

1. Внесем необходимую надпись в ячейку **A14**, как это изображено на рис. 6.10.
2. В ячейку **A15** введем формулу: `=A8-A$13`, которая позволяет вычесть минимальный элемент первого столбца редуцированной по строкам матрицы затрат из первого элемента этой же строки.
3. Скопируем формулу, введенную в ячейку **A15**, в ячейки **B15:E15** и **A16:E19**.

После выполнения этих действий в ячейках **A15:E19** будут записаны значения элементов редуцированной по строкам и столбцам матрицы затрат. Внешний вид рабочего листа MS Office Excel 2003 с редуцированной матрицей затрат задачи о назначении венгерским методом на первой итерации будет иметь следующий вид (рис. 6.17).

Далее на первой итерации алгоритма венгерского метода необходимо выполнить процедуру вычеркивания строк и столбцов. С этой целью последовательно вычеркиваются: вторая строка и третий столбец, четвертая строка и четвертый столбец, третья строка и первый столбец, первая строка и пятый столбец, пятая строка и второй столбец. Заметим, что операцию вычеркивания

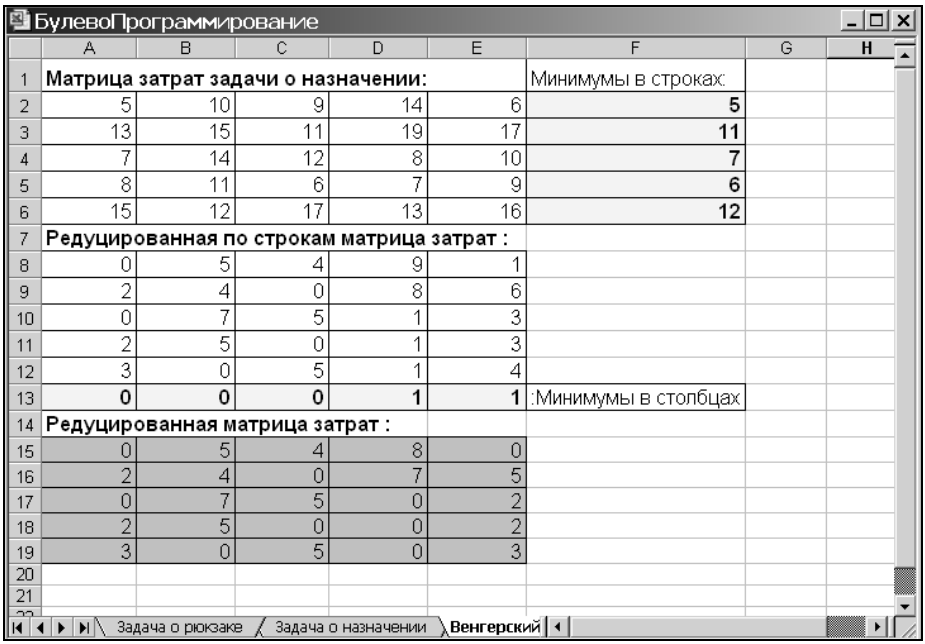


БулевоПрограммирование

	A	B	C	D	E	F	G	H
1	**Матрица затрат задачи о назначении:**					Минимумы в строках:		
2	5	10	9	14	6	**5**		
3	13	15	11	19	17	**11**		
4	7	14	12	8	10	**7**		
5	8	11	6	7	9	**6**		
6	15	12	17	13	16	**12**		
7	**Редуцированная по строкам матрица затрат :**							
8	0	5	4	9	1			
9	2	4	0	8	6			
10	0	7	5	1	3			
11	2	5	0	1	3			
12	3	0	5	1	4			
13	**0**	**0**	**0**	**1**	**1**	:Минимумы в столбцах		
14	**Редуцированная матрица затрат :**							
15	0	5	4	8	0			
16	2	4	0	7	5			
17	0	7	5	0	2			
18	2	5	0	0	2			
19	3	0	5	0	3			
20								
21								
22								

Задача о рюкзаке | Задача о назначении | Венгерский ме

Рис. 6.17. Редуцированная матрица затрат задачи о назначении на первой итерации



БулевоПрограммирование

	A	B	C	D	E	F	G	H
1	**Матрица затрат задачи о назначении:**					Минимумы в строках:		
2	5	10	9	14	6	**5**		
3	13	15	11	19	17	**11**		
4	7	14	12	8	10	**7**		
5	8	11	6	7	9	**6**		
6	15	12	17	13	16	**12**		
7	**Редуцированная по строкам матрица затрат :**							
8	0	5	4	9	1			
9	2	4	0	8	6			
10	0	7	5	1	3			
11	2	5	0	1	3			
12	3	0	5	1	4			
13	**0**	**0**	**0**	**1**	**1**	:Минимумы в столбцах		
14	**Редуцированная матрица затрат :**							
15	0	5	4	8	0			
16	2	4	0	7	5			
17	0	7	5	0	2			
18	2	5	0	0	2			
19	3	0	5	0	3			
20								
21								

Задача о рюкзаке | Задача о назначении | Венгерский

Рис. 6.18. Результат вычеркивания строк и столбцов
в редуцированной матрице затрат на первой итерации

в программе MS Office Excel удобно выполнять посредством заливки серым цветом соответствующих ячеек редуцированной матрицы затрат. Внешний вид рабочего листа MS Office Excel 2003 после вычеркивания строк и столбцов в редуцированной матрице затрат на первой итерации будет иметь следующий вид (рис. 6.18).

Поскольку в полученной после вычеркивания строк и столбцов редуцированной матрице затрат на первой итерации отсутствуют невычеркнутые элементы, то найденное назначение является полным, а соответствующее решение — допустимым. Этой последовательности вычеркнутых строк и столбцов соответствуют следующие значения переменных: $x_{15} = 1$, $x_{23} = 1$, $x_{31} = 1$, $x_{44} = 1$, $x_{52} = 1$, остальные переменные равны 0. Поскольку данное назначение является полным, поэтому найденное решение является оптимальным. На этом выполнение алгоритма венгерского метода может быть закончено. Найденному оптимальному решению соответствует минимальное значение целевой функции: $f_{opt} = 43$. Сравнение найденных оптимальных решений задачи о назначении с помощью программы MS Excel и алгоритма венгерского метода показывают их полное совпадение, что может свидетельствовать о достоверности соответствующих результатов.

6.5. Упражнения

В качестве упражнений для самостоятельного решения предлагаются задачи, аналогичные типовым задачам оптимизации, рассмотренным в данной главе. Они содержат конкретные значения исходных данных, что позволяет получить их количественное решение с помощью программы MS Excel. Те из читателей, кто сочтет для себя интересным найти решение данных задач несколькими способами, получат возможность убедиться в правильности полученных решений.

6.5.1. Двумерная задача о рюкзаке

Имеются 7 грузовых контейнеров: красный, оранжевый, желтый, зеленый, голубой, синий и фиолетовый, которые предполагается транспортировать в трюме морского судна. Каждый из контейнеров имеет 2 характеристики: масса контейнера в *тоннах* и объем в $м^3$. Дополнительно известна стоимость каждого из контейнеров (например, в *тыс. рублей*). Значения этих характеристик для рассматриваемых контейнеров представлены в следующей таблице (табл. 6.2).

Таблица 6.2. *Значения отдельных характеристик для контейнеров*

Контейнер/ Характеристика	Красный	Оранжевый	Желтый	Зеленый	Голубой	Синий	Фиолетовый
Масса	10	12	15	18	16	11	20
Объем	26	45	25	35	24	28	32
Стоимость	20	45	40	50	30	25	35

Общая грузоподъемность судна равна 60 *т*, а максимальный объем трюма ограничен величиной 120 $м^3$. Требуется определить те контейнеры, которые следует погрузить в трюм морского судна, так чтобы обеспечить максимальную стоимость общего груза и при этом не допустить перегрузки судна и превышения объема трюма.

6.5.2. Задача водопроводчика

Для конкретного плана прокладки труб под землей в пределах района производства ремонтных работ (рис. 6.19) определить минимальное число плит, которые требуется приподнять, чтобы установить по одному вентильному перекрытию на каждой трубе.

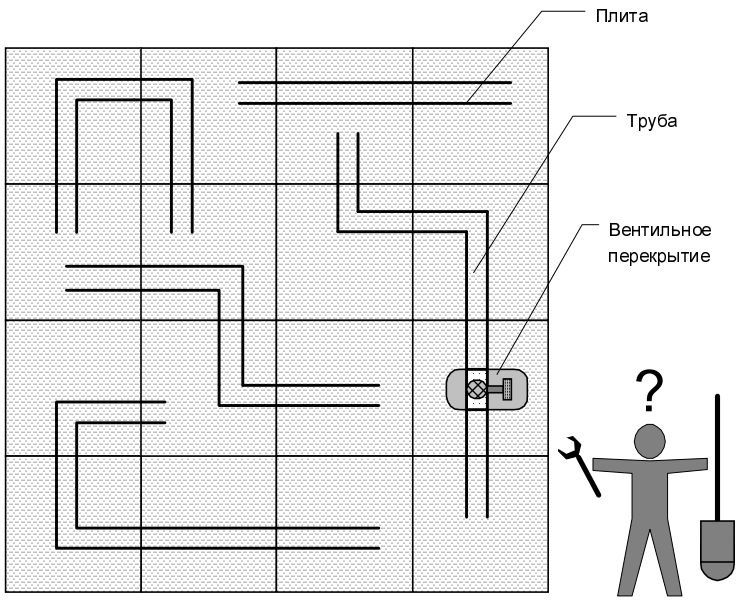


Рис. 6.19. План прокладки труб в задаче водопроводчика

Примечание

В качестве более сложной задачи водопроводчика предлагается для данной схемы прокладки труб определить минимальное число плит, которые требуется приподнять, чтобы установить по два вентильных перекрытия на каждой трубе. При этом оба вентиля для одной трубы не могут устанавливаться под одной плитой.

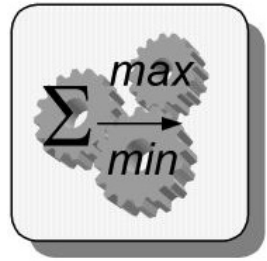
6.5.3. Задача о назначении

Рассмотрим несимметричную задачу о назначении, когда количество кандидатов превышает количество вакантных должностей. В этом случае в качестве кандидатов выступают рекруты под отдельными номерами, а в качестве работ — вакантные должности в некоторой фирме, например, менеджер, программист, бизнес-аналитик. Затраты на замещение этих должностей кандидатами, связанные с необходимостью их предварительного обучения и стажировки, заданы в следующей таблице (табл. 6.3).

Таблица 6.3. *Затраты на замещение должностей рекрутами (в тыс. рублей)*

Рекруты/ Должности	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Менеджер	9	10	7	14	6	12	8	6
Программист	11	15	13	11	17	15	12	20
Бизнес-аналитик	7	14	12	8	10	13	19	7

Требуется из всех кандидатов на замещение вакантных должностей выбрать трех рекрутов, так чтобы общие затраты на их предварительное обучение и стажировку были наименьшими.



Глава 7

Задачи оптимизации на графах

Многие прикладные задачи оптимизации могут быть сформулированы в форме той или иной задачи оптимизации на графах. Наряду с этим в теории графов многие интересные задачи связаны с решением задач оптимизации. В связи с этим изучение общих свойств задач оптимизации на графах приобретает самостоятельное значение, а изучение методов их решения традиционно относят к необходимым элементам современного образования, формирующим так называемый алгоритмический способ мышления.

Из достаточно большого числа типовых задач оптимизации на графах в настоящей главе рассматриваются только те, которые стали в некотором смысле классическими для задач данного класса:

- задача нахождения оптимальных покрывающих деревьев;
- задача нахождения кратчайшего пути в графе;
- задача нахождения критического пути в сетевом графе;
- задача нахождения максимального потока в графе.

Для каждой из них представлена математическая постановка задачи в форме модели булева или целочисленного программирования, которая является необходимым элементом их решения с помощью программы MS Excel. В то же время приводится описание специальных алгоритмов их решения, которые учитывают специфические особенности постановки этих задач оптимизации на графах.

Рассмотрение специальных алгоритмов решения типовых задач оптимизации на графах позволяет не только проверить правильность решения отдельных индивидуальных задач, полученных с помощью программы MS Excel, но и служит основой для программного решения этих задач оптимизации с помощью встроенных средств программы MS Excel. Соответствующие программы на языке Visual Basic for Applications (VBA) являются дополнительным и более удобным средством их решения и рассматриваются в *главе 11*.

7.1. Общая характеристика задач оптимизации на графах

К классу задач оптимизации на графах относятся такие задачи оптимизации, которые формулируются в форме нахождения специальных объектов на графе, обеспечивающих оптимальное значение некоторой целевой функции. При этом вид целевой функции и ограничений заранее не указываются и определяются спецификой конкретной задачи оптимизации. В общем случае задачи оптимизации на графах не предполагают формулировку исходной задачи в форме задачи математического программирования, однако успех в решении соответствующих задач тесно связан с выбором удачной модели для ее математической постановки.

7.1.1. Математическая постановка задачи оптимизации на графах

Задачи оптимизации на графах в своей исходной постановке используют общее понятие ориентированного или неориентированного графа, определения и свойства которых приводятся в *приложении 1*. В дополнение к общим свойствам графа того или иного вида применительно к типовой задаче оптимизации на графах вводятся в рассмотрение специальные объекты, такие как пути, деревья и потоки. При этом каждому такому объекту однозначно ставится в соответствие некоторое количественное значение целевой функции, рассчитываемое на основе конкретного графа, рассматриваемого в качестве исходных данных задачи оптимизации. Обязательным условием для этих объектов является наличие некоторого конструктивного способа их перечисления, в противном случае соответствующая задача оптимизации может оказаться неразрешимой с вычислительной точки зрения.

Тем самым отдельная задача оптимизации на графах формулируется как нахождение такого специального объекта, которому соответствует максимальное или минимальное значение целевой функции. Более строгое определение данного класса задач можно получить на основе рассмотрения общей математической постановки задачи оптимизации на графах и ее связи с задачами целочисленного и булева программирования.

В общем случае математическая постановка задачи оптимизации на графах может быть сформулирована в следующем виде:

$$f(x) \to \max_{x \in \Delta_\beta} \quad \text{или} \quad f(x) \to \min_{x \in \Delta_\beta}, \tag{7.1.1}$$

где переменная x условно обозначает некоторый специальный объект графа, а множество допустимых альтернатив Δ_β содержит все возможные специальные объекты рассматриваемого вида. При этом никаких дополнительных предположений о характере целевой функции или ограничений не делается. Однако исходя из того факта, что в практических задачах оптимизации исходные графы являются конечными, множество допустимых альтернатив в типовых задачах оптимизации на графах также является конечным. Это обстоятельство позволяет в отдельных случаях находить решение той или иной конкретной задачи методом простого перебора всех специальных объектов того или иного вида.

При рассмотрении математической постановки задачи оптимизации на графах следует иметь в виду, что для отдельных графов конкретного вида множество допустимых альтернатив соответствующей задачи оптимизации для тех или иных специальных объектов может оказаться пустым. Например, поиск оптимальных покрывающих деревьев или путей для несвязных графов теряет свой смысл. Именно по этой причине в исходной постановке задач оптимизации на графах необходимо указывать дополнительные свойства графа, которым он должен удовлетворять, чтобы соответствующая задача оптимизации имела допустимые решения.

7.1.2. Основные методы решения задач оптимизации на графах

Хотя общая математическая постановка задачи оптимизации на графах (7.1.1) не дает никакой информации относительно возможных методов ее решения, тем не менее, все методы решения таких задач можно условно разделить на два класса.

1. Большинство известных задач оптимизации на графах могут быть сформулированы в форме математической модели целочисленного или булева программирования. В этом случае выбор способа их решения полностью определяется математическими свойствами соответствующей постановки задачи. Более того, возможность решения практических задач оптимизации на графах с помощью программы MS Excel непосредственно зависит от возможности ее формулировки в виде задачи целочисленного или булева программирования.
2. Задачи оптимизации на графах могут быть решены с использованием специальных алгоритмов, которые учитывают специфические особенности тех или иных объектов графа и конечную мощность множества допустимых альтернатив. В этой связи задачи оптимизации на графах концептуально оказываются тесно связанными с задачами комбинаторной оптими-

зации, методы решения которых рассматриваются в *главе 8*. Хотя для нахождения точного решения задач оптимизации на графах могут использоваться общие алгоритмы типа метода ветвей и границ и метода динамического программирования, наиболее эффективными с вычислительной точки зрения оказываются специальные алгоритмы. Именно такие алгоритмы решения типовых задач оптимизации на графах рассматриваются в настоящей главе и положены в основу разработки программ на VBA, которые представлены в *главе 11*.

Применительно к решению практических задач оптимизации на графах с большим количеством элементов множества допустимых альтернатив были разработаны также специальные алгоритмы нахождения приближенного решения, которые позволяют находить одно или несколько локально-оптимальных решений. Однако как уже отмечалось ранее, использование приближенных методов оправдано лишь тогда, когда по тем или иным причинам нахождение точного решения соответствующих задач оказывается невозможным.

Поскольку программа MS Excel позволяет находить точное решение задач целочисленного и булева программирования, тем самым имеется возможность решения с ее помощью и задач оптимизации на графах. При этом общий порядок подготовки исходных данных и поиска решения аналогичен рассмотренному ранее для решения других типовых задач оптимизации. Для оценки точности получаемых решений целесообразно выполнить сравнение полученных различными способами оптимальных решений отдельных практических задач. Именно с этой целью в данной главе приводится описание способов решения задач о максимальном и минимальном покрывающем дереве в графе и задачи о кратчайшем пути.

7.2. Задача о минимальном покрывающем дереве в графе

Содержательная постановка задачи о минимальном покрывающем дереве в графе приводится в *разд. 1.2.9*. В настоящей главе рассматривается ее модификация, используемая для формальной записи условий и решения соответствующей индивидуальной задачи оптимизации в виде модели булева программирования.

7.2.1. Математическая постановка задачи

Рассмотрим неориентированный связный граф: $G=(V, E, h)$, в котором $V=\{v_1, v_2,..., v_n\}$ — конечное множество вершин, $E=\{e_1, e_2,..., e_n\}$ — конечное множество ребер, $h: E \to \boldsymbol{Z}_+$ — весовая функция ребер. Для математиче-

ской постановки задачи удобно обозначить отдельные значения весовой функции ребер через: $c_{ij} = h(e_k)$, где ребро $e_k \in E$ соответствует паре вершин $\{v_i, v_j\} \subset V$. Согласно содержательной постановке рассматриваемой задачи отдельные значения: $c_{ij} = h(\{v_i, v_j\})$ могут интерпретироваться как длина, затраты или стоимость участка (i, j) исходного графа.

Стоимость любого подмножества ребер $E_k \subset E$ в графе G равна сумме весов ребер, входящих в это подмножество. Требуется определить такое подмножество ребер, которое образует покрывающее или остовное дерево в графе G и обладает минимальной стоимостью.

Для формальной записи условий задачи о минимальном покрывающем дереве в графе в виде модели булева программирования следует воспользоваться двумя условиями покрывающего дерева в графе:

1. Каждая из вершин исходного графа должна иметь хотя бы одно инцидентное ей ребро, входящее в минимальное покрывающее дерево. В противном случае такие вершины в искомом дереве окажутся изолированными, и, следовательно, дерево не будет являться покрывающим.
2. Общее количество ребер в минимальном покрывающем дереве должно быть в точности равно $n - 1$, где n — общее количество вершин исходного графа. Действительно, если некоторое дерево содержит меньше $n - 1$ ребер, то оно не будет покрывающим, если же дерево содержит больше $n - 1$ ребер, то оно будет содержать цикл.

Примечание

Чтобы убедиться в справедливости отмеченных свойств, можно обратиться к специальной литературе по теории графов, список которых приведен в конце книги. Заинтересованные читатели могут ограничиться проверкой данных свойств на конкретных примерах графов.

Введем в рассмотрение m булевых переменных, которые для удобства обозначаются через x_{ij} и интерпретируются следующим образом. Переменная $x_{ij} = 1$, если ребро $e_k \in E$, которому соответствует пара вершин $\{v_i, v_j\}$, входит в искомое покрывающее дерево минимальной стоимости, и $x_{ij} = 0$, в противном случае, т. е. если ребро $\{v_i, v_j\}$ не входит в оптимальное покрывающее дерево. Заметим, что количество рассматриваемых булевых переменных конечно и равно m, где m — количество ребер исходного графа.

Тогда в общем случае математическая постановка задачи о минимальном покрывающем дереве в графе может быть сформулирована следующим образом:

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}x_{ij} \to \min_{x\in\Delta_\beta}, \tag{7.2.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств и неравенств:

$$\begin{cases} \sum_{j=2}^{n} x_{1j} \geq 1; & (7.2.2)\\ \sum_{i=1}^{k-1} x_{ik} + \sum_{j=k+1}^{n} x_{kj} \geq 1\ (\forall k \in \{2,..., n-1\}); & (7.2.3)\\ \sum_{i=1}^{n-1} x_{in} \geq 1; & (7.2.4)\\ \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} x_{ij} = n-1; & (7.2.5)\\ x_{ij} \in \{0, 1\}, \text{ где } \{v_i, v_j\} = e_k \in E\ (\forall k \in \{1, 2,..., m\}). & (7.2.6) \end{cases}$$

Заметим, что те переменные x_{ij}, для которых весовая функция ребер h не определена или равна 0, вообще не входят в математическую постановку рассматриваемой задачи (7.2.1)—(7.2.6). При этом первых три ограничения (7.2.2)—(7.2.4) требуют выполнения первого из отмеченных ранее свойств, т. е. в искомом покрывающем дереве не должно быть изолированных вершин. Четвертое ограничение (7.2.5) требует выполнения второго из отмеченных ранее свойств, т. е. искомое покрывающее дерево должно содержать ровно $n-1$ ребер. Общее количество ограничений (7.2.2)—(7.2.5) равно n. Наконец, последнее ограничение (7.2.6) требует, чтобы переменные принимали только булевы значения.

Если в постановке задачи о минимальном покрывающем дереве в графе (7.2.1)—(7.2.6) в выражении целевой функции (7.2.1) операцию отыскания минимума заменить операцией отыскания максимума, то может быть получена математическая постановка соответствующей задачи о максимальном покрывающем дереве в графе. Ввиду того, что эти две задачи оптимизации имеют одинаковую структуру, методы их решения целесообразно рассматривать совместно.

7.2.2. Решение задач о минимальном и максимальном покрывающем дереве в графе с помощью программы MS Excel

Для решения задачи о минимальном покрывающем дереве в графе с помощью программы MS Excel необходимо задать конкретные значения параметрам исходной задачи. С этой целью рассмотрим задачу разработки проекта транспортной сети, которая должна соединить 7 населенных пунктов в некотором географическом районе. Данный район может быть представлен в виде схемы, формально представляющей собой неориентированный связный граф, состоящий из 7 вершин и 12 ребер (рис. 7.1). Стоимость прокладки автодороги между двумя соседними населенными пунктами, выраженная, например, в *млн. рублей*, равна значению весовой функции для каждого ребра, которое указано рядом с изображением этого ребра в графе.

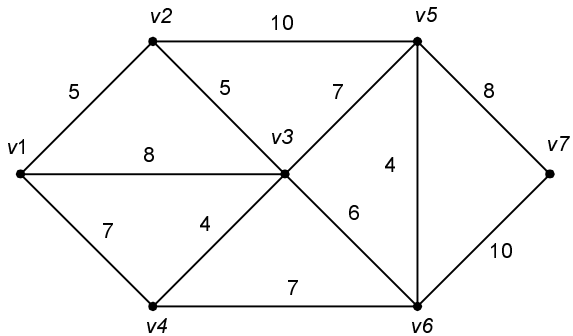


Рис. 7.1. Исходный граф индивидуальной задачи о покрывающем дереве

Требуется разработать такой проект, чтобы общая стоимость реализации проекта была минимальной. При этом должно быть выполнено обязательное условие — из любого населенного пункта по построенным автодорогам можно было бы попасть в любой другой населенный пункт данного географического района.

Переменными математической модели данной индивидуальной задачи о минимальном покрывающем дереве являются 12 переменных: x_{12}, x_{13}, x_{14}, x_{23}, x_{25}, x_{34}, x_{35}, x_{36}, x_{46}, x_{56}, x_{57}, x_{67}. Каждая из этих переменных x_{ij} принимает значение 1, если ребро $\{i, j\}$ входит в минимальное покрывающее дерево, и 0 — в противном случае. Тогда математическая постановка рассматриваемой индивидуальной задачи о минимальном покрывающем дереве может быть записана в следующем виде:

$$5x_{12} + 8x_{13} + 7x_{14} + 5x_{23} + 10x_{25} + 4x_{34} + 7x_{35} + 6x_{36} + \\ +7x_{46} + 4x_{56} + 8x_{57} + 10x_{67} \to \min_{x \in \Delta_\beta}, \qquad (7.2.7)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} x_{12}+x_{13}+x_{14}\ge 1; \\ x_{12}+x_{23}+x_{25}\ge 1; \\ x_{13}+x_{23}+x_{34}+x_{35}+x_{36}\ge 1; \\ x_{14}+x_{34}+x_{46}\ge 1; \\ x_{25}+x_{35}+x_{56}+x_{57}\ge 1; \\ x_{36}+x_{46}+x_{56}+x_{67}\ge 1; \\ x_{57}+x_{67}\ge 1; \\ x_{12}+x_{13}+x_{14}+x_{23}+x_{25}+x_{34}+x_{35}+x_{36}+x_{46}+x_{56}+x_{57}+x_{67}=6; \\ x_{12},x_{13},x_{14},x_{23},x_{25},x_{34},x_{35},x_{36},x_{46},x_{56},x_{57},x_{67}\in\{0,1\}. \end{cases} \quad (7.2.8)$$

Для решения данной задачи с помощью программы MS Excel создадим новую книгу с именем Оптимизация на графах и изменим имя ее первого рабочего листа на Покрывающее дерево. Для решения поставленной задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:F1**, **A14** (рис. 7.2). Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи.
2. В ячейки **A2:A13** введем индексы начальных вершин, а в ячейки **B2:B13** — индексы конечных вершин всех имеющихся ребер исходного графа.
3. В ячейки **C2:C13** введем значения коэффициентов целевой функции (7.2.7).
4. В ячейку **F2** введем формулу: `=СУММПРОИЗВ(C2:C13; D2:D13)`, которая представляет целевую функцию (7.2.7).
5. В ячейку **E2** введем значение левой части первого ограничения (7.2.8): `=СУММ(D2:D4)`; в ячейку **E3** введем значение левой части второго ограничения: `=СУММ(D2; D5:D6)`; в ячейку **E4** введем значение левой части третьего ограничения: `=СУММ(D3;D5; D7:D9)`; в ячейку **E5** введем значение левой части четвертого ограничения: `=СУММ(D4; D7; D10)`; в ячейку **E6** введем значение левой части пятого ограничения: `=СУММ(D6; D8; D11:D12)`; в ячейку **E7** введем значение левой части шестого ограничения: `=СУММ(D9:D11; D13)`; в ячейку **E8** введем значение левой части седьмого ограничения: `=СУММ(D12:D13)`.
6. В ячейку **D14** введем формулу: `=СУММ(D2:D13)`, которая представляет левую часть восьмого ограничения (7.2.8).

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о минимальном покрывающем дереве в графе имеет следующий вид (рис. 7.2).

ОптимизацияНаГрафах

	A	B	C	D	E	F
1	*vi*	*vj*	*cij*	Переменные:	Ограничения:	Значение ЦФ:
2	1	2	5		=СУММ(D2:D4)	=СУММПРОИЗВ(C2:C13;D2:D13)
3	1	3	8		=СУММ(D2;D5:D6)	
4	1	4	7		=СУММ(D3;D5;D7:D9)	
5	2	3	5		=СУММ(D4;D7;D10)	
6	2	5	10		=СУММ(D6;D8;D11:D12)	
7	3	4	4		=СУММ(D9:D11;D13)	
8	3	5	7		=СУММ(D12:D13)	
9	3	6	6			
10	4	6	7			
11	5	6	4			
12	5	7	8			
13	6	7	10			
14	Ограничение общее:			=СУММ(D2:D13)		
15						
16						
17						
18						
19						
20						

Покрывающее дерево

Рис. 7.2. Исходные данные для решения задачи о минимальном покрывающем дереве в графе

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$F$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$D$2:$D$13`.
4. Задать первые 7 ограничений для рассматриваемой задачи (7.2.8). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **E2$E$8**, который должен отобразиться в поле с именем **Ссылка на ячейку**;

- в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";
- в качестве значения правой части этой группы ограничений ввести с клавиатуры значение 1;
- для добавления первой группы ограничений в дополнительном окне нажать кнопку с надписью **Добавить**.

5. Добавить восьмое ограничение из второй группы для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$D$14**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры значение 6;
   - для добавления этого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
6. Добавить последнее ограничение на булевы значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$D$2:$D$13**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "двоичное";
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
7. В окне дополнительных параметров поиска решения выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения имеет следующий вид (рис. 7.3).

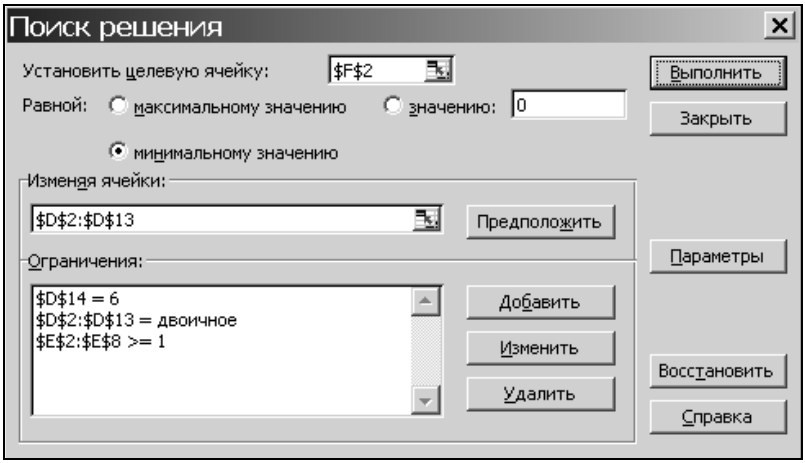


Рис. 7.3. Ограничения на значения переменных и параметры мастера поиска решения для задачи о минимальном покрывающем дереве в графе

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 7.4).

ОптимизацияНаГрафах

	A	B	C	D	E	F	G	H	I	J
1	*vi*	*vj*	*cij*	Переменные:	Ограничения:	Значение ЦФ:				
2	1	2	5	**1**	1	**32**				
3	1	3	8	**0**	2					
4	1	4	7	**0**	3					
5	2	3	5	**1**	1					
6	2	5	10	**0**	2					
7	3	4	4	**1**	2					
8	3	5	7	**0**	1					
9	3	6	6	**1**						
10	4	6	7	**0**						
11	5	6	4	**1**						
12	5	7	8	**1**						
13	6	7	10	**0**						
14	**Ограничение общее:**			6						
15										
16										
17										

Покрывающее дерево

Рис. 7.4. Результат количественного решения задачи о минимальном покрывающем дереве в графе

Результатом решения булевой задачи о минимальном покрывающем дереве в графе являются найденные оптимальные значения переменных: $x_{12} = 1$,

$x_{23} = 1$, $x_{34} = 1$, $x_{36} = 1$, $x_{56} = 1$, $x_{57} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствует значение целевой функции: $f_{opt} = 32$.

Анализ найденного решения показывает, что минимальное покрывающее дерево исходного графа (рис. 7.1) содержит следующие ребра: (1, 2), (2, 3), (3, 4), (3, 6), (5, 6), (5, 7). Тем самым найден оптимальный проект транспортной сети, который должен включать в себя построение автодорог между населенными пунктами: 1 и 2, 2 и 3, 3 и 4, 3 и 6, 5 и 6, 5 и 7 (рис. 7.5). При этом общая стоимость реализации проекта будет минимальной и равна 32 *млн. руб*.

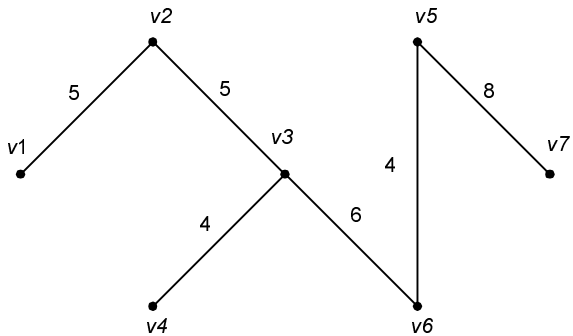


Рис. 7.5. Минимальное покрывающее дерево исходного графа

Легко проверить, что в этом случае из любого населенного пункта по построенным автодорогам можно будет попасть в любой другой населенный пункт данного географического района. Так, например, чтобы добраться из населенного пункта 4 в населенный пункт 7, необходимо будет последовательно миновать соседние населенные пункты 3, 6, 5 и, наконец, 7.

7.2.3. Решение задачи о максимальном покрывающем дереве в графе с помощью программы MS Excel

Задача о минимальном покрывающем дереве в графе (7.2.7) и (7.2.8) может быть легко преобразована к варианту задачи о максимальном покрывающем дереве в графе, при этом характер задачи и ее сложность не изменяются. Соответствующая содержательная задача может быть сформулирована следующим образом.

При маршрутизации пакетов передачи данных в вычислительных сетях желательно выбирать каналы передачи данных между соседними маршрутизаторами с максимальной скоростью или пропускной способностью. Именно

в этом случае будет обеспечена максимальная скорость передачи данных между любыми двумя абонентами. Считается, что два абонента или узла сети соединены между собой, если существует прямой или транзитный канал связи между ними. Топология вычислительной сети и пропускная способность линий связи, выраженная, например, в *млн. бит в сек*, могут быть представлены в виде связного неориентированного графа (см. рис. 7.1). Требуется определить вариант маршрутизации узлов данной вычислительной сети с максимальной пропускной способностью.

Математическая постановка задачи о максимальном покрывающем дереве в графе аналогична математической постановке задачи о минимальном покрывающем дереве (7.2.7) и (7.2.8) за исключением того, что в выражении для целевой функции (7.2.7) операция нахождения минимума должна быть операцией нахождения максимума. Более того, для решения задачи о максимальном покрывающем дереве в графе с помощью программы MS Excel можно воспользоваться исходными данными, которые были подготовлены для решения задачи о минимальном покрывающем дереве *(см. разд. 7.2.2)* и уже имеются на рабочем листе Покрывающее дерево.

Для нахождения решения рассматриваемой индивидуальной задачи о максимальном покрывающем дереве в графе необходимо лишь изменить вариант поиска решения в первом окне спецификации параметров мастера поиска решения, а именно для группы **Равной:** выбрать вариант — **максимальному значению**. В этом случае после выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 7.6).

ОптимизацияНаГрафах

	A	B	C	D	E	F	G	H	I	J
1	*vi*	*vj*	*cij*	Переменные:	Ограничения:	Значение ЦФ:				
2	1	2	5	0	2	50				
3	1	3	8	1	1					
4	1	4	7	1	2					
5	2	3	5	0	1					
6	2	5	10	1	3					
7	3	4	4	0	1					
8	3	5	7	1	2					
9	3	6	6	0						
10	4	6	7	0						
11	5	6	4	0						
12	5	7	8	1						
13	6	7	10	1						
14	Ограничение общее:			6						
15										
16										
17										

Покрывающее дерево

Рис. 7.6. Результат количественного решения задачи о максимальном покрывающем дереве в графе

Результатом решения булевой задачи о максимальном покрывающем дереве в графе являются найденные оптимальные значения переменных: $x_{13} = 1$, $x_{14} = 1$, $x_{25} = 1$, $x_{35} = 1$, $x_{57} = 1$, $x_{67} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствует значение целевой функции: $f_{opt} = 50$.

Анализ найденного решения показывает, что максимальное покрывающее дерево исходного графа (рис. 7.1) содержит следующие ребра: (1, 3), (1, 4), (2, 5), (3, 5), (5, 7), (6, 7). Тем самым найден оптимальный вариант маршрутизации вычислительной сети, который должен включать в себя линии передачи данных между узлами: 1 и 3, 1 и 4, 2 и 5, 3 и 5, 5 и 7, 6 и 7 (рис. 7.7). При этом общая пропускная способность оптимального варианта маршрутизации будет максимальной и равна 50 *млн. бит в сек.*

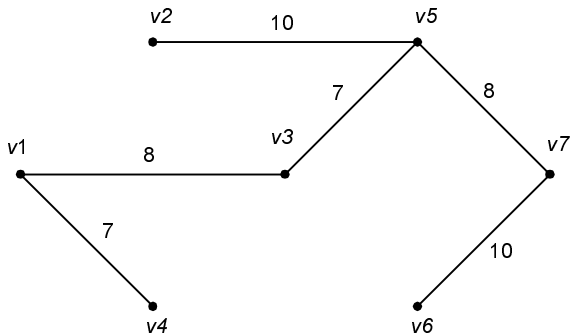


Рис. 7.7. Максимальное покрывающее дерево исходного графа

Следует заметить, что оптимальные решения как задачи о минимальном покрывающем дереве, так и задачи о максимальном покрывающем дереве в графе могут быть найдены методом полного перебора. В этом случае необходимо проверить на допустимость, т. е. условие того, что дерево является покрывающим, $C_{12}^{6} = \dfrac{12!}{(6! \cdot 6!)} = 924$ возможных комбинаций из 6 ребер исходного графа (см. рис. 7.1). Здесь C_m^{n-1} — число сочетаний из m исходных ребер по $n - 1$ ребер, которые может содержать покрывающее дерево. Далее необходимо вычислить для каждого из покрывающих деревьев значение целевой функции (7.2.7) и выбрать в качестве оптимального то из них, которому соответствует минимальное или максимальное значение целевой функции.

Однако для нахождения оптимальных решений задач о минимальном и максимальном покрывающем дереве предложен специальный метод, который получил экзотическое название жадного алгоритма.

7.2.4. Решение задач о максимальном и минимальном покрывающем дереве с помощью жадного алгоритма

Для оценки точности и правильности результатов решения задачи о максимальном покрывающем дереве в графе, полученных с помощью программных средств, можно воспользоваться специально разработанным для решения задач данного класса так называемым *жадным алгоритмом*. Название алгоритма происходит от особенности выбора ребер на каждой итерации алгоритма, а именно: всегда выбирается ребро с максимальным весом.

Для описания жадного алгоритма предварительно вводятся в рассмотрение 2 множества вершин: $\overline{V}$ — множество соединенных покрывающим деревом вершин исходного графа и V_0 — множество не соединенных покрывающим деревом вершин исходного графа. По определению: $\overline{V} \cup V_0 = V$, $\overline{V} \cap V_0 = \varnothing$, где $V = \{v_1, v_2, \ldots, v_n\}$ — множество вершин исходного графа $G = (V, E, h)$. Множество ребер, входящих в максимальное покрывающее дерево, обозначим через E_{max}.

Жадный алгоритм имеет итеративный характер и заключается в выполнении следующих действий (шагов):

1. *Предварительное определение множеств соединенных и несоединенных вершин*. До начала выполнения основных итераций алгоритма определить введенные в рассмотрение множества вершин следующим образом: $\overline{V} = \{v_1\}$ и $V_0 = V / \{v_1\}$, где вершина v_1 выбирается, в общем случае, произвольным образом и может быть заменена любой другой вершиной исходного графа. Множество ребер, входящих в максимальное покрывающее дерево, до начала выполнения алгоритма устанавливается равным: $E_{max} = \varnothing$.
2. *Основная итерация алгоритма.* Среди множества ребер, соединяющих вершины из множеств $\overline{V}$ и V_0, выбрать ребро с максимальным весом $c_{ij} = h(e_k)$, где ребро $e_k \in E$ соответствует паре вершин $\{v_i, v_j\}$. При этом должно выполняться обязательное условие: $v_i \in V$ и $v_j \in V_0$. После нахождения ребра с максимальным весом оно добавляется во множество E_{max}, которое становится равным: $E'_{max} = E_{max} \cup e_k$. Одновременно с этим вершина v_j исключается из множества V_0 и добавляется в множество $\overline{V}$, которые становятся равными: $V_0 = V_0 / \{v_j\}$ и $\overline{V}' = V \cup \{v_j\}$.
3. *Проверка условия окончания алгоритма*. Проверить выполнение условия: $V'_0 = \varnothing$. Если это условие выполняется, то все вершины исходного графа соединены максимальным покрывающим деревом E'_{max}, и выполнение алгоритма на этом заканчивается. Если же это условие не выполняется, то установить: $E_{max} = E'_{max}$, $V_0 = V'_0$, $\overline{V} = \overline{V}'$ и перейти к выполнению действий шага 2.

Примечание

Если на шаге 2 жадного алгоритма имеется несколько ребер $e_k \in E$ с одинаковым значением максимального веса, удовлетворяющих условию: $v_i \in V$ и $v_j \in V_0$, следует выбрать любое из них. При этом произвольный выбор ребра с максимальным весом не оказывает влияния на конечный результат решения задачи.

Рассмотренная схема жадного алгоритма может быть изображена графически в форме диаграммы деятельности языка UML (рис. 7.8).

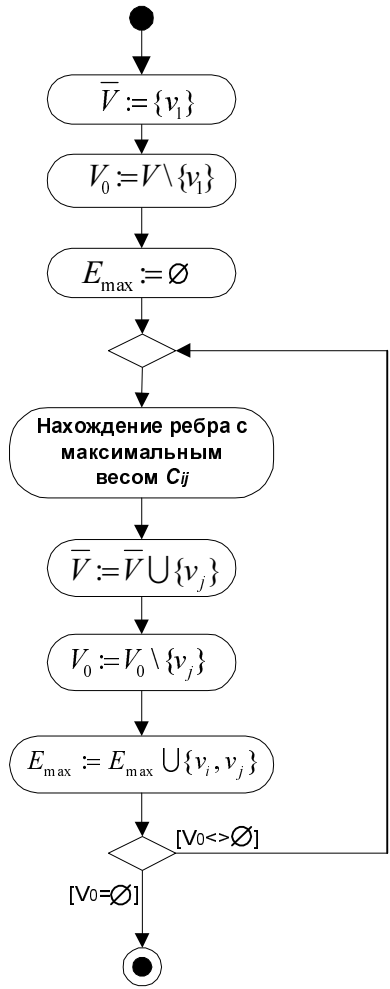


Рис. 7.8. Диаграмма деятельности жадного алгоритма для нахождения максимального покрывающего дерева

Нетрудно заметить, что в силу конечности вершин исходного графа, рассмотренный жадный алгоритм является конечным. Более того, поскольку заранее известно, что максимальное покрывающее дерево содержит $n-1$ ребер, общее количество итераций данного алгоритма, связанных с выполняемыми на шаге 2 действиями, конечно и равно: $n-1$.

Проиллюстрируем использование рассмотренного жадного алгоритма для решения индивидуальной задачи о максимальном покрывающем дереве, исходный граф которой изображен на рис. 7.1. Для этого в соответствии с описанным алгоритмом следует выполнить следующие действия:

Шаг 1. Установить: $\overline{V} = \{v_1\}, V_0 = \{v_2, v_3, v_4, v_5, v_6, v_7\}$ и $E_{\max} = \varnothing$.

Шаг 2 (первая итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_1, v_3\}$. Установить: $\overline{V}' = \{v_1, v_3\}, V_0' = \{v_2, v_4, v_5, v_6, v_7\}$ и $E'_{\max} = \{\{v_1, v_3\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (вторая итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_3, v_5\}$. Установить: $\overline{V}' = \{v_1, v_3, v_5\}, V_0' = \{v_2, v_4, v_6, v_7\}$ и $E'_{\max} = \{\{v_1, v_3\}, \{v_3, v_5\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (третья итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_5, v_2\}$. Установить: $\overline{V}' = \{v_1, v_2, v_3, v_5\}, V_0' = \{v_4, v_6, v_7\}$ и $E'_{\max} = \{\{v_1, v_3\}, \{v_3, v_5\}, \{v_5, v_2\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (четвертая итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_5, v_7\}$. Установить: $\overline{V}' = \{v_1, v_2, v_3, v_5, v_7\}, V_0' = \{v_4, v_6\}$ и $E'_{\max} = \{\{v_1, v_3\}, \{v_3, v_5\}, \{v_2, v_5\}, \{v_5, v_7\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (пятая итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_7, v_6\}$. Установить: $\overline{V}' = \{v_1, v_2, v_3, v_5, v_6, v_7\}, V_0' = \{v_4\}$ и $E'_{\max} = \{\{v_1, v_3\}, \{v_3, v_5\}, \{v_2, v_5\}, \{v_5, v_7\}, \{v_7, v_6\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (шестая итерация). На данной итерации имеется 2 ребра, удовлетворяющих условию шага 2, а именно: $\{v_1, v_4\}$ и $\{v_6, v_4\}$. В качестве искомого ребра выберем первое из них: $\{v_1, v_4\}$. Установить: $\overline{V}' = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$, $V_0' = \varnothing$ и $E'_{\max} = \{\{v_1, v_3\}, \{v_3, v_5\}, \{v_2, v_5\}, \{v_5, v_7\}, \{v_6, v_7\}, \{v_1, v_4\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ выполняется, то на этом заканчивается выполнение алгоритма в целом. В качестве результата принимается максимальное покрывающее дерево: $E'_{\max} = \{\{v_1, v_3\}, \{v_3, v_5\}, \{v_2, v_5\}, \{v_5, v_7\}, \{v_6, v_7\}, \{v_1, v_4\}\}$.

Сравнивая полученный результат с результатом решения данной индивидуальной задачи с помощью программы MS Excel (рис. 7.6), получаем их полное совпадение, что является свидетельством в пользу правильности соответствующих результатов.

Примечание

Если на шаге 2 шестой итерации вместо ребра $\{v_1, v_4\}$ выбрать ребро $\{v_6, v_4\}$, то получим второе оптимальное решение соответствующей индивидуальной задачи о максимальном покрывающем дереве в графе: $E'_{\max} = \{\{v_1, v_3\}, \{v_3, v_5\}, \{v_2, v_5\}, \{v_5, v_7\}, \{v_6, v_7\}, \{v_1, v_4\}\}$ с тем же значением целевой функции: $f_{\text{opt}} = 50$. Поскольку это решение не было найдено средством поиска решений программы MS Excel, то данный факт может свидетельствовать о неспособности программы MS Excel находить другие оптимальные решения задач оптимизации в случае, если их несколько. Для исправления этой ситуации можно воспользоваться средством программирования MS Excel и реализовать жадный алгоритм в форме программы на языке VBA. Соответствующие программы для решения задач оптимизации на графах рассматриваются в *главе 11*.

Следует заметить, что рассмотренный жадный алгоритм может быть легко модифицирован для нахождения минимального покрывающего дерева в графе. При этом модификация относится только к выполнению действий на шаге 2. А именно вместо выбора ребра с *максимальным* весом следует выбирать ребро с *минимальным* весом, а обязательное условие остается без изменения. Что касается множества ребер $E_{\max}$, входящих в максимальное покрывающее дерево, то вместо него вводится в рассмотрение множество ребер $E_{\min}$, входящих в минимальное покрывающее дерево, обладающее идентичными свойствами.

Проиллюстрируем использование *модифицированного жадного алгоритма* для решения индивидуальной задачи о минимальном покрывающем дереве, исходный граф которой изображен на рис. 7.1. Для этого в соответствии с описанным алгоритмом следует выполнить следующие действия:

Шаг 1. Установить: $\overline{V} = \{v_1\}, V_0 = \{v_2, v_3, v_4, v_5, v_6, v_7\}$ и $E_{\min} = \varnothing$.

Шаг 2 (первая итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_1, v_2\}$. Установить: $\overline{V}' = \{v_1, v_2\}, V_0' = \{v_3, v_4, v_5, v_6, v_7\}$ и $E'_{\min} = \{\{v_1, v_2\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (вторая итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_2, v_3\}$. Установить: $\overline{V}' = \{v_1, v_2, v_3\}, V_0' = \{v_4, v_5, v_6, v_7\}$ и $E'_{\min} = \{\{v_1, v_2\}, \{v_2, v_3\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (третья итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_3, v_4\}$. Установить: $\overline{V}' = \{v_1, v_2, v_3, v_4\}, V_0' = \{v_5, v_6, v_7\}$ и $E'_{\min} = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (четвертая итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_3, v_6\}$. Установить: $\overline{V}' = \{v_1, v_2, v_3, v_4, v_6\}, V_0' = \{v_5, v_7\}$ и $E'_{\min} = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_3, v_6\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (пятая итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_6, v_5\}$. Установить: $\overline{V}' = \{v_1, v_2, v_3, v_4, v_5, v_6\}, V_0' = \{v_7\}$ и $E'_{\min} = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_3, v_6\}, \{v_6, v_5\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (шестая итерация). В качестве ребра, удовлетворяющего условию шага 2, выбрать ребро $\{v_5, v_7\}$. Установить: $\overline{V}' = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}, V_0' = \varnothing$ и $E'_{\min} = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_3, v_6\}, \{v_6, v_5\}, \{v_5, v_7\}\}$.

Шаг 3. Поскольку условие окончания алгоритма: $V_0' = \varnothing$ выполняется, то выполнение алгоритма заканчивается. В качестве результата принимается минимальное покрывающее дерево: $E'_{\min} = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_3, v_6\}, \{v_6, v_5\}, \{v_5, v_7\}\}$.

Сравнивая полученный результат с результатом решения данной индивидуальной задачи с помощью программы MS Excel (рис. 7.4), получаем их полное совпадение, что является свидетельством в пользу правильности соответствующих результатов.

Примечание

Область применения жадного алгоритма отнюдь не исчерпывается решением задач нахождения минимальных и максимальных покрывающих деревьев в графах, а распространяется на более широкий класс задач специального вида — задач со структурой *матроидов*. Доказательство корректности данного алгоритма и примеры решения других задач оптимизации со структурой матроидов можно найти в специальной литературе, приведенной в конце книги.

7.3. Задача о минимальном пути в графе

Содержательная постановка задачи о минимальном пути в графе приводится в *разд. 1.2.5*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи оптимизации в виде модели булева программирования.

7.3.1. Математическая постановка задачи

Рассмотрим ориентированный граф: $G = (V, E, h)$, в котором $V = \{v_1, v_2, ..., v_n\}$ — конечное множество вершин, $E = \{e_1, e_2, ..., e_m\}$ — конечное множество дуг, $h : E \to \boldsymbol{Z}_+$ — весовая функция дуг. Для математической постановки задачи удобно обозначить отдельные значения весовой функции дуг через: $c_{ij} = h(e_k)$, где дуга $e_k \in E$ соответствует упорядоченной паре вершин (v_i, v_j). Согласно содержательной постановке рассматриваемой задачи значения: $c_{ij} = h((v_i, v_j))$ могут интерпретироваться как длина участка, затраты или стоимость переезда из *i*-го города в *j*-й город.

Длина любого маршрута в графе равна сумме весов дуг, входящих в этот маршрут. Дополнительно в графе фиксируются две вершины: *начальная* вершина v_s и *конечная* вершина v_t. В предположении, что исходный граф G является связным, т. е. вершина v_t потенциально достижима из v_s, требуется

определить маршрут минимальной длины из начальной вершины v_s в конечную вершину v_t.

Введем в рассмотрение следующие булевы переменные x_{ij}, которые интерпретируются следующим образом. Переменная $x_{ij} = 1$, если дуга (v_i, v_j) входит в искомый маршрут минимальной длины, и $x_{ij} = 0$, в противном случае, т. е. если дуга (v_i, v_j) не входит в оптимальный маршрут. Тогда в общем случае математическая постановка задачи о минимальном маршруте или пути в ориентированном графе может быть сформулирована следующим образом:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min_{x\in\Delta_\beta}, \tag{7.3.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} \sum_{j=1}^{n} x_{sj} - \sum_{i=1}^{n} x_{is} = 1; & (7.3.2)\\ \sum_{j=1}^{n} x_{tj} - \sum_{i=1}^{n} x_{it} = -1; & (7.3.3)\\ \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = 0\ (\forall i \in \{1, 2,..., n\}, i \neq s, i \neq t); & (7.3.4)\\ x_{ij} \in \{0, 1\}\ (\forall i, j \in \{1, 2,..., n\}). & (7.3.5) \end{cases}$$

При этом первое ограничение (7.3.2) требует выполнения следующего условия — искомый путь должен начинаться в вершине v_s, ограничение (7.3.3) требует выполнения следующего условия — искомый путь должен заканчиваться в вершине v_t. Третье ограничение (7.3.4) гарантирует связность минимального пути, т. е. искомый путь должен проходить через промежуточные вершины графа. Общее количество ограничений (7.3.2)—(7.3.4) равно n. Наконец, последнее ограничение (7.3.5) требует, чтобы переменные принимали только булевы значения.

Примечание

Заметим, что те коэффициенты целевой функции c_{ij}, для которых весовая функция дуг h исходного графа не определена или равна 0, в общей математической постановке рассматриваемой задачи (7.3.1)—(7.3.5) следует положить равными $+\infty$, т. е. достаточно большому положительному значению.

Если в постановке задачи о минимальном пути в графе (7.3.1)—(7.3.5) в выражении целевой функции (7.3.1) операцию отыскания минимума заменить операцией отыскания максимума, то может быть получена математическая постановка соответствующей задачи о максимальном пути в ориентированном графе. Однако последний вариант задачи получил наибольшую известность в виде задачи о критическом пути в сети, постановка и методы решения которой рассматриваются в *разд. 7.4*.

Примечание

В литературе встречаются самые разнообразные варианты задачи о минимальном пути в графе, а именно: задача нахождения k минимальных маршрутов (путей) в ориентированном или неориентированном графе, где k — некоторое натуральное число или задача нахождения минимальных маршрутов (путей) в соответствующем графе между всеми парами вершин. Для решения этих задач разработаны специальные методы, которые зачастую оказываются более эффективными, чем поиск решения программой MS Excel.

7.3.2. Решение задачи о минимальном пути в ориентированном графе с помощью программы MS Excel

Для решения задачи о минимальном пути в ориентированном графе с помощью программы MS Excel необходимо задать конкретные значения параметрам исходной задачи. С этой целью рассмотрим задачу нахождения минимального пути в транспортной сети, которая соединяет 8 населенных пунктов в некотором географическом районе. Район может быть задан в виде схемы, формально представляющей собой ориентированный связный граф, состоящий из 8 вершин и 15 дуг (рис. 7.9). Длина автодороги между двумя соседними

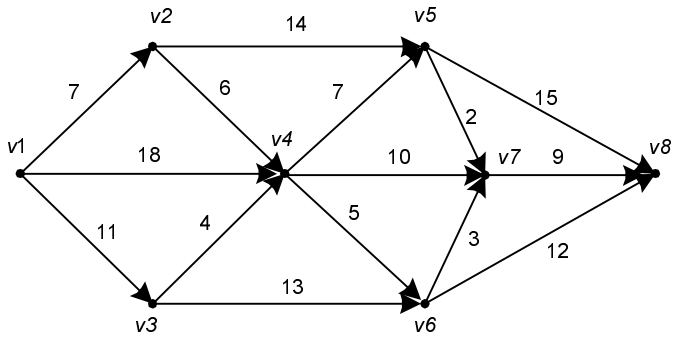


Рис. 7.9. Исходный ориентированный граф индивидуальной задачи о минимальном пути

населенными пунктами, выраженная, например, в *км*, равна для каждой дуги значению весовой функции, которое указано рядом с изображением этой дуги в графе.

Требуется найти маршрут, соединяющий начальный пункт 1, которому соответствует вершина $v_1 = v_s$, с конечным пунктом 8, которому соответствует вершина $v_8 = v_t$, так чтобы общая длина пути была минимальной.

Переменными математической модели данной индивидуальной задачи о минимальном пути в ориентированном графе являются 15 переменных: x_{12}, x_{13}, x_{14}, x_{24}, x_{25}, x_{34}, x_{36}, x_{45}, x_{46}, x_{47}, x_{57}, x_{58}, x_{67}, x_{68}, x_{78}. Каждая из этих переменных x_{ij} принимает значение 1, если дуга (i, j) входит в минимальный путь, и 0 — в противном случае. Тогда математическая постановка рассматриваемой индивидуальной задачи о минимальном пути в ориентированном графе может быть записана в следующем виде:

$$7x_{12} + 11x_{13} + 18x_{14} + 6x_{24} + 14x_{25} + 4x_{34} + 13x_{36} + 7x_{45} + \\ + 5x_{46} + 10x_{47} + 2x_{57} + 15x_{58} + 3x_{67} + 12x_{68} + 9x_{78} \to \min_{x \in \Delta_\beta}, \tag{7.3.6}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств и неравенств:

$$\begin{cases} x_{12} + x_{13} + x_{14} = 1; \\ x_{58} + x_{68} + x_{78} = 1; \\ x_{12} - x_{24} - x_{25} = 0; \\ x_{13} - x_{34} - x_{36} = 0; \\ x_{14} + x_{24} + x_{34} - x_{45} - x_{46} - x_{47} = 0; \\ x_{25} + x_{45} - x_{57} - x_{58} = 0; \\ x_{36} + x_{46} - x_{67} - x_{68} = 0; \\ x_{47} + x_{57} + x_{67} - x_{78} = 0; \\ x_{12}, x_{13}, x_{14}, x_{24}, x_{25}, x_{34}, x_{36}, x_{45}, \\ x_{46}, x_{47}, x_{57}, x_{58}, x_{67}, x_{68}, x_{78} \in \{0, 1\}. \end{cases} \tag{7.3.7}$$

Заметим, что те переменные x_{ij}, для которых весовая функция дуг h не определена или равна 0, не входят в математическую постановку рассматриваемой задачи (7.3.6) и (7.3.7). Для решения данной задачи с помощью программы MS Excel создадим в книге с именем Оптимизация на графах новый рабочий лист с именем Минимальный путь. Для решения поставленной задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:F1** (рис. 7.10). Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи.

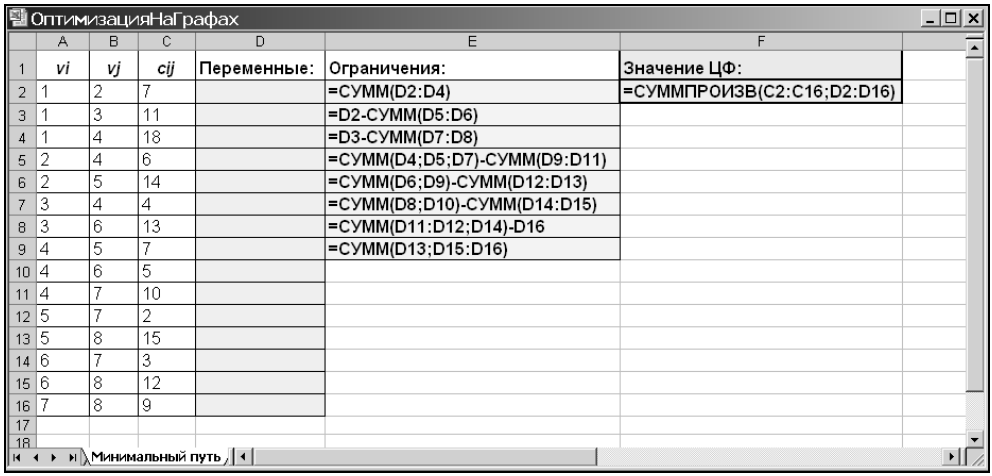
2. В ячейки **A2:A16** введем индексы начальных вершин, а в ячейки **B2:B16** — индексы конечных вершин всех имеющихся дуг исходного графа.
3. В ячейки **C2:C16** введем значения коэффициентов целевой функции (7.3.6).
4. В ячейку **F2** введем формулу: `=СУММПРОИЗВ(C2:C16; D2:D16)`, которая представляет собой целевую функцию (7.3.6).
5. В ячейку **E2** введем формулу: `=СУММ(D2:D4)`, которая представляет собой левую часть первого ограничения (7.3.7).
6. В ячейку **E9** введем формулу: `=СУММ(D13;D15:D16)`, которая представляет собой левую часть второго ограничения (7.3.7).
7. В ячейку **E3** введем значение левой части третьего ограничения: `=D2-СУММ(D5:D6)`.
8. В ячейку **E4** введем значение левой части четвертого ограничения: `=D3-СУММ(D7:D8)`.
9. В ячейку **E5** введем значение левой части пятого ограничения: `=СУММ(D4;D5;D7)-СУММ(D9:D11)`.
10. В ячейку **E6** введем значение левой части шестого ограничения: `=СУММ(D6;D9)-СУММ(D12:D13)`.
11. В ячейку **E7** введем значение левой части седьмого ограничения: `=СУММ(D8;D10)-СУММ(D14:D15)`.
12. И, наконец, в ячейку **E8** введем значение левой части восьмого ограничения: `=СУММ(D11:D12;D14)-D16`.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о минимальном пути в графе имеет вид, представленный на рис. 7.10.

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$F$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.



ОптимизацияНаГрафах

	A	B	C	D	E	F
1	*vi*	*vj*	*cij*	Переменные:	Ограничения:	Значение ЦФ:
2	1	2	7		=СУММ(D2:D4)	=СУММПРОИЗВ(C2:C16;D2:D16)
3	1	3	11		=D2-СУММ(D5:D6)	
4	1	4	18		=D3-СУММ(D7:D8)	
5	2	4	6		=СУММ(D4;D5;D7)-СУММ(D9:D11)	
6	2	5	14		=СУММ(D6;D9)-СУММ(D12:D13)	
7	3	4	4		=СУММ(D8;D10)-СУММ(D14:D15)	
8	3	6	13		=СУММ(D11:D12;D14)-D16	
9	4	5	7		=СУММ(D13;D15:D16)	
10	4	6	5			
11	4	7	10			
12	5	7	2			
13	5	8	15			
14	6	7	3			
15	6	8	12			
16	7	8	9			
17						
18						

Минимальный путь

Рис. 7.10. Исходные данные для решения задачи о минимальном пути в графе

3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$D$2:$D$16`.
4. Задать первое ограничение для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$E$2**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры значение 1;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Аналогичным образом добавить второе ограничение (7.3.7), используя в качестве исходной ячейки **$E$9**.
6. Задать группу ограничений для промежуточных вершин рассматриваемой задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;

- в появившемся дополнительном окне выбрать диапазон ячеек **$E$3:$E$8**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
- в качестве знака ограничения из выпадающего списка выбрать равенство "=";
- в качестве значения правой части ограничения ввести с клавиатуры значение 0;
- для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

7. Задать последнее ограничение на булевы значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$D$2:$D$16**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение **"двоичное"**;
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
8. В окне дополнительных параметров поиска решения выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения имеет следующий вид (рис. 7.11).

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 7.4).

Результатом решения задачи о минимальном пути в графе являются найденные оптимальные значения переменных: $x_{12} = 1$, $x_{24} = 1$, $x_{46} = 1$, $x_{67} = 1$, $x_{78} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствует значение целевой функции: $f_{opt} = 30$.

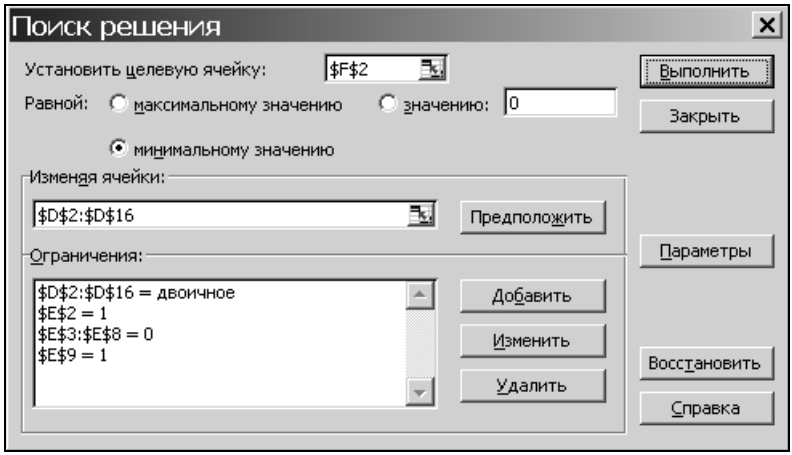


Рис. 7.11. Ограничения значений переменных и параметры мастера поиска решения для задачи о минимальном пути в графе

ОптимизацияНаГрафах

	A	B	C	D	E	F	G	H	I	J
1	*vi*	*vj*	*cij*	Переменные:	Ограничения:	Значение ЦФ:				
2	1	2	7	1	1	30				
3	1	3	11	0	0					
4	1	4	18	0	0					
5	2	4	6	1	0					
6	2	5	14	0	0					
7	3	4	4	0	0					
8	3	6	13	0	0					
9	4	5	7	0	1					
10	4	6	5	1						
11	4	7	10	0						
12	5	7	2	0						
13	5	8	15	0						
14	6	7	3	1						
15	6	8	12	0						
16	7	8	9	1						
17										
18										
19										

Покрывающее дерево | Минимальный путь

Рис. 7.12. Результат количественного решения задачи о минимальном пути в графе

Анализ найденного решения показывает, что минимальный путь в исходном ориентированном графе (рис. 7.9), соединяющий вершину 1 с вершиной 8, содержит следующие дуги: (1, 2), (2, 4), (4, 6), (6, 7), (7, 8). Тем самым найден оптимальный маршрут перемещения из исходного населенного пункта в конечный, который включает в себя последовательное перемещение между соседними населенными пунктами: из 1 в 2, из 2 в 4, из 4 в 6, из 6 в 7, из 7 в 8 (рис. 7.13). При этом общая длина пути будет минимальной и равна 30 *км*.

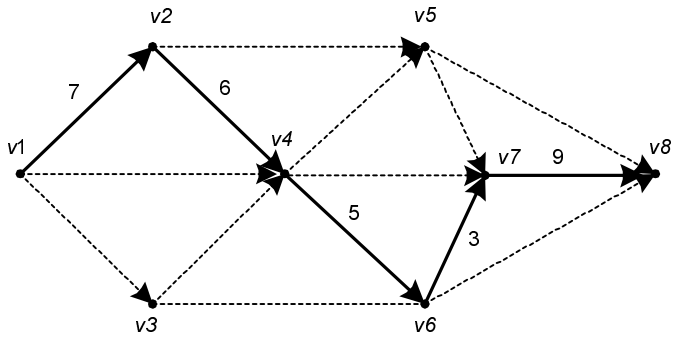


Рис. 7.13. Минимальный путь в исходном графе между вершинами 1 и 8

7.3.3. Решение задачи о минимальном пути в графе с помощью алгоритма пометок Дейкстры

Для оценки точности и правильности результатов решения задач о минимальном пути в ориентированном графе, полученных с помощью программы MS Excel, можно воспользоваться специально разработанным для решения задач данного класса так называемым *алгоритмом пометок Дейкстры*. Этот алгоритм, предложенный в 1959 г. Дейкстрой Э. В. (Dijksatra E. W.), является одним из наиболее эффективных методов решения данной задачи, учитывающих специфику рассматриваемой задачи оптимизации на графах.

Алгоритм пометок Дейкстры имеет итеративный характер и позволяет достичь даже большего, а именно — найти все минимальные пути из начальной вершины во все остальные для неориентированного графа. Сущность алгоритма заключается в построении на каждой итерации растущего ориентированного дерева минимальных путей из начальной вершины во все оставшиеся, включая и конечную вершину, в которую необходимо попасть при планировании перемещения по искомому графу.

Для описания алгоритма пометок вводятся в рассмотрение 2 множества помеченных значений: $L = \{l_1, l_2, \ldots, l_n\}$ — множество значений *временных* пометок вершин V исходного графа и $\overline{L} = \left\{\overline{l}_1, \overline{l}_2, ..., \overline{l}_n\right\}$ — множество значений *постоянных* пометок вершин V исходного графа $G = (V, E, h)$. По определению, временная пометка l_i некоторой вершины $v_i \in V$ означает некоторую предварительную оценку длины минимального пути из начальной вершины $v_s \in V$ в вершину $v_i \in V$. Постоянная пометка $\overline{l}_i$ некоторой вершины $v_i \in V$ означает окончательную оценку длины минимального пути из начальной вершины $v_s \in V$ в вершину $v_i \in V$. Поскольку окончательная оценка длины

минимального пути не может быть улучшена на последующих итерациях алгоритма пометок, значение $\bar{l}_i$ является искомым результатом решения рассматриваемой задачи.

Очевидно, множествам значений временных и постоянных пометок вершин можно поставить в соответствие множество вершин графа с временными пометками V_0 и множество вершин графа с постоянными пометками $\bar{V}$. При этом должно выполняться обязательное условие: $\bar{V} \cup V_0 = V, \bar{V} \cap V_0 = \varnothing$, где $V = \{v_1, v_2, ..., v_n\}$ — множество вершин исходного графа $G = (V, E, h)$. В дополнение к этому множество дуг, входящих в минимальный путь из начальной вершины v_s в конечную вершину v_t, обозначим через $E_{\min}$.

Алгоритм расстановки пометок Дейкстры или просто *алгоритм пометок* имеет итеративный характер и заключается в выполнении следующих действий (шагов):

1. *Предварительное определение значений временных и постоянных пометок.* До начала выполнения основных итераций алгоритма определить следующие значения: для начальной вершины установить постоянную пометку $\bar{l}_s = 0$, а для всех остальных вершин — временные пометки: $l_i = \infty \left(\forall v_i \in V \setminus \{v_s\}\right)$. Для удобства установить $v_s = v_1$ и $k = 1$, т. е. $\bar{V} = \{v_1\}$ и $V_0 = \{v_2, v_3, ..., v_n\}$. Множество дуг, входящих в минимальный путь, до начала выполнения алгоритма устанавливается равным: $E_{\min} = \varnothing$. После чего можно перейти к выполнению действий шага 2.

2. *Обновление временных пометок.* Среди вершин $v_j \in V_k \subset V_0$, инцидентных вершине v_k с постоянной пометкой $\bar{l}_k$ и имеющих временные пометки l_j, изменить эти временные пометки на новые значения l'_j с использованием следующей формулы:

$$l_j' = \min_{v_j \in V_k} \{l_j, \bar{l}_k + c_{kj}\} \qquad (7.3.8)$$

   После чего перейти к выполнению действий шага 3.

3. *Превращение временной пометки в постоянную.* Среди всех вершин графа V_0, имеющих временные пометки, выбрать вершину с минимальным значением временной пометки l'_j. Сделать эту пометку постоянной, т. е. $\bar{l}_j = l'_j$, тем самым изменить множества вершин с постоянными и временными пометками следующим образом: вершина v_j исключается

из множества V_0 и добавляется в множество $\overline{V}$, которые становятся равными: $V_0' = V_0 / \{v_j\}$ и $V' = V \cup \{v_j\}$. Установить $k = j$ и перейти к выполнению действий шага 4.

4. *Проверка условия окончания основных итераций.* При нахождении только минимального пути из начальной вершины v_s в конечную вершину v_t, проверить выполнение условия: $k = t$. Если это условие выполняется, то конечная вершина v_t исходного графа имеет постоянную пометку $\overline{l}_t$, которая равна длине минимального пути в нее из начальной вершины v_s. На этом выполнение основных итераций алгоритма заканчивается и выполняется переход к выполнению действий последнего шага 5, на котором определяется множество дуг $E_{\min}$, входящих в минимальный путь. Если же условие $k = t$ не выполняется, то установить: $V_0 = V_0', \overline{V} = \overline{V}'$ и перейти к выполнению действий шага 2.

5. *Определение структуры минимального пути.* Для каждой пары вершин, входящих в множество $\overline{V}$ с постоянными пометками, проверить выполнение следующего условия: $\overline{l}_j - \overline{l}_i = c_{ij}$. Если это условие выполняется, то дугу $(v_i, v_j) \in E$, где $v_i, v_j \in \overline{V}$, следует включить в множество $E_{\min}$. После окончания проверки сформулированного условия выполнения для последней пары вершин из множества $\overline{V}$ выполнение алгоритма заканчивается.

Примечание

Если на шаге 3 алгоритма пометок имеется несколько вершин с одинаковым значением минимальных временных пометок l_j', то следует выбрать любое из них. При этом произвольный выбор дуги с минимальным значением не оказывает принципиального влияния на конечный результат решения задачи. Проверку условия на шаге 5 целесообразно выполнять последовательно, начиная от конечной вершины графа, при этом выстраивать соответствующий путь.

Следует заметить, что в случае определения множества минимальных путей из начальной вершины во все остальные в качестве базового условия на шаге 4 следует проверять условие: $V_0' = \varnothing$. В случае его истинности принимается решение о завершении основных итераций алгоритма, а в случае ложности — осуществляется переход к выполнению действий шага 2 описанного алгоритма.

Рассмотренная схема алгоритма пометок может быть изображена графически в форме диаграммы деятельности языка UML (рис. 7.14).

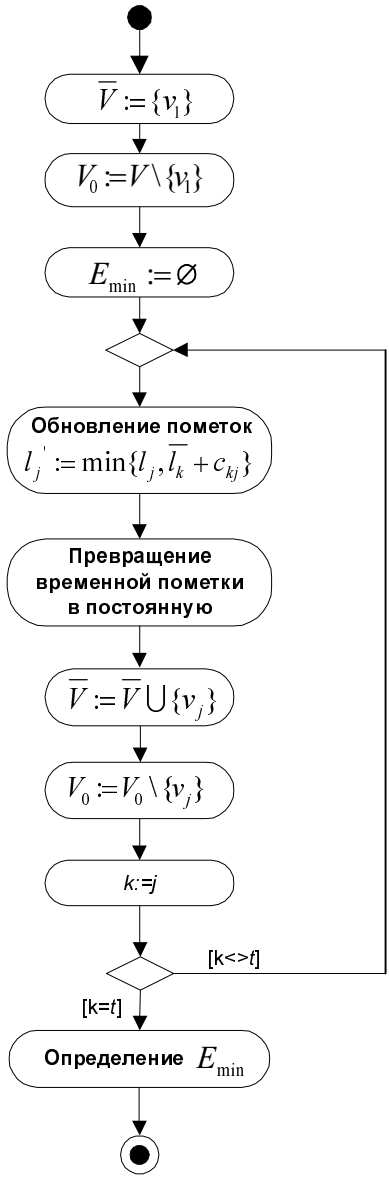


Рис. 7.14. Диаграмма деятельности алгоритма пометок для нахождения минимального пути

Нетрудно заметить, что в силу конечности вершин исходного графа, рассмотренный алгоритм пометок является конечным. Более того, поскольку

заранее известно, что на каждой основной итерации в точности одна вершина исходного графа получает постоянную пометку, общее количество итераций данного алгоритма, связанных с выполняемыми на шагах 2 и 3 действиями, конечно и не превышает: $n - 1$.

Проиллюстрируем использование рассмотренного алгоритма пометок для решения индивидуальной задачи о минимальном пути в ориентированном графе (рис. 7.9). Для этого в соответствии с описанным алгоритмом следует выполнить следующие действия:

Шаг 1. Установить для первой вершины постоянную пометку $\overline{l}_1 = 0$, а для всех остальных вершин — временные пометки: $l_i = \infty \left(\forall v_i \in V \setminus \{v_1\}\right)$, при этом: $V = \{v_1\}$, $V_0 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$, $V=\{v_1\}$, а также: $k = 1$ и $E_{\min} = \varnothing$.

Шаг 2 (первая итерация). Для вершины v_1 множество инцидентных вершин с временными пометками равно: $V_1 = \{v_2, v_3, v_4\}$. Используя формулу (7.3.8), находятся значения временных пометок этих вершин, которые равны: $l_2 = 7$, $l_3 = 11$, $l_4 = 18$.

Шаг 3 (первая итерация). Из всех значений временных пометок выбирается минимальное значение: $l_2 = 7$. Пометка вершины v_2 из временной становится постоянной и равной: $\overline{l}_2 = 7$. При этом изменяется значение $k = 2$.

Шаг 4 (первая итерация). Поскольку условие окончания алгоритма: $k = 8$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (вторая итерация). Для вершины v_2 множество инцидентных вершин с временными пометками равно: $V_2 = \{v_4, v_5\}$. Используя формулу (7.3.8), находятся значения временных пометок этих вершин, которые равны: $l_4 = 13$, $l_5 = 21$.

Шаг 3 (вторая итерация). Из всех значений временных пометок выбирается минимальное значение: $l_4 = 13$. Пометка вершины v_4 из временной становится постоянной и равной: $\overline{l}_4 = 13$. При этом изменяется значение $k = 4$.

Шаг 4 (вторая итерация). Поскольку условие окончания алгоритма: $k = 8$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (третья итерация). Для вершины v_4 множество инцидентных вершин с временными пометками равно: $V_4 = \{v_5, v_6, v_7\}$. Используя формулу (7.3.8), находятся значения временных пометок этих вершин, которые равны: $l_5 = 20$, $l_6 = 18$, $l_7 = 23$.

Шаг 3 (третья итерация). Из всех значений временных пометок выбирается минимальное значение: $l_6 = 18$. Пометка вершины v_6 из временной становится постоянной и равной: $\overline{l}_6 = 13$. При этом изменяется значение $k = 6$.

Шаг 4 (третья итерация). Поскольку условие окончания алгоритма: $k = 8$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (четвертая итерация). Для вершины v_6 множество инцидентных вершин с временными пометками равно: $V_6 = \{v_7, v_8\}$. Используя формулу (7.3.8), находятся значения временных пометок этих вершин, которые равны: $l_7 = 21$, $l_8 = 30$.

Шаг 3 (четвертая итерация). Из всех значений временных пометок выбирается минимальное значение: $l_7 = 21$. Пометка вершины v_7 из временной становится постоянной и равной: $\overline{l}_7 = 21$. При этом изменяется значение $k = 7$.

Шаг 4 (четвертая итерация). Поскольку условие окончания алгоритма: $k = 8$ не выполняется, перейти к выполнению шага 2.

Шаг 2 (пятая итерация). Для вершины v_7 множество инцидентных вершин с временными пометками равно: $V_7 = \{v_8\}$. Используя формулу (7.3.8), находится значение временной пометки вершины v_8, которое равно: $l_8 = 30$.

Шаг 3 (пятая итерация). Поскольку значение временной пометки единственное, оно и принимается за минимальное: $l_8 = 30$. Пометка вершины v_8 из временной становится постоянной и равной: $\overline{l}_8 = 30$. При этом изменяется значение $k = 8$.

Шаг 4 (пятая итерация). Поскольку условие окончания алгоритма: $k = 8$ выполняется, перейти к выполнению шага 5.

Шаг 5. Напомним, что постоянные пометки вершин, найденные на основных итерациях алгоритма, равны: $\overline{l}_1 = 0$, $\overline{l}_2 = 7$, $\overline{l}_4 = 13$, $\overline{l}_6 = 13$, $\overline{l}_7 = 21$, $\overline{l}_8 = 30$. Выполняя проверку условия: $\overline{l}_j - \overline{l}_i = c_{ij}$ для каждой пары вершин, входящих в множество $\overline{V}$ с постоянными пометками, получим два множества дуг, образующих 2 минимальных пути: $E'_{\min} = \{(v_1, v_2), (v_2, v_4), (v_4, v_6), (v_6, v_7), (v_7, v_8)\}$ и $E''_{\min} = \{(v_1, v_2), (v_2, v_4), (v_4, v_6), (v_6, v_8)\}$. На этом выполнение алгоритма заканчивается.

Сравнивая полученный результат с результатом решения данной индивидуальной задачи с помощью программы MS Excel (рис. 7.12), можно констатировать тот факт, что программа MS Excel определила только первый из них. В то же время второй путь (рис. 7.15) также имеет значение минимума целевой функции: $f_{opt} = 30$, в чем можно убедиться непосредственной проверкой. Отсюда следует вывод: алгоритм пометок Дейкстры позволяет определить все минимальные пути между начальной и конечной вершинами в ориентированном графе в том случае, если таких минимальных путей несколько.

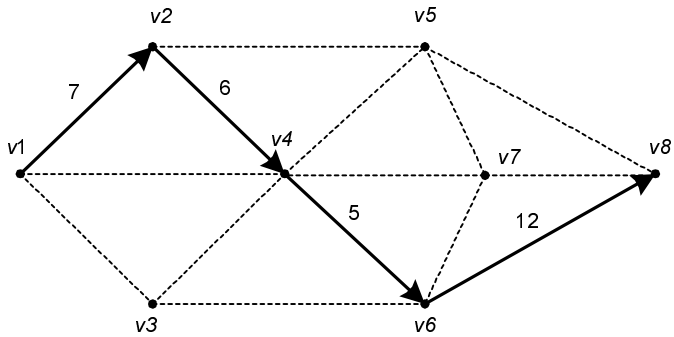


Рис. 7.15. Второе оптимальное решение задачи о минимальном пути в графе между вершинами 1 и 8, полученное с помощью алгоритма пометок

Поскольку второе оптимальное решение не было найдено средством поиска решений программы MS Excel, то данный факт также свидетельствует о неспособности программы MS Excel находить другие оптимальные решения задач оптимизации, если их несколько. Для исправления этой ситуации можно воспользоваться средством программирования MS Excel и реализовать алгоритм пометок в форме программы на языке VBA. Соответствующие программы для решения задач оптимизации на графах рассматриваются в *главе 11*.

Примечание

Заинтересованным читателям в качестве упражнения предлагается самостоятельно найти минимальные пути из начальной вершины этого же исходного графа во все остальные.

Может создаться впечатление, что рассмотренный алгоритм пометок может быть модифицирован для нахождения максимального пути в графе. Это действительно так, несмотря на то, что эти две задачи при схожих формулировках существенно различаются по своей сложности. На этом основании, а также по причине того, что задача нахождения максимального пути в графе имеет самостоятельное прикладное значение в форме задачи нахождения максимального пути графа бизнес-процесса или критического пути в сетевом графе, она рассматривается в *разд. 7.4*.

Примечание

В связи с нахождением минимальных и максимальных путей в графе можно предложить наглядную интерпретацию практического решения первой из них в форме нитей с завязанными узелками. А именно для исходного графа берутся отрезки нитей, длина каждого из которых пропорциональна весу соответст-

вующего ребра исходного графа. Далее эти отрезки нитей завязываются в форме узлов, которые соответствуют вершинам исходного графа. При этом узел, соответствующий начальной вершине, берется в левую руку, а узел, соответствующий конечной вершине, берется в правую руку. После чего руки разводятся в стороны до тех пор, пока нити не натянутся. Нетрудно заметить, что натянутым нитям будет соответствовать путь минимальной длины в исходном графе. Хотя эта интерпретация представляется не более чем шуткой, она может служить практическим способом решения задач нахождения минимального пути в графе между любой парой вершин. Удивительно, что предложить подобный способ для решения задачи нахождения максимального пути в графе не удается. Не является ли данное обстоятельство следствием более глубоких различий между этими двумя видами задач?

В завершение также следует обратить внимание на тот факт, что рассмотренный алгоритм пометок может быть использован также для нахождения минимальных путей как в ориентированном, так и в неориентированном графе. В то же время рассмотренный способ решения данной задачи с помощью программы MS Excel, основанный на модели булевой оптимизации (7.3.6) и (7.3.7), предполагает фиксацию некоторой ориентации ребер для определения исходных переменных. Данный факт ограничивает класс возможных задач соответствующего типа, которые могут быть решены с помощью программы MS Excel, и делает актуальным написание специальной программы, реализующей схему вычислений алгоритма пометок.

7.4. Задача нахождения максимального пути в ориентированном графе

Хотя задача нахождения максимального пути в ориентированном графе по своему характеру во многом похожа на задачу о минимальном пути в графе, она имеет самостоятельное значение в контексте более общей прикладной задачи нахождения критического пути выполнения бизнес-процесса. В настоящем разделе рассматривается ее содержательная постановка, необходимая для формальной записи условий соответствующей задачи оптимизации в виде модели булева программирования.

7.4.1. Содержательная постановка задачи нахождения критического пути выполнения бизнес-процесса

Данная задача является одной из основных при моделировании различных бизнес-процессов, а также при планировании и управлении проектами вы-

полнения работ самого различного целевого назначения. Сущность задачи нахождения критического пути выполнения бизнес-процесса заключается в следующем.

Многие реальные проекты, такие как строительство дома и транспортной сети города, изготовление и сборка машин и механизмов, разработка технических устройств и программного обеспечения, обработка заказов в торговле и логистике, а также процессы приготовления кофе и обучения в институте могут быть детализированы в форме выполнения большого количества различных операций или работ. Некоторые из этих операций могут выполняться одновременно или параллельно, другие — только последовательно, когда та или иная операция может начаться только после окончания других операций.

Например, при строительстве дома можно параллельно выполнять работы по внутренней отделке помещений и озеленению прилегающей к дому территории. При разработке программного обеспечения можно одновременно выполнять написание программ для одних модулей и тестирование других модулей.

В то же время последовательное выполнение операций бизнес-процесса требует согласования времени начала и окончания отдельных работ. Например, при строительстве дома выполнение работы по внутренней отделке помещений может начаться только после того, как будут закончены работы по возведению стен и крыши дома. При разработке программного обеспечения написание программ для отдельных модулей может быть начато после спецификации требований к ним, например, в форме вариантов использования. Даже процесс приготовления кофе с помощью кофеварки предполагает, что прежде чем кофеварка будет включена, в нее следует залить воду, а в фильтр насыпать молотый кофе.

В общем случае модель бизнес-процессов, отражающая последовательность и логическую взаимосвязь выполнения отдельных операций или работ, может быть представлена в форме некоторого конечного ориентированного графа. При этом отдельные операции могут быть представлены как в виде вершин этого графа, так и в виде дуг. Поскольку второй случай интерпретации работ в виде дуг ориентированного графа более удобен для выполнения расчетов общего времени выполнения бизнес-процесса, он традиционно используется при рассмотрении содержательной постановки нахождения критического пути бизнес-процесса.

Таким образом, исходной информацией для моделирования бизнес-процессов является ориентированный граф выполнения операций, каждая дуга которого интерпретируется как отдельная операция или работа этого бизнес-процесса, а вершина — как некоторое событие, связанное с заверше-

нием выполнения тех или иных операций. При этом временная длительность выполнения отдельных операций задается в форме веса соответствующей дуги. Исходя из общей логики выполнения бизнес-процессов, вводится следующее условие — графическая модель отдельного бизнес-процесса должна иметь единственное начальное событие, которое инициирует начало его выполнения, и единственное конечное событие, которое фиксирует момент окончания его выполнения. Применительно к ориентированному графу бизнес-процесса это условие означает, что в данном графе должна быть единственная вершина, из которой *выходят* дуги, и единственная вершина, в которую *входят* дуги.

Дополнительно требуется, чтобы рассматриваемый ориентированный граф модели бизнес-процесса не содержал циклов и был связным, т. е. его конечная вершина была достижима из начальной вершины. Ориентированный граф, удовлетворяющий перечисленным условиям, называется *сетевым графом* или просто *сетью*.

Одна из основных задач моделирования бизнес-процессов, а также более частная задача планирования и управления проектом заключается не только в построении сетевого графа бизнес-процесса, адекватно отражающего общую логику и технологию выполнения операций, но и в оценке общей длительности этого бизнес-процесса.

Следует заметить, что если все операции некоторого бизнес-процесса выполняются последовательно, то его общая длительность равна алгебраической сумме интервалов времени выполнения отдельных операций. Если же операции бизнес-процесса выполняются параллельно, то общая длительность бизнес-процесса, очевидно, равна максимальному интервалу времени параллельно выполняемых операций. Отсюда следует вывод: длительность выполнения бизнес-процесса, представленного в общем случае моделью сетевого графа, равна пути максимальной длины, соединяющего начальную вершину этого графа с его конечной вершиной. Такой путь получил специальное название — *критического пути* в сетевом графе.

Нахождение критического пути в сетевом графе позволяет выявить операции бизнес-процесса, которые наиболее критичны ко времени своего выполнения. Действительно, увеличение времени выполнения операций, лежащих на критическом пути, приводит к однозначному увеличению общего времени выполнения бизнес-процесса. Тем самым управление бизнес-процессом приобретает приоритетный характер и направлено на предотвращение незапланированных задержек с выполнением в первую очередь тех операций, которые входят в критический путь соответствующего сетевого графа. С другой стороны, уменьшение времени выполнения таких операций за счет внутрен-

них резервов бизнес-системы способно привести к сокращению общего времени выполнения всей совокупности операций, что является одной из целей оптимизации или *реинжиниринга* бизнес-процессов.

Таким образом, операции критического пути получают более высокий приоритет по сравнению с остальными операциями бизнес-процесса, а задача построения сетевого графа бизнес-процесса и нахождения критического пути в этом сетевом графе становится важным элементом моделирования бизнес-процессов.

Примечание

Задача нахождения критического пути в сетевом графе традиционно относится к проблематике сетевого планирования и управления проектами, которая дополнительно включает нахождение целого ряда специальных характеристик сети, таких как расчет ранних и поздних сроков наступления событий, резервов времени выполнения операций и других. При необходимости заинтересованные читатели могут более детально познакомиться с этими вопросами с помощью рекомендованной литературы.

7.4.2. Математическая постановка задачи

Рассмотрим ориентированный граф: $G = (V, E, h)$, в котором $V = \{v_1, v_2, ..., v_n\}$ — конечное множество вершин, $E = \{e_1, e_2, ..., e_m\}$ — конечное множество дуг, $h : E \to \boldsymbol{Z}_+$ — весовая функция дуг. Для математической постановки задачи удобно обозначить отдельные значения весовой функции дуг через: $c_{ij} = h(e_k)$, где дуга $e_k \in E$ соответствует упорядоченной паре вершин (v_i, v_j). Согласно содержательной постановке задачи нахождения максимального пути в графе значения $c_{ij} = h((v_i, v_j))$ могут интерпретироваться как длина участка, затраты или стоимость переезда из *i*-го в *j*-й город. Применительно к задаче нахождения критического пути в сетевом графе каждая дуга (v_i, v_j) интерпретируется как отдельная операция бизнес-процесса, а значения c_{ij} — как временная длительность выполнения соответствующей операции (v_i, v_j).

Дополнительно в графе фиксируются две вершины: *начальная* вершина v_s и *конечная* вершина v_t. При этом длина любого маршрута в графе равна сумме весов дуг, входящих в этот маршрут. В предположении, что исходный сетевой граф G является связным, т. е. вершина v_t потенциально достижима

из v_s и не содержит циклов, требуется определить маршрут максимальной длины из начальной вершины v_s в конечную вершину v_t.

Как и в задаче о минимальном пути в ориентированном графе, введем в рассмотрение булевы переменные x_{ij}, которые интерпретируются следующим образом. Переменная $x_{ij} = 1$, если дуга (v_i, v_j) входит в искомый маршрут максимальной длины, и $x_{ij} = 0$, в противном случае, т. е. если дуга (v_i, v_j) не входит в оптимальный маршрут. Тогда в общем случае математическая постановка задачи о максимальном маршруте в ориентированном графе или критическом пути в сети может быть сформулирована следующим образом:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to \max_{x\in\Delta_\beta}, \qquad (7.4.1)$$

где множество допустимых альтернатив Δ_β формируется системой ограничений, которая полностью тождественна системе ограничений математической модели задачи о минимальном пути в ориентированном графе (7.3.2)—(7.3.5). Поскольку последняя была рассмотрена ранее в *разд. 7.3.1*, здесь она не приводится. Тем самым, говоря о математической постановке задачи нахождения максимального маршрута в ориентированном графе, будем иметь

в виду модель (7.4.1), (7.3.2)—(7.3.5).

7.4.3. Решение задачи нахождения критического пути в сетевом графе с помощью программы MS Excel

Для решения задачи нахождения критического пути в сетевом графе с помощью программы MS Excel необходимо задать конкретные значения параметрам индивидуальной задачи. С этой целью для удобства рассмотрим задачу нахождения максимального пути в сетевом графе, структура которого полностью совпадает с графом, являющимся исходным для решения задачи о минимальном пути (рис. 7.9). При этом изменится только лишь интерпретация вершин и дуг, а именно — каждая дуга данного сетевого графа будет означать отдельную операцию некоторого бизнес-процесса, а вершина — событие, связанное с моментом начала или окончания этих операций. Длительность выполнения операций, выраженная, например, в *часах*, равна значению весовой функции для каждой дуги, которое указано рядом с изображением этой дуги в графе.

Требуется найти критический путь, соединяющий начальное событие 1, которому соответствует вершина $v_1 = v_s$, с конечным событием 8, которому соответствует вершина $v_8 = v_t$, так чтобы общая длина пути была максимальной.

Так же, как в задаче о минимальном пути, переменными математической модели данной индивидуальной задачи о критическом пути в сетевом графе являются 15 переменных: x_{12}, x_{13}, x_{14}, x_{24}, x_{25}, x_{34}, x_{36}, x_{45}, x_{46}, x_{47}, x_{57}, x_{58}, x_{67}, x_{68}, x_{78}. Каждая из этих переменных x_{ij} принимает значение 1, если дуга (i, j) входит в критический путь, и 0 — в противном случае. Тогда математическая постановка рассматриваемой индивидуальной задачи о критическом пути в сетевом графе может быть записана в следующем виде:

$$7x_{12} + 11x_{13} + 18x_{14} + 6x_{24} + 14x_{25} + 4x_{34} + 13x_{36} + 7x_{45} + \\ + 5x_{46} + 10x_{47} + 2x_{57} + 15x_{58} + 3x_{67} + 12x_{68} + 9x_{78} \to \max_{x \in \Delta_\beta}, \qquad (7.4.2)$$

где множество допустимых альтернатив Δ_β формируется системой ограничений, которая полностью тождественна системе ограничений математической модели задачи о минимальном пути в ориентированном графе (7.3.7). Поскольку последняя была рассмотрена ранее в *разд. 7.3.2*, здесь она не приводится. Тем самым, говоря о решении индивидуальной задачи нахождения критического пути в сетевом графе, будем иметь в виду математическую модель (7.4.1), (7.3.7).

Для решения задачи нахождения критического пути с помощью программы MS Excel в книге с именем Оптимизация на графах создать новый рабочий лист с именем Критический путь. Скопируем в него данные из ранее созданного листа с именем Минимальный путь, который использовался в *разд. 7.3.2* для решения индивидуальной задачи о нахождении минимального пути в ориентированном графе (7.3.6) и (7.3.7). Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о критическом пути в сетевом графе имеет вид, тождественный изображенному на рис. 7.10.

Примечание

Если исходные данные и соответствующий лист с именем Минимальный путь отсутствуют в книге Оптимизация на графах, то следует выполнить действия 1—12 по предварительной подготовке исходных данных для решения рассматриваемой задачи оптимизации.

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить действия, аналогичные описанным в *разд. 7.3.2*. Однако при выборе варианта оптимизации следует выполнить следующее действие: для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**. Общий вид

диалогового окна спецификации параметров мастера поиска решения имеет следующий вид (рис. 7.16).

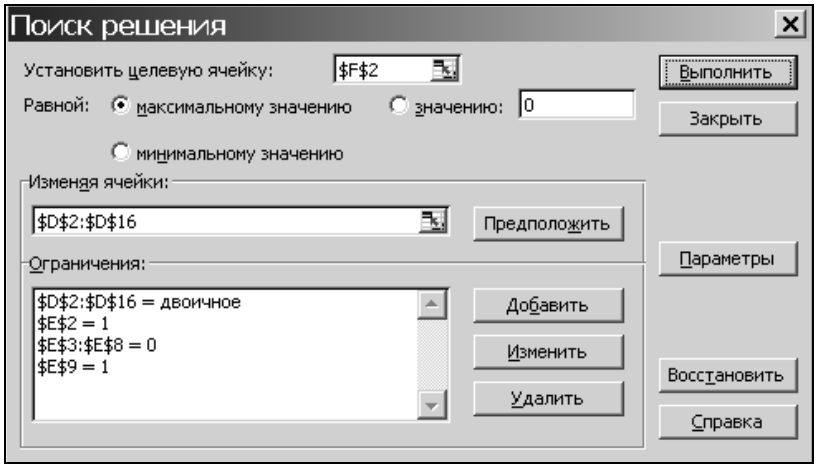


Рис. 7.16. Ограничения на значения переменных и параметры мастера поиска решения для задачи о критическом пути в сетевом графе

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 7.17).

ОптимизацияНаГрафах

	A	B	C	D	E	F	G	H	I	J
1	*vi*	*vj*	*cij*	Переменные:	Ограничения:	Значение ЦФ:				
2	1	2	7	0	1	40				
3	1	3	11	0	0					
4	1	4	18	1	0					
5	2	4	6	0	0					
6	2	5	14	0	0					
7	3	4	4	0	0					
8	3	6	13	0	0					
9	4	5	7	1	1					
10	4	6	5	0						
11	4	7	10	0						
12	5	7	2	0						
13	5	8	15	1						
14	6	7	3	0						
15	6	8	12	0						
16	7	8	9	0						
17										
18										
19										

Минимальный путь | Критический путь

Рис. 7.17. Результат количественного решения задачи о критическом пути в сетевом графе

Результатом решения задачи о минимальном пути в графе являются найденные оптимальные значения переменных: $x_{14} = 1$, $x_{45} = 1$, $x_{58} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствует значение целевой функции: $f_{opt} = 40$.

Анализ найденного решения показывает, что критический путь в исходном сетевом графе (см. рис. 7.9), соединяющий вершину 1 с вершиной 8, содержит следующие дуги: (1, 4), (4, 5), (5, 8). Эти дуги соответствуют критическим операциям моделируемого бизнес-процесса (рис. 7.18). При этом общая продолжительность критического пути будет максимальной и равна 40 *час.*, что соответствует общей плановой продолжительности бизнес-процесса.

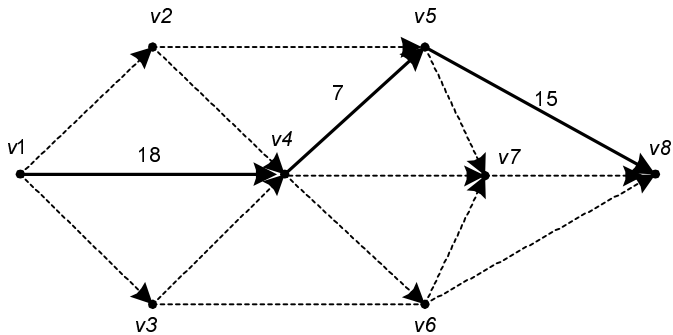


Рис. 7.18. Критический путь в исходном сетевом графе между вершинами 1 и 8

Критический характер операций (1, 4), (4, 5) и (5, 8) проявляется в том, что увеличение любой из них, например, на 1 *час*. приведет к увеличению общей продолжительности реализации бизнес-процесса, которая станет равной 41 *час*. Что касается других операций, не лежащих на критическом пути, то они имеют так называемые резервы времени, которые допускают некоторое увеличение их продолжительности без изменения общей продолжительности соответствующего бизнес-процесса.

Примечание

В общем контексте моделирования бизнес-процессов, а также сетевого планирования и управления проектами, после нахождения критического пути в сетевом графе становится возможным анализ дополнительных характеристик данного сетевого графа, таких как ранних и поздних сроков наступления событий, резервов времени выполнения операций и некоторых других. Решение этой дополнительной задачи предлагается выполнить заинтересованным читателям самостоятельно в качестве упражнения.

7.4.4. Решение задачи нахождения критического пути в сетевом графе с помощью алгоритма расстановки постоянных пометок

Для оценки точности и правильности результатов решения задач о нахождении критического пути в сетевом графе, полученных с помощью программы MS Excel, можно модифицировать рассмотренный ранее в *разд. 7.3.3* алгоритм пометок Дейкстры. Необходимость модификации алгоритма пометок связана с тем, чтобы исключить возможность изменения локально оптимальных участков максимального пути на последующих итерациях алгоритма.

С этой целью необходимо выполнить предварительное упорядочение всех вершин исходного сетевого графа таким образом, чтобы для всех вершин выполнялось следующее формальное условие:

$$r_j > r_i \text{ тогда и только тогда, когда } c_{ij} \in E, \qquad (7.4.3)$$

где r_i, r_j — ранги вершин $v_i, v_j \in V$, соответственно, при этом величины r_i и r_j могут быть произвольными натуральными числами. После нахождения рангов всех вершин исходного сетевого графа алгоритм расстановки постоянных пометок можно представить в виде итеративной процедуры, в рамках которой, с одной стороны, обновление временных пометок выполняется в порядке возрастания рангов вершин, а с другой стороны, для изменения временных пометок используется видоизмененная формула (7.3.8).

Алгоритм расстановки постоянных пометок также имеет итеративный характер и позволяет найти все максимальные пути из начальной вершины во все остальные для ориентированного графа без циклов. Для описания алгоритма расстановки постоянных пометок вводится в рассмотрение только одно множество: $\overline{L} = \{\bar{l}_1, \bar{l}_2, ..., \bar{l}_n\}$ — множество значений *постоянных* пометок вершин V исходного сетевого графа $G = (V, E, h)$.

По определению, постоянная пометка $\bar{l}_i$ некоторой вершины $v_i \in V$ означает окончательную оценку длины максимального пути из начальной вершины $v_s \in V$ в вершину $v_i \in V$. Поскольку окончательная оценка длины максимального пути не может быть улучшена на последующих итерациях алгоритма пометок, значение $\bar{l}_t$ является искомым результатом решения рассматриваемой задачи. Дополнительно вводится в рассмотрение множество рангов вершин $V_r = \{r_1, r_2, ..., r_n\}$, удовлетворяющих условию (7.4.3).

Примечание

Что касается наличия в исходном сетевом графе дуг с весом 0, то в контексте нахождения критического пути изменять эти значения нет необходимости. В дополнение к этому множество дуг, входящих в максимальный путь из начальной вершины v_s в конечную вершину v_t, обозначим через $E_{\max}$.

Алгоритм расстановки постоянных пометок заключается в последовательном выполнении двух отдельных алгоритмов: алгоритма определения рангов вершин сетевого графа и модифицированного алгоритма пометок.

Алгоритм определения рангов вершин сетевого графа имеет итеративный характер и заключается в выполнении следующих действий:

1. *Предварительное определение значений рангов вершин.* До начала выполнения данного алгоритма определить следующие значения рангов: для начальной вершины присвоить ранг $r_s = 1$, а для всех остальных вершин ранги не определены. Для удобства установить: $v_s = v_1$ и $v_t = v_n$, а также определить предварительно множество $\overline{V} = \{v_1\}$ как множество вершин, имеющих значения рангов, и $V_0 = \{v_2, v_3, ..., v_n\}$ как множество вершин, не имеющих значений рангов. После чего можно перейти к выполнению действий шага 2.
2. *Определение ранга вершины.* Выбрать любую вершину v_i из множества V_0, для которой отсутствует значение ранга и для которой отсутствуют дуги типа: $(v_i, v_k) \in E$ для любой вершины $v_k \in \overline{V}$. Присвоить этой вершине ранг: $r_i = \max\{r_j\} + 1$, где максимум берется по всем заданным значениям рангов вершин. Перейти к выполнению действий шага 3.
3. *Изменение множества вершин с рангами.* Изменить множества вершин, имеющих и не имеющих рангов, следующим образом: вершина v_j исключается из множества V_0 и добавляется в множество $\overline{V}$, которые становятся равными: $V_0' = V_0 \setminus \{v_j\}$ и $V' = V \cup \{v_j\}$. Перейти к выполнению действий шага 4.
4. *Проверка условия окончания алгоритма.* Проверить выполнение условия: $V_0' = \varnothing$. Если это условие выполняется, то все вершины исходного сетевого графа имеют ранги, и выполнение данного алгоритма может быть закончено. Если же условие $V_0' = \varnothing$ не выполняется, то установить: $V_0 = V_0', \overline{V} = \overline{V}'$ ' и перейти к выполнению действий шага 2.

Поскольку исходный сетевой граф по определению конечен и не содержит циклов, то рассмотренный алгоритм определения рангов вершин конечен и требует в точности $n-1$ основных итераций. В результате выполнения этого алгоритма будет определено множество рангов вершин $V_r=\{r_1,r_2,...,r_n\}$, удовлетворяющих условию (7.4.3) и необходимое для модифицированного алгоритма пометок.

Модифицированный алгоритм пометок, предназначенный для нахождения максимального пути в ориентированном графе без циклов, имеет итеративный характер и заключается в выполнении следующих действий:

1. *Предварительное определение значения постоянной пометки*. До начала выполнения основных итераций алгоритма определить следующие значения: для начальной вершины установить постоянную пометку $\bar{l}_s=0$, а для всех остальных вершин постоянные пометки не определены. Для удобства установить $v_s=v_1$ и $r=2$, т. е. $\overline{V}=\{v_1\}$ и $V_0=\{v_2,v_3,...,v_n\}$. Множество дуг, входящих в максимальный путь, до начала выполнения алгоритма устанавливается равным: $E_{\max}=\varnothing$. После чего можно перейти к выполнению действий шага 2.

2. *Определение постоянной пометки вершины*. Для вершины $v_i\in V_0$, имеющей ранг $r=r_i$, определить постоянную пометку $\bar{l}_i$ с использованием следующей формулы:

$$\bar{l}_i=\max_{v_k\in\overline{V}}\{\bar{l}_k+c_{ki}\} \qquad (7.4.4)$$

   Изменить множества вершин с постоянными пометками и с отсутствием пометок следующим образом: вершина v_i исключается из множества V_0 и добавляется в множество $\overline{V}$, соответствующие множества вершин становятся равными: $V_0'=V_0\setminus\{v_i\}$ и $\overline{V}'=\overline{V}\cup\{v_i\}$ соответственно. Перейти к выполнению действий шага 3.

3. *Проверка условия окончания основных итераций*. Проверить выполнение условия: $r=n$. Если это условие выполняется, то конечная вершина v_t исходного графа имеет постоянную пометку $\bar{l}_t$, которая равна длине максимального пути в нее из начальной вершины v_s. На этом выполнение основных итераций алгоритма заканчивается и выполняется переход к выполнению действий последнего шага 4, на котором определяется множество дуг $E_{\max}$, входящих в максимальный путь. Если же условие $r=n$ не выполняется, то установить: $V_0=V_0',\overline{V}=\overline{V}',r=i+1$ и перейти к выполнению действий шага 2.

4. *Определение структуры минимального пути*. Для каждой пары вершин, входящих в множество $\overline{V}$ с постоянными пометками, проверить выполнение следующего условия: $\bar{l}_j - \bar{l}_i = c_{ij}$. Если это условие выполняется, то дугу $(v_i, v_j) \in E$, где $v_i, v_j \in \overline{V}$, следует включить в множество $E_{\max}$. После окончания проверки сформулированного условия выполнения для последней пары вершин из множества $\overline{V}$ выполнение алгоритма заканчивается.

Примечание

Проверку условия на шаге 4 целесообразно выполнять последовательно, начиная от конечной вершины графа, при этом выстраивать соответствующий путь.

Нетрудно заметить, что в силу конечности вершин исходного графа, рассмотренный модифицированный алгоритм пометок в целом также является конечным. Более того, поскольку заранее известно, что на каждой основной итерации в точности одна вершина исходного графа получает постоянную пометку, общее количество итераций данного алгоритма, связанных с выполняемыми на шаге 2 действиями, конечно и не превышает значения: $n - 1$.

Алгоритм расстановки постоянных пометок в целом позволяет определить множество максимальных путей из начальной вершины во все остальные вершины сетевого графа. Рассмотренная схема алгоритма расстановки постоянных пометок может быть изображена графически в форме диаграммы деятельности языка UML (рис. 7.19).

Проиллюстрируем использование рассмотренного алгоритма расстановки постоянных пометок для решения индивидуальной задачи о максимальном пути в сетевом графе (рис. 7.9). Для этого в соответствии с описанным алгоритмом следует предварительно определить ранги вершин. Эта процедура, выполняемая с помощью алгоритма определения рангов вершин, достаточно тривиальна и поэтому детально не рассматривается. В качестве результата могут быть получены следующие значения рангов вершин: $r_1 = 1$, $r_2 = 2$, $r_3 = 3$, $r_4 = 4$, $r_5 = 5$, $r_6 = 6$, $r_7 = 7$, $r_8 = 8$, которые совпадают с индексами соответствующих вершин исходного сетевого графа.

Более подробно рассмотрим выполнение модифицированного алгоритма пометок. Для этого в соответствии с описанным алгоритмом выполняются следующие действия.

Шаг 1. Установить для первой вершины постоянную пометку $\bar{l}_1 = 0$, а для всех остальных вершин пометки не определены, при этом: $\overline{V} = \{v_1\}$, $V_0 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$, а также: $r = 2$ и $E_{\max} = \varnothing$.

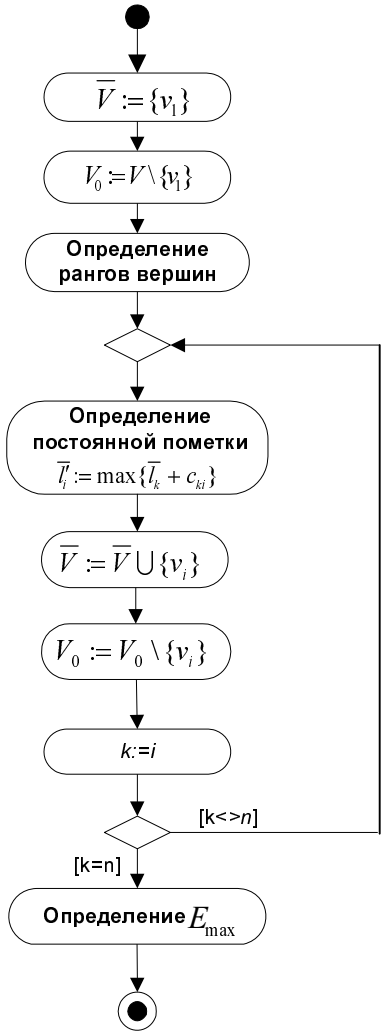


Рис. 7.19. Диаграмма деятельности алгоритма расстановки постоянных пометок для нахождения критического пути

Шаг 2 (первая итерация). Для вершины v_2 с использованием формулы (7.4.4) находится значение постоянной пометки: $\overline{l}_2 = 7$.

Шаг 3 (первая итерация). Поскольку условие окончания алгоритма: $r = 8$ не выполняется, установить $r = 3$ и перейти к выполнению шага 2.

Шаг 2 (вторая итерация). Для вершины v_3 с использованием формулы (7.4.4) находится значение постоянной пометки: $\overline{l}_2 = 7$.

Шаг 3 (вторая итерация). Поскольку условие окончания алгоритма: $r = 8$ не выполняется, установить $r = 4$ и перейти к выполнению шага 2.

Шаг 2 (третья итерация). Для вершины v_4 с использованием формулы (7.4.4) находится значение постоянной пометки: $\overline{l}_4 = 18$.

Шаг 3 (третья итерация). Поскольку условие окончания алгоритма: $r = 8$ не выполняется, установить $r = 5$ и перейти к выполнению шага 2.

Шаг 2 (четвертая итерация). Для вершины v_5 с использованием формулы (7.4.4) находится значение постоянной пометки: $\overline{l}_5 = 25$.

Шаг 3 (четвертая итерация). Поскольку условие окончания алгоритма: $r = 8$ не выполняется, установить $r = 6$ и перейти к выполнению шага 2.

Шаг 2 (пятая итерация). Для вершины v_6 с использованием формулы (7.4.4) находится значение постоянной пометки: $\overline{l}_6 = 24$.

Шаг 3 (пятая итерация). Поскольку условие окончания алгоритма: $r = 8$ не выполняется, установить $r = 7$ и перейти к выполнению шага 5.

Шаг 2 (шестая итерация). Для вершины v_7 с использованием формулы (7.4.4) находится значение постоянной пометки: $\overline{l}_7 = 28$.

Шаг 3 (шестая итерация). Поскольку условие окончания алгоритма: $r = 8$ не выполняется, установить $r = 8$ и перейти к выполнению шага 5.

Шаг 2 (седьмая итерация). Для вершины v_8 с использованием формулы (7.4.4) находится значение постоянной пометки: $\overline{l}_8 = 40$.

Шаг 3 (седьмая итерация). Поскольку условие окончания алгоритма: $r = 8$ выполняется, следует закончить основные итерации и перейти к выполнению шага 4.

Шаг 4. Напомним, что постоянные пометки вершин, найденные на основных итерациях алгоритма, равны: $\overline{l}_1 = 0$, $\overline{l}_2 = 7$, $\overline{l}_3 = 11$, $\overline{l}_4 = 18$, $\overline{l}_5 = 25$, $\overline{l}_6 = 24$, $\overline{l}_7 = 28$, $\overline{l}_8 = 40$. Выполняя проверку условия: $\overline{l}_j - l_i = c_{ij}$ для каждой пары вершин, входящих в множество $\overline{V}$ с постоянными пометками, получим множество дуг, образующих единственный критический путь: $E_{\min} = \{(v_1, v_4), (v_4, v_5)(v_5, v_8)\}$. На этом выполнение алгоритма в целом заканчивается.

Сравнивая полученный результат с результатом решения данной индивидуальной задачи с помощью программы MS Excel (рис. 7.17), можно констатировать их полное совпадение, что может свидетельствовать в пользу правильности полученного результата.

Примечание

Строгое обоснование алгоритма расстановки постоянных пометок для решения задачи нахождения критического пути в сетевом графе может быть выполнено на основе принципа оптимальности Р. Беллмана. В этом контексте рассмотренный алгоритм можно считать специальным случаем метода динамического программирования. Заинтересованные читатели могут более детально познакомиться с этими вопросами с помощью рекомендованной литературы.

7.5. Задача о максимальном потоке в сети

Содержательная постановка задачи о максимальном потоке в сети приводится в *разд. 1.2.10*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи оптимизации в виде модели целочисленного линейного программирования.

7.5.1. Математическая постановка задачи

Пусть $G = (V, E, h)$ — ориентированный граф, в котором $V = \{v_1, v_2, ..., v_n\}$ — конечное множество вершин, $E = \{e_1, e_2, ..., e_m\}$ — конечное множество дуг, $h : E \to \boldsymbol{Z}_+$ — весовая функция дуг, которая интерпретируется как пропускная способность дуги. Дополнительно в графе фиксируются две вершины: начальная вершина v_s, которая называется *исток*, и конечная вершина v_t, которая называется *сток*. В предположении, что исходный граф G является связным, т. е. вершина v_t потенциально достижима из v_s, и не содержит циклов, требуется определить поток максимального объема, протекающий из начальной вершины v_s в конечную вершину v_t.

Введем в рассмотрение следующие неотрицательные целочисленные переменные — x_{ij}, которые интерпретируются как величина потока, проходящего по дуге $(v_i, v_j) \in E$. Тогда в общем случае математическая постановка задачи о максимальном потоке в сети может быть сформулирована следующим образом:

$$\sum_{j=1}^{n} x_{sj} \to \max_{x \in \Delta_\beta}, \tag{7.5.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} \sum_{j=1}^{n} x_{sj} - \sum_{i=1}^{n} x_{it} = 0; & (7.5.2) \\ \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = 0 \ (\forall i \in \{1, 2,..., n\}, i \neq s, i \neq t); & (7.5.3) \\ 0 \leq x_{ij} \leq c_{ij} \ (\forall i, j \in \{1, 2,..., n\}); & (7.5.4) \\ x_{ij} \in \boldsymbol{Z}_{+}^{1} \ (\forall i, j \in \{1, 2,..., n\}). & (7.5.5) \end{cases}$$

При этом первое ограничение (7.5.2) требует выполнения следующего условия: величина потока, выходящего из вершины v_s (истока), должна быть равна величине потока, входящего в вершину v_t (сток). Вторая группа ограничений (7.5.3) гарантирует выполнение следующего условия: любой частичный поток, входящий в каждую промежуточную вершину графа, должен быть равен потоку, выходящему из этой вершины. Общее количество ограничений (7.5.2) и (7.5.3) равно $n - 1$. Третья группа ограничений (7.5.4) требует выполнения следующего условия: величина потока, протекающего по дуге $(v_i, v_j) \in E$, должна быть неотрицательной и не должна превышать пропускной способности этой дуги c_{ij}.

Наконец, последнее ограничение (7.5.5) требует, чтобы все переменные принимали только неотрицательные целочисленные значения. Напомним также, что ориентированный связный граф, не содержащий циклов, принято называть сетью.

Примечание

В литературе встречаются различные варианты задачи о максимальном потоке в сети, а именно: задача нахождения максимального потока с несколькими источниками и стоками, задача нахождения потока минимальной стоимости и другие. Для решения этих задач разработаны специальные методы, которые зачастую оказываются более эффективными, чем поиск решения программой MS Excel.

7.5.2. Решение задачи о максимальном потоке в сети с помощью программы MS Excel

Для решения задачи о максимальном потоке в сети с помощью программы MS Excel необходимо задать конкретные значения параметрам индивидуальной задачи. С этой целью рассмотрим задачу нахождения максимального

потока в сети, которая представляет модель системы магистральных трубопроводов, связывающих источник добычи некоторого жидкого продукта (нефти, сжиженного газа) с предприятием по его промышленной переработке. Данная модель может быть представлена в виде схемы, формально представляющей собой ориентированный связный граф без циклов, состоящий из 6 вершин и 10 дуг (рис. 7.20). Предельные значения пропускной способности каждого участка рассматриваемой системы между двумя соседними компрессорными станциями, выраженные, например, в *т/час*, равны значению весовой функции для каждой дуги, которые указаны рядом с изображением этой дуги в графе.

В предположении, что источник, которому соответствует вершина $v_1 = v_s$, обладает достаточными запасами продукта, требуется определить количество транспортируемого продукта по каждому из участков трубопроводной системы до стока, которому соответствует вершина $v_6 = v_t$, так чтобы количество доставленного на сток продукта было максимальным.

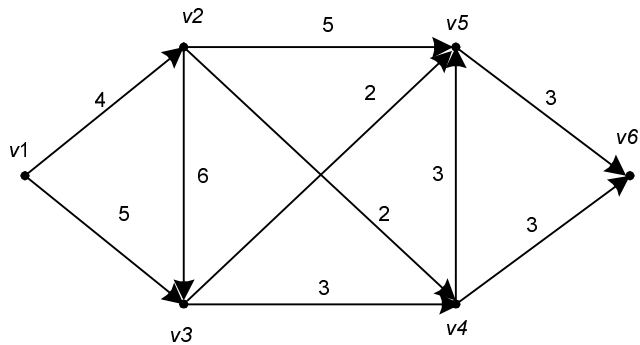


Рис. 7.20. Исходный ориентированный граф индивидуальной задачи о максимальном потоке в сети

Переменными математической модели данной индивидуальной задачи о максимальном потоке в сети являются 10 переменных: x_{12}, x_{13}, x_{23}, x_{24}, x_{25}, x_{34}, x_{35}, x_{45}, x_{46}, x_{56}. Каждая из этих переменных x_{ij} может принимать неотрицательное целочисленное значение, не превышающее пропускной способности дуги c_{ij} и соответствующее величине потока продукта, транспортируемого по отдельному трубопроводу, связывающему компрессорные станции — вершины сети. Тогда математическая постановка рассматриваемой индивидуальной задачи о максимальном потоке в сети может быть записана в следующем виде:

$$x_{12} + x_{13} \to \min_{x \in \Delta_\beta}, \tag{7.5.6}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств и неравенств:

$$\begin{cases} x_{12} + x_{13} - x_{46} - x_{56} = 0; \\ x_{12} - x_{23} - x_{24} - x_{25} = 0; \\ x_{13} + x_{23} - x_{34} - x_{35} = 0; \\ x_{24} + x_{34} - x_{45} - x_{46} = 0; \\ x_{25} + x_{35} + x_{45} - x_{56} = 0; \\ 0 \le x_{12} \le 4,\ 0 \le x_{13} \le 5,\ 0 \le x_{23} \le 6,\ 0 \le x_{24} \le 2; \\ 0 \le x_{25} \le 5,\ 0 \le x_{34} \le 3,\ 0 \le x_{35} \le 2,\ 0 \le x_{45} \le 3; \\ 0 \le x_{46} \le 3,\ 0 \le x_{56} \le 3; \\ x_{12}, x_{13}, x_{23}, x_{24}, x_{25}, x_{34}, x_{35}, x_{45}, x_{46}, x_{56} \in \boldsymbol{Z}_+^1. \end{cases} \quad (7.5.7)$$

Заметим, что те переменные x_{ij}, для которых весовая функция дуг h не определена или равна 0, не входят в математическую постановку рассматриваемой задачи (7.5.6) и (7.5.7).

Для решения данной индивидуальной задачи с помощью программы MS Excel создадим в книге с именем Оптимизация на графах новый рабочий лист с именем Максимальный поток. После этого следует выполнить следующие действия:

1. Внесем необходимые надписи в ячейки **A1:F1** (рис. 7.21). Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи.
2. В ячейки **A2:A11** введем индексы начальных вершин, а в ячейки **B2:B11** — индексы конечных вершин всех имеющихся дуг исходного графа.
3. В ячейки **C2:C11** введем значения пропускных способностей дуг исходного графа.
4. В ячейку **F2** введем формулу: `=СУММ(D2:D3)`, которая представляет целевую функцию (7.5.6).
5. В ячейку **E2** введем формулу: `=СУММ(D2:D3)-СУММ(D10:D11)`, которая представляет собой левую часть первого ограничения (7.5.7).
6. В ячейку **E3** введем значение левой части второго ограничения: `=D2-СУММ(D4:D6)`.
7. В ячейку **E4** введем значение левой части третьего ограничения: `=D3-СУММ(D7:D8)`.

8. В ячейку **E5** введем значение левой части четвертого ограничения: `=СУММ(D5;D7)-СУММ(D9:D10)`.

9. И, наконец, в ячейку **E6** введем значение левой части пятого ограничения: `=СУММ(D6;D8:D9)-D11`.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о минимальном пути в графе имеет следующий вид (рис. 7.21).

ОптимизацияНаГрафах

	A	B	C	D	E	F
1	*vi*	*vj*	*cij*	Переменные:	Ограничения:	Значение ЦФ:
2	1	2	4		=СУММ(D2:D3)-СУММ(D10:D11)	=СУММ(D2:D3)
3	1	3	5		=D2-СУММ(D4:D6)	
4	2	3	6		=D3-СУММ(D7:D8)	
5	2	4	2		=СУММ(D5;D7)-СУММ(D9:D10)	
6	2	5	5		=СУММ(D6;D8:D9)-D11	
7	3	4	3			
8	3	5	2			
9	4	5	3			
10	4	6	3			
11	5	6	3			
12						
13						
14						
15						
16						

Максимальный поток

Рис. 7.21. Исходные данные для решения задачи о максимальном потоке в сети

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$F$2`.

2. Для группы **Равной:** выбрать вариант поиска решения — **максимальному значению**.

3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$D$2:$D$11`.

4. Задать первых 5 ограничений для рассматриваемой задачи. С этой целью выполнить следующие действия:

   - для задания этих ограничений в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;

- в появившемся дополнительном окне выбрать диапазон ячеек **$E$2:$E$11**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
- в качестве знака ограничения из выпадающего списка выбрать равенство "=";
- в качестве значения правой части ограничения ввести с клавиатуры значение 0;
- для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

5. Задать первое из 10 ограничений на пропускные способности дуг (7.5.7). С этой целью выполнить следующие действия:
   - для задания ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$D$2**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения в появившемся дополнительном окне выбрать ячейку **$C$2**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
6. Аналогичным образом добавить остальные 9 ограничений (7.5.7), используя в качестве исходных ячеек **$D$3: $D$11** и **$C$3: $C$11**.
7. Задать ограничение на целочисленные значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$D$2:$D$11**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**цел**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "целочисленные";

- для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

8. В окне дополнительных параметров поиска решения выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения показан на рис. 7.22.

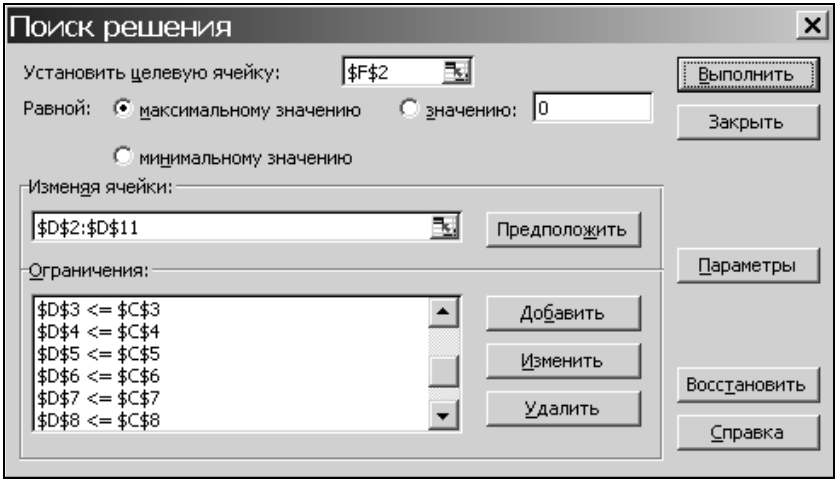


Рис. 7.22. Ограничения значений переменных и параметры мастера поиска решения для задачи о максимальном потоке в сети

ОптимизацияНаГрафах

	A	B	C	D	E	F	G	H	I
1	*vi*	*vj*	*cij*	Переменные:	Ограничения:	Значение ЦФ:			
2	1	2	4	2	0	6			
3	1	3	5	4	0				
4	2	3	6	0	0				
5	2	4	2	2	0				
6	2	5	5	0	0				
7	3	4	3	3					
8	3	5	2	1					
9	4	5	3	2					
10	4	6	3	3					
11	5	6	3	3					
12									
13									
14									
15									
16									

Максимальный поток

Рис. 7.23. Результат количественного решения задачи о максимальном потоке в сети

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 7.23).

Результатом решения задачи о максимальном потоке в сети являются найденные оптимальные значения переменных: $x_{12} = 2$, $x_{13} = 4$, $x_{24} = 2$, $x_{34} = 3$, $x_{35} = 1$, $x_{45} = 2$, $x_{46} = 3$, $x_{56} = 3$, остальные переменные равны 0. Найденному оптимальному решению соответствует значение целевой функции: $f_{opt} = 6$.

Анализ найденного решения показывает, что максимальный поток в сети (см. рис. 7.20), проходящий из вершины 1 в вершину 6, протекает по следующим дугам: по дуге (1, 2) в количестве 2 *т/час*, по дуге (1, 3) в количестве 4 *т/час*, по дуге (2, 4) в количестве 2 *т/час*, по дуге (3, 4) в количестве 3 *т/час*, по дуге (3, 5) в количестве 1 *т/час*, по дуге (4, 5) в количестве 2 *т/час*, по дуге (4, 6) в количестве 3 *т/час*, по дуге (5, 6) в количестве 3 *т/час*.

Тем самым найден оптимальный план транспортировки продукта от пункта его добычи до пункта его переработки (рис. 7.24). При этом общая величина потока транспортируемого продукта для рассматриваемой сети будет максимальна и равна 6 *т/час*.

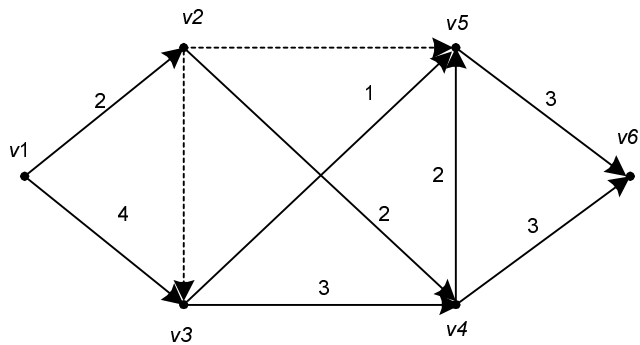


Рис. 7.24. Максимальный поток в исходной сети между вершинами 1 и 6

Простейшая проверка найденного решения может быть связана с рассмотрением так называемых разрезов графа сети (рис. 7.24). При этом в общем случае под *разрезом* ориентированного графа понимается такое подмножество его дуг, которое содержит минимальное количество дуг, удаление которых из исходного графа делает полученный в результате граф несвязным.

Например, множество дуг {(2, 4), (2, 5), (3, 4) и (3, 5)} удовлетворяет определению разреза, поскольку их удаление разделяет исходный граф на две несвязные части. Имеет место следующее свойство, которому должен удов-

летворять максимальный поток в сети: максимальный поток в сети равен потоку, протекающему по дугам любого разреза этой сети, отделяющего исток от стока. При этом поток по дугам, проходящий в направлении от истока к стоку, учитывается со знаком "+", а поток по дугам, проходящий в направлении от стока к истоку, учитывается со знаком "–".

Для проверки этого условия применительно к найденному решению задачи о максимальном потоке в сети (см. рис. 7.24) рассмотрим некоторый разрез, например, {(2, 4), (2, 5), (3, 4) и (3, 5)}. Нетрудно убедиться в том, что величина потока, проходящего через дуги этого разреза, равна 6, что в точности равно величине найденного максимального потока. Для дополнительной проверки получаемых с помощью программы MS Excel решений данной типовой задачи оптимизации на графах можно воспользоваться специальным алгоритмом пометок Форда — Фалкерсона, который рассматривается в *разд. 7.5.3*.

7.5.3. Решение задачи о максимальном потоке в сети с помощью алгоритма пометок Форда — Фалкерсона

Для оценки точности и правильности результатов решения задач о максимальном потоке в сети, полученных с помощью программы MS Excel, можно воспользоваться специально разработанным для решения задач данного класса так называемым *алгоритмом пометок Форда — Фалкерсона*. Этот алгоритм, предложенный в 1957 г. Фордом и Фалкерсоном, является одним из наиболее эффективных методов решения задачи о максимальном потоке в сети, учитывающих специфику рассматриваемой задачи оптимизации на графах.

Алгоритм пометок Форда — Фалкерсона имеет итеративный характер и позволяет найти максимальный поток в сети, выходящий из начальной вершины и входящий в конечную вершину сети. Сущность алгоритма заключается в построении на каждой итерации некоторого частичного потока, также выходящего из начальной вершины и входящего в конечную вершину исходной сети. Все такие частичные потоки в совокупности и представляют искомый максимальный поток.

Для описания алгоритма пометок Форда — Фалкерсона вводятся в рассмотрение 2 множества значений: $\tilde{N}_0 = \{c_{ij}\}$ — множество дуг, образуемых некоторый поток, протекающий через дуги $(i, j) \in E, (i, j \in \{1, 2, ..., n\})$ исходной сети $G = (V, E, h)$, и $D_{\max} = \{d_{ij}\}$ — множество значений максимального потока, протекающего через каждую дугу $(i, j) \in E, (\forall i, j \in \{1, 2, ..., n\})$ исходной

сети $G=(V, E, h)$. По определению, частичный и максимальный потоки выходят из начальной вершины $v_s \in V$ и входят в конечную вершину $v_i \in V$ сети $G=(V, E, h)$.

Отдельное множество значений $C_0 = \{c_{ij}\}$ соответствует некоторому связному ориентированному пути, который начинается в начальной вершине $v_s \in V$ и заканчивается в конечной вершине $v_t \in V$. При этом величина потока, которая может быть передана по этому пути, равна минимальной пропускной способности c_{ij} среди всех дуг, образующих этот связный ориентированный путь. При этом в качестве матрицы смежности исходной сети $G=(V, E, h)$ удобно рассматривать матрицу $C=\{c_{ij}\}$, которая содержит пропускные способности дуг: $h(i, j) \in E, (\forall i, j \in \{1, 2,..., n\})$ исходной сети $G=(V, E, h)$. До начала выполнения алгоритма предполагается, что $C_0 = \varnothing$ и $D_{max} = \varnothing$, при этом соответствующие элементы этих множеств удобно считать равными 0.

Алгоритм расстановки пометок Форда — Фалкерсона или просто *алгоритм пометок Форда — Фалкерсона* имеет итеративный характер и заключается в выполнении следующих действий:

1. *Нахождение ориентированного пути.* С использованием некоторой процедуры перебора найти некоторый ориентированный путь C_0, выходящий из начальной вершины и входящий в конечную вершину сети, которая задана матрицей смежности C. Перейти к выполнению действий шага 2.
2. *Определение величины частичного потока.* В качестве величины частичного потока, протекающего по ориентированному пути C_0, присвоить значение: $c_0 = \boldsymbol{min}\{c_{ij}\}$, где минимум берется по всем значениям пропускных способностей дуг, входящих в множество C_0. Перейти к выполнению действий шага 3.
3. *Изменение значений множеств D_{max} и C.* Изменить значения максимального потока, образующих множество D_{max}, следующим образом: $d'_{ij} = d_{ij} + c_0$. Изменить значения пропускных способностей дуг, образующих матрицу смежности C, следующим образом: $c'_{ij} = c_{ij} - c_0$. При этом изменяются только значения дуг, входящих в множество C_0. Перейти к выполнению действий шага 4.
4. *Проверка условия окончания алгоритма.* Проверить выполнение условия: в сети G', образуемой матрицей смежности C', конечная вершина v_t является достижимой из начальной вершины v_s. Если это условие не выполняется, то в сети G' отсутствует ориентированный путь C_0, выходящий

из начальной вершины и входящий в конечную вершину сети, и выполнение данного алгоритма может быть закончено. Если же данное условие выполняется, то установить: $D_{max} = D'_{max}, C = C', C_0 = \varnothing$ и перейти к выполнению действий шага 1.

Поскольку исходный сетевой граф по определению конечен и не содержит циклов, то рассмотренный алгоритм пометок Форда — Фалкерсона также конечен. Однако определить в общем случае точное количество необходимых для его выполнения итераций не представляется возможным, поскольку заранее неизвестно количество различных ориентированных путей, которые связывают начальную вершину сети с ее конечной вершиной. Можно только оценить максимальное количество возможных итераций, которое служит оценкой трудоемкости данного алгоритма в худшем случае.

В результате выполнения этого алгоритма будет определено $D_{max} = \{d_{ij}\}$ множество значений максимального потока, протекающего через каждую дугу $(i, j) \in E$ $(\forall i, j \in \{1, 2, \ldots, n\})$ исходной сети $G = (V, E, h)$.

Таким образом, алгоритм пометок Форда — Фалкерсона позволяет определить множество значений максимального потока, протекающего через каждую дугу исходной сети. Рассмотренная схема алгоритма может быть изображена графически в форме диаграммы деятельности языка UML (рис. 7.25).

Проиллюстрируем использование рассмотренного алгоритма пометок Форда — Фалкерсона для решения индивидуальной задачи о максимальном потоке в сетевом графе (см. рис. 7.20). Для этого в соответствии с описанным алгоритмом выполняются следующие действия:

Шаг 1 (первая итерация). Для исходной матрицы смежности C определяется ориентированный путь C_0, выходящий из начальной вершины и входящий в конечную вершину сети, который содержит следующие пропускные способности дуг: $C_0 = \{c_{12}, c_{25}, c_{56}\}$.

Шаг 2 (первая итерация). Величина частичного потока по найденному пути $C_0 = \boldsymbol{min}\{4, 5, 3\} = 3$.

Шаг 3 (первая итерация). Новые значения множества D_{max} равны: $\{d_{12} = 3, d_{25} = 3, d_{56} = 3\}$. Новые значения матрицы смежности C равны: $\{c_{12} = 1, c_{13} = 5, c_{23} = 6, c_{24} = 2, c_{25} = 2, c_{34} = 3, c_{35} = 2, c_{45} = 3, c_{46} = 3\}$. Остальные значения этих матриц равны 0. Данной матрице смежности соответствует новый сетевой граф (рис. 7.26).

Шаг 4 (первая итерация). Поскольку условие окончания алгоритма не выполняется, что видно из визуального анализа рис. 7.26, следует перейти к выполнению шага 1.

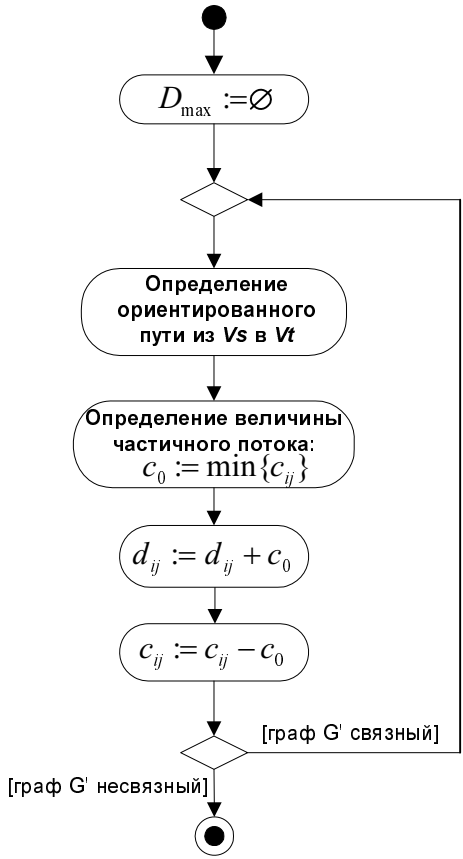


Рис. 7.25. Диаграмма деятельности алгоритма пометок Форда — Фалкерсона для нахождения критического пути

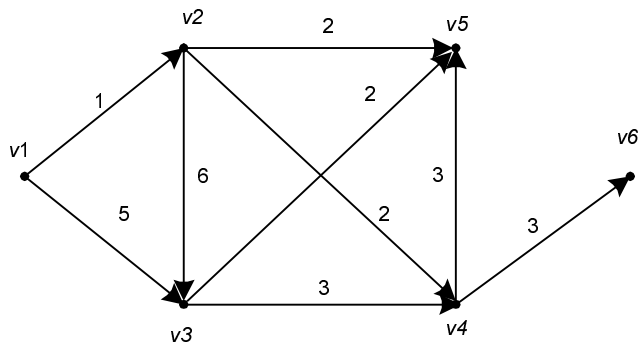


Рис. 7.26. Новый сетевой граф индивидуальной задачи о максимальном потоке в сети, полученный на первой итерации алгоритма

Шаг 1 (вторая итерация). Для новой матрицы смежности C определяется ориентированный путь C_0, выходящий из начальной вершины и входящий в конечную вершину сети, который содержит следующие пропускные способности дуг: $C_0 = \{c_{13}, c_{34}, c_{46}\}$.

Шаг 2 (вторая итерация). Величина частичного потока по пути C_0 равна: ***min***$\{5, 3, 3\}=3$.

Шаг 3 (вторая итерация). Новые значения множества D_{max} равны: $\{d_{12} = 3, d_{25} = 3, d_{56} = 3, d_{13} = 3, d_{34} = 3, d_{46} = 3\}$. Новые значения матрицы смежности C равны: $\{c_{12} = 1, c_{13} = 2, c_{23} = 3, c_{25} = 5, c_{35} = 2\}$. Остальные значения этих матриц равны 0. Данной матрице смежности соответствует новый сетевой граф (рис. 7.27).

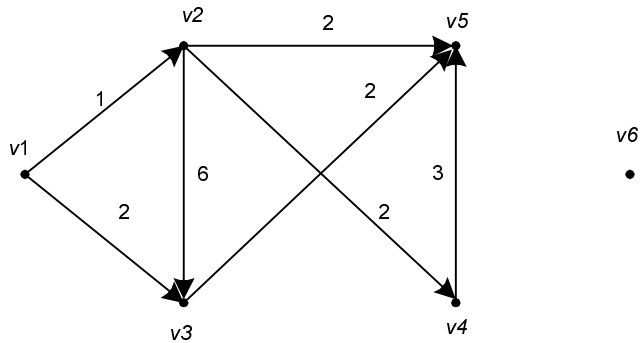


Рис. 7.27. Новый сетевой граф индивидуальной задачи о максимальном потоке в сети, полученный на второй итерации алгоритма

Шаг 4 (вторая итерация). Поскольку условие окончания алгоритма выполняется, а также из визуального анализа рис. 7.27 видно, что отсутствует связный ориентированный путь из начальной вершины в конечную вершину, то на этом выполнение данного алгоритма заканчивается.

Результатом решения задачи о максимальном потоке в сети с помощью алгоритма Форда — Фалкерсона являются найденные оптимальные значения максимального потока D_{max}, протекающего через каждую дугу исходной сети: $d_{12} = 3$, $d_{25} = 3$, $d_{56} = 3$, $d_{13} = 3$, $d_{34} = 3$, $d_{46} = 3$, остальные значения равны 0. Найденному оптимальному решению соответствует значение целевой функции: $f_{opt} = 6$.

Анализ найденного решения показывает, что максимальный поток в сети (см. рис. 7.20), проходящий из вершины 1 в вершину 6, протекает по следующим дугам: по дуге (1, 2) в количестве 3 *т/час*, по дуге (1, 3) в количестве 3 *т/час*, по дуге (2, 5) в количестве 3 *т/час*, по дуге (3, 4) в количестве 3 *т/час*,

по дуге (4, 6) в количестве 3 *т/час*, по дуге (5, 6) в количестве 3 *т/час*. При этом общая величина потока транспортируемого продукта для рассматриваемой сети также равна 6 *т/час*.

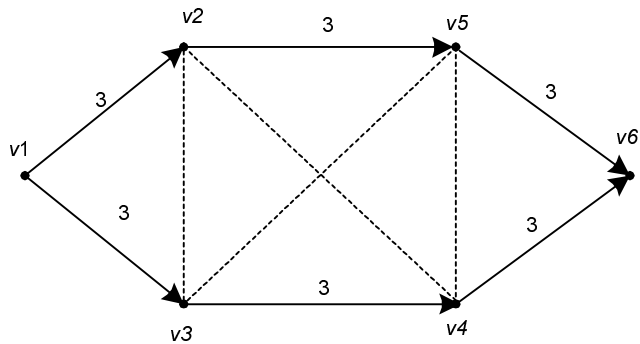


Рис. 7.28. Максимальный поток в исходной сети между вершинами 1 и 6, найденный с помощью алгоритма пометок Форда — Фалкерсона

Сравнивая полученный результат с результатом решения данной индивидуальной задачи с помощью программы MS Excel (см. рис. 7.23), можно констатировать, что данный оптимальный план транспортировки продукта от пункта его добычи до пункта его переработки (рис. 7.28) отличается от оптимального плана, полученного с помощью программы MS Excel. В действительности, для рассмотренной индивидуальной задачи существуют и другие оптимальные решения с тем же максимальным значением целевой функции. Таким решением, например, является поток: $d_{12} = 3$, $d_{25} = 1$, $d_{24} = 2$, $d_{56} = 3$, $d_{13} = 3$, $d_{34} = 1$, $d_{35} = 2$, $d_{46} = 3$, остальные значения равны 0. Это решение также является оптимальным, поскольку ему соответствует значение целевой функции: $f_{opt} = 6$.

Тем самым можно сделать вывод, что для данной индивидуальной задачи имеется несколько оптимальных решений с максимальным значением целевой функции $f_{opt} = 6$, причем ни программа MS Excel, ни алгоритм пометок Форда — Фалкерсона не позволяют найти все эти решения. В случае необходимости для этой цели следует использовать другие методы и алгоритмы, которые специально ориентированы на решение этой типовой задачи оптимизации на графах.

Примечание

Кроме алгоритма пометок Форда — Фалкерсона для решения задачи о максимальном потоке в сети предложены и другие алгоритмы, наиболее эффектив-

ными из которых с вычислительной точки зрения являются алгоритмы Е. А. Диница (1970, Россия) и А. В. Карзанова (1974, Россия). Заинтересованные читатели могут более детально познакомиться с этими алгоритмами с помощью рекомендованной литературы.

7.6. Упражнения

В качестве упражнений для самостоятельного решения предлагаются задачи, аналогичные типовым задачам оптимизации на графах, рассмотренным в данной главе. Предлагаемые в качестве упражнений задачи оптимизации содержат конкретные значения исходных данных в форме заданного графа, что позволяет получить их количественное решение с помощью программы MS Excel. Те из читателей, кто сочтет для себя интересным найти решение данных задач несколькими способами, получат возможность убедиться в правильности полученных решений.

7.6.1. Задача о минимальном и максимальном покрывающем дереве в графе

Для графа, изображенного на рис. 7.29, определить минимальное и максимальное покрывающее дерево.

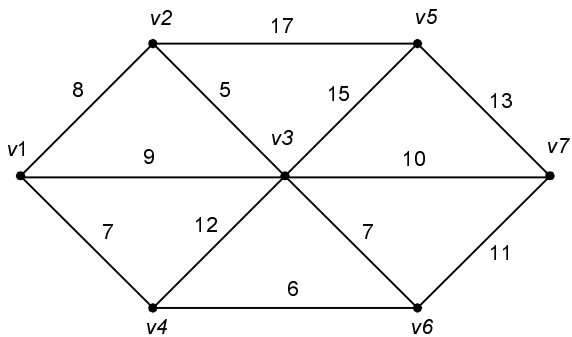


Рис. 7.29. Исходный граф для задачи о нахождении минимального и максимального покрывающего дерева

7.6.2. Задача о минимальном и максимальном пути в ориентированном графе

Для графа, изображенного на рис. 7.30, определить пути минимальной и максимальной длины из начальной вершины с номером 1 в конечную вершину с номером 10.

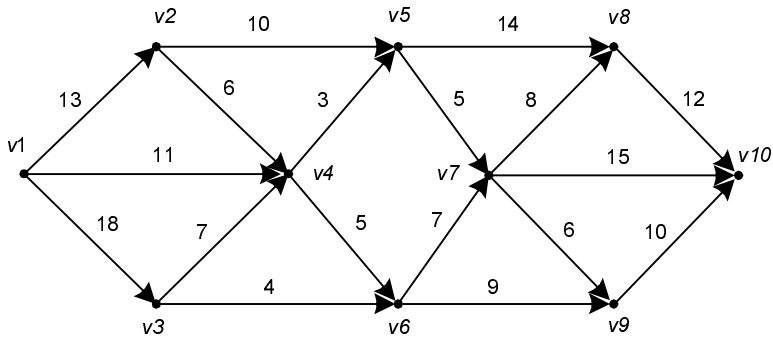


Рис. 7.30. Исходный ориентированный граф для задачи о нахождении минимального и максимального пути

7.6.3. Задача о максимальном потоке в сети

Для системы магистральных трубопроводов, связывающих источник добычи нефти с предприятием по ее промышленной переработке, представленной в форме графа (рис. 7.31), определить количество транспортируемой нефти по каждому из участков трубопроводной системы, так чтобы количество доставленной на предприятие переработки нефти было максимальным. Предельные значения пропускной способности каждого участка рассматриваемой системы указаны рядом с обозначением соответствующей дуги в графе сети. При этом предполагается, что источник добычи нефти (вершина с номером 1) обладает достаточными запасами нефти для ее транспортировки, а предприятие по переработке нефти (вершина с номером 7) обладает достаточными возможностями по ее переработке.

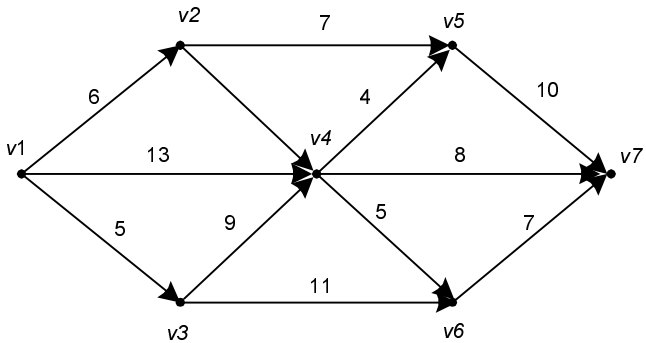
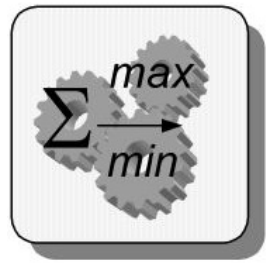


Рис. 7.31. Исходная сеть для задачи о нахождении максимального потока



Глава 8

Задачи комбинаторной оптимизации

Многие интересные задачи оптимизации могут быть сформулированы в форме задач комбинаторной оптимизации. Формальная постановка и методы решения этих задач базируются на понятиях комбинаторики или комбинаторного анализа. В связи с этим изучение общих свойств задач комбинаторной оптимизации приобретает самостоятельное значение, а методы нахождения их точного решения традиционно относят к наиболее сложным вопросам практики построения вычислительных алгоритмов.

8.1. Общая характеристика задач комбинаторной оптимизации

В общем случае к *комбинаторным задачам* относятся такие математические задачи, формулировка которых и поиск решения связан с предметом специального раздела математики — комбинаторным анализом или комбинаторикой. В рамках этого раздела изучаются математические свойства специальных *комбинаторных объектов*, основными из которых являются: перестановки, размещения и сочетания *(см. приложение 1)*. При этом многие комбинаторные задачи имеют развлекательное содержание и форму головоломок.

Одной из классических комбинаторных задач, фигурирующей еще в мифах Древнего Востока, является задача построения магического квадрата. Ряд комбинаторных задач был рассмотрен Л. Эйлером. Одна из них — задача об офицерах, состоящая в том, чтобы 36 офицеров 6-ти различных воинских званий и из 6-ти различных полков расположить в ячейках квадрата 6×6, так чтобы каждая колонна и каждая шеренга содержали одновременно одного и только одного офицера каждого воинского звания и каждого полка. Эта задача эквивалентна задаче построения латинского квадрата порядка 6. Другая комбинаторная задача, также рассмотренная Л. Эйлером, — известная задача

о кёнигсбергских мостах, которая положила начало новому разделу математики — теории графов.

К задачам *комбинаторной оптимизации* относятся такие задачи оптимизации, которые связаны с нахождением оптимального значения некоторой целевой функции на множестве некоторых комбинаторных объектов. Хотя отдельные задачи комбинаторной оптимизации в своей исходной постановке формулируются как задачи оптимизации на графах, характерным их свойством является наличие у множества допустимых альтернатив некоторой комбинаторной структуры. Наличие подобной структуры позволяет не только априори оценить мощность исходного множества достижимых альтернатив, но и обоснованно подойти к разработке вычислительных методов решения соответствующих задач оптимизации.

Типовыми задачами комбинаторной оптимизации являются: задача о разбиении, задачи о разбиении и покрытии, задачи размещения различных геометрических фигур на плоскости и в пространстве, задачи нахождения комбинаторных объектов, обладающих некоторыми экстремальными свойствами.

8.1.1. Математическая постановка задачи комбинаторной оптимизации

Задачи комбинаторной оптимизации в своей исходной постановке могут формулироваться непосредственно в форме нахождения комбинаторного объекта того или иного типа, доставляющего оптимальное значение некоторой целевой функции. Однако многие прикладные задачи комбинаторной оптимизации первоначально формулируются в форме некоторой задачи оптимизации на графах. Подобная задача оптимизации на графах оказывается тесно связанной с тем или иным комбинаторным объектом, что и позволяет отнести ее к классу задач комбинаторной оптимизации.

В общем случае задача комбинаторной оптимизации обладает тем свойством, что в рамках ее постановки каждому комбинаторному объекту определенного типа ставится в соответствие некоторое количественное значение целевой функции, рассматриваемой в качестве элемента исходных данных. Поскольку для всех комбинаторных объектов имеется конструктивный способ их перечисления, задачи комбинаторной оптимизации оказываются разрешимыми с вычислительной точки зрения.

Таким образом, задача комбинаторной оптимизации формулируется как нахождение такого специального комбинаторного объекта, который доставляет максимальное или минимальное значение целевой функции. Более строгое определение данного класса задач можно получить на основе рассмотрения

общей математической постановки задачи комбинаторной оптимизации и ее связи с задачами булева программирования.

В общем случае математическая постановка задачи комбинаторной оптимизации может быть сформулирована в следующем виде:

$$f(x) \to \max_{x \in \Delta_\beta} \quad \text{или} \quad f(x) \to \min_{x \in \Delta_\beta}, \tag{8.1.1}$$

где переменная x условно обозначает некоторый специальный комбинаторный объект, такой как: перестановка, размещение или сочетание, а множество допустимых альтернатив Δ_β содержит все возможные комбинаторные объекты рассматриваемого типа. При этом никаких дополнительных предположений о характере целевой функции или ограничений не делается. Тем не менее, в практических задачах комбинаторной оптимизации множество допустимых альтернатив типовых задач является конечным. Это обстоятельство позволяет в простых случаях находить решение той или иной конкретной задачи комбинаторной оптимизации методом простого перебора всех комбинаторных объектов того или иного типа.

При рассмотрении математической постановки задачи комбинаторной оптимизации следует также иметь в виду, что для отдельных индивидуальных задач множество допустимых альтернатив может оказаться пустым. Например, поиск оптимального полного замкнутого пути для несвязных графов теряет свой смысл. Именно по этой причине в исходной постановке задач комбинаторной оптимизации необходимо указывать дополнительные свойства, которым должны удовлетворять ее исходные данные, чтобы соответствующая задача оптимизации имела допустимые решения.

8.1.2. Основные методы решения задач комбинаторной оптимизации

Хотя общая математическая постановка задачи комбинаторной оптимизации (8.1.1) не дает никакой информации относительно возможных методов ее решения, тем не менее, все методы решения задач комбинаторной оптимизации можно условно разделить на три класса.

- Во-первых, многие известные задачи комбинаторной оптимизации формулируются в форме некоторой задачи оптимизации на графах. Как было отмечено в *главе 7*, один из основных способов решения задачи оптимизации на графах связан с постановкой этих задач в форме математической модели целочисленного или булева программирования. В этом случае выбор способа их решения полностью определяется математическими свойствами соответствующей постановки задачи. В этом контексте возмож-

ность решения практических задач комбинаторной оптимизации с помощью программы MS Excel непосредственно зависит от возможности ее формулировки в виде задачи целочисленного или булева программирования.

- Во-вторых, задачи комбинаторной оптимизации могут быть решены с использованием специальных алгоритмов, которые учитывают специфические особенности тех или иных комбинаторных объектов и конечную мощность множества допустимых альтернатив. В связи с этим для нахождения точного решения задач комбинаторной оптимизации могут использоваться общие алгоритмы типа метода ветвей и границ и метода динамического программирования. Именно такие специальные алгоритмы решения типовых задач комбинаторной оптимизации рассматриваются в настоящей главе.
- В-третьих, применительно к решению практических задач комбинаторной оптимизации становится актуальной разработка специальных алгоритмов нахождения приближенного решения. Алгоритмы, позволяющие находить одно или несколько локально-оптимальных решений, имеют широкое применение при решении задач комбинаторной оптимизации, поскольку нахождение точного решения соответствующих задач за приемлемое время уже для задач небольшой размерности оказывается невозможным даже с привлечением быстродействующих компьютеров. Подобные алгоритмы положены в основу разработки программ на VBA, которые рассматриваются в *главе 12*.

Поскольку программа MS Excel позволяет находить точное решение задач целочисленного и булева программирования, тем самым имеется возможность решения с ее помощью и задач комбинаторной оптимизации. При этом общий порядок подготовки исходных данных и поиск решения полностью аналогичен порядку, рассмотренному ранее для решения других типовых задач оптимизации. Для оценки точности получаемых решений также целесообразно выполнить сравнение полученных различными способами оптимальных решений отдельных практических задач. Именно с этой целью в книге приводится описание различных способов решения задач коммивояжера и задачи о разбиении.

8.2. Задача коммивояжера

Задача коммивояжера, содержательная постановка которой приводится в *разд. 1.2.6*, является классическим примером задачи комбинаторной оптимизации. Хотя ее постановка в виде задачи оптимизации на графах достаточно проста, нахождение ее точного решения является достаточно трудоемким процессом с вычислительной точки зрения. Именно по этой причине методам и алгоритмам ее решения посвящена обширная литература. В этом смысле

задача коммивояжера может служить тестовой задачей для проверки вычислительных алгоритмов решения задач не только комбинаторной оптимизации, но и целочисленного программирования в целом. В настоящей главе рассматривается исходная формулировка задачи коммивояжера в виде задачи оптимизации на графах, используемая для формальной записи условий и решения соответствующей задачи оптимизации в виде модели булева программирования.

8.2.1. Математическая постановка задачи

Исходная постановка задачи коммивояжера формулируется в форме задачи оптимизации на графах. С этой целью рассмотрим связный ориентированный граф: $G = (V, E, h)$, в котором $V = \{v_1, v_2, \ldots, v_n\}$ — конечное множество вершин, $E = \{e_1, e_2, \ldots, e_m\}$ — конечное множество дуг, $h: E \to \mathbf{Z}_+$ — весовая функция дуг. Для математической постановки задачи удобно обозначить отдельные значения весовой функции дуг через: $c_{ij} = h(e_k)$, где дуга $e_k \in E$ соответствует упорядоченной паре вершин (v_i, v_j). Согласно содержательной постановке рассматриваемой задачи отдельные значения: $c_{ij} = h(v_i, v_j)$ интерпретируются как длина участка (i, j) исходного графа.

Длина любого подмножества дуг $E_k \subset E$ в графе G равна сумме весов дуг, входящих в это подмножество. Требуется определить такое подмножество дуг, которое образует в графе G замкнутый путь, проходит через каждую вершину ровно один раз и обладает минимальной длиной.

Для формальной записи условий задачи коммивояжера в виде модели булева программирования следует отметить следующие особенности искомого маршрута в графе:

1. Во-первых, каждая из вершин исходного графа должна иметь в искомом маршруте ровно одну инцидентную ей дугу, которая является входящей для этой вершины, и ровно одну инцидентную ей дугу, которая является выходящей для этой вершины. В противном случае такие вершины окажутся изолированными или тупиковыми и, следовательно, путь не будет проходить через все вершины исходного графа.
2. Во-вторых, общее количество дуг в искомом пути должно быть в точности равно n, где n — общее количество вершин исходного графа. Действительно, если некоторый путь содержит меньше n дуг, то он не будет проходить через все вершины или являться циклическим. Если же искомый путь содержит больше n дуг, то он не будет удовлетворять условию прохождения каждой вершины ровно один раз.
3. В-третьих, что наиболее важно, искомый путь должен представлять собой единственный цикл и не должен распадаться на отдельные циклы с коли-

чеством дуг меньше n, где n — общее количество вершин исходного графа. Это условие имеет комбинаторный характер.

Введем в рассмотрение следующие булевы переменные x_{ij}, которые интерпретируются следующим образом. Переменная $x_{ij} = 1$, если дуга (v_i, v_j) входит в искомый маршрут минимальной длины, т. е. коммивояжер непосредственно переезжает из i-го города в j-й город, и $x_{ij} = 0$, если дуга (v_i, v_j) не входит в оптимальный маршрут, т. е. если коммивояжер непосредственно не переезжает из i-го города в j-й город.

Тогда в общем случае математическая постановка задачи коммивояжера может быть сформулирована следующим образом:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min_{x\in\Delta_\beta}, \tag{8.2.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств и неравенств:

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = 1 \ (\forall i \in \{1, 2,..., n\}); & (8.2.2)\\ \sum_{i=1}^{n} x_{ij} = 1 \ (\forall j \in \{1, 2,..., n\}); & (8.2.3)\\ u_i - u_j + n \cdot x_{ij} \le n - 1 \ (\forall i, j \in \{2, 3,..., n\}, i \ne j); & (8.2.4)\\ x_{ij} \in \{0, 1\}, \ (\forall i, j \in \{1, 2,..., n\}); & (8.2.5)\\ u_i \in R^1, \ (\forall i \in \{2, 3,..., n\}). & (8.2.6) \end{cases}$$

В данной математической модели задачи коммивояжера используются также вспомогательные переменные: $u_i \ (\forall i \in \{2, 3,..., n\})$, которые могут принимать любые действительные значения. При этом ограничения (8.2.2) и (8.2.3) обеспечивают выполнение первых двух указанных ранее условий — искомый путь должен проходить через каждую вершину графа ровно один раз. Ограничения (8.2.4) обеспечивают выполнение третьего из указанных ранее условий — искомый путь не должен распадаться на отдельные циклы. Ограничения (8.2.5) обеспечивают выполнение условия — переменные x_{ij} должны принимать булевы значения, а ограничения (8.2.6) обеспечивают выполнение условия — переменные u_j должны принимать вещественные значения.

Нетрудно показать, что общее количество ограничений (8.2.2)—(8.2.4) равно $2n + (n-1)(n-2) = n^2 - n + 2$.

Следует заметить, что те коэффициенты целевой функции c_{ij}, для которых весовая функция ребер h исходного графа не определена или равна 0, в математической постановке рассматриваемой задачи (8.2.1)—(8.2.6) следует положить равными $+\infty$, т. е. достаточно большому положительному значению.

Примечание

Математическая постановка задачи коммивояжера в форме (8.2.1)—(8.2.6) предложена А. Таккером (1960 г.) и содержит наименьшее число ограничений. Математическая модель данной задачи, предложенная М. Фладом (1956 г.), основанная на использовании 3-индексных переменных, имеет n^3 переменных и $2n^2$ ограничений. Заинтересованные читатели могут познакомиться с этими и другими вариантами постановки задачи коммивояжера в специальной литературе.

Классическая задача коммивояжера формулируется как *несимметричная*, т. е. для случая несимметричного исходного графа, при котором $c_{ij} \neq c_{ji}$. Если же $c_{ij} = c_{ji}, (\forall i, j \in \{1, 2, ..., n\})$, то соответствующая задача коммивояжера называется *симметричной*. Нетрудно увидеть, что исходный граф симметричной задачи коммивояжера является неориентированным.

Если в постановке задачи коммивояжера (8.2.1)—(8.2.6) в выражении целевой функции (8.2.1) операцию отыскания минимума заменить операцией отыскания максимума, то может быть получена математическая постановка соответствующей задачи о нахождении полного циклического пути в графе максимальной длины. Однако последний вариант задачи не получил широкой известности и поэтому в книге не рассматривается.

Примечание

В литературе встречаются и другие варианты задачи коммивояжера. Так, например, если $c_{ij} = d_{ij}$ и d_{ij} удовлетворяют аксиомам метрики, то задача коммивояжера называется *метрической*. Иногда встречается вариант задачи коммивояжера, в котором требуется отыскать k полных циклических маршрутов минимальной длины в ориентированном или неориентированном графе, где k — некоторое натуральное число.

Нетрудно показать, что задача коммивояжера может быть сформулирована и как задача комбинаторной оптимизации. В этом случае, не ограничивая общности рассматриваемой задачи, можно считать, что путь коммивояжера начинается и заканчивается в вершине с номером 1. Тогда каждому полному пути в исходном графе будет соответствовать отдельная перестановка из n элементов (v_{i1}, v_{i2}, …, v_{in}) вершин множества V или, что эквивалентно, перестановка из n элементов (i_1, i_2, …, i_n) числового множества $A = \{1, 2,\ldots, n\}$. Для получения полного замкнутого пути следует каждую перестановку дополнить первым ее элементом: (i_1, i_2, …, i_n, i_1), где $i_1 = 1$.

Тогда математическая постановка задачи коммивояжера в форме задачи комбинаторной оптимизации может быть сформулирована в следующем виде:

$$f(a_{i1}, a_{i2},..., a_{in}, a_{i1}) \to \min_{x \in \Delta_\beta}, \tag{8.2.7}$$

где множество допустимых альтернатив Δ_β содержит все возможные циклические перестановки вида (i_1, i_2,…, i_n, i_1) элементов числового множества {1, 2, …, n}, а для расчета значений целевой функции $f(x)$ может быть использована некоторая матрица значений величин попарной связности этих элементов, аналогичная матрице смежности взвешенного графа.

8.2.2. Решение задачи коммивояжера с помощью программы MS Excel

Для решения задачи коммивояжера с помощью программы MS Excel необходимо задать конкретные значения параметрам исходной задачи. С этой целью рассмотрим задачу коммивояжера для транспортной сети, которая соединяет 6 населенных пунктов в некотором географическом районе. Данный район может быть представлен в виде схемы, формально представляющей собой ориентированный связный граф, состоящий из 6 вершин, которые связаны между собой дугами в прямом и обратном направлении (рис. 8.1). Длина участка автодороги между двумя соседними населенными пунктами, выраженная, например, в *км*, равна значению весовой функции для каждой дуги. Это значение указано рядом с изображением соответствующей дуги в графе.

Требуется найти такой полный замкнутый путь, начинающийся в вершине с номером 1 и заканчивающийся в вершине с номером 6, чтобы общая длина пути была минимальной.

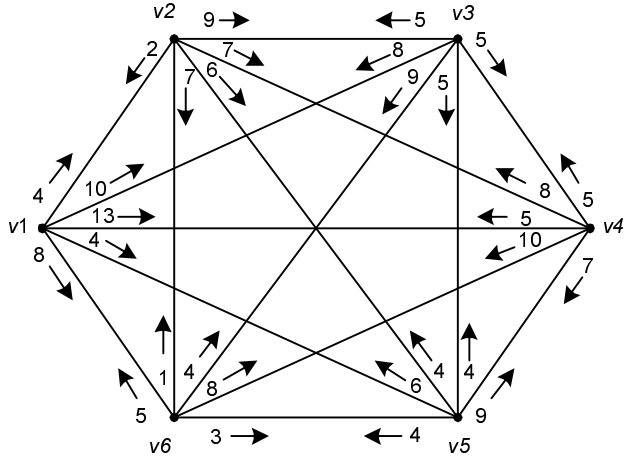


Рис. 8.1. Исходный граф индивидуальной задачи коммивояжера

Переменными математической модели данной индивидуальной задачи коммивояжера являются, во-первых, 36 переменных: x_{ij} ($\forall i, j \in \{1, 2, \ldots, 6\}$), каждая из которых x_{ij} принимает значение 1, если дуга (i, j) входит в минимальный полный путь, и 0 — в противном случае и, во-вторых, 5 вспомогательных переменных: u_i ($\forall i \in \{2, 3, 4, 5, 6\}$), которые могут принимать любые действительные значения. Дополнительно для удобства вычислений следует установить значения весов c_{ii} равными некоторому достаточно большому положительному числу, например, $c_{ii} = 100$ ($\forall i \in \{1, 2, \ldots, 6\}$).

Тогда математическая постановка рассматриваемой индивидуальной задачи коммивояжера может быть записана в следующем виде:

$$\begin{aligned}
&100x_{11} + 4x_{12} + 10x_{13} + 13x_{14} + 4x_{15} + 8x_{16} + \\
&+ 100x_{22} + 2x_{21} + 9x_{23} + 7x_{24} + 6x_{25} + 7x_{26} + \\
&+ 100x_{33} + 8x_{31} + 5x_{32} + 5x_{34} + 5x_{35} + 9x_{36} + \\
&+ 100x_{44} + 5x_{41} + 8x_{42} + 5x_{43} + 7x_{45} + 10x_{46} + \\
&+ 100x_{55} + 6x_{51} + 4x_{52} + 4x_{53} + 9x_{54} + 4x_{56} + \\
&+ 100x_{66} + 5x_{61} + x_{62} + 4x_{63} + 8x_{64} + 3x_{65} \to \min\limits_{x \in \Delta_\beta},
\end{aligned} \tag{8.2.8}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$
\begin{cases}
x_{11}+x_{12}+x_{13}+x_{14}+x_{15}+x_{16}=1;\\
x_{21}+x_{22}+x_{23}+x_{24}+x_{25}+x_{26}=1;\\
x_{31}+x_{32}+x_{33}+x_{34}+x_{35}+x_{36}=1;\\
x_{41}+x_{42}+x_{43}+x_{44}+x_{45}+x_{46}=1;\\
x_{51}+x_{52}+x_{53}+x_{54}+x_{55}+x_{56}=1;\\
x_{61}+x_{62}+x_{63}+x_{64}+x_{65}+x_{66}=1;\\
x_{11}+x_{21}+x_{31}+x_{41}+x_{51}+x_{61}=1;\\
x_{12}+x_{22}+x_{32}+x_{42}+x_{52}+x_{62}=1;\\
x_{13}+x_{23}+x_{33}+x_{43}+x_{53}+x_{63}=1;\\
x_{14}+x_{24}+x_{34}+x_{44}+x_{54}+x_{64}=1;\\
x_{15}+x_{25}+x_{35}+x_{45}+x_{55}+x_{65}=1;\\
x_{16}+x_{26}+x_{36}+x_{46}+x_{56}+x_{66}=1;\\
u_2-u_3+6x_{23}\le 5;\\
\ldots\\
u_6-u_5+6x_{65}\le 5;\\
x_{11},x_{12},x_{13},x_{14},x_{15},x_{16},\ldots,x_{61},x_{62},x_{63},x_{64},x_{65},x_{66}\in\{0,1\};\\
u_1,u_2,u_3,u_4,u_5,u_6\in \boldsymbol{R}^1.
\end{cases}
\tag{8.2.9}
$$

Следует заметить, что в математической постановке индивидуальной задачи коммивояжера (8.2.8) и (8.2.9) первые 6 ограничений соответствуют ограничениям (8.2.2), следующие 6 ограничений соответствуют ограничениям (8.2.3), из 20 ограничений вида (8.2.4) приведены только первое и последнее. При этом предполагается, что переменные: $x_{ii}=0$ ($\forall i\in\{1, 2,\ldots, 6\}$) и поэтому не включаются в постановку задачи (8.2.7) и (8.2.8).

Для решения данной задачи с помощью программы MS Excel создадим новую книгу с именем Комбинаторная оптимизация и изменим имя ее первого рабочего листа на Задача коммивояжера. Для решения поставленной задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A8:A22**, **B1**, **H1**, **B8:H8** (рис. 8.2). Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи.
2. В ячейки **B2:G7** введем веса дуг исходного графа (рис. 8.1), которые представляют собой значения коэффициентов целевой функции (8.2.8).
3. В ячейку **H2** введем формулу: `=СУММПРОИЗВ(B2:G7;B9:G14)`, которая представляет целевую функцию (8.2.8).

4. В ячейку **H9** введем значение левой части первого ограничения (8.2.9): `=СУММ(B9:G9)`, в ячейку **H10** введем значение левой части второго ограничения: `=СУММ(B10:G10)`, в ячейку **H11** введем значение левой части третьего ограничения: `=СУММ(B11:G11)`, в ячейку **H12** введем значение левой части четвертого ограничения: `=СУММ(B12:G12)`, в ячейку **H13** введем значение левой части пятого ограничения: `=СУММ(B13:G13)`, в ячейку **H14** введем значение левой части шестого ограничения: `=СУММ(B14:G14)`.
5. В ячейку **B15** введем значение левой части седьмого ограничения (8.2.9): `=СУММ(B9:B14)`, в ячейку **C15** введем значение левой части восьмого ограничения: `=СУММ(C9:C14)`, в ячейку **D15** введем значение левой части девятого ограничения: `=СУММ(D9:D14)`, в ячейку **E15** введем значение левой части десятого ограничения: `=СУММ(E9:E14)`, в ячейку **F15** введем значение левой части одиннадцатого ограничения: `=СУММ(F9:F14)`, в ячейку **G15** введем значение левой части двенадцатого ограничения: `=СУММ(G9:G14)`.
6. В ячейку **B15** введем значение левой части седьмого ограничения (8.2.9): `=СУММ(B9:B14)`, в ячейку **C15** введем значение левой части восьмого ограничения: `=СУММ(C9:C14)`, в ячейку **D15** введем значение левой части девятого ограничения: `=СУММ(D9:D14)`, в ячейку **E15** введем значение левой части десятого ограничения: `=СУММ(E9:E14)`, в ячейку **F15** введем значение левой части одиннадцатого ограничения: `=СУММ(F9:F14)`, в ячейку **G15** введем значение левой части двенадцатого ограничения: `=СУММ(G9:G14)`.
7. В ячейки **C18:G22** введем формулы, которые соответствуют ограничениям типа (8.2.4), при этом в ячейки **C18**, **D19**, **E20**, **F21**, **G22** для удобства выполнения дальнейших расчетов записывается значение 0.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи коммивояжера имеет следующий вид (рис. 8.2).

	A	B	C	D	E	F	G	H
1				Коэффициенты целевой функции:				Значение ЦФ:
2		100	4	10	13	4	8	=СУММПРОИЗВ(B2:G7;B9:G14)
3		2	100	9	7	6	7	
4		8	5	100	5	5	9	
5		5	8	5	100	7	10	
6		6	4	4	9	100	4	
7		5	1	4	8	3	100	
8	**Переменные: *Xij***	***Xi1***	***Xi2***	***Xi3***	***Xi4***	***Xi5***	***Xi6***	Ограничения 1:
9	***X1j***							=СУММ(B9:G9)
10	***X2j***							=СУММ(B10:G10)
11	***X3j***							=СУММ(B11:G11)
12	***X4j***							=СУММ(B12:G12)
13	***X5j***							=СУММ(B13:G13)
14	***X6j***							=СУММ(B14:G14)
15	Ограничения 2:	=СУММ(B9:B14)	=СУММ(C9:C14)	=СУММ(D9:D14)	=СУММ(E9:E14)	=СУММ(F9:F14)	=СУММ(G9:G14)	
16	**Переменные: *Ui***							
17				Значения ограничений:				
18	***u2-uj+6x2j***		0	=$C$16-D16+6*D10	=$C$16-E16+6*E10	=$C$16-F16+6*F10	=$C$16-G16+6*G10	
19	***u3-uj+6x3j***		=$D$16-C16+6*C11	0	=$D$16-E16+6*E11	=$D$16-F16+6*F11	=$D$16-G16+6*G11	
20	***u4-uj+6x3j***		=$E$16-C16+6*C12	=$E$16-D16+6*D12	0	=$E$16-F16+6*F12	=$E$16-G16+6*G12	
21	***u5-uj+6x5j***		=$F$16-C16+6*C13	=$F$16-D16+6*D13	=$F$16-E16+6*E13	0	=$F$16-G16+6*G13	
22	***u6-uj+6x6j***		=$G$16-C16+6*C14	=$G$16-D16+6*D14	=$G$16-E16+6*E14	=$G$16-F16+6*F14	0	
23								
24								

Рис. 8.2. Исходные данные для решения задачи коммивояжера

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$H$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес диапазонов ячеек `$B$9:$G$14; C16:G16`.
4. Добавить ограничения первой группы для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$H$9:$H$14**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры значение 1;
   - для добавления этого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Добавить ограничения второй группы для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$15:$G$15**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры значение 1;
   - для добавления этого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

6. Добавить ограничения второй группы для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$C$18:$G$22**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения ввести с клавиатуры значение 5;
   - для добавления этого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
7. Добавить ограничение на булевы значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$9:$G$14**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "двоичное";
   - для добавления этого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
8. В окне дополнительных параметров поиска решения выбрать отметки **Линейная модель** и **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения имеет следующий вид (рис. 8.3).

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 8.4).

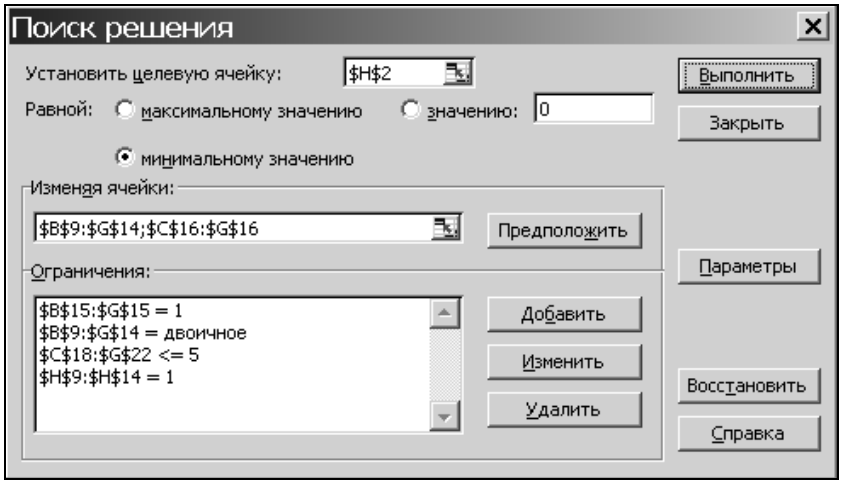


Рис. 8.3. Ограничения значений переменных и параметры мастера поиска решения для задачи коммивояжера

КомбинаторнаяОптимизация

	A	B	C	D	E	F	G	H
1			Коэффициенты целевой функции:					Значение ЦФ:
2		100	4	10	13	4	8	26
3		2	100	9	7	6	7	
4		8	5	100	5	5	9	
5		5	8	5	100	7	10	
6		6	4	4	9	100	4	
7		5	1	4	8	3	100	
8	**Переменные: *Xij***	***Xi1***	***Xi2***	***Xi3***	***Xi4***	***Xi5***	***Xi6***	Ограничения 1:
9	***X1j***	**0**	**0**	**0**	**0**	**1**	**0**	1
10	***X2j***	**1**	**0**	**0**	**0**	**0**	**0**	1
11	***X3j***	**0**	**0**	**0**	**1**	**0**	**0**	1
12	***X4j***	**0**	**0**	**0**	**0**	**0**	**1**	1
13	***X5j***	**0**	**0**	**1**	**0**	**0**	**0**	1
14	***X6j***	**0**	**1**	**0**	**0**	**0**	**0**	1
15	Ограничения 2:	1	1	1	1	1	1	
16	**Переменные: *Ui***		5	1	3	0	4	
17			**Значения ограничений:**					
18	***u2-uj+6x2j***		0	4	2	5	1	
19	***u3-uj+6x3j***		-4	0	4	1	-3	
20	***u4-uj+6x3j***		-2	2	0	3	5	
21	***u5-uj+6x5j***		-5	5	-3	0	-4	
22	***u6-uj+6x6j***		5	3	1	4	0	

Задача коммивояжера

Рис. 8.4. Результат количественного решения задачи коммивояжера

Результатом решения данной индивидуальной задачи коммивояжера являются найденные оптимальные значения переменных: $x_{15} = 1$, $x_{21} = 1$, $x_{34} = 1$, $x_{46} = 1$, $x_{53} = 1$, $x_{62} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствует значение целевой функции: $f_{opt} = 26$.

Анализ найденного решения показывает, что замкнутый маршрут минимальной длины, проходящий через все вершины ориентированного графа (см. рис. 8.1), содержит следующие дуги: (1, 5), (2, 1), (3, 4), (4, 6), (5, 3), (6, 2). Тем самым найден оптимальный полный замкнутый маршрут, начинающийся и заканчивающийся в вершине с номером 1 и включающий в себя последовательное посещение населенных пунктов: из 1 в 5, из 5 в 3, из 3 в 4, из 4 в 6, из 6 в 2, из 2 в 1 (рис. 8.5). При этом общая длина этого пути будет минимальной и равна 26 *км*.

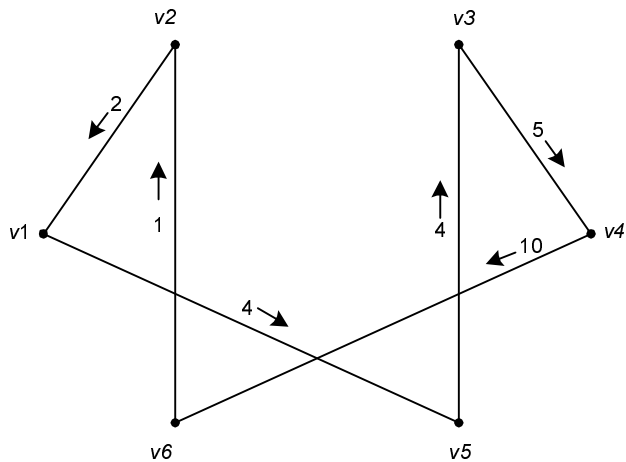


Рис. 8.5. Полный замкнутый путь минимальной длины в исходном графе

Для дополнительной проверки получаемых с помощью программы MS Excel решений данной задачи комбинаторной оптимизации можно воспользоваться специальным алгоритмом динамического программирования, который рассматривается в следующем разделе. В то же время при решении практических задач коммивояжера с помощью программы MS Excel могут возникнуть вычислительные проблемы. Именно по этой причине особенности решения этой задачи оптимизации с помощью специальных программ VBA, реализующих приближенные методы нахождения оптимальных решений, рассматриваются в *главе 12*.

8.2.3. Решение задачи коммивояжера с помощью алгоритма динамического программирования

Общая схема метода динамического программирования для решения задач булева программирования была рассмотрена в *разд. 6.2.3*. В то же время следует отметить, что применительно к решению типовых задач комбинаторного

программирования требуется уточнение данного метода, учитывающее специфику и комбинаторный характер каждой типовой задачи.

Применительно к задаче коммивояжера в теоретико-графовой постановке для формулировки принципа оптимальности Р. Беллмана следует некоторым образом определить многошаговую схему последовательного принятия решения. С этой целью рассмотрим процесс принятия решения о перемещении из некоторой вершины v_i при условии, что до этого уже пройдены вершины: v_{i1}, v_{i2}, …, v_{ik}. Очевидно, на этом этапе может быть принято решение о перемещении только в одну из вершин множества $V\backslash\{v_{i1}, v_{i2}, \ldots, v_{ik}, v_i\}$. Причем попрежнему предполагается, что коммивояжер начинает и заканчивает свой путь в вершине с номером 1.

Для решения задачи коммивояжера с помощью метода динамического программирования необходимо конкретизировать принцип оптимальности Р. Беллмана. С этой целью для задачи коммивояжера в комбинаторной постановке запишем основное функциональное уравнение, отражающее многошаговый характер общей схемы метода динамического программирования.

Основное функциональное уравнение, которое отражает принцип оптимальности применительно к решению задачи коммивояжера (8.2.7), имеет следующий вид:

$$h_k(1; i_1, i_2, \ldots, i_k; \mathrm{i}_{k+1}) = \\ = \min_{a_k \in \{2,3,\ldots,n\}\backslash\{i_1, i_2, \ldots, i_k\}} \{(h_{k-1}(1; i_1, i_2, \ldots, i_k; \mathrm{a}_k) + c_{a_k i_{k+1}})\}. \qquad (8.2.10)$$

где рассматриваются только такие перестановки вида (1; i_1, i_2,…, i_k; i_{k+1}) из элементов числового множества {1, 2,…, n}, в которых первый элемент всегда равен 1, а последний — некоторому числу $i_{k+1} \in \{2, 3,\ldots, n\}\backslash\{i_1, i_2,\ldots, i_k, a_k\}$. Последнее условие отвечает требованию однократного посещения коммивояжером каждого из населенных пунктов заданного региона. Следует заметить, что для $k = n - 1$ значение $i_n = 1$. Значения величин связности элементов c_{ij} соответствуют весам дуг матрицы смежности исходного графа. При этом значение k изменяется от 0 до $n - 1$.

Здесь функция $h_0(1; i)$ по определению равна c_{1i}, а функция $h_{n-1}(1; i_1, i_2, \ldots, i_{n-1}; i_n)$ определяет значение минимального пути, начинающегося в вершине с номером 1 и заканчивающегося в вершине с номером n.

Тогда решение задачи коммивояжера для циклического пути находится с использованием следующей формулы:

$$x^* = \arg\min_{i \in \{2,3,\ldots,n\}\backslash\{i_1, i_2, \ldots, i_{n-1}\}} (h_{n-1}(1; i_1, i_2, \ldots, i_{n-1}; \mathrm{i}) + c_{i1}). \qquad (8.2.11)$$

Метод динамического программирования, ориентированный на решение задачи коммивояжера в комбинаторной постановке, основывается на рекуррентном характере соотношений (8.2.10) и (8.2.11). Алгоритм метода динамического программирования имеет итеративный характер и заключается в выполнении следующих действий:

1. *Предварительное определение значений функций* h_0. До начала выполнения основных итераций алгоритма задаются значения функций $h_0(1; i) = c_{1i}$ ($\forall i \in \{2, 3,\ldots, n\}$). После чего следует перейти к выполнению действий шага 2.
2. *Прямая последовательность расчета.* С использованием рекуррентных соотношений (8.2.10) последовательно рассчитываются значения функций h_k для значений k от 1 до $n-1$, где n — количество вершин исходного графа. Одновременно с этим находятся и запоминаются условно оптимальные значения перестановок $(1; i_1, i_2,\ldots, i_k; i_{k+1})$ для всех значений i от 2 до n. После чего следует перейти к выполнению действий шага 3.
3. *Нахождение оптимального значения целевой функции.* Среди всех найденных на предыдущем шаге значений функций h_{n-1} с использованием выражения (8.2.11) находится минимальное значение. Это значение равно оптимальному значению целевой функции (8.2.7), которому соответствует оптимальное решение — полная циклическая перестановка: $x^* = (1; i_1, i_2,\ldots, i_k; i_{k+1})$. На этом выполнение алгоритма заканчивается.

Рассмотренный алгоритм метода динамического программирования может быть изображен графически в форме следующей диаграммы деятельности языка UML (рис. 8.6).

Нетрудно заметить, что в силу конечности общего количества вершин исходного графа рассмотренный алгоритм метода динамического программирования является конечным. Тем не менее, комбинаторный характер данной задачи ограничивает практические возможности нахождения точного решения задач этого класса с помощью метода динамического программирования.

Проиллюстрируем использование рассмотренного алгоритма метода динамического программирования для решения индивидуальной задачи коммивояжера, исходный граф которой изображен на рис. 8.1. В соответствии с описанной ранее схемой алгоритма следует выполнить следующие практические действия.

Шаг 1. До начала выполнения основных итераций алгоритма задаются значения функций: $h_0(1; 2) = 4$, $h_0(1; 3) = 10$, $h_0(1; 4) = 13$, $h_0(1; 5) = 4$, $h_0(1; 6) = 8$. После этого следует перейти к выполнению действий первой итерации шага 2.

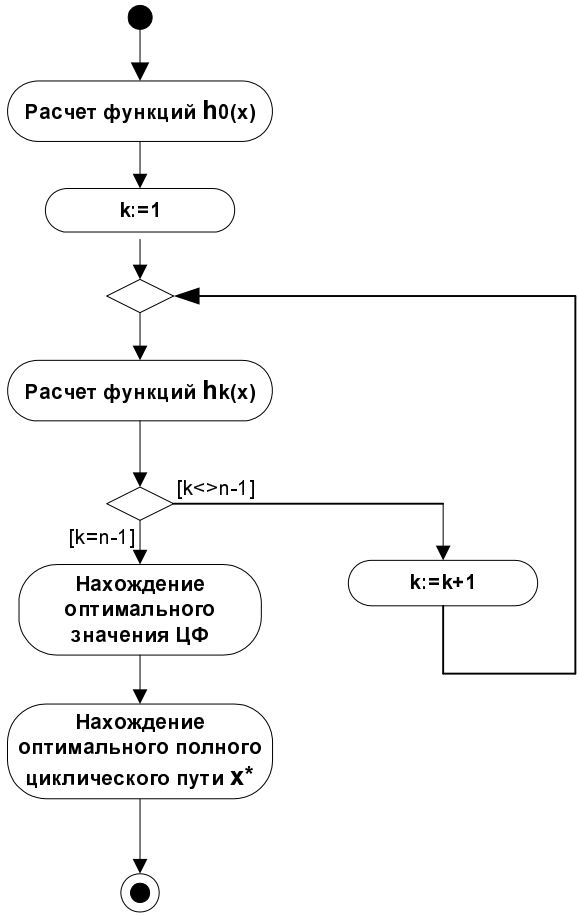


Рис. 8.6. Диаграмма деятельности алгоритма метода динамического программирования для решения задачи коммивояжера

Шаг 2 (первая итерация). С использованием рекуррентных соотношений (8.2.10) последовательно рассчитываются значения функций h_1 для значений $k = 1$. Соответствующие значения функций равны: $h_1(1; 3; 2) = 15$, $h_1(1; 4; 2) = 21$, $h_1(1; 5; 2) = 8$, $h_1(1; 6; 2) = 9$, $h_1(1; 2; 3) = 13$, $h_1(1; 4; 3) = 18$, $h_1(1; 5; 3) = 8$, $h_1(1; 6; 3) = 12$, $h_1(1; 2; 4) = 11$, $h_1(1; 3; 4) = 15$, $h_1(1; 5; 4) = 13$, $h_1(1; 6; 4) = 16$, $h_1(1; 2; 5) = 10$, $h_1(1; 3; 5) = 15$, $h_1(1; 4; 5) = 20$, $h_1(1; 6; 5) = 11$, $h_1(1; 2; 6) = 11$, $h_1(1; 3; 6) = 19$, $h_1(1; 4; 6) = 23$, $h_1(1; 5; 6) = 8$. После этого следует перейти к выполнению действий второй итерации шага 2.

Шаг 2 (вторая итерация). С использованием рекуррентных соотношений (8.2.10) последовательно рассчитываются значения функций h_2 для значений

$k = 2$. Соответствующие значения функций равны: $h_2(1; 3, 4; 2) = 23$, $h_2(1; 5, 3; 2) = 13$, $h_2(1; 6, 3; 2) = 17$, $h_2(1; 5, 4; 2) = 21$, $h_2(1; 4, 6; 2) = 24$, $h_2(1; 5, 6; 2) = 9$, $h_2(1; 2, 4; 3) = 16$, $h_2(1; 2, 5; 3) = 14$, $h_2(1; 2, 6; 3) = 15$, $h_2(1; 5, 4; 3) = 18$, $h_2(1; 6, 4; 3) = 21$, $h_2(1; 5, 6; 3) = 12$, $h_2(1; 2, 3; 4) = 18$, $h_2(1; 5, 2; 4) = 15$, $h_2(1; 6, 2; 4) = 16$, $h_2(1; 5, 3; 4) = 13$, $h_2(1; 6, 3; 4) = 17$, $h_2(1; 5, 6; 4) = 16$, $h_2(1; 2, 3; 5) = 18$, $h_2(1; 2, 4; 5) = 18$, $h_2(1; 2, 6; 5) = 14$, $h_2(1; 3, 4; 5) = 22$, $h_2(1; 6, 3; 5) = 17$, $h_2(1; 6, 4; 5) = 23$, $h_2(1; 2, 3; 6) = 22$, $h_2(1; 2, 4; 6) = 21$, $h_2(1; 2, 5; 6) = 14$, $h_2(1; 3, 4; 6) = 25$, $h_2(1; 5, 3; 6) = 17$, $h_2(1; 5, 4; 6) = 23$. После этого следует перейти к выполнению действий третьей итерации шага 2.

Шаг 2 (третья итерация). С использованием рекуррентных соотношений (8.2.10) последовательно рассчитываются значения функций h_3 для значений $k = 3$. Соответствующие значения функций равны: $h_3(1; 5, 3, 4; 2) = 21$, $h_3(1; 6, 3, 4; 2) = 25$, $h_3(1; 5, 6, 3; 2) = 17$, $h_3(1; 4, 5, 6; 2) = 24$, $h_3(1; 5, 2, 4; 3) = 20$, $h_3(1; 6, 2, 4; 3) = 21$, $h_3(1; 2, 5, 6; 3) = 18$, $h_3(1; 5, 6, 4; 3) = 21$, $h_3(1; 2, 5, 3; 4) = 19$, $h_3(1; 2, 6, 3; 4) = 20$, $h_3(1; 5, 6, 2; 4) = 16$, $h_3(1; 5, 6, 3; 4) = 17$, $h_3(1; 2, 4, 3; 5) = 21$, $h_3(1; 2, 6, 3; 5) = 20$, $h_3(1; 6, 2, 4; 5) = 23$, $h_3(1; 6, 3, 4; 5) = 24$, $h_3(1; 2, 4, 3; 6) = 25$, $h_3(1; 3, 5, 2; 6) = 20$, $h_3(1; 2, 4, 5; 6) = 22$, $h_3(1; 5, 3, 4; 6) = 23$. После этого следует перейти к выполнению действий четвертой итерации шага 2.

Шаг 2 (четвертая итерация). С использованием рекуррентных соотношений (8.2.10) последовательно рассчитываются значения функций h_4 для значений $k = 4$. Соответствующие значения функций равны: $h_4(1; 5, 3, 4, 6; 2) = 24$, $h_4(1; 5, 6, 2, 4; 3) = 21$, $h_4(1; 2, 5, 6, 3; 4) = 23$, $h_4(1; 6, 2, 4, 3; 5) = 26$, $h_4(1; 2, 4, 3, 5; 6) = 25$. После этого следует перейти к выполнению действий пятой итерации шага 2.

Шаг 2 (пятая итерация). С использованием рекуррентных соотношений (8.2.10) последовательно рассчитываются значения функций h_5 для значений $k = 5$. Соответствующие значения функций равны: $h_5(1; 5, 3, 4, 6, 2; 1) = 26$, $h_5(1; 5, 6, 2, 4, 3; 1) = 29$, $h_5(1; 2, 5, 6, 3, 4; 1) = 28$, $h_5(1; 6, 2, 4, 3, 5; 1) = 32$, $h_5(1; 2, 4, 3, 5, 6; 1) = 30$. После этого следует перейти к выполнению действий пятой итерации шага 3.

Шаг 3 (пятая итерация). Среди всех найденных на предыдущем шаге значений функций h_5 с использованием выражения (8.2.11) находится минимальное значение. Это значение равно: $h_5(1; 5, 3, 4, 6, 2; 1) = 26$, которому соответствует оптимальное решение — полная циклическая перестановка: $x^* = (1, 5, 3, 4, 6, 2, 1)$. На этом выполнение алгоритма заканчивается.

Таким образом, результатом решения комбинаторной задачи коммивояжера методом динамического программирования является найденное оптимальное значение перестановки: $x^* = (1, 5, 3, 4, 6, 2, 1)$, которой соответствует значение целевой функции: $f_{opt} = 26$ (*км*). Анализ результатов решения задачи коммивояжера с помощью программы MS Excel и методом динамического программирования показывает их полное совпадение, что служит веским аргументом в пользу их достоверности.

Далее в *главе 12* рассматриваются алгоритм и программа приближенного решения задачи коммивояжера, которая может быть использована для практического нахождения решений задач данного класса с достаточно большим числом городов ($n > 40$). В последнем случае пользователь также избавляется от необходимости выполнения рутинных операций по записи системы ограничений в программе MS Excel.

8.3. Задача о разбиении

Многие прикладные задачи оптимизации могут быть сформулированы в форме задачи о разбиении некоторого конечного множества объектов на фиксированное число подмножеств или классов разбиения, так чтобы обеспечивалось оптимальное значение некоторого критерия. Одна из наиболее известных задач этого типа связана с проектированием радиоэлектронной аппаратуры.

8.3.1. Содержательная постановка задачи

Сущность задачи проектирования радиоэлектронной аппаратуры заключается в следующем. Научно-производственное предприятие проектирует некоторое радиоэлектронное устройство, например, локатор или ТВ-передатчик, которое согласно проектной документации должно состоять из отдельных аппаратурных модулей. Для изготовления этих модулей используются радиоэлектронные компоненты. Известен общий состав этих компонентов, необходимых для изготовления одного требуемого устройства, и электрическая схема их соединения. При этом каждый радиоэлектронный компонент обладает некоторой мощностью потребления энергии, а отдельные компоненты должны быть соединены между собой электрическими шинами, количество проводников в которых определяется электрической схемой их соединения. Задана предельная величина мощности всех компонентов в модуле. Причем эту предельную мощность при размещении компонентов в одном модуле по техническим условиям нельзя превышать.

Задача состоит в том, чтобы все имеющиеся компоненты разбить на отдельные модули, так чтобы общая мощность компонентов в каждом модуле

не превышала заданной предельной величины, и при этом размещаемые в каждом из модулей компоненты имели максимальное по количеству проводников электрических шин для связи между собой.

8.3.2. Математическая постановка задачи

Исходная постановка задачи о разбиении может быть сформулирована в форме следующей задачи оптимизации на графах. С этой целью рассмотрим связный неориентированный граф: $G=(V, E, d, h)$, в котором: $V=\{v_1, v_2, ..., v_n\}$ — конечное множество вершин, $E=\{e_1, e_2, ..., e_m\}$ — конечное множество ребер, $d: V \to \boldsymbol{Z}_+$ — весовая функция вершин, $h: E \to \boldsymbol{Z}_+$ — весовая функция ребер. Дополнительно задано некоторое положительное число w, интерпретируемое как максимально допустимый вес вершин в каждом из подмножеств разбиения, а также натуральное r — максимальное количество подмножеств разбиения.

Требуется определить такое разбиение вершин графа на подмножества, при котором обеспечивается максимум суммы весов ребер для всех вершин, входящих в каждое из подмножеств разбиения, и удовлетворяются следующие два условия:

1. Во-первых, каждая из вершин исходного графа может принадлежать только одному подмножеству разбиения. В противном случае решение не будет удовлетворять определению разбиения.
2. Во-вторых, общая сумма весов вершин в каждом из подмножеств искомого разбиения не должна превышать заданного максимально допустимого веса вершин подмножества w.

Для формулировки задачи о разбиении в виде модели булева программирования введем в рассмотрение следующие булевы переменные: $x_{ik} = 1$, если вершина v_i входит в k-е подмножество разбиения, и $x_{ik} = 0$ в противном случае, т. е. если вершина v_i не входит в k-е подмножество разбиения.

Тогда в общем случае математическая постановка задачи о разбиении может быть сформулирована следующим образом:

$$\frac{1}{2}\sum_{k=1}^{r}\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ik} x_{jk} \to \max_{x \in \Delta_\beta}, \tag{8.3.1}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств и неравенств:

$$\begin{cases} \sum_{k=1}^{r} x_{ik} = 1, (\forall i \in \{1, 2,..., n\}); & (8.3.2) \\ \sum_{i=1}^{n} d_i x_{ik} \le w, (\forall k \in \{1, 2,..., r\}); & (8.3.3) \\ x_{ik} \in \{0, 1\}, (\forall i \in \{1, 2,..., n\}, \forall k \in \{1, 2,..., r\}). & (8.3.4) \end{cases}$$

Здесь $c_{ij} = h(v_i, v_j)$ — вес ребра $\{v_i, v_j\}$, которому соответствует $e_l \in E$; $d(v_i)$ — вес вершины $v_i \in V$; r — общее количество подмножеств разбиения.

В данной математической модели задачи разбиения графа ограничения (8.3.1) обеспечивают выполнение первого из двух указанных ранее условий. Ограничения (8.3.3) обеспечивают выполнение второго из указанных ранее условий. Ограничения (8.2.4) обеспечивают выполнение общего условия — переменные x_{ik} должны принимать булевы значения. Нетрудно показать, что общее количество ограничений (8.3.2) и (8.2.3) равно: $n + r$.

Задача о разбиении может быть также сформулирована как задача комбинаторной оптимизации. Каждому допустимому решению задачи о разбиении соответствует некоторое разбиение $R = \{A_1, A_2,\ldots, A_r\}$, где каждый класс разбиения A_i $(\forall i \in \{1, 2,\ldots, r\})$ состоит из вершин исходного графа $G = (V, E, d, h)$ или, что эквивалентно, из элементов числового множества $A = \{1, 2,\ldots, n\}$.

Тогда математическая постановка задачи о разбиении в форме задачи комбинаторной оптимизации может быть сформулирована в следующем виде:

$$f(R) \to \max_{R \in \Delta_\beta}, \qquad (8.3.5)$$

где множество допустимых альтернатив Δ_β содержит все возможные разбиения множества вершин исходного графа или числового множества $A = \{1, 2,\ldots, n\}$ на r классов, которые удовлетворяют следующим ограничениям:

$$d(v_{i1}) + d(v_{i2}) + \ldots + d(v_{iq}) \le w\ (\forall A_i \in R), \qquad (8.3.6)$$

где каждый класс A_i $(\forall i \in \{1, 2,\ldots, r\})$, входящий в допустимое разбиение R, состоит из вершин $A_i = \{v_{i1}, v_{i2},\ldots, v_{iq}\}$ или из элементов $A_i = \{i_1, i_2,\ldots, i_q\}$. При этом для каждого разбиения предполагаются выполненными общие условия непересекаемости классов и принадлежности каждого элемента исходного множества одному из классов допустимого разбиения *(см. приложение 1)*. Значения целевой функции $f(R)$ определяются как общая сумма величин связности каждой пары элементов, попадающих в один класс допустимого разбиения, аналогично выражению (8.3.1).

8.3.3. Решение задачи о разбиении с помощью программы MS Excel

Для решения задачи о разбиении с помощью программы MS Excel необходимо задать конкретные значения параметрам исходной задачи. С этой целью рассмотрим задачу о разбиении 8 радиоэлектронных компонентов на модули. Электрическая схема соединения этих компонентов формально представлена в виде графа (рис. 8.7). В этом случае каждому из 8 компонентов поставлена в соответствие отдельная вершина графа, а электрическому соединению компонентов в форме шины — соответствующее ребро графа. Количество электрических проводников в шине указано рядом с соответствующим ребром графа, а потребляемая мощность компонента, измеряемая, например, в *КВт*, указана рядом с обозначением вершины в скобках. При этом предельная величина мощности компонентов, размещаемых на отдельном модуле, не должна превышать 12 *КВт*.

Требуется найти такое разбиение всех 8 компонентов на модули, так чтобы общая мощность компонентов в каждом модуле не превышала заданной предельной величины 12 *КВт*, и при этом размещаемые в каждом из модулей компоненты имели по максимальному количеству проводников электрических шин для связи между собой.

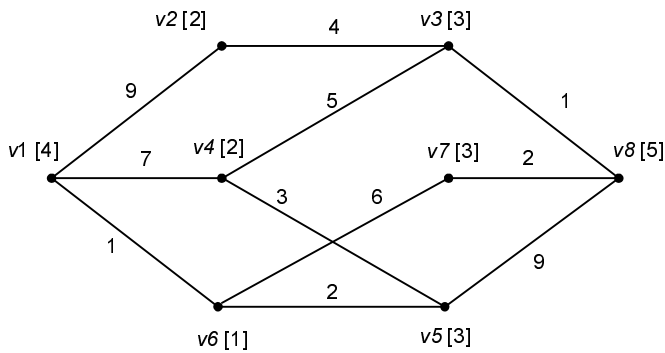


Рис. 8.7. Исходный граф индивидуальной задачи о разбиении

Поскольку в рассматриваемой индивидуальной задаче о разбиении не указано общее число подмножеств разбиения r, то в общем случае при решении данной задачи с помощью программы MS Excel удобно считать $r = n = 8$. Тогда в качестве переменных математической модели данной индивидуальной задачи о разбиении следует рассмотреть 64 переменных: x_{ik} $(\forall i, k \in \{1, 2, \ldots, 8\})$.

В этом случае математическая постановка рассматриваемой индивидуальной задачи о разбиении может быть записана в следующем виде:

$$\begin{aligned}&9x_{11}x_{21}+7x_{11}x_{41}+x_{11}x_{61}+4x_{21}x_{31}+5x_{31}x_{41}+x_{31}x_{81}+\\&+3x_{41}x_{51}+2x_{51}x_{61}+9x_{51}x_{81}+6x_{61}x_{71}+2x_{71}x_{81}+\ldots\\&\ldots+9x_{18}x_{28}+7x_{18}x_{48}+x_{18}x_{68}+4x_{28}x_{38}+5x_{38}x_{48}+x_{38}x_{88}+\\&+3x_{41}x_{51}+2x_{51}x_{61}+9x_{51}x_{81}+6x_{61}x_{71}+2x_{71}x_{81}\to\max\limits_{x\in\Delta_\beta},\end{aligned}\tag{8.3.7}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа равенств и неравенств:

$$\begin{cases}x_{11}+x_{12}+x_{13}+x_{14}+x_{15}+x_{16}+x_{17}+x_{18}=1;\\x_{21}+x_{22}+x_{23}+x_{24}+x_{25}+x_{26}+x_{27}+x_{28}=1;\\x_{31}+x_{32}+x_{33}+x_{34}+x_{35}+x_{36}+x_{37}+x_{38}=1;\\x_{41}+x_{42}+x_{43}+x_{44}+x_{45}+x_{46}+x_{47}+x_{48}=1;\\x_{51}+x_{52}+x_{53}+x_{54}+x_{55}+x_{56}+x_{57}+x_{58}=1;\\x_{61}+x_{62}+x_{63}+x_{64}+x_{65}+x_{66}+x_{67}+x_{68}=1;\\x_{71}+x_{72}+x_{73}+x_{74}+x_{75}+x_{76}+x_{77}+x_{78}=1;\\x_{81}+x_{82}+x_{83}+x_{84}+x_{85}+x_{86}+x_{87}+x_{88}=1;\\4x_{11}+2x_{21}+3x_{31}+2x_{41}+3x_{51}+x_{61}+3x_{71}+5x_{81}\le 12;\\4x_{12}+2x_{22}+3x_{32}+2x_{42}+3x_{52}+x_{62}+3x_{72}+5x_{82}\le 12;\\4x_{13}+2x_{23}+3x_{33}+2x_{43}+3x_{53}+x_{63}+3x_{73}+5x_{83}\le 12;\\4x_{14}+2x_{24}+3x_{34}+2x_{44}+3x_{54}+x_{64}+3x_{74}+5x_{84}\le 12;\\4x_{15}+2x_{25}+3x_{35}+2x_{45}+3x_{55}+x_{65}+3x_{75}+5x_{85}\le 12;\\4x_{16}+2x_{26}+3x_{36}+2x_{46}+3x_{56}+x_{66}+3x_{76}+5x_{86}\le 12;\\4x_{17}+2x_{27}+3x_{37}+2x_{47}+3x_{57}+x_{67}+3x_{77}+5x_{87}\le 12;\\4x_{18}+2x_{28}+3x_{38}+2x_{48}+3x_{58}+x_{68}+3x_{78}+5x_{88}\le 12;\\x_{11},x_{12},x_{13},x_{14},x_{15},x_{16},x_{17},x_{18}\ldots,x_{81},x_{82},x_{83},x_{84},x_{85},x_{86},x_{87},x_{88}\in\{0,1\}.\end{cases}\tag{8.3.8}$$

Следует заметить, что в математической постановке индивидуальной задачи о разбиении (8.3.7) и (8.3.8) в выражении для целевой функции (8.3.7) записаны только те произведения переменных, которые соответствуют ребрам исходного графа, имеющим ненулевой вес. Первые 8 ограничений (8.3.8) соответствуют ограничениям (8.3.2), следующие 8 ограничений (8.3.8) соответствуют ограничениям (8.3.3).

Для решения данной задачи с помощью программы MS Excel создадим в книге с именем Комбинаторная оптимизация новый рабочий лист с именем Оптимальное разбиение. Для решения поставленной задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:E1**, **I1**, **N1**, **P1** (рис. 8.8). Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи.
2. В ячейки **A2:A12** и **B2:B12** введем номера вершин, которые соединены ребрами в исходном графе.
3. В ячейки **C2:C12** введем веса ребер исходного графа (рис. 8.7), которые представляют собой значения коэффициентов целевой функции (8.3.7).
4. В ячейки **E2:E12** введем номера всех вершин исходного графа.
5. В ячейки **F2:F12** введем веса вершин исходного графа (рис. 8.7), которые представляют собой значения коэффициентов второй группы ограничений (8.3.8).
6. В ячейку **O2** введем формулу: `=СУММ(G2:N2)`, которая представляет левую часть первого ограничения (8.3.8).
7. Аналогично введем формулы для остальных ограничений первой группы. А именно в ячейку **O3** введем значение левой части второго ограничения: `=СУММ(G3:N3)`; в ячейку **O4** введем значение левой части третьего ограничения: `=СУММ(G4:N4)`; в ячейку **O5** введем значение левой части четвертого ограничения: `=СУММ(G5:N5)`; в ячейку **O6** введем значение левой части пятого ограничения: `=СУММ(G6:N6)`; в ячейку **O7** введем значение левой части шестого ограничения: `=СУММ(G7:N7)`; в ячейку **O8** введем значение левой части шестого ограничения: `=СУММ(G8:N8)`; в ячейку **O9** введем значение левой части шестого ограничения: `=СУММ(G9:N9)`.
8. В ячейку **G10** введем значение левой части первого ограничения второй группы (8.3.8): `=СУММПРОИЗВ($F$2:$F$9;G2:G9)`.
9. Аналогично введем формулы для остальных ограничений второй группы. А именно, в ячейку **H10** введем значение левой части второго ограничения: `=СУММПРОИЗВ($F$2:$F$9;H2:H9)`; в ячейку **I10** введем значение левой части третьего ограничения: `=СУММПРОИЗВ($F$2:$F$9;I2:I9)`; в ячейку **J10** введем значение левой части четвертого ограничения: `=СУММПРОИЗВ($F$2:$F$9; J2:J9)`; в ячейку **K10** введем значение левой части пятого ограничения: `=СУММПРОИЗВ($F$2:$F$9;K2:K9)`; в ячейку **L10** введем значение левой части шестого ограничения: `=СУММПРОИЗВ($F$2:$F$9;L2:L9)`; в ячейку **M10** введем значение левой

части седьмого ограничения: `=СУММПРОИЗВ($F$2:$F$9;M2:M9)`; в ячейку **N10** введем значение левой части восьмого ограничения: `=СУММПРОИЗВ($F$2:$F$9;N2:N9)`.

10. В ячейку **D2** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$2:$N$2;G3:N3)`, которой соответствует попадание вершин с номерами 1 и 2 в один класс искомого разбиения.
11. В ячейку **D3** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$2:$N$2;G5:N5)`, которой соответствует попадание вершин с номерами 1 и 4 в один класс искомого разбиения.
12. В ячейку **D4** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$2:$N$2;G7:N7)`, которой соответствует попадание вершин с номерами 1 и 6 в один класс искомого разбиения.
13. В ячейку **D5** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$3:$N$3;G4:N4)`, которой соответствует попадание вершин с номерами 2 и 3 в один класс искомого разбиения.
14. В ячейку **D6** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$4:$N$4;G5:N5)`, которой соответствует попадание вершин с номерами 3 и 4 в один класс искомого разбиения.
15. В ячейку **D7** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$4:$N$4;G9:N9)`, которой соответствует попадание вершин с номерами 3 и 8 в один класс искомого разбиения.
16. В ячейку **D8** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$5:$N$5;G6:N6)`, которой соответствует попадание вершин с номерами 4 и 5 в один класс искомого разбиения.
17. В ячейку **D9** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$6:$N$6;G7:N7)`, которой соответствует попадание вершин с номерами 5 и 6 в один класс искомого разбиения.
18. В ячейку **D10** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$6:$N$6;G9:N9)`, которой соответствует попадание вершин с номерами 5 и 8 в один класс искомого разбиения.
19. В ячейку **D11** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$7:$N$7;G8:N8)`, которой соответствует попадание вершин с номерами 6 и 7 в один класс искомого разбиения.
20. В ячейку **D12** введем значение частичной суммы произведений переменных: `=СУММПРОИЗВ($G$8:$N$8;G9:N9)`, которой соответствует попадание вершин с номерами 7 и 8 в один класс искомого разбиения.

21. В ячейку **P2** введем формулу: `=СУММПРОИЗВ(C2:C12;D2:D12)`, которая представляет целевую функцию (8.3.7).
22. В ячейку **F10** введем значение 12, соответствующее правой части второй группы ограничений (8.3.8).

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о разбиении имеет следующий вид (рис. 8.8).

КомбинаторнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	*vi*	*vj*	*cij*	*Xik*Xjk*	*№ и Вес вершины:*				Переменные:						Ограничения:	Значение ЦФ:
2	1	2	9	0	1	**4**									0	0
3	1	4	7	0	2	**2**									0	
4	1	6	1	0	3	**3**									0	
5	2	3	4	0	4	**2**									0	
6	3	4	5	0	5	**3**									0	
7	3	8	1	0	6	**1**									0	
8	4	5	3	0	7	**3**									0	
9	5	6	2	0	8	**5**									0	
10	5	8	9	0		12	0	0	0	0	0	0	0	0		
11	6	7	6	0												
12	7	8	2	0												
13																
14																

Оптимальное разбиение

Рис. 8.8. Исходные данные для решения задачи о разбиении

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$P$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **максимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$G$2:$N$9`.
4. Задать ограничения первой группы для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$O$2:$O$9**, который должен отобразиться в поле с именем **Ссылка на ячейку**;

- в качестве знака ограничения из выпадающего списка выбрать равенство "=";
- в качестве значения правой части ограничения ввести с клавиатуры значение 1;
- для добавления этого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

5. Задать ограничения второй группы для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$G$10:$N$10**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения выбрать ячейку **$F$10**;
   - для добавления этого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
6. Задать ограничение на булевы значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$G$2:$N$9**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "двоичное";
   - для добавления этого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
7. В окне дополнительных параметров поиска решения следует обязательно снять отметку **Линейная модель** и выбрать отметку **Неотрицательные значения**.

Общий вид диалогового окна спецификации параметров мастера поиска решения для данной задачи оптимизации представлен на рис. 8.9.

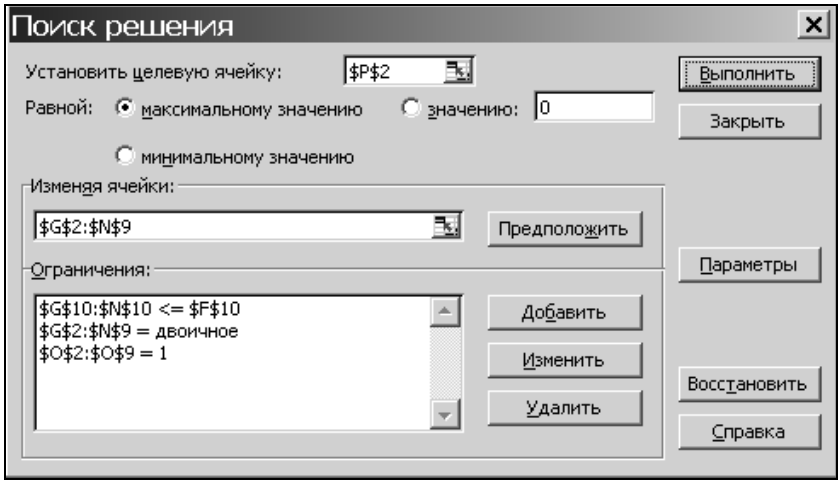


Рис. 8.9. Ограничения значений переменных и параметры мастера поиска решения для задачи о разбиении

После задания ограничений и целевой функции можно приступить к поиску численного решения, для чего следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 8.10).

КомбинаторнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	*vi*	*vj*	*cij*	*Xik*Xjk*	*№ и Вес вершины:*				Переменные:					Ограничения:		Значение ЦФ:
2	1	2	9	1	1	4	0	0	0	0	1	0	0	0	1	44
3	1	4	7	1	2	2	0	0	0	0	1	0	0	0	1	
4	1	6	1	0	3	3	0	0	0	0	1	0	0	0	1	
5	2	3	4	1	4	2	0	0	0	0	1	0	0	0	1	
6	3	4	5	1	5	3	0	0	0	0	0	0	0	1	1	
7	3	8	1	0	6	1	0	0	0	0	0	0	0	1	1	
8	4	5	3	0	7	3	0	0	0	0	0	0	0	1	1	
9	5	6	2	1	8	5	0	0	0	0	0	0	0	1	1	
10	5	8	9	1		12	0	0	0	0	11	0	0	12		
11	6	7	6	1												
12	7	8	2	1												
13																
14																

Оптимальное разбиение

Рис. 8.10. Результат количественного решения задачи о разбиении

Результатом решения рассматриваемой индивидуальной задачи о разбиении являются найденные оптимальные значения переменных: $x_{15} = 1$, $x_{25} = 1$, $x_{35} = 1$, $x_{45} = 1$, $x_{58} = 1$, $x_{68} = 1$, $x_{78} = 1$, $x_{88} = 1$, остальные переменные равны 0.

Найденному оптимальному решению соответствует значение целевой функции: $f_{opt} = 44$.

Анализ найденного решения показывает, что оптимальное разбиение всех вершин исходного графа (см. рис. 8.7) содержит два класса, первый из которых содержит вершины с номерами: 1, 2, 3 и 4, а второй — вершины с номерами: 5, 6, 7 и 8 (рис. 8.11). Этому разбиению вершин графа соответствует разбиение исходных 8 радиоэлектронных компонентов на 2 модуля, в первом из которых должны быть размещены компоненты с номерами: 1, 2, 3 и 4, а во втором — компоненты с номерами: 5, 6, 7 и 8. При этом общая мощность компонентов в каждом модуле не будет превышать заданной предельной величины 12 *КВт*, а размещаемые в каждом из модулей компоненты имеют по максимальному количеству проводников электрических шин для связи между собой. При этом общее количество этих проводников будет максимальным и равным 44.

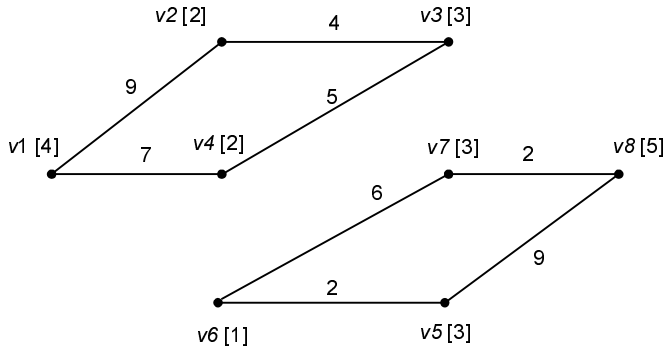


Рис. 8.11. Оптимальное разбиение вершин в исходном взвешенном графе

Для дополнительной проверки получаемых с помощью программы MS Excel решений данной задачи комбинаторной оптимизации можно воспользоваться специальными методами нахождения точного решения, например, методом ветвей и границ или методом динамического программирования, который рассматривается в следующем разделе. В то же время при решении практических задач о разбиении с помощью программы MS Excel могут возникнуть вычислительные проблемы. Именно по этой причине особенности решения этой задачи оптимизации с помощью специальных программ VBA, реализующих приближенные методы нахождения оптимальных решений, рассматриваются в *главе 12*.

8.3.4. Решение задачи о разбиении с помощью алгоритма динамического программирования

Применительно к задаче о разбиении в теоретико-графовой постановке для формулировки принципа оптимальности Р. Беллмана, согласно традиционной методике, следует определить некоторую многошаговую схему последовательного принятия решения. Метод динамического программирования, который используется для решения задачи о разбиении в комбинаторной постановке (8.3.5) и (8.3.6), основан на последовательном применении некоторой процедуры порождения допустимых разбиений множества вершин исходного графа. В этом случае следует определить интерпретации понятий состояния и управления, которые используются при описании общей схемы данного метода.

Под *состоянием* на некотором шаге $k \in \{1, 2, \ldots, n\}$ порождения допустимых разбиений понимается допустимое разбиение подмножества вершин $A_k \subseteq V$ исходного графа, где $A_k = \{v_1, v_2, \ldots, v_k\}$. Каждому из состояний соответствует отдельное значение целевой функции (8.3.5). Тогда под *управлением* при переходе из допустимого, с точки зрения ограничений вида (8.3.6), состояния на шаге k к допустимому состоянию на шаге $k+1 \le n$ следует понимать включение вершины $v_{k+1} \in V$ в одно из классов разбиения, соответствующего состоянию на шаге k. Общее количество шагов процедуры порождения допустимых разбиений равно $n = \boldsymbol{card}(A)$. Допустимое разбиение множества вершин исходного графа на последнем шаге процедуры, которому соответствует максимальное значение критериальной функции по сравнению с другими допустимыми разбиениями на последнем шаге, является глобально оптимальным, а найденный экстремум является точным.

Использование рассмотренной процедуры порождения допустимых разбиений основано на предварительном определении целесообразности включения вершины $v_{k+1} \in V$ в каждое из подмножеств всех допустимых разбиений на шаге $k \in \{1, 2, \ldots, n-1\}$. В этом случае решение о возможности включения задачи вершины $v_{k+1} \in V$ в некоторый класс $A_j^k \subseteq A_k \subseteq V$ принимается в том случае, если выполняется хотя бы одно из следующих условий:

- среди вершин подмножества A_j^k существует хотя бы одна $v_j \in A_j^k$, для которой $h(v_{k+1}, v_j) \neq 0$;
- среди вершин подмножества A_i^k существует хотя бы одна $v_j \in A_i^k$, для которой найдется последовательность вершин $\{v_{l1}, v_{l2}, \ldots, v_{lk}\}$, где все

$l_1, l_2, ..., l_k > k$, так что выполнены условия: $h(v_j, v_{l1}) \neq 0$, $h(v_{l1}, v_{l2}) \neq 0$;...; $h(v_{lk}, v_{jlk+1}) \neq 0$.

Здесь под $h(v_i, v_j)$ понимается значение весовой функции ребра, соединяющего вершины v_i и v_j исходного графа.

Для получения информации о целесообразности включения отдельных вершин исходного графа в классы разбиений предварительно используется специальная вычислительная процедура, которая обеспечивает нахождение последовательностей задач, удовлетворяющих второму условию для заданных значений: $k \in \{n, n-1, \ldots, 2, 1\}$. Метод динамического программирования в этом случае соответствует последовательному применению процедуры определения целесообразности включения вершин исходного графа в классы разбиений и процедуры порождения допустимых разбиений множества вершин исходного графа.

Алгоритм метода динамического программирования для нахождения решения задачи о разбиении в комбинаторной постановке представляет собой итеративный процесс выполнения следующих действий:

1. *Формирование матрицы смежности.* Положить $k = n - 1$, где $n = \mathbf{card}(V)$. Для всех $v_i \in V$ $(I \leq k)$ проверить условие: $h(v_n, v_i) = 0$. Если оно выполнено, то положить $p(v_n, v_i) = 0$ $(\forall v_i \in V)$. В противном случае, положить $p(v_n, v_i) = 1$.
2. *Определение путей в матрице смежности.* Для каждой из вершин $v_i \in V$ такой, что $i < k$, определить целесообразность объединения вершин $v_k, v_i \in V$ в один класс разбиения. Если выполнено условие: $h(v_k, v_i) + h(v_{k+1}, v_k) + p(v_{k+1}, v_i) = 0$, то принять *нецелесообразным* объединение вершин $v_i, v_j \in V$ и положить $p(v_k, v_i) = 0$. В противном случае, принять объединение данных задач *целесообразным* и положить $p(v_k, v_i) = 1$.
3. Если выполнено условие $k > 2$, то уменьшить k на единицу и перейти к выполнению шага 2. В противном случае, если $k = 2$, следует перейти к выполнению шага 4.
4. *Формирование допустимых разбиений.* Сформировать допустимые разбиения вершин подмножества $\{v_1, v_2, \ldots, v_k\} \subset V$ с учетом целесообразности их объединения в один класс тогда и только тогда, когда выполнено условие: $p(v_i, v_i) = 1$. В противном случае, для всех допустимых разбиений вершины $v_i, v_j \in V$ должны включаться в разные классы. При этом все разбиения, содержащие классы вершин, для которых не выполняется ограничение (8.3.6), следует считать недопустимыми и исключить из дальнейшего рассмотрения.

5. Для каждого из сформированных допустимых разбиений множества вершин $\{v_1, v_2,\ldots, v_k\}\subset V$ определить соответствующее значение целевой функции (8.3.5).
6. Если выполнено условие: $k = n$, то обозначить множество сформированных допустимых разбиений через Δ_s и перейти к выполнению шага 7. В противном случае, если $k < n$, следует увеличить значение k на единицу и перейти к выполнению шага 4.
7. Определение оптимального разбиения. Среди разбиений множества Δ_s найти такое, которому соответствует максимальное значение целевой функции. Найденное значение является точным решением задачи о разбиении, после чего следует закончить работу алгоритма.

Примечание

Следует заметить, что нахождение значений $p(v_i, v_j)$ $(\forall v_i, v_j\in V)$ соответствует введению в рассмотрение некоторого нового графа $G'(V, E', p)$ с тем же множеством вершин, что и у исходного графа $G(V, E)$. Множество ребер нового графа $G'(V, E', p)$ определяется матрицей смежности, которая образована найденными значениями: $p(v_i, v_j)$ $(\forall v_i, v_j\in V)$. Именно эта матрица определяется на шаге 1 и используется на последующих шагах 2—4 описанного алгоритма.

Рассмотренный алгоритм метода динамического программирования может быть изображен графически в форме следующей диаграммы деятельности языка UML (рис. 8.12).

Нетрудно заметить, что в силу конечности общего количества вершин исходного графа, рассмотренный алгоритм метода динамического программирования также является конечным. Тем не менее, комбинаторный характер данной задачи ограничивает практические возможности нахождения точного решения задач этого класса с помощью метода динамического программирования.

Проиллюстрируем использование рассмотренного алгоритма метода динамического программирования для решения индивидуальной задачи о разбиении, исходный граф которой изображен на рис. 8.7. При этом для удобства в качестве вершин, входящих в отдельные классы разбиений, будем указывать их порядковый номер, что является эквивалентной формой записи. В соответствии с описанной ранее схемой алгоритма следует выполнить следующие практические действия.

Шаг 1. До начала выполнения основных итераций алгоритма находятся следующие значения: $p(8, 1) = 0$, $p(8, 2) = 0$, $p(8, 3) = 1$, $p(8, 4) = 0$, $p(8, 5) = 1$, $p(8, 6) = 0$, $p(8, 7) = 1$. После этого следует перейти к выполнению действий первой итерации шага 2.

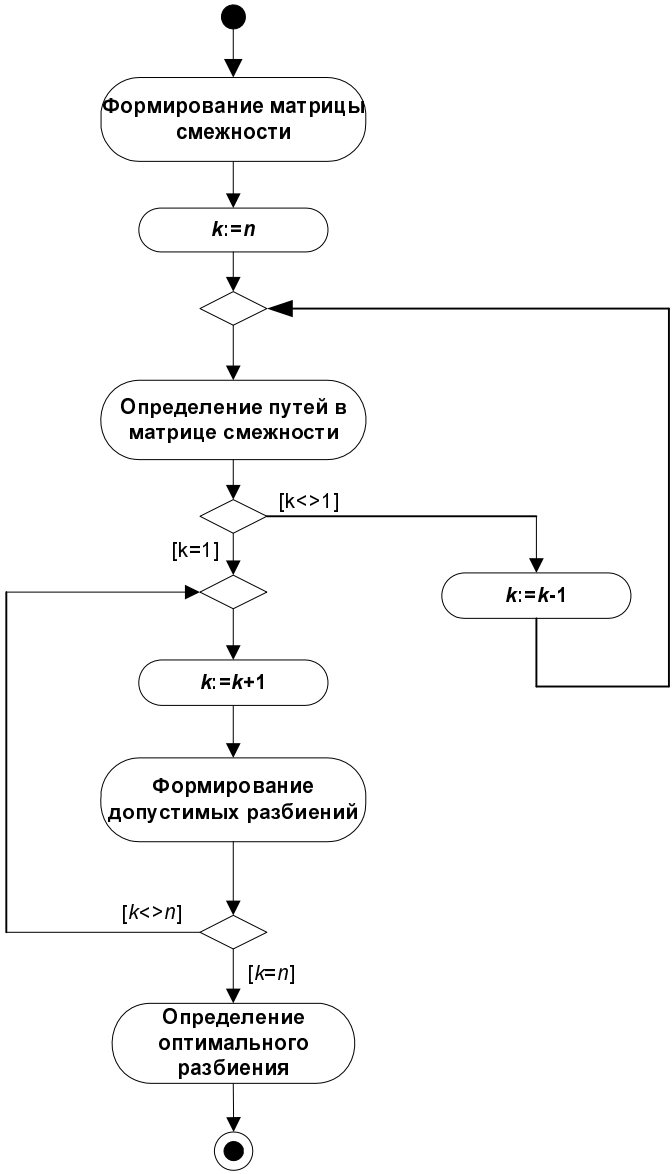


Рис. 8.12. Диаграмма деятельности алгоритма метода динамического программирования для решения задачи о разбиении

Шаг 2 (первая итерация). Последовательно рассчитываются значения функций p для $k = 7$. Соответствующие значения функций равны: $p(7, 1) = 0$, $p(7, 2) = 0$, $p(7, 3) = 1$, $p(7, 4) = 0$, $p(7, 5) = 1$, $p(7, 6) = 1$. После этого перейти к выполнению действий шага 3.

Шаг 3. Поскольку выполнено условие $k > 2$, то следует перейти к выполнению действий второй итерации шага 2.

Шаг 2 (вторая итерация). Последовательно рассчитываются значения функций p для $k = 6$. Соответствующие значения функций равны: $p(6, 1) = 1$, $p(6, 2) = 1$, $p(6, 3) = 1$, $p(6, 4) = 1$, $p(6, 5) = 1$. После этого перейти к выполнению действий шага 3.

Шаг 3. Поскольку выполнено условие $k > 2$, то следует перейти к выполнению действий третьей итерации шага 2.

Шаг 2 (третья итерация). Последовательно рассчитываются значения функций p для $k = 5$. Соответствующие значения функций равны: $p(5, 1) = 1$, $p(5, 2) = 1$, $p(5, 3) = 1$, $p(5, 4) = 1$. После этого перейти к выполнению действий шага 3.

Шаг 3. Поскольку выполнено условие $k > 2$, то следует перейти к выполнению действий четвертой итерации шага 2.

Шаг 2 (четвертая итерация). Последовательно рассчитываются значения функций p для $k = 4$. Соответствующие значения функций равны: $p(4, 1) = 1$, $p(4, 2) = 1$, $p(4, 3) = 1$. После этого перейти к выполнению действий шага 3.

Шаг 3. Поскольку выполнено условие $k > 2$, то следует перейти к выполнению действий пятой итерации шага 2.

Шаг 2 (пятая итерация). Последовательно рассчитываются значения функций p для $k = 3$. Соответствующие значения функций равны: $p(3, 1) = 1$, $p(3, 2) = 1$. После этого перейти к выполнению действий шага 3.

Шаг 3. Поскольку выполнено условие $k > 2$, то следует перейти к выполнению действий шестой итерации шага 2.

Шаг 2 (шестая итерация). Последовательно рассчитываются значения функций p для $k = 2$. Соответствующее единственное значение функции равно: $p(2, 1) = 1$. После этого перейти к выполнению действий шага 3.

Шаг 3. Поскольку выполнено условие $k = 2$, то следует перейти к выполнению действий шага 4.

Шаг 4 (первая итерация). Допустимые разбиения вершин подмножества $\{v_1, v_2\} \subset V$ равны: $R_1^1 = \{\{1\}, \{2\}\}$ и $R_2^1 = \{\{1, 2\}\}$. После этого перейти к выполнению действий шага 5.

Шаг 5 (первая итерация). Значения целевой функции для сформированных допустимых разбиений вершин подмножества $\{v_1, v_2\} \subset V$ равны: $f\left(R_1^1\right) = 0$ и $f\left(R_2^1\right) = 9$. После этого перейти к выполнению действий шага 6.

Шаг 6. Поскольку $k < 8$, следует увеличить значение k на единицу и перейти к выполнению второй итерации шага 4.

Шаг 4 (вторая итерация). Допустимые разбиения вершин подмножества $\{v_1, v_2, v_3\} \subset V$ равны: $R_1^2 = \{\{1\}, \{2\}, \{3\}\}$, $R_2^2 = \{\{1, 2\}, \{3\}\}$, $R_3^2 = \{\{1, 3\}, \{2\}\}$, $R_4^2 = \{\{1\}, \{2, 3\}\}$, $R_5^2 = \{\{1, 2, 3\}\}$. После этого перейти к выполнению действий шага 5.

Шаг 5 (вторая итерация). Значения целевой функции для сформированных допустимых разбиений вершин подмножества $\{v_1, v_2, v_3\} \subset V$ равны: $f\left(R_1^2\right) = 0$, $f\left(R_2^2\right) = 9$, $f\left(R_3^2\right) = 0$, $f\left(R_4^2\right) = 4$, $f\left(R_5^2\right) = 13$. После этого перейти к выполнению действий шага 6.

Шаг 6. Поскольку $k < 8$, следует увеличить значение k на единицу и перейти к выполнению третьей итерации шага 4.

Примечание

Поскольку выполнение последующих итераций алгоритма представляет собой рутинную процедуру, она здесь не приводится. Заинтересованным читателям предлагается выполнить итерации 3—8 самостоятельно в качестве упражнения. Далее приводится результат выполнения только завершающей восьмой итерации.

Шаг 4 (восьмая итерация). Допустимые разбиения всех вершин исходного графа: $V=\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ равны: $R_1^8 = \{\{1\}, \{2\}, \{3, 8\}, \{4\}, \{5\}, \{6\}, \{7\}\}$, $R_2^8 = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5, 8\}, \{6\}, \{7\}\}$, $R_3^8 = \{\{1, 2\}, \{3, 8\}, \{4\}, \{5\}, \{6\}, \{7\}\}$, $R_4^8 = \{\{1, 2\}, \{3\}, \{4\}, \{5, 8\}, \{6\}, \{7\}\}$, $R_5^8 = \{\{1, 2, 3\}, \{4\}, \{5, 8\}, \{6\}, \{7\}\}$, $R_6^8 = \{\{1, 2, 4\}, \{3\}, \{5, 8\}, \{6\}, \{7\}\}$, $R_7^8 = \{\{1, 2, 3, 4\}, \{5, 8\}, \{6\}, \{7\}\}$, $R_8^8 = \{\{1\}, \{2, 3, 4, 8\}, \{5\}, \{6\}, \{7\}\}$, $R_9^8 = \{\{1, 2, 3, 4, 6\}, \{5, 8\}, \{7\}\}$, $R_{10}^8 = \{\{1, 2, 3, 4\}, \{5, 6, 8\}, \{7\}\}$, $R_{11}^8 = \{\{1, 2, 3, 4\}, \{5, 8\}, \{6, 7\}\}$, $R_{12}^8 = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}$. После этого перейти к выполнению действий шага 5.

Шаг 5 (восьмая итерация). Значения целевой функции для сформированных допустимых разбиений вершин множества V равны: $f\left(R_1^8\right) = 1$, $f\left(R_2^8\right) = 9$, $f\left(R_3^8\right) = 10$, $f\left(R_4^8\right) = 15$, $f\left(R_5^8\right) = 22$, $f\left(R_6^8\right) = 25$, $f\left(R_7^8\right) = 34$, $f\left(R_8^8\right) = 10$, $f\left(R_9^8\right) = 35$, $f\left(R_{10}^8\right) = 36$, $f\left(R_{11}^8\right) = 40$, $f\left(R_{12}^8\right) = 44$. После этого следует перейти к выполнению действий шага 6.

Шаг 6. Поскольку $k = 8$, следует перейти к выполнению шага 7.

Шаг 7. Среди всех найденных на предыдущем шаге разбиений оптимальным является то, для которого целевая функция принимает максимальное значение: $f\left(R_{12}^{8}\right) = 44$. Соответствующее разбиение равно: $R_{12}^{8} = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}$. На этом выполнение алгоритма заканчивается.

Таким образом, результатом решения комбинаторной задачи о разбиении методом динамического программирования является найденное оптимальное значение — разбиение: $R_{\text{opt}} = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}$, которому соответствует значение целевой функции: $f_{\text{opt}} = 44$. Анализ результатов решения задачи о разбиении с помощью программы MS Excel и методом динамического программирования показывает их полное совпадение, что служит веским аргументом в пользу их достоверности.

Примечание

Следует отметить, что хотя трудоемкость рассмотренного алгоритма для случаев, когда существуют такие вершины в исходном графе: v_i, $v_j \in V$, для которых выполнено условие: $h(v_i, v_j) = 0$, отличается от полного перебора, выполненные вычислительные эксперименты показывают близкую к экспоненциальной зависимость требуемого объема оперативной памяти ЭВМ от размерности исходных данных. Последнее обстоятельство существенно ограничивает возможность применения данного алгоритма для точного решения практических задач о разбиении.

Далее в *главе 12* рассматриваются алгоритм и программа приближенного решения задачи о разбиении, которая может быть использована для практического нахождения решений задач данного класса для исходных графов с достаточно большим числом вершин ($n > 40$). В последнем случае пользователь также избавляется от необходимости выполнения рутинных операций по записи системы ограничений в программе MS Excel.

8.4. Упражнения

В качестве упражнений для самостоятельного решения предлагаются задачи, аналогичные типовым задачам комбинаторной оптимизации, рассмотренным в данной главе. Предлагаемые в качестве упражнений задачи оптимизации содержат конкретные значения исходных данных в форме матриц смежности взвешенных графов, что позволяет получить их количественное решение с помощью программы MS Excel. Те из читателей, кто сочтет для себя интересным найти решение данных задач несколькими способами, получат возможность убедиться в правильности полученных решений.

8.4.1. Задача коммивояжера

Для ориентированного графа, матрица весов дуг которого представлена в табл. 8.1, определить оптимальный замкнутый путь, проходящий через каждую вершину ровно один раз и обладающий минимальной общей длиной.

Таблица 8.1. *Матрица весов дуг исходного графа для задачи коммивояжера*

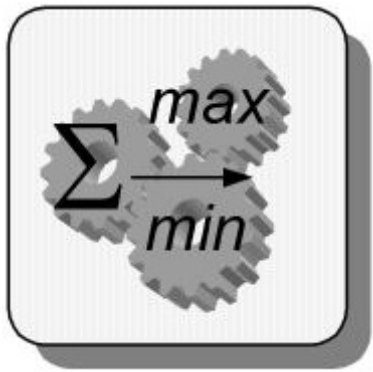
Вершины исходного графа	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
V_1	0	15	7	10	9	21	5	11
V_2	17	0	10	15	7	12	6	9
V_3	11	9	0	13	25	14	8	10
V_4	12	7	13	0	21	24	10	17
V_5	23	8	9	13	0	15	21	16
V_6	17	21	8	11	13	0	10	14
V_7	9	11	20	15	10	17	0	8
V_8	7	12	17	10	9	11	22	0

8.4.2. Задача о разбиении

Для неориентированного графа, матрица весов ребер и веса вершин которого представлена в табл. 8.2, определить оптимальное разбиение вершин, общий вес вершин в классах которого не превосходит величины 15, а общая сумма попарных весов ребер для вершин во всех классах является максимальной.

Таблица 8.2. *Матрица весов дуг исходного графа для задачи о разбиении*

Вершины исходного графа	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	**Вес вершин**
V_1	0	10	7	0	9	0	8	12	3
V_2	10	0	0	15	0	12	0	9	4
V_3	7	0	0	0	0	14	11	0	7
V_4	0	15	0	0	10	0	9	0	2
V_5	9	0	0	10	0	15	0	16	6
V_6	0	12	14	0	15	0	9	0	4
V_7	8	0	11	9	0	9	0	13	5
V_8	12	9	0	0	16	0	13	0	3



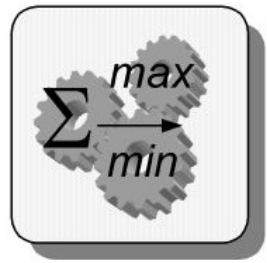
Часть IV

Задачи многокритериальной оптимизации

Глава 9. Задачи многокритериального линейного и целочисленного программирования

Глава 10. Задачи многокритериальной булевой оптимизации

При постановке многих практических задач оптимизации возникает ситуация, когда невозможно представить в форме единственной целевой функции желаемый результат их решения. Это происходит в тех случаях, когда в качестве отдельных критериев оптимизации выступают две или более характеристики рассматриваемых в качестве решений элементов или объектов. Соответствующие задачи оптимизации отличаются по своему характеру от рассмотренных ранее тем, что вместо единственной целевой функции в них присутствуют несколько целевых функций. Это обстоятельство послужило основой для включения подобных задач в отдельный класс задач оптимизации, имеющих самостоятельный предмет исследования и важное научно-прикладное значение. Именно задачи этого класса и методы их решения с помощью программы MS Excel рассматриваются в *части IV*.



Глава 9

Задачи многокритериального линейного и целочисленного программирования

К классу задач многокритериального линейного программирования относятся такие задачи непрерывной оптимизации, в которых, с одной стороны, имеется несколько целевых функций, а с другой стороны, все целевые функции являются линейными функциями своих аргументов. При этом все ограничения также могут быть представлены в форме линейных функций. Задачи многокритериальной оптимизации этого класса и методы их решения являются предметом обсуждения в *главе 9*.

9.1. Общая характеристика задач многокритериальной оптимизации

В общем случае под *задачей многокритериальной оптимизации* понимается такая задача оптимизации, в которой имеется несколько целевых функций. При этом по аналогии с обычными задачами оптимизации ограничения могут присутствовать, — и тогда соответствующая задача многокритериальной оптимизации называется задачей с ограничениями; либо отсутствовать, — и тогда соответствующая задача многокритериальной оптимизации называется задачей без ограничений. Поскольку задачи многокритериальной оптимизации без ограничений являются частным случаем соответствующих задач с ограничениями, в дальнейшем, если дополнительно не отмечено, рассматриваются только задачи с ограничениями. Более строгое определение данного класса задач можно получить на основе рассмотрения общей математической постановки задачи многокритериальной оптимизации, которое приводится далее.

9.1.1. Математическая постановка задачи многокритериальной оптимизации

В общем случае математическая постановка задачи многокритериальной оптимизации с одной переменной может быть сформулирована в следующем виде:

$$f_i(x) \to \max_{x \in \Delta_\beta} \quad \text{или} \quad f_i(x) \to \min_{x \in \Delta_\beta} \ (\forall i \in L = \{1, 2, ..., l\}, \tag{9.1.1}$$

где $$\Delta_\beta = \{\Delta \mid g_k(x) \le (=) 0\}, \qquad (k \in \{1, 2, ..., m\}). \tag{9.1.2}$$

При этом не вводится никаких дополнительных предположений о характере целевой функции и левых частей ограничений. В случае $l = 1$ задача многокритериальной оптимизации становится обычной задачей оптимизации вида (1.5.1) и (1.5.2). В общем случае задачу оптимизации вида (9.1.1) и (9.1.2) называют также задачей *l-критериальной* оптимизации, задачей оптимизации с l критериями или просто задачей многокритериальной оптимизации. В частном случае при $l = 2$ соответствующую задачу оптимизации часто называют задачей *двухкритериальной* оптимизации или задачей оптимизации с двумя критериями. В приложениях двухкритериальные задачи оптимизации встречаются наиболее часто.

Примечание

Как уже отмечалось ранее при рассмотрении задач однокритериальной оптимизации, минимизация целевой функции $f(x)$ эквивалентна максимизации функции: $-f(x)$. Поэтому для удобства изложения без ограничения общности в последующем будем предполагать, что при рассмотрении общей постановки задачи многокритериальной оптимизации (9.1.1) и (9.1.2) необходимо найти максимум всех целевых функций (9.1.1). В случае наличия нескольких переменных может быть сформулирована аналогичная постановка задачи многокритериальной оптимизации (9.1.1) и (9.1.2), в которой переменная x заменена множеством переменных $\{x_1, x_2, .., x_n\}$.

Если рассмотреть некоторые дополнительные предположения о характере целевой функции и левых частей ограничений, то можно получить специальные классы задач многокритериальной оптимизации. Так, например, если предположить, что все целевые функции $f_i(x)$ являются нелинейными, при этом левые части ограничений $g_k(x)$ могут быть как линейными, так и нелинейными функциями относительно единственного аргумента x, то соответствующий класс задач оптимизации называется задачами *нелинейной многокритериальной оптимизации*. При этом переменная x может быть как единственной, так и не единственной в подобных задачах оптимизации. Что

касается ее значений, то в качестве исходного множества значений переменных принимается множество действительных чисел $\boldsymbol{R}^1$, т. е. $x \in \boldsymbol{R}^1$.

Аналогичным образом могут быть определены и другие специальные классы многокритериальных задач оптимизации, такие как задачи многокритериального линейного и целочисленного программирования, а также задачи многокритериальной булевой и комбинаторной оптимизации. В этом контексте определяющим фактором являются математические свойства критериальных функций и ограничений, такие как линейность или целочисленность.

Принципиальным отличием задач многокритериальной оптимизации от задач оптимизации с единственной целевой функцией является само понятие решения. Дело в том, что при формулировке и анализе задач многокритериальной оптимизации понятие решения в традиционном смысле может отсутствовать или не иметь смысла. В практических задачах нахождение оптимального решения по одному из критериев, как правило, не соответствует оптимальному решению по другим критериям. Тем самым на практике имеет место *противоречивость* критериальных функций, связанная с невозможностью найти оптимальное допустимое решение задачи (9.1.1) и (9.1.2) сразу по всем целевым функциям.

Примечание

В связи с этим как не вспомнить о желании обычных покупателей приобретать вещи наилучшего качества за наименьшую цену. Аналогичная ситуация встречается при выполнении проектов, когда заказчик требует выполнить некоторые работы за наименьшее время при наименьшей стоимости, а исполнитель заинтересован не только в максимальном финансировании проекта, но и в более продолжительном его выполнении. Тем самым конфликт интересов на практике может быть связан с многокритериальным характером соответствующих задач оптимизации. Можно даже высказать утверждение, что бесконфликтные задачи многокритериальной оптимизации не представляют ни практического, ни математического интереса, поскольку решение в этом случае получается тривиальным образом.

В общем случае понятие решения задачи многокритериальной оптимизации (9.1.1) и (9.1.2) тесно связано с анализом множества допустимых альтернатив. С этой целью на множестве допустимых альтернатив вводится некоторое специальное отношение, получившее название *отношение доминирования по Парето*. Оно названо в честь итальянского экономиста В. Парето, который впервые ввел его в рассмотрение при изучении экономических задач.

Формально это отношение доминирования по Парето определяется следующим образом: некоторое допустимое решение $x_i \in \Delta_\beta$ задачи многокритери-

альной оптимизации в форме максимизации целевых функций (9.1.1) *доминирует по Парето* допустимое решение $x_j \in \Delta_\beta$, если для всех целевых функций значения $f_r(x_j)$ не превосходят значений $f_r(x_i)$, при этом хотя бы для одной целевой функции f_s значение $f_s(x_i)$ превосходит значение $f_s(x_j)$. Если отношение доминирования по Парето обозначить через $\succ$, то соответствующие условия формально можно записать в следующем виде:

$$x_i \succ x_j \text{ тогда и только тогда, когда } f_s(x_i) \geq f_s(x_j), (\forall s \in L) \quad (9.1.3)$$
$$\text{и при этом } \exists r \in L, \text{ такое, что : } f_r(x_i) > f_r(x_j).$$

Здесь x_i, $x_j \in \Delta_\beta$ — произвольные допустимые альтернативы из множества Δ_β для задачи многокритериальной оптимизации в форме максимизации целевых функций (9.1.1).

Принципиальным для задачи многокритериальной оптимизации является тот факт, что доминируемые по Парето допустимые альтернативы могут быть исключены из рассмотрения, поскольку для каждой из них по определению всегда найдется допустимая альтернатива со значениями целевых функций не меньшими, чем значения этих функций для доминируемой альтернативы.

В общем случае отношение доминирования по Парето является отношением частичного строгого порядка на множестве допустимых альтернатив Δ_β. При этом наибольший интерес представляют именно те допустимые альтернативы, которые нельзя сравнить между собой по данному отношению. Такое множество образует собственное подмножество во множестве Δ_β и имеет название: множество недоминируемых по Парето альтернатив $\Delta_\beta^{nd} \subset \Delta_\beta$ или просто — множество недоминируемых альтернатив.

Формально недоминируемые альтернативы множества Δ_β^{nd} должны удовлетворять следующему условию:

$$x_i, x_j \in \Delta_\beta^{nd} \text{ тогда и только тогда, когда не имеет места:} \quad (9.1.4)$$
$$\text{ни } x_i \succ x_j, \text{ ни } x_j \succ x_i, \text{ и при этом не существует}$$
$$x_p, x_q \in \Delta_\beta^{nd}, \text{таких что : } x_p \succ x_i \text{ или } x_q \succ x_j.$$

Проверка данного условия может служить конструктивным способом определения множества недоминируемых альтернатив Δ_β^{nd} в задачах многокритериальной оптимизации с конечным множеством допустимых альтернатив. Действительно, выполнив проверку условия (9.1.4) для всех пар альтернатив множества Δ_β и исключив из рассмотрения доминируемые по Парето альтернативы, оставшиеся альтернативы в точности образуют множество Δ_β^{nd}.

Что качается задач многокритериальной оптимизации с бесконечным множеством допустимых альтернатив, то в этом случае для определения множества Δ_{β}^{nd} необходимо принять некоторые дополнительные предположения о характере целевых функций и ограничений соответствующих задач многокритериальной оптимизации.

В общем случае, все недоминируемые по Парето альтернативы являются эквивалентными между собой с точки зрения исходной постановки задачи многокритериальной оптимизации (9.1.1) и (9.1.2). Для выбора единственной альтернативы, которая должна служить итоговым решением задачи многокритериальной оптимизации, необходимы некоторые дополнительные предположения о свойствах искомого решения или о предпочтениях лиц или экспертов, принимающих окончательное решение. При этом в отдельных случаях подобные предположения могут включаться в исходную постановку задачи (9.1.1) и (9.1.2).

В общем случае понятие решения задачи многокритериальной оптимизации включает в себя в качестве составного элемента предварительное определение множества недоминируемых альтернатив и последующий анализ этого множества с целью выбора окончательной единственной альтернативы в качестве итогового решения задачи многокритериальной оптимизации. Поскольку нахождение окончательного решения возможно только при наличии дополнительных предположений относительно свойств этого решения, заданных в форме предпочтения лиц или экспертов, принимающих решения, то в качестве базового решения задач многокритериальной оптимизации принимается нахождение множества недоминируемых альтернатив Δ_{β}^{nd}.

Таким образом, множество недоминируемых альтернатив может служить как окончательным решением задач многокритериальной оптимизации в случае отсутствия дополнительных предположений о свойствах окончательного решения, так и основой для принятия окончательного решения в случае наличия подобных предположений. Именно по этой причине при решении практических задач многокритериальной оптимизации, когда необходимо получение некоторого окончательного решения поставленной задачи, следует дополнить постановку задачи (9.1.1) и (9.1.2) информацией относительно свойств этого решения. В противном случае получение окончательного единственного решения оказывается принципиально невозможным, о чем следует помнить всем системным аналитикам, приступающим к решению задач многокритериальной оптимизации.

9.1.2. Основные подходы и методы решения задач многокритериальной оптимизации

В общем случае можно предложить два подхода к решению задач многокритериальной оптимизации. С одной стороны, та или иная задача многокритериальной оптимизации может быть решена аналитически. При этом под *аналитическим решением* задачи многокритериальной оптимизации понимают установление некоторой функциональной зависимости между исходными данными задачи и точным ее решением в форме множества недоминируемых альтернатив или окончательного решения, допускающей нахождение множества Δ_β^{nd} или итогового решения по известным значениям аргументов.

Так, например, базовым способом нахождения аналитических решений в форме построения множества недоминируемых альтернатив Δ_β^{nd} для задач многокритериальной оптимизации вида (9.1.1) и (9.1.2) является решение задач однокритериальной оптимизации типа:

$$\alpha_1 f_1(x) + \alpha_2 f_2(x) + ... + \alpha_l f_l(x) \to \max_{x \in \Delta_\beta}, \qquad (9.1.5)$$

где $$\Delta_\beta = \{\Delta \mid g_k(x) \le (=) 0\}, \qquad (k \in \{1, 2, ..., m\}). \qquad (9.1.6)$$

При этом значения $\{\alpha_1, \alpha_2, ..., \alpha_l\}$, называемые *весовыми коэффициентами целевых функций*, должны удовлетворять следующему условию: $\alpha_1, \alpha_2, ..., \alpha_l \in [0, 1]$ и $\alpha_1+\alpha_2+...+\alpha_l = 1$. При этом имеет место замечательный факт, а именно: множество недоминируемых альтернатив Δ_β^{nd} для задачи многокритериальной оптимизации в общей постановке (9.1.1) и (9.1.2) с непрерывными и выпуклыми целевыми функциями, а также компактным и выпуклым множеством допустимых альтернатив может быть получено в результате решения задачи однокритериальной оптимизации (9.1.5) и (9.1.6) для всех возможных комбинаций весовых коэффициентов целевых функций, т. е. для всех возможных вещественных значений: $\alpha_1, \alpha_2, ..., \alpha_l \in [0, 1]$, таких что $\alpha_1 + \alpha_2 + \ldots + \alpha_l = 1$.

К сожалению, данный способ нахождения множества недоминируемых альтернатив Δ_β^{nd} не находит практического применения, поскольку является довольно трудоемким с вычислительной точки зрения. В то же время на нем основан широко распространенный способ решения задач многокритериальной оптимизации, получивший название *метода аддитивной свертки* целевых функций. Действительно, если в дополнение к постановке задачи (9.1.1) и (9.1.2) задать некоторые значения весовых коэффициентов целевых функций,

удовлетворяющих условию: $\alpha_1, \alpha_2, \ldots, \alpha_l \in [0, 1]$ и $\alpha_1 + \alpha_2 + \ldots + \alpha_l = 1$, то в результате решения единственной задачи однокритериальной оптимизации (9.1.5) и (9.1.6) может быть получено итоговое решение исходной задачи. При этом создается впечатление о ненужности построения и анализа собственно множества недоминируемых альтернатив.

В связи с этим следует высказать несколько замечаний общего характера.

1. Во-первых, любое оптимальное решение задачи (9.1.5) и (9.1.6) принадлежит множеству недоминируемых альтернатив, а значит и решение задачи многокритериальной оптимизации, полученное методом аддитивной свертки целевых функций при выполнении исходных предположений относительно непрерывности целевых функций и компактности множества допустимых альтернатив, дает в результате некоторое недоминируемое по Парето решение.
2. Во-вторых, априорное задание некоторых значений весовых коэффициентов целевых функций, удовлетворяющих условию: $\alpha_1, \alpha_2, \ldots, \alpha_l \in [0, 1]$ и $\alpha_1 + \alpha_2 + \ldots + \alpha_l = 1$, является дополнительной информацией к постановке общей задачи многокритериальной оптимизации (9.1.1) и (9.1.2), а значит, метод аддитивной свертки целевых функций не может служить универсальным средством решения задачи многокритериальной оптимизации. К сожалению, данный факт зачастую просто игнорируется во многих работах при решении соответствующих практических задач.
3. В-третьих, в результате целого ряда специальных исследований психофизиологических факторов, оказывающих влияние на процесс принятия решения человеком, были получены выводы о том, что априорное задание значений весовых коэффициентов целевых функций является для экспертов достаточно трудоемкой операцией, приводящей к ненадежным результатам. В связи с этим подлежит сомнению любое решение задачи многокритериальной оптимизации, полученное с помощью метода аддитивной свертки целевых функций.
4. Наконец, в-четвертых, сведение задачи многокритериальной оптимизации (9.1.1) и (9.1.2) к решению задачи однокритериальной оптимизации (9.1.5) и (9.1.6) по своей сути не вносит ничего нового в процесс анализа и решения исходной задачи (9.1.1) и (9.1.2). Действительно, в этом случае следует просто воспользоваться известными методами решения задач однокритериальной оптимизации того или иного класса. По этой причине данный способ не представляет самостоятельного интереса, а его абсолютизация является следствием низкой квалификации системных аналитиков.

Среди других подходов к решению задач многокритериальной оптимизации наибольшее применение нашли так называемые метод уступок и метод ми-

нимизации отклонения от идеальной точки, которые по своему характеру могут быть применены для решения задач многокритериальной оптимизации различных классов. Именно эти методы используются далее при решении практических задач многокритериальной оптимизации отдельных классов в *главах 9* и *10*.

Поскольку аналитическое решение задач многокритериальной оптимизации возможно только для простейших задач с дополнительными предположениями о характере целевой функции и ограничений, то альтернативой аналитическому решению является алгоритмическое или вычислительное решение задач многокритериальной оптимизации.

При этом под *алгоритмическим решением* задачи многокритериальной оптимизации понимают разработку или конструирование такой вычислительной процедуры, которая позволяет на основе известных исходных данных задачи находить ее решение в форме множества недоминируемых альтернатив или итогового решения. Соответствующая процедура может быть задана некоторым формальным образом и зафиксирована в форме алгоритма, т. е. формального предписания выполнить точно определенную последовательность действий, направленных на решение поставленной задачи многокритериальной оптимизации. Как правило, современные методы алгоритмического решения задач многокритериальной оптимизации предполагают использование компьютеров и соответствующих программ.

Кроме аналитического и алгоритмического подходов к решению задач многокритериальной оптимизации применительно к простейшим задачам рассматривают также способ *графического решения*, который основан на изображении графиков целевых функций и ограничений на плоскости или в трехмерном пространстве с последующим визуальным нахождением множества недоминируемых альтернатив или итогового решения. Наиболее часто графический способ решения используется для иллюстрации особенностей тех или иных методов решения задач многокритериального линейного и целочисленного программирования.

Методы алгоритмического решения задач многокритериальной оптимизации также делятся на две категории: методы нахождения точного решения и методы нахождения приближенного решения. Методы и алгоритмы первой группы позволяют за конечное время найти точное решение задачи многокритериальной оптимизации.

Методы и алгоритмы *приближенного решения* задач многокритериальной оптимизации позволяют находить некоторую аппроксимацию множества недоминируемых альтернатив, одно или несколько локально-оптимальных итоговых решений. В связи с этим нахождение точного решения задачи много-

критериальной оптимизации во всех случаях является наиболее предпочтительным. Если же точное решение по каким-либо причинам найти невозможно, то следует попытаться найти приближенное решение. При этом общей рекомендацией для нахождения приближенных решений также является применение нескольких методов или использование одного метода с различными параметрами с целью получения нескольких приближенных решений и выбора из них наибольшего для задач максимизации или наименьшего для задач минимизации.

Поскольку в программе MS Excel реализованы приближенные методы решения задач однокритериальной оптимизации с достаточно высокой степенью точности, то процесс решения задач многокритериальной оптимизации с помощью этой программы может быть связан с применением методов уступок и метода минимизации отклонения от идеальной точки. Для оценки точности получаемых решений на основе сравнения аналитических и алгоритмических способов их решения следует использовать несколько различных методов и подходов. Однако следует помнить, что выбор адекватного метода решения задачи многокритериальной оптимизации напрямую зависит не только от постановки исходной задачи (9.1.1) и (9.1.2), но и от наличия дополнительной информации о свойствах окончательного решения.

9.1.3. Метод уступок для решения задач многокритериальной оптимизации

Метод уступок основан на введении некоторого предварительного упорядочения целевых функций по важности и допустимых отклонений от их оптимальных значений с последующим решением однокритериальных задач оптимизации известными аналитическими или алгоритмическими методами. Поскольку данный метод имеет в некотором смысле универсальный характер, он может быть описан независимо от класса задач оптимизации, которые могут быть решены с его помощью.

Не уменьшая общности дальнейшего изложения, рассмотрим постановку задачи многокритериальной оптимизации в следующей форме:

$$f_i(x) \to \min_{x \in \Delta_\beta} \qquad (\forall i \in L = \{1, 2, ..., l\}, \tag{9.1.7}$$

где $$\Delta_\beta = \{\Delta \mid g_k(x) \geq 0\}, \qquad (k \in \{1, 2, ..., m\}). \tag{9.1.8}$$

Дополнительно предполагается, что все критериальные функции *линейно упорядочены* по важности, например, в порядке возрастания их индексов. В этом случае целевая функция $f_1(x)$ является наиболее важной, а целевая функция $f_l(x)$ — наименее важной. Если это условие не выполняется, то сле-

дует выполнить переиндексацию целевых функций, так чтобы постановка задачи многокритериальной оптимизации (9.1.7) и (9.1.8) соответствовала этому требованию.

Наконец, для применения метода уступок должно быть задано множество положительных действительных чисел: $\Theta = \{\delta_1, \delta_2, \ldots, \delta_l\}$, каждое из которых δ_i $(\forall i \in L)$ интерпретируется как величина *допустимой уступки* по целевой функции $f_i(x)$.

Алгоритм метода уступок, ориентированный на решение задач многокритериальной оптимизации в постановке (9.1.7) и (9.1.8), имеет итеративный характер и заключается в выполнении следующих действий:

1. *Предварительное задание исходных данных*. В качестве индекса целевой функции установить $i = 1$, а в качестве множества допустимых альтернатив принять исходное множество Δ_β, образуемое системой ограничений (9.1.8). После этого следует перейти к выполнению действий шага 2.
2. *Решение однокритериальной задачи*. Одним из методов решить однокритериальную задачу оптимизации: $f_i(x) \to \min\limits_{x \in \Delta_\beta}$. Найденное оптимальное значение целевой функции обозначим через: $b_i = f_i^{opt}$. После этого следует перейти к выполнению действий шага 3.
3. *Проверка условия окончания расчетов*. Если выполняется условие: $i = l$, то следует закончить выполнение алгоритма, приняв в качестве результата решения исходной задачи многокритериальной оптимизации значение: $x_{opt} = \arg\min f_i^{opt}$. Если данное условие не выполнено, т. е. $i < l$, то увеличить i на 1 и перейти к выполнению действий шага 4.
4. *Формирование нового множества допустимых альтернатив*. С этой целью следует к предыдущему множеству допустимых альтернатив Δ_β добавить дополнительное ограничение: $f_{i-1}(x) \le b_{i-1} + \delta_{i-1}$, которое интерпретируется как уступка относительно оптимального значения этой целевой функции. Полученное новое множество Δ_β считать за базовое множество допустимых альтернатив для целевой функции $f_i(x)$. После этого следует перейти к выполнению действий шага 2.

Рассмотренный алгоритм метода уступок может быть изображен графически в форме следующей диаграммы деятельности языка UML (рис. 9.1).

Нетрудно заметить, что в силу конечности общего количества целевых функций для исходной постановки задачи многокритериальной оптимизации (9.1.7) и (9.1.8), рассмотренный алгоритм метода уступок является конечным в случае конечности метода, который применяется для решения однокрите-

риальных задач оптимизации. Иллюстрация использования рассмотренного алгоритма метода уступок для решения индивидуальных практических задач многокритериальной оптимизации приводится далее в *главах 9* и *10*.

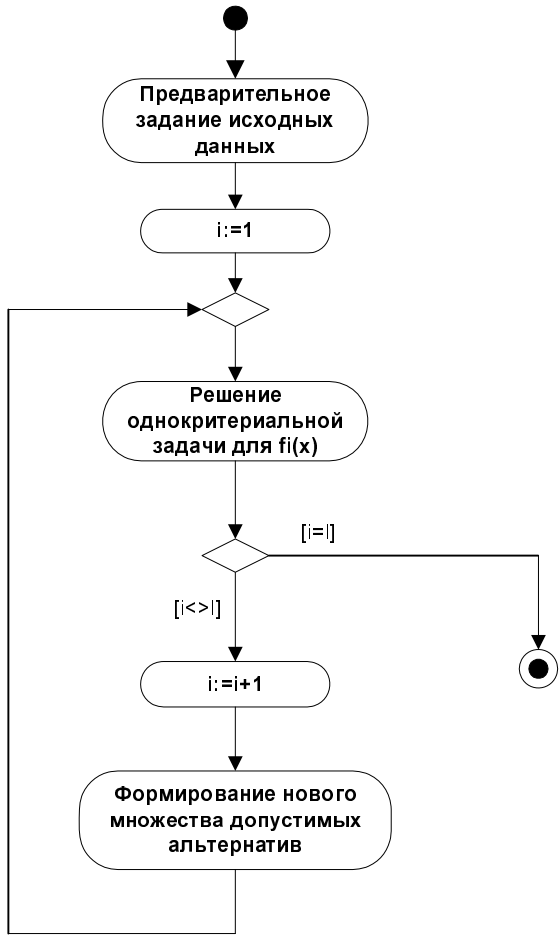


Рис. 9.1. Диаграмма деятельности метода уступок для решения многокритериальных задач оптимизации

Примечание

Следует заметить, что в случае постановки исходной задачи многокритериальной оптимизации в форме нахождения максимальных значений отдельных целевых функций вида (9.1.5) и (9.1.6) дополнительные ограничения должны иметь следующий вид: $f_{i-1}(x) \geq b_{i-1} - \delta_{i-1}$.

9.1.4. Метод минимального отклонения от идеальной точки

Метод минимального отклонения от идеальной точки является разновидностью общего метода свертки целевых функций, однако существенно отличается от него по характеру интерпретации итогового результата.

Основная идея метода заключается в том, чтобы предварительно найти так называемую идеальную точку задачи многокритериальной оптимизации, а после этого решить некоторую новую задачу однокритериальной оптимизации. При этом в качестве новой задачи оптимизации рассматривается задача минимизации отклонения от найденной идеальной точки в некоторой заданной метрике. Полученный результат принимается за окончательное решение исходной задачи многокритериальной оптимизации.

Под *идеальной точкой* задачи многокритериальной оптимизации в общей постановке (9.1.7) и (9.1.8) без каких бы то ни было дополнительных предположений понимается совокупность оптимальных значений: $f_1^{opt}, f_2^{opt},..., f_l^{opt}$ отдельных целевых функций на исходном множестве допустимых альтернатив Δ_β. При этом самой альтернативы, которой бы соответствовал набор значений $f_1^{opt}, f_2^{opt},..., f_l^{opt}$ как правило, не существует, или же она не принадлежит множеству допустимых альтернатив Δ_β. Именно во втором случае *"идеальной точкой"* называют саму альтернативу x^*, для которой выполняются условия: $f_1^{opt} = f_1(x^*), f_2^{opt} = f_2(x^*), ..., f_l^{opt} = f_l(x^*)$.

В качестве метрики, используемой для расчета количественного отклонения от идеальной точки, наиболее часто применяется метрика Евклида.

Алгоритм метода минимального отклонения от идеальной точки, ориентированный на решение задач многокритериальной оптимизации в постановке (9.1.7) и (9.1.8), имеет итеративный характер и заключается в выполнении следующих действий:

1. *Предварительное нахождение идеальной точки.* Одним из методов решить совокупность однокритериальных задач оптимизации: $f_i(x) \to \min\limits_{x \in \Delta_\beta} (\forall i \in L)$. Найденное оптимальное значение для каждой целевой функции обозначить через: $b_i = f_i^{opt}(\forall i \in L)$. После этого следует перейти к выполнению действий шага 2.
2. *Формирование новой целевой функции.* В качестве новой целевой функции следует рассмотреть функцию: $f(x) = (f_1(x) - b_1)^2 + (f_2(x) - b_2)^2 + \ldots + (f_l(x) - b_l)^2$, которая интерпретируется как отклонение от идеальной точки. После этого следует перейти к выполнению действий шага 3.

3. *Решение новой задачи оптимизации*. Одним из методов решить новую однокритериальную задачу оптимизации: $f(x) \to \min\limits_{x \in \Delta_\beta}$. Найденное оптимальное значение принять в качестве результата решения исходной задачи многокритериальной оптимизации: $x_{opt} = \arg\min f(x)$. На этом следует закончить выполнение алгоритма.

Рассмотренный алгоритм метода минимального отклонения может быть изображен графически в форме следующей диаграммы деятельности языка UML (рис. 9.2).

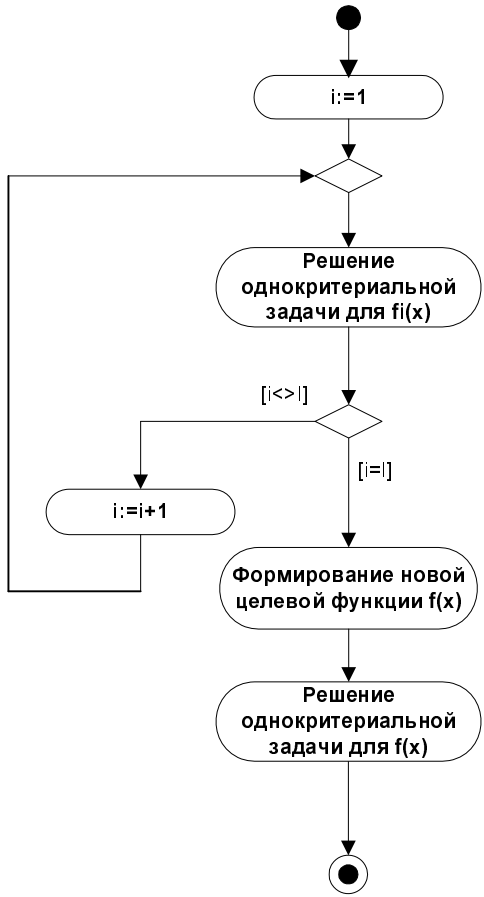


Рис. 9.2. Диаграмма деятельности метода минимального отклонения для решения многокритериальных задач оптимизации

Нетрудно заметить, что в силу конечности общего количества целевых функций для исходной постановки задачи многокритериальной оптимизации (9.1.7) и (9.1.8), рассмотренный алгоритм метода минимального отклонения является конечным в случае конечности метода, который применяется для решения однокритериальных задач оптимизации. Иллюстрация использования рассмотренного алгоритма метода минимального отклонения от идеальной точки для решения индивидуальных практических задач многокритериальной оптимизации приводится далее в *главах 9* и *10*.

Примечание

Следует заметить, что в случае постановки исходной задачи многокритериальной оптимизации в форме нахождения максимальных значений отдельных целевых функций вида (9.1.5) и (9.1.6) новая целевая функция и соответствующая задача оптимизации останутся без изменений.

9.2. Задача об оптимальной диете с двумя целевыми функциями

Содержательная постановка задачи об оптимальной диете приводится в *разд. 1.2.3*, а методы решения однокритериальной задачи рассматриваются в *разд. 4.2*. В настоящей главе рассматривается ее уточнение, необходимое для формальной записи условий соответствующей задачи многокритериальной оптимизации.

9.2.1. Математическая постановка задачи и подходы к ее решению

Для математической постановки данной задачи с двумя целевыми функциями необходимо определить переменные соответствующей задачи многокритериальной оптимизации, задать целевые функции и специфицировать ограничения, позволяющие представить исходную задачу как стандартную задачу многокритериального линейного программирования. В общем случае задача об оптимальной диете с двумя целевыми функциями может быть сформулирована следующим образом.

Имеется n видов продуктов питания, в которых содержится m типов питательных веществ (белки, жиры, углеводы). В одной весовой единице продукта i-го типа ($i \in \{1, 2,\ldots, n\}$) содержится a_{ij} единиц питательного вещества j-го вида ($j \in \{1, 2,\ldots, m\}$). Известна минимальная суточная потребность b_j ($j \in \{1, 2,\ldots, m\}$) человека в каждом из видов питательных веществ. Задана

калорийность c_i и стоимость d_i одной весовой единицы i-го продукта ($i \in \{1, 2,\ldots, n\}$). Требуется определить оптимальный состав рациона продуктов такой, чтобы каждое питательное вещество содержалось в нем в необходимом количестве, обеспечивающем суточную потребность человека, и при этом суммарная калорийность и стоимость рациона была минимальной.

Введем в рассмотрение следующие переменные: x_i — весовое количество продукта питания i-го типа в суточном рационе. Тогда в общем случае математическая постановка задачи об оптимальной диете может быть сформулирована следующим образом.

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \min_{x \in \Delta_\beta}; \tag{9.2.1}$$

$$d_1x_1 + d_2x_2 + \ldots + d_nx_n \to \min_{x \in \Delta_\beta}, \tag{9.2.2}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \ge b_1; \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \ge b_2; \\ \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \ge b_m \end{cases} \tag{9.2.3}$$

$$\text{и} \quad x_1, x_2, \ldots, x_n \ge 0.$$

Хотя математически задача об оптимальной диете с двумя целевыми функциями формулируется как задача о минимизации целевых функций, это не имеет принципиального значения для ее последующего решения. В связи с этим следует помнить, что, с учетом сделанного ранее замечания, всегда можно перейти от задачи максимизации целевой функции к эквивалентной ей задаче минимизации целевой функции и наоборот.

В качестве основных подходов к решению задач многокритериального линейного программирования, ориентированных на использование программы MS Excel, рассмотрим метод уступок и метод минимизации отклонения от идеальной точки.

9.2.2. Решение многокритериальной задачи об оптимальной диете с помощью программы MS Excel методом уступок

Для решения задачи об оптимальной диете с двумя целевыми функциями с помощью программы MS Excel рассмотренным ранее методом уступок

необходимо задать конкретные значения параметрам исходной задачи. Для определенности предположим, что в качестве исходных типов продуктов рассматриваются: хлеб, мясо, сыр, бананы, огурцы, помидоры, виноград ($n = 7$), а в качестве питательных веществ рассматриваются белки, жиры, углеводы ($m = 3$). Калорийность одной весовой единицы каждого из продуктов следующая: $c_1 = 2060$, $c_2 = 2430$, $c_3 = 3600$, $c_4 = 890$, $c_5 = 140$, $c_6 = 230$, $c_7 = 650$. Стоимость одной весовой единицы каждого из продуктов следующая: $d_1 = 12$, $d_2 = 100$, $d_3 = 160$, $d_4 = 24$, $d_5 = 40$, $d_6 = 30$, $d_7 = 80$.

Содержание питательных веществ в каждом из продуктов может быть задано в форме следующей таблицы (табл. 9.1).

Таблица 9.1. *Содержание питательных веществ в продуктах питания*

Продукты/ Питательные вещества	**Хлеб ржаной**	**Мясо баранина**	**Сыр "Российский"**	**Бананы**	**Огурцы**	**Помидоры**	**Виноград**
Белки	61	220	230	15	8	11	6
Жиры	12	172	290	1	1	2	2
Углеводы	420	0	0	212	26	38	155

Минимальная суточная потребность в питательных веществах следующая: в белках $b_1 = 100$, в жирах $b_2 = 70$, в углеводах $b_3 = 400$. Предполагается, что первая целевая функция является более важной, чем вторая. Величина уступок будет задана в последующем после нахождения оптимального значения первой целевой функции.

Примечание

Не уменьшая общности решаемой задачи, можно считать, что калорийность продуктов измеряется в *ккал/кг*, их стоимость — в *руб/кг*, суточная потребность в питательных веществах — в *граммах*, а содержание питательных веществ в продуктах — в *грамм/кг*. Что касается стоимости продуктов, то указывается некоторая средняя стоимость без учета сезонных колебаний цен и их различия в зависимости от качества соответствующих продуктов.

Для решения данной задачи с помощью программы MS Excel создадим новую книгу с именем Многокритериальная оптимизация и изменим имя ее первого рабочего листа: Задача о диете. Решение данной задачи многокритериальной оптимизации методом уступок будет состоять из 2-х этапов.

На первом этапе необходимо решить обычную задачу оптимизации, используя в качестве критериальной функции целевую функцию (9.2.1). Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *главе 4* при решении однокритериальной задачи о диете:

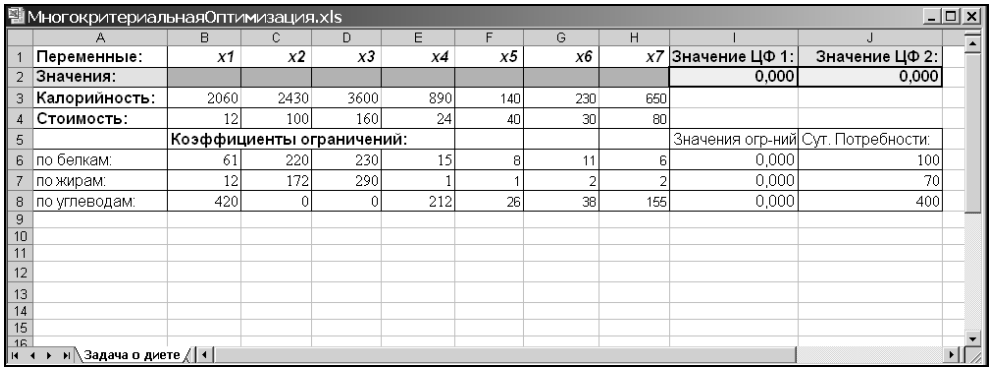
1. Внесем необходимые надписи в ячейки **A1:J1**, **A2:A8**, **B5**, **I5**, **J5**. Следует отметить, что конкретное содержание этих надписей не оказывает никакого влияния на решение рассматриваемой задачи линейного программирования.
2. В ячейки **B3:H3** введем значения коэффициентов первой целевой функции: $c_1 = 2060$, $c_2 = 2430$, $c_3 = 3600$, $c_4 = 890$, $c_5 = 140$, $c_6 = 230$, $c_7 = 650$.
3. В ячейки **B4:H4** введем значения коэффициентов второй целевой функции: $d_1 = 12$, $d_2 = 100$, $d_3 = 160$, $d_4 = 24$, $d_5 = 40$, $d_6 = 30$, $d_7 = 80$.
4. В ячейку **I2** введем формулу: `=СУММПРОИЗВ($B$2:$H$2;$B$3:$H$3)`, которая представляет первую целевую функцию (9.2.1).
5. В ячейку **J2** введем формулу: `=СУММПРОИЗВ($B$2:$H$2;$B$4:$H$4)`, которая представляет вторую целевую функцию (9.2.2).
6. В ячейки **B6:H8** введем значения коэффициентов ограничений, взятых из табл. 9.1.
7. В ячейки **J6:J8** введем значения правых частей ограничений, соответствующих минимальной суточной потребности в питательных веществах: в белках $b_1 = 100$, жирах $b_2 = 70$ и углеводах $b_3 = 400$.
8. В ячейку **I6** введем формулу: `=СУММПРОИЗВ($B$2:$H$2;B5:H5)`, которая представляет левую часть первого ограничения (9.2.3).
9. Скопируем формулу, введенную в ячейку **I6**, в ячейки **I7** и **I8**.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о диете методом уступок на первом этапе имеет следующий вид (рис. 9.3).

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$I$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.



МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	***x1***	***x2***	***x3***	***x4***	***x5***	***x6***	***x7***	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**								0,000	0,000
3	**Калорийность:**	2060	2430	3600	890	140	230	650		
4	**Стоимость:**	12	100	160	24	40	30	80		
5		**Коэффициенты ограничений:**							Значения огр-ний	Сут. Потребности:
6	по белкам:	61	220	230	15	8	11	6	0,000	100
7	по жирам:	12	172	290	1	1	2	2	0,000	70
8	по углеводам:	420	0	0	212	26	38	155	0,000	400
9										
10										
11										
12										
13										
14										
15										
16										

Задача о диете

Рис. 9.3. Исходные данные для решения задачи об оптимальной диете методом уступок на первом этапе

3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$H$2`.
4. Задать 3 ограничения, представляющие минимальные суточные потребности в питательных веществах. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$I$5**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";
   - в качестве значения правой части ограничения выбрать ячейку **$J$5**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**;
   - аналогичным образом задать оставшиеся два ограничения.
5. Задать ограничение на допустимые значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$H$2**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";

- в качестве значения правой части ограничения в поле с именем **Ограничение:** ввести значение 0;
- для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

В окне дополнительных параметров поиска решения следует выбрать отметки **Линейная модель** и **Неотрицательные значения**. Общий вид окна мастера поиска решения с заданными параметрами представлен на рис. 9.4.

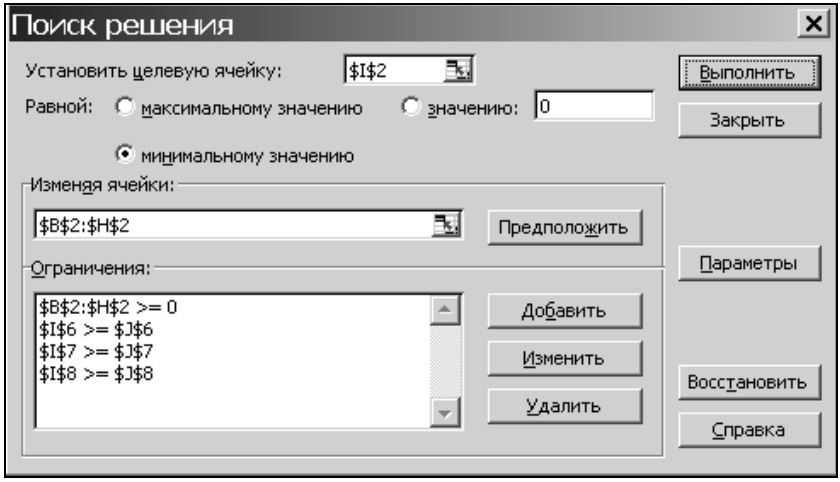


Рис. 9.4. Параметры мастера поиска решения и базовые ограничения первого этапа для двухкритериальной задачи об оптимальной диете

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**	0,000	0,211	0,109	1,887	0,000	0,000	0,000	2587,140	83,942
3	**Калорийность:**	2060	2430	3600	890	140	230	650		
4	**Стоимость:**	12	100	160	24	40	30	80		
5		**Коэффициенты ограничений:**							Значения огр-ний	Сут. Потребности:
6	по белкам:	61	220	230	15	8	11	6	100,000	100
7	по жирам:	12	172	290	1	1	2	2	70,000	70
8	по углеводам:	420	0	0	212	26	38	155	400,000	400
9										
10										
11										
12										
13										
14										
15										
16										

Задача о диете

Рис. 9.5. Результат количественного решения двухкритериальной задачи об оптимальной диете на первом этапе

После задания ограничений и целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 9.5).

Результатом решения двухкритериальной задачи об оптимальной диете на первом этапе является найденное значение первой целевой функции: $f_1^{opt} \cong 2587{,}140$. При выполнении расчетов для ячеек **B2:I2** был выбран числовой формат с 3 знаками после запятой. Анализ найденного решения показывает, что общая калорийность найденной диеты будет приближенно равна 2590 *ккал*, причем стоимость этой диеты составляет приблизительно 84 *руб*.

Предположим, что, исходя из медицинских рекомендаций, оказывается возможной уступка по первой целевой функции, равная 100 *ккал*. Тем самым можно перейти ко второму этапу решения данной задачи оптимизации методом уступок.

С этой целью следует рассмотреть дополнительное ограничение следующего вида:

$$c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 + c_6x_6 + c_7x_7 \le 2690, \qquad (9.2.4)$$

в котором правая часть равна значению суммы: f_1^{opt}+100, а знак ограничения выбран таким образом, чтобы соответствовать решению задачи минимизации целевой функции.

Для окончательного решения двухкритериальной задачи о диете внесем следующие дополнительные данные в лист с именем Задача о диете:

1. Внесем дополнительную надпись в ячейку **A9**.
2. Скопируем формулу из ячейки **I2** в ячейку **I9**, которая представляет левую часть дополнительного ограничения (9.2.4).
3. В ячейку **J9** введем значение 2490, которое представляет правую часть дополнительного ограничения (9.2.4).
4. Удалим значения переменных из ячеек **B2:H2**, полученные в результате решения задачи на первом этапе.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о диете методом уступок на втором этапе представлен на рис. 9.6.

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	***x1***	***x2***	***x3***	***x4***	***x5***	***x6***	***x7***	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**								0,000	0,000
3	**Калорийность:**	2060	2430	3600	890	140	230	650		
4	**Стоимость:**	12	100	160	24	40	30	80		
5		**Коэффициенты ограничений:**							Значения огр-ний	Сут. Потребности:
6	по белкам:	61	220	230	15	8	11	6	0,000	100
7	по жирам:	12	172	290	1	1	2	2	0,000	70
8	по углеводам:	420	0	0	212	26	38	155	0,000	400
9	**Дополнительное ограничение:**								0,000	2690
10										
11										
12										
13										
14										
15										

Задача о диете

Рис. 9.6. Исходные данные для решения задачи
об оптимальной диете методом уступок на втором этапе

После появления диалогового окна **Поиск решения** следует выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$J$2`.
2. Для группы **Равной:** оставить вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$B$2:$H$2`.
4. Добавить дополнительное ограничение (9.2.4). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$I$9**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения выбрать ячейку **$J$9**;
   - для добавления нового ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
5. В окне дополнительных параметров мастера поиска решения следует оставить выбранными отметки **Линейная модель** и **Неотрицательные значения**. Общий вид окна мастера поиска решения с заданными параметрами представлен на рис. 9.7.

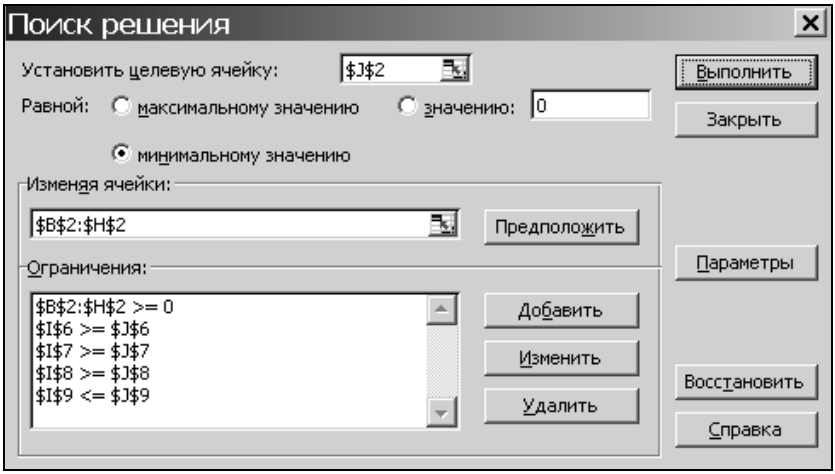


Рис. 9.7. Параметры мастера поиска решения и базовые ограничения второго этапа для двухкритериальной задачи об оптимальной диете

После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи о диете, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 9.8).

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**	0,952	0,000	0,202	0,000	0,000	0,000	0,000	2688,998	43,744
3	**Калорийность:**	2060	2430	3600	890	140	230	650		
4	**Стоимость:**	12	100	160	24	40	30	80		
5		**Коэффициенты ограничений:**							Значения огр-ний	Сут. Потребности:
6	по белкам:	61	220	230	15	8	11	6	104,548	100
7	по жирам:	12	172	290	1	1	2	2	70,000	70
8	по углеводам:	420	0	0	212	26	38	155	400,000	400
9	**Дополнительное ограничение:**								2688,998	2690

Задача о диете

Рис. 9.8. Результат окончательного решения двухкритериальной задачи об оптимальной диете методом уступок

Результатом решения задачи об оптимальной диете являются найденные оптимальные значения переменных: $x_1 \cong 0{,}952$, $x_2 = 0$, $x_3 \cong 0{,}202$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$, $x_7 = 0$, которым соответствуют значения целевых функций: $f_1^{opt} \cong 2688{,}998$ и $f_2^{opt} \cong 43{,}744$.

Анализ найденного решения показывает, что для удовлетворения суточной потребности в питательных веществах (белки, жиры, углеводы) следует использовать 952 *г* хлеба и 202 *г* сыра, совсем отказавшись от всего остального. При этом общая калорийность найденной оптимальной диеты будет приближенно равна 2689 *ккал*, а общая стоимость продуктов приближенно составит 43 *руб.* 75 *коп*.

Примечание

Интерпретируя полученные результаты, можно убедиться в том, что необходимо весьма осторожно использовать стоимостные критерии при решении задач оптимизации. Иначе придется ограничить себя диетой из сыра с хлебом. Заинтересованным читателям в качестве упражнения предлагается решить рассмотренную двухкритериальную задачу о диете методом уступок, изменив порядок важности критериев. Полученные результаты сравнить.

9.2.3. Решение двухкритериальной задачи о диете с помощью программы MS Excel методом минимального отклонения

Для решения задачи об оптимальной диете с двумя целевыми функциями с помощью программы MS Excel рассмотренным ранее методом минимального отклонения от идеальной точки будем использовать те же конкретные значения параметров рассматриваемой конкретной задачи о диете.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная оптимизация создадим новый рабочий лист с именем Задача о диете №2. Решение данной задачи многокритериальной оптимизации методом минимального отклонения от идеальной точки будет состоять из 2-х этапов. На первом этапе необходимо решить две обычных задачи оптимизации, используя в качестве критериальных функций, соответственно, целевые функции (9.2.1) и (9.2.2).

Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 9.2.2* при решении двухкритериальной задачи о диете методом уступок. Чтобы сократить дублирование действий, скопируем данные из листа с именем Задача о диете в лист с именем Задача о диете №2.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о диете методом минимального отклонения от идеальной точки на первом этапе должен быть аналогичным изображенному на рис. 9.3.

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения и задать соответствующие параметры. Поскольку последовательность действий по заданию параметров мастера поиска решения однокритериальной задачи о диете уже неоднократно рассматривались ранее, здесь она не приводится. Общий вид окна мастера поиска решения с заданными параметрами для решения задачи о диете по первому критерию представлен на рис. 9.4.

После задания ограничений и первой целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет вид, изображенный ранее на рис. 9.5. Результатом решения однокритериальной задачи об оптимальной диете на первом этапе является найденное значение первой целевой функции: $f_1^{opt} \cong 2587{,}140$.

Диалоговое окно задания параметров для мастера поиска решения задачи о диете по второму критерию представлено на рис. 9.9.

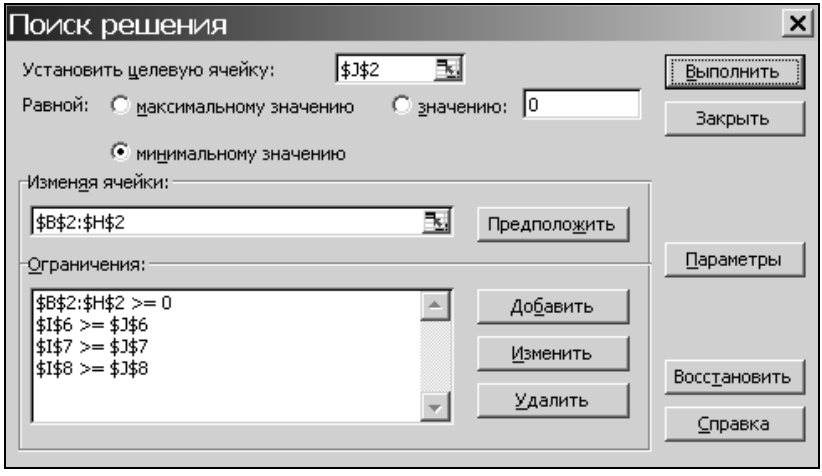


Рис. 9.9. Параметры мастера поиска решения и базовые ограничения для второго критерия двухкритериальной задачи о диете

После задания ограничений и второй целевой функции можно приступить к поиску численного решения по второму критерию, для этого следует предварительно удалить ранее найденные значения переменных и нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено соответствующее количественное решение. Результатом решения однокритериальной задачи об оптимальной диете на первом этапе является найденное значение второй целевой функции: $f_2^{opt} \cong 43{,}744$. Заметим, что это

значение соответствует оптимальному значению второй целевой функции, полученному при решении данной задачи о диете методом уступок.

Для окончательного решения двухкритериальной задачи о диете на втором этапе внесем следующие дополнительные данные в лист с именем Задача о диете №2:

1. Введем дополнительную надпись в ячейку **A9**.
2. В ячейку **I3** введем найденное на первом этапе оптимальное значение первой целевой функции $f_1^{opt} \cong 2587{,}14$.
3. В ячейку **J3** введем найденное на первом этапе оптимальное значение второй целевой функции $f_2^{opt} \cong 43{,}744$.
4. В ячейку **I3** введем формулу: `=(I2-I3)^2+(J2-J3)^2`, которая представляет минимальное отклонение от найденной идеальной точки, полученной в результате решения задачи на первом этапе.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о диете методом минимального отклонения от идеальной точки на втором этапе имеет следующий вид (рис. 9.10).

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	***x1***	***x2***	***x3***	***x4***	***x5***	***x6***	***x7***	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**								**0,000**	**0,000**
3	**Калорийность:**	2060	2430	3600	890	140	230	650	**2587,14**	**43,744**
4	**Стоимость:**	12	100	160	24	40	30	80		
5		**Коэффициенты ограничений:**							Значения огр-ний	Сут. Потребности:
6	по белкам:	61	220	230	15	8	11	6	0,000	100
7	по жирам:	12	172	290	1	1	2	2	0,000	70
8	по углеводам:	420	0	0	212	26	38	155	0,000	400
9	**Итоговая целевая функция:**								**6695206,917**	
10										
11										
12										
13										
14										
15										

Задача о диете №2

Рис. 9.10. Исходные данные для решения задачи об оптимальной диете методом минимального отклонения от идеальной точки на втором этапе

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$I$9`.

2. Для группы **Равной:** оставить вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$B$2:$H$2`.

В окне дополнительных параметров мастера поиска решения следует оставить выбранной отметку **Неотрицательные значения** и убрать отметку **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами представлен на рис. 9.11.

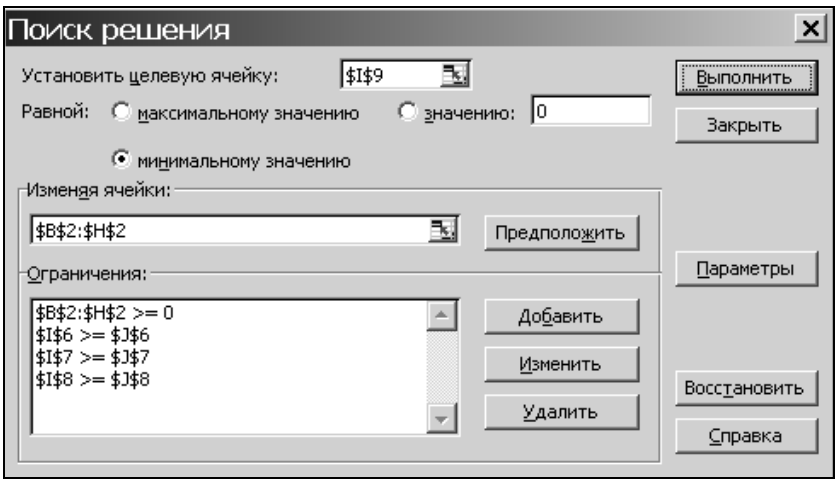


Рис. 9.11. Параметры мастера поиска решения и базовые ограничения для окончательного решения двухкритериальной задачи об оптимальной диете методом минимального отклонения от идеальной точки

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**	0,172	0,163	0,132	1,546	0,000	0,000	0,000	2602,650	76,627
3	**Калорийность:**	2060	2430	3600	890	140	230	650	**2587,14**	**43,744**
4	**Стоимость:**	12	100	160	24	40	30	80		
5		**Коэффициенты ограничений:**							Значения огр-ний	Сут. Потребности:
6	по белкам:	61	220	230	15	8	11	6	100,000	100
7	по жирам:	12	172	290	1	1	2	2	70,000	70
8	по углеводам:	420	0	0	212	26	38	155	400,000	400
9	**Итоговая целевая функция:**								**1321,829**	

Задача о диете №2

Рис. 9.12. Результат окончательного решения двухкритериальной задачи об оптимальной диете методом минимального отклонения от идеальной точки

После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи о диете, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 9.12).

Результатом решения задачи об оптимальной диете методом минимального отклонения от идеальной точки являются найденные оптимальные значения переменных: $x_1 \cong 0{,}172$, $x_2 \cong 0{,}163$, $x_3 \cong 0{,}132$, $x_4 \cong 1{,}546$, $x_5 = 0$, $x_6 = 0$, $x_7 = 0$, которым соответствуют значения целевых функций: $f_1^{opt} \cong 2602{,}02$ и $f_2^{opt} \cong 76{,}627$.

Анализ найденного решения показывает, что для удовлетворения суточной потребности в питательных веществах (белки, жиры, углеводы) следует использовать 172 *г* хлеба, 163 *г* мяса баранины, 132 *г* сыра и 1,546 *кг* бананов, совсем отказавшись от огурцов, помидоров и винограда. При этом общая калорийность найденной оптимальной диеты будет равна 2602,65 *ккал*, а общая стоимость продуктов приближенно составит 76 *руб.* 63 *коп*.

На примере решения данной задачи многокритериальной оптимизации следует обратить внимание на проблему соизмеримости абсолютных значений отдельных целевых функций и на проблему определения физической размерности итоговой целевой функции.

Причиной первой из проблем является различная субъективная значимость 1 единицы количественного значения различных целевых функций. Так, например, применительно к рассматриваемой задаче с точки зрения субъекта, планирующего выбор продуктов для своей диеты, 1 *ккал* рациона продуктов может быть вовсе не тождествен 1 *руб.* их стоимости. В то же время методы свертки целевых функций, разновидностью которых является метод минимального отклонения от идеальной точки, игнорируют данное обстоятельство, результатом этого может стать неадекватное решение той или иной задачи многокритериальной оптимизации.

Вторая проблема связана с тем, что если для исходных критериальных функций, как правило, известна их физическая размерность (например, *ккал*, *руб.*), то для итоговой целевой функции, которая представляет собой ту или иную свертку исходных, определить физическую размерность невозможно. Как результат — вопрос: "Что мы оптимизируем в итоге?" остается без ответа, если попытаться дать на него содержательный ответ.

Выходом из данной ситуации представляется усложнение и модификация методов свертки посредством перехода к некоторым безразмерным относительным критериальным функциям. Однако на этом пути встают новые проблемы, связанные со сложностью субъективных оценок безразмерного шкалирования отдельных целевых функций для лиц, принимающих решения.

Действительно, если один рацион продуктов отличается от другого на 5 *руб.*, то это вполне поддается оценке с точки зрения субъективного предпочтения. Напротив, если утверждать, что один рацион продуктов отличается от другого на 5% от диапазона абсолютного изменения их стоимости на множестве допустимых альтернатив, то эта информация вряд ли может служить достаточной для адекватной оценки с точки зрения субъективного предпочтения.

Несмотря на отмеченные недостатки методов свертки как универсального способа решения задач многокритериальной оптимизации и отдавая дань сложившейся традиции, далее приводится описание ставшего уже в некоторой степени "классическим" метода аддитивной свертки на основе задания весов отдельных целевых функций применительно для решения двухкритериальной задачи о диете.

9.2.4. Решение двухкритериальной задачи о диете с помощью программы MS Excel методом аддитивной свертки

Для решения задачи об оптимальной диете с двумя целевыми функциями с помощью программы MS Excel методом аддитивной свертки на основе задания весов отдельных целевых функций будем использовать те же конкретные значения параметров рассматриваемой ранее задачи о диете. Дополнительно следует задать количественные значения *весов* исходным целевым функциям, которые удовлетворяют условиям, отмеченным в *разд. 9.1.2*. Если предположить, что с точки зрения субъективных предпочтений лица, принимающего решение, первый критерий менее важен, чем второй, то такими весами могут быть, например, $\alpha_1 = 0{,}2$ и $\alpha_2 = 0{,}8$.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная оптимизация создадим новый рабочий лист с именем Задача о диете №3. Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 9.2.2* при решении двухкритериальной задачи о диете методом уступок. В то же время, чтобы сократить дублирование действий, скопируем данные из листа с именем Задача о диете №2 (рис. 9.10) в лист с именем Задача о диете №3.

Для решения двухкритериальной задачи о диете методом аддитивной свертки на основе задания весов отдельных целевых функций внесем следующие изменения в лист с именем Задача о диете №3:

1. Удалим данные из ячеек **I3** и **J3**, необходимость использования которых отпала.

2. В ячейку **I3** введем формулу: `=0,2*I2+0,8*J2`, которая представляет аддитивную свертку исходных целевых функций с использованием заданных ранее весов критериев.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о диете методом аддитивной свертки имеет следующий вид (рис. 9.13).

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	***x1***	***x2***	***x3***	***x4***	***x5***	***x6***	***x7***	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**								**0,000**	**0,000**
3	**Калорийность:**	2060	2430	3600	890	140	230	650		
4	**Стоимость:**	12	100	160	24	40	30	80		
5		**Коэффициенты ограничений:**							Значения огр-ний	Сут. Потребности:
6	по белкам:	61	220	230	15	8	11	6	0,000	100
7	по жирам:	12	172	290	1	1	2	2	0,000	70
8	по углеводам:	420	0	0	212	26	38	155	0,000	400
9	**Итоговая целевая функция:**								**0,000**	
10										
11										
12										
13										
14										
15										

Задача о диете №3

Рис. 9.13. Исходные данные для решения задачи об оптимальной диете методом аддитивной свертки

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**. В диалоговом окне **Поиск решения** следует задать целевую ячейку, диапазон изменения ячеек и ограничения задачи таким же образом, как было рассмотрено в *разд. 9.2.3* при решении задачи методом минимального отклонения от идеальной точки. В окне дополнительных параметров мастера поиска решения следует выбрать отметки **Неотрицательные значения** и **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами аналогичен окну параметров мастера поиска, представленному на рис. 9.11.

После задания ограничений и целевой функции на втором этапе можно приступить к поиску решения двухкритериальной задачи о диете, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 9.14).

Результатом решения задачи об оптимальной диете методом аддитивной свертки на основе задания весов отдельных целевых функций являются найденные оптимальные значения переменных: $x_1 \cong 0{,}757$, $x_2 = 0$, $x_3 \cong 0{,}209$, $x_4 \cong 0{,}386$, $x_5 = 0$, $x_6 = 0$, $x_7 = 0$, которым соответствуют значения целевых функций: $f_1^{opt} \cong 2655{,}393$ и $f_2^{opt} \cong 51{,}749$.

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**	**0,757**	**0,000**	**0,209**	**0,386**	**0,000**	**0,000**	**0,000**	**2655,393**	**51,749**
3	**Калорийность:**	2060	2430	3600	890	140	230	650		
4	**Стоимость:**	12	100	160	24	40	30	80		
5		**Коэффициенты ограничений:**							Значения огр-ний	Сут. Потребности:
6	по белкам:	61	220	230	15	8	11	6	100,000	100
7	по жирам:	12	172	290	1	1	2	2	70,000	70
8	по углеводам:	420	0	0	212	26	38	155	400,000	400
9	**Итоговая целевая функция:**								**572,478**	
10										
11										
12										
13										
14										
15										

Задача о диете №3

Рис. 9.14. Результат окончательного решения двухкритериальной задачи об оптимальной диете методом аддитивной свертки

Примечание

Анализ найденного решения и его содержательную интерпретацию предлагается сделать читателям самостоятельно в качестве упражнения. Хотелось бы только заметить, что в последнем случае составитель диеты в дополнение к сыру и хлебу может себя побаловать и бананами. В качестве упражнения предлагается также решить данную задачу методом аддитивной свертки на основе задания весов отдельных целевых функций для весов, например, $\alpha_1 = 0{,}8$ и $\alpha_2 = 0{,}2$, а полученные результаты сравнить.

9.3. Задача о производстве красок с двумя целевыми функциями

Задача о производстве красок с одной целевой функцией была рассмотрена ранее в *разд. 4.3*. Простота двухкритериального варианта этой задачи может послужить основой для иллюстрации графического способа нахождения множества недоминируемых альтернатив. В этом случае оказывается возможным не только наглядно представить форму множества допустимых альтернатив, но и построить множество Парето, которое является основой нахождения окончательного оптимального решения задач многокритериальной оптимизации. В то же время, говоря о простоте и наглядности графического способа построения множества Парето, следует отметить его ограниченный характер, поскольку он может быть использован только для задач линейного программирования с двумя переменными и произвольным числом ограничений.

9.3.1. Математическая постановка двухкритериальной задачи о производстве красок

Как было рассмотрено ранее в *разд. 4.3*, производственное предприятие выпускает два вида краски ($n = 2$), одна из которых предназначена для внутренних работ, а другая — для наружных. Для производства этих видов краски используются три типа исходных красителей и химических веществ ($m = 3$) — индиго, железный купорос и свежегашеная известь. На производство одной весовой единицы краски i-го вида ($i \in \{1, 2\}$) требуется a_{ij} единиц исходного красителя j-го вида ($j \in \{1, 2, 3\}$). Расход этих красителей для получения каждого вида краски приводится в табл. 4.2 *(см. главу 4)*. Запасы исходных красителей на складе предприятия ограничены следующими значениями: индиго $b_1 = 10$, железный купорос $b_2 = 7$, свежегашеная известь $b_3 = 5$.

В качестве стоимостного критерия рассмотрим две категории стоимости каждого вида краски:

- стоимость на внутреннем или национальном рынке для оптовых покупателей: $c_1 = 250$ и $c_2 = 230$, измеряемая, например, в *руб. за кг*;
- стоимость на внешнем или зарубежном рынке для оптовых покупателей: $d_1 = 13$ и $c_2 = 6$, измеряемая, например, в *у. е. за кг*.

Требуется разработать такой план выпуска красок для внутреннего и внешнего рынков, чтобы стоимость всей партии была максимальной.

Исходными переменными математической модели задачи о производстве красок являются: x_1 — объем выпуска первого вида краски и x_2 — объем выпуска второго вида краски. Тогда математическая постановка рассматриваемой индивидуальной двухкритериальной задачи о производстве красок может быть записана в следующем виде:

$$250x_1 + 230x_2 \to \max_{x \in \Delta_\beta}; \tag{9.3.1}$$

$$13x_1 + 6x_2 \to \max_{x \in \Delta_\beta}, \tag{9.3.2}$$

где множество допустимых альтернатив Δ_β, как и ранее, формируется следующей системой ограничений типа неравенств:

$$\begin{cases} 0,1x_1 + 0,2x_2 \le 10; \\ 0,2x_1 + 0,1x_2 \le 7; \\ 0,15x_1 + 0,05x_2 \le 5 \end{cases} \tag{9.3.3}$$

$$\text{и} \quad x_1, x_2 \ge 0.$$

9.3.2. Графический способ построения множества Парето для двухкритериальной задачи о производстве красок

Для графического решения задачи о производстве красок с помощью программы MS Excel в книге с именем Многокритериальная оптимизация создадим новый рабочий лист с именем Задача о красках. Далее необходимо построить на плоскости диаграмму, содержащую изображение области допустимых альтернатив, соответствующей системе ограничений (9.3.3) и целевых функций (9.3.1) и (9.3.2).

Для построения области допустимых альтернатив следует воспользоваться методикой построения, рассмотренной в *разд. 4.3.2*. Применительно к задаче о красках с двумя целевыми функциями (9.3.1)—(9.3.3) на одной диаграмме следует построить 5 графиков линейных функций. При этом в качестве исходных функций для построения области допустимых альтернатив необходимо использовать 3 следующих функции: $y_1 = 50 - 0{,}5 \cdot x_1$; $y_2 = 70 - 2 \cdot x_1$ и $y_3 = 100 - 3 \cdot x_1$, где в качестве зависимой переменной y с соответствующим индексом используется вторая переменная данной задачи. В качестве четвертой функции следует использовать первую целевую линейную функцию (9.3.1), а в качестве пятой — вторую целевую линейную функцию (9.3.2).

Для подготовки исходных данных с целью предварительного построения области допустимых альтернатив для исключения дублирования действий скопируем данные из листа с именем Задача о красках рабочей книги Линейное программирование (см. рис. 4.5) в лист с именем Задача о красках рабочей книги Многокритериальная оптимизация. Тем самым предполагаются выполненными действия 1—6, описанные при подготовке исходных данных для построения множества допустимых альтернатив в *разд. 4.3.2*.

После копирования данных следует выполнить такие действия:

1. Изменить текст в ячейке **A5** и ввести дополнительный текст в ячейку **A6**.
2. В ячейку **B6** ввести формулу: `=80-(13/6)*B1`, которая соответствует второй целевой функции.
3. Скопировать формулу из ячейки **B6** в ячейку **C6**, которая должна иметь вид: `=80-(13/6)*C1`.

Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 9.15).

Для построения графиков необходимо воспользоваться мастером диаграмм, который может быть вызван с помощью кнопки стандартной панели инструментов или операции главного меню: **Вставка** | **Диаграмма**.

МногокритериальнаяОптимизация.xls

	A	B	C
1	Значения переменной x_1:	0	34
2	Значения функции f_1:	=50-0,5*B1	=50-0,5*C1
3	Значения функции f2:	=70-2*B1	=70-2*C1
4	Значения функции f_3:	=100-3*B1	=100-3*C1
5	Значения целевой функции 1:	=60-(25/23)*B1	=60-(25/23)*C1
6	Значения целевой функции 2:	=80-(13/6)*B1	=80-(13/6)*C1
7			
8			
9			

Задача о красках

Рис. 9.15. Исходные данные для построения графиков функций линейных ограничений и целевых функций

На первом шаге построения диаграммы необходимо выбрать тип диаграммы и ее вид. С этой целью следует выделить на вкладке **Стандартные** в левом списке тип диаграммы **График**, а в правом списке с графическими миниатюрами — первый вид графика, который отражает развитие процесса во времени или по категориям. После выбора типа и разновидности диаграммы следует нажать кнопку **Далее** и перейти ко второму шагу построения диаграммы с помощью мастера диаграмм.

На втором шаге построения диаграммы необходимо выбрать ячейки с данными, которые должны быть отображены на четырех графиках. Применительно к рассматриваемому примеру — это значения функций, которые содержатся в диапазоне ячеек **B2:C6**. Для указания этих значений следует установить переключатель **Ряды в:** в положение — **строках**. После этого нажать кнопку **Диапазон**, расположенную в правой части поля ввода, и выделить диапазон ячеек **B2:C6**. Далее на этом же шаге работы мастера диаграмм следует задать подписи по горизонтальной оси. С этой целью необходимо перейти на вкладку **Ряд** и выполнить щелчок кнопкой, расположенной в правой части поля ввода с именем **Подписи оси *X***. Для указания соответствующего источника данных следует на рабочем листе с помощью мыши или клавиатуры выделить диапазон значений функции `B1:C1`.

После редактирования свойств диаграммы на третьем и четвертом шагах работы мастера будет построена диаграмма, содержащая графики пяти линейных функций следующего вида (рис. 9.16).

Область допустимых альтернатив образуется пересечением пяти полуплоскостей, каждая из которых соответствует отдельному ограничению (9.3.3),

включая и ограничения неотрицательности переменных задачи. Таким образом, область допустимых альтернатив задачи о красках представляет собой выпуклый многогранник на плоскости. Каждая точка из этой области, которая на рис. 9.16 изображена серым цветом, является некоторым допустимым решением рассматриваемой задачи о красках.

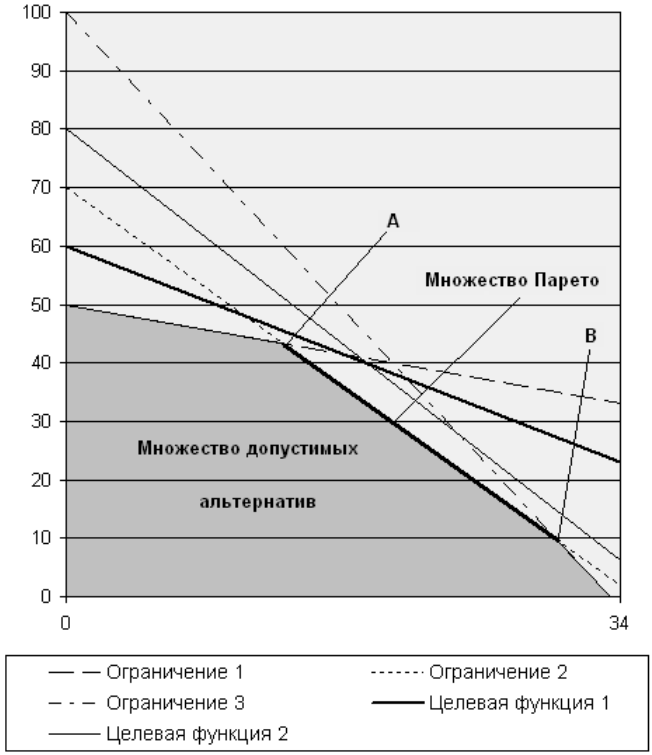


Рис. 9.16. Результат построения множества Парето для двухкритериальной задачи о красках

Множество недоминируемых альтернатив или множество Парето для рассматриваемой задачи представляет собой отрезок прямой линии, соответствующей уравнению: $y_2 = 70 - 2 \cdot x_1$ и лежащей между точками A и B. При этом первая точка A, соответствующая максимуму первой целевой функции (9.3.1), находится в точке пересечения первого и второго ограничений. Вторая точка B, соответствующая максимуму второй целевой функции (9.3.2), находится в точке пересечения второго и третьего ограничений.

Для нахождения значений переменных в точках A и B, а также значений целевых функций в этих точках необходимо найти координаты точек пересече-

ния соответствующих прямых. С этой целью необходимо решить 2 системы линейных уравнений следующего вида:

$$\begin{cases} 0{,}1x_1 + 0{,}2x_2 = 10; \\ 0{,}2x_1 + 0{,}1x_2 = 7, \end{cases} \tag{9.3.4}$$

$$\begin{cases} 0{,}2x_1 + 0{,}1x_2 = 7; \\ 0{,}15x_1 + 0{,}05x_2 = 5. \end{cases} \tag{9.3.5}$$

Для нахождения количественных значений координат точек A и B следует выполнить действия, аналогичные действиям 1—4, рассмотренным в *разд. 4.3.2*. После этого дважды воспользоваться встроенным инструментом **Подбор параметра**. Этот инструмент может быть вызван с помощью операции главного меню: **Сервис | Подбор параметра**. После задания необходимых значений свойств мастера подбора параметра и выполнения расчетов будут найдены значения переменных в точках A и B. Эти значения для точки A будут содержаться в ячейках **B7** и **C7**: $x_1 = 13{,}333$ и $x_2 = 43{,}333$, а для точки B — в ячейках **B10** и **C10**: $x_1 = 30$ и $x_2 = 10$.

МногокритериальнаяОптимизация.xls

	A	B	C	D
1	Значения переменной x_1:	0	34	
2	Значения функции f_1:	50	33	
3	Значения функции f2:	70	2	
4	Значения функции f_3:	100	-2	
5	Значения целевой функции 1:	60	23,043	
6	Значения целевой функции 2:	80	6,333	
7	Значения переменных в точке A:	13,333	43,333	
8	Значение разности функций:	0		
9	Значение целевых функций в точке A:	13300	433,333	
10	Значения переменных в точке B:	30	10	
11	Значение разности функций:	0		
12	Значение целевых функции в точке B:	9800	450	
13				
14				
15				
16				
17				
18				

Задача о краск

Рис. 9.17. Результат нахождения переменных точек A и B из множества Парето для двухкритериальной задачи о красках с помощью мастера подбора параметра

Соответствующие значения целевых функций для точки A содержатся в ячейках **B9** и **C9**: $f_1^A = 13300$ и $f_2^A = 433{,}333$, а для точки B — содержатся в ячейках **B11** и **C11**: $f_1^B = 9800$ и $f_2^B = 450$ (рис. 9.17).

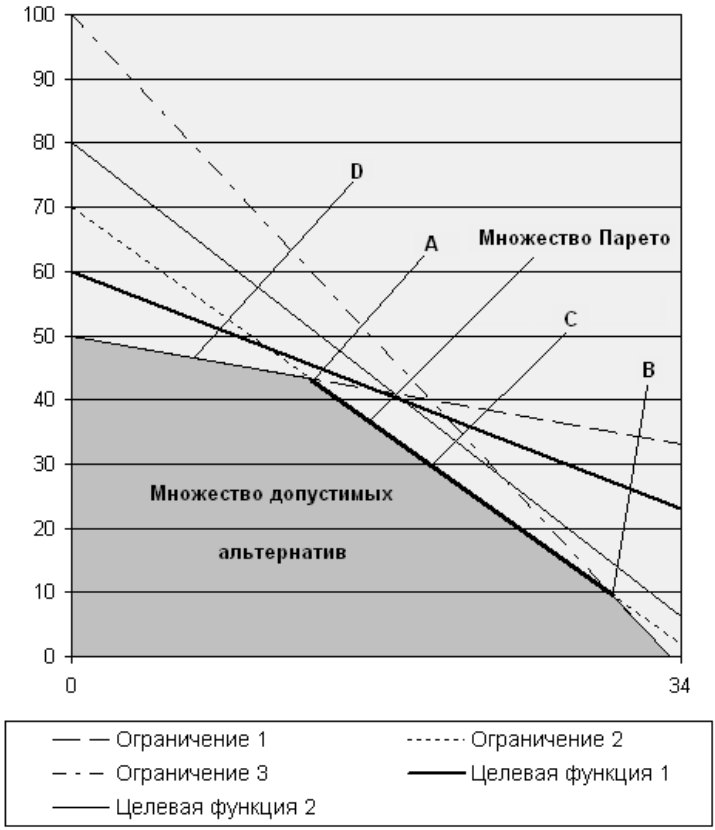


Рис. 9.18. Множество Парето для двухкритериальной задачи о красках и дополнительные допустимые альтернативы

Анализ полученных результатов показывает, что точки A и B действительно являются недоминируемыми, поскольку если для первой целевой функции выполняется условие: $f_1^A > f_1^B$, то для второй целевой функции выполняется обратное неравенство: $f_2^A < f_2^B$. Более того, все точки, лежащие на отрезке AB, будут удовлетворять аналогичному свойству. В этом можно убедиться непосредственной проверкой. Так, например, для некоторой промежуточной точки C (рис. 9.18) с координатами: $x_1 = 20$ и $x_2 = 30$ соответствующие значе-

ния целевых функций равны: $f_1^C = 11\,900$ и $f_2^C = 440$. Сравнивая значения целевых функций для точек A и C, получаем: $f_1^A > f_1^C$, но $f_2^A < f_2^C$. Сравнивая значения целевых функций для точек B и C, получаем: $f_1^B < f_1^C$, но $f_2^B > f_2^C$. Тем самым оказывается справедливым утверждение о том, что все точки отрезка AB являются недоминируемыми альтернативами.

Проверим, что никаких других точек, кроме отрезка AB, множество Парето не содержит. С этой целью возьмем для непосредственной проверки, например, допустимую альтернативу со значениями переменных: $x_1 = 10$ и $x_2 = 45$. Точка D (рис. 9.18) с этими координатами лежит на прямой, соответствующей первому ограничению рассматриваемой задачи. Для точки D значения целевых функций равны: $f_1^D = 12\,850$ и $f_2^D = 400$. Нетрудно проверить, что $f_1^A > f_1^D$ и одновременно $f_2^A > f_2^D$. Поскольку альтернатива A доминирует по Парето альтернативу D, то последняя не входит в множество недоминируемых альтернатив.

Примечание

Более строгое доказательство свойств множеств Парето для задач линейного программирования можно найти в специальной литературе, список которой приведен в конце книги. Заинтересованные читатели в качестве упражнения могут решить рассмотренную двухкритериальную задачу о красках с помощью метода уступок и метода минимального отклонения от идеальной точки.

Результатом построения множества Парето в общем случае является подготовка материала для лица, принимающего решение, с целью окончательного выбора решения задачи многокритериальной оптимизации. Действительно, окончательным решением рассмотренной двухкритериальной задачи о красках может быть только такая альтернатива, которая принадлежит множеству Парето или графически расположена на отрезке AB. В противном случае любое другое решение заведомо не будет наилучшим, поскольку будет доминируемым одной из альтернатив множества Парето. Это свойство имеет место и в общем случае, что делает актуальным построение и анализ множества Парето в задачах и многокритериальной оптимизации, и других классов.

В то же время для выбора окончательного решения из множества Парето необходима дополнительная информация о предпочтениях лиц, принимающих решение. Соответствующие аспекты решения задач многокритериальной оптимизации образуют специальную область научно-прикладных исследований, которая выходит за рамки тематики настоящей книги. Следует лишь отметить, что методы и алгоритмы построения предпочтений лиц, прини-

мающих решения, для некоторых классов задач многокритериальной оптимизации реализованы в специальных программных инструментариях, ориентированных, как правило, на решение узкого класса прикладных задач.

9.4. Двухкритериальная задача о рюкзаке

Содержательная постановка задачи о рюкзаке приводится в *разд. 1.2.7*, а способы ее решения — в *разд. 5.2*. В настоящей главе рассматривается вариант постановки и решения двухкритериальной задачи о рюкзаке методом уступок и методом минимального отклонения от идеальной точки. Заметим, что этот вариант задачи многокритериальной оптимизации несколько отличается по содержанию, чем рассмотренная ранее одномерная задача о рюкзаке.

9.4.1. Математическая постановка двухкритериальной задачи о рюкзаке

В общем случае переносимый в рюкзаке груз может включать n видов предметов, при этом каждый предмет вида $i \in \{1, 2, \ldots, n\}$ обладает некоторой массой c_i, например, в *кг*, и объемом d_i, например, в $дм^3$ или *литрах*. Для каждого вида предмета турист, исходя из своих субъективных предпочтений, определяет его индивидуальную ценность a_i во время перехода. Ценность может быть определена с точки зрения калорийности, если в качестве предметов берутся продукты питания. При этом общая ценность всех предметов не должна быть ниже некоторой заданной величины, например, b.

Требуется определить количество предметов каждого вида, которые следует положить в рюкзак, чтобы их общая масса и объем были минимальными, а ценность была бы не ниже фиксированного значения.

Для решения задачи о рюкзаке с помощью программы MS Excel необходимо задать конкретные значения параметрам исходной задачи. Для определенности предположим, что в качестве исходных видов предметов рассматриваются продукты питания: хлеб, сухари, тушеная говядина, лосось в масле, сыр, сушеные бананы, сахар рафинад ($n = 7$), а в качестве их характеристик — объем и масса каждого продукта. Субъективная ценность для туриста *одного экземпляра* каждого из продуктов следующая: $a_1 = 20$, $a_2 = 30$, $a_3 = 50$, $a_4 = 30$, $a_5 = 40$, $a_6 = 20$, $a_7 = 70$. Известна масса одного экземпляра каждого из продуктов: $c_1 = 0{,}8$ *кг*, $c_2 = 0{,}4$ *кг*, $c_3 = 0{,}3$ *кг*, $c_4 = 0{,}2$ *кг*, $c_5 = 0{,}5$ *кг*, $c_6 = 0{,}4$ *кг*, $c_7 = 0{,}6$ *кг* и объем одного экземпляра каждого из продуктов: $d_1 = 1{,}2$ *л*, $d_2 = 1{,}5$ *л*, $d_3 = 0{,}4$ *л*, $d_4 = 0{,}3$ *л*, $d_5 = 0{,}8$ *л*, $d_6 = 0{,}9$ *л*, $d_7 = 0{,}5$ *л*.

Предполагается, что общая ценность продуктов в рюкзаке не должна быть ниже величины 2000.

Исходными переменными математической модели одномерной задачи о рюкзаке являются: x_i — количество экземпляров продуктов i-го вида, которые следует положить в рюкзак. Тогда математическая постановка рассматриваемой индивидуальной задачи о рюкзаке может быть записана в следующем виде.

$$0,8x_1 + 0,4x_2 + 0,3x_3 + 0,2x_4 + 0,5x_5 + 0,4x_6 + 0,6x_7 \to \min_{x\in\Delta_\beta}, \quad (9.4.1)$$

$$1,2x_1 + 1,5x_2 + 0,4x_3 + 0,3x_4 + 0,8x_5 + 0,9x_6 + 0,5x_7 \to \min_{x\in\Delta_\beta}, \quad (9.4.2)$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} 20x_1 + 30x_2 + 50x_3 + 30x_4 + 40x_5 + 20x_6 + 70x_7 \geq 2000 \\ x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7 \geq 0 \\ x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7 - \text{целые числа.} \end{cases} \quad (9.4.3)$$

Данная задача относится к классу задач многокритериального целочисленного линейного программирования, которая может быть решена с помощью рассмотренных ранее методов уступок и минимального отклонения от идеальной точки. При этом общая схема данных методов останется прежней, изменятся лишь особенности поиска решений соответствующих однокритериальных задач оптимизации.

9.4.2. Решение двухкритериальной задачи о рюкзаке с помощью программы MS Excel методом уступок

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная оптимизация создадим новый рабочий лист с именем Задача о рюкзаке. Решение данной задачи многокритериальной оптимизации методом уступок будет состоять из 2-х этапов. На первом этапе необходимо решить обычную задачу оптимизации, используя в качестве критериальной функции первую целевую функцию (9.4.1) в предположении, что она является более важной, чем вторая целевая функция. Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *главе 5* при решении однокритериальной задачи о рюкзаке:

1. Внесем необходимые надписи в ячейки **A1:J1**, **A2:A6**, **B5**, **I5**, **J5**. Следует отметить, что конкретное содержание этих надписей не оказывает

никакого влияния на решение рассматриваемой задачи линейного программирования.

2. В ячейки **B3:H3** введем значения коэффициентов первой целевой функции: $c_1 = 0{,}8$, $c_2 = 0{,}4$, $c_3 = 0{,}3$, $c_4 = 0{,}2$, $c_5 = 0{,}5$, $c_6 = 0{,}4$, $c_7 = 0{,}6$.
3. В ячейки **B4:H4** введем значения коэффициентов второй целевой функции: $d_1 = 1{,}2$, $d_2 = 1{,}5$, $d_3 = 0{,}4$, $d_4 = 0{,}3$, $d_5 = 0{,}8$, $d_6 = 0{,}9$, $d_7 = 0{,}5$.
4. В ячейку **I2** введем формулу: `=СУММПРОИЗВ($B$2:$H$2;$B3:$H3)`, которая представляет первую целевую функцию (9.4.1).
5. В ячейку **J2** введем формулу: `=СУММПРОИЗВ($B$2:$H$2;$B4:$H4)`, которая представляет вторую целевую функцию (9.4.2).
   - В ячейки **B6:H6** введем значения коэффициентов ограничения (9.4.3): $a_1 = 20$, $a_2 = 30$, $a_3 = 50$, $a_4 = 30$, $a_5 = 40$, $a_6 = 20$, $a_7 = 70$.
   - В ячейку **I6** введем формулу: `=СУММПРОИЗВ($B$2:$H$2; B6:H6)`, которая представляет левую часть ограничения (9.4.3).
   - В ячейку **J6** введем значение правой части ограничения, соответствующего максимальному объему продуктов в рюкзаке: $b = 2000$.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения двухкритериальной задачи о рюкзаке методом уступок на первом этапе имеет следующий вид (рис. 9.19).

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**								=СУММПРОИЗВ($B$2:$H$2;$B3:$H3)	=СУММПРОИЗВ($B$2:$H$2;$B$4:H
3	**Масса:**	0,8	0,4	0,3	0,2	0,5	0,4	0,6		
4	**Объем:**	1,2	1,5	0,4	0,3	0,8	0,9	0,5		
5		**Коэффициенты ограничений:**							Значения огр-ний:	Общая ценность:
6	по ценности:	20	30	50	30	40	20	70	=СУММПРОИЗВ($B$2:$H$2;B6:H6)	2000
7	**Дополнительное ограничение:**								**=СУММПРОИЗВ($B$2:$H$2;$B3:$H3)**	15
8										
9										
10										
11										
12										
13										
14										

Задача о рюкзаке

Рис. 9.19. Исходные данные для решения двухкритериальной задачи о рюкзаке методом уступок на первом этапе

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$I$2`.

2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$H$2`.
4. Задать первое ограничение для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$I$6**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";
   - в качестве значения правой части ограничения выбрать ячейку **$J$6**;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Задать ограничение на неотрицательность значений переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$H$2**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** ввести значение 0;
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
6. Задать ограничение на целочисленность значений переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$H$2**, который должен отобразиться в поле с именем **Ссылка на ячейку**;

- в качестве знака ограничения из выпадающего списка выбрать строку "**цел**";
- в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**целое**";
- для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

В окне дополнительных параметров поиска решения следует выбрать отметки **Линейная модель** и **Неотрицательные значения**. Общий вид окна мастера поиска решения на первом этапе с заданными параметрами представлен на рис. 9.20.

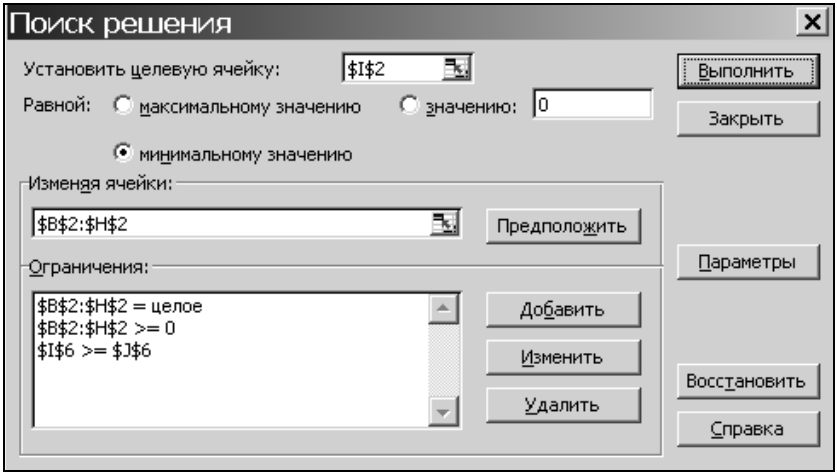


Рис. 9.20. Параметры мастера поиска решения и базовые ограничения первого этапа для двухкритериальной задачи о рюкзаке

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K
1	**Переменные:**	***x1***	***x2***	***x3***	***x4***	***x5***	***x6***	***x7***	**Значение ЦФ 1:**	**Значение ЦФ 2:**	
2	**Значения:**	**0**	**0**	**40**	**0**	**0**	**0**	**0**	**12,000**	**16,000**	
3	**Масса:**	0,8	0,4	0,3	0,2	0,5	0,4	0,6			
4	**Объем:**	1,2	1,5	0,4	0,3	0,8	0,9	0,5			
5		**Коэффициенты ограничений:**							Значения огр-ний:	Общая ценность:	
6	по ценности:	20	30	50	30	40	20	70	2000,000	2000	
7											
8											
9											
10											
11											

Задача о рюкзаке

Рис. 9.21. Результат количественного решения двухкритериальной задачи о рюкзаке на первом этапе

После задания ограничений и целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 9.21).

Результатом решения двухкритериальной задачи о рюкзаке на первом этапе является найденное значение первой целевой функции: $f_1^{opt} = 12$. Анализ найденного решения показывает, что общая масса груза будет равна 12 *кг*. Ценность этого груза, состоящего из 40 банок тушеной говядины, в точности составит 2000.

Предположим, что, исходя из субъективных предположений туриста, оказывается возможной уступка по первой целевой функции, равная 3 *кг*. Тем самым можно перейти ко второму этапу решения данной задачи оптимизации методом уступок.

С этой целью следует рассмотреть дополнительное ограничение следующего вида:

$$0{,}8x_1 + 0{,}4x_2 + 0{,}3x_3 + 0{,}2x_4 + 0{,}5x_5 + 0{,}4x_6 + 0{,}6x_7 \le 15\,, \qquad (9.4.4)$$

в котором правая часть равна значению суммы: $f_1^{opt} + 3$, а знак ограничения выбран таким образом, чтобы соответствовать решенной задачи минимизации по первой целевой функции.

Для окончательного решения двухкритериальной задачи о диете внесем следующие дополнительные данные в лист с именем Задача о рюкзаке:

1. Введем дополнительную надпись в ячейку **A7**.
2. Скопируем формулу из ячейки **I2** в ячейку **I7**, которая представляет левую часть дополнительного ограничения (9.4.4).
3. В ячейку **J7** введем значение 15, которое представляет правую часть дополнительного ограничения (9.4.4).
4. Удалим значения переменных из ячеек **B2:H2**, полученные в результате решения задачи на первом этапе.

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**. После появления диалогового окна **Поиск решения** следует выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$J$2`.
2. Для группы **Равной:** оставить вариант поиска решения — **минимальному значению**.

3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$B$2:$H$2`.
4. Задать дополнительное ограничение (9.4.4). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$I$7**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения выбрать ячейку **$J$7**;
   - для добавления нового ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

В окне дополнительных параметров мастера поиска решения следует оставить выбранными отметки **Линейная модель** и **Неотрицательные значения**.

После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи о рюкзаке, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 9.22).

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**	
2	**Значения:**	**0**	**0**	**17**	**1**	**0**	**0**	**16**	**14,900**	**15,100**	
3	**Масса:**	0,8	0,4	0,3	0,2	0,5	0,4	0,6			
4	**Объем:**	1,2	1,5	0,4	0,3	0,8	0,9	0,5			
5		**Коэффициенты ограничений:**							Значения огр-ний:	Общая ценность:	
6	по ценности:	20	30	50	30	40	20	70	2000,000	2000	
7	**Дополнительное ограничение:**								**14,900**	15	
8											
9											
10											
11											
12											

Задача о рюкзаке

Рис. 9.22. Результат окончательного решения двухкритериальной задачи о рюкзаке методом уступок

Результатом решения двухкритериальной задачи о рюкзаке являются найденные оптимальные значения переменных: $x_1 = 0$, $x_2 = 0$, $x_3 = 17$, $x_4 = 1$, $x_5 = 0$, $x_6 = 6$, $x_7 = 16$, которым соответствуют значения целевых функций: $f_1^{opt} = 14{,}9$ и $f_2^{opt} = 15{,}1$.

Анализ найденного решения показывает, что в поход следует взять 17 банок тушеной говядины, 1 банку лосося в масле и 16 пачек сахара рафинада, совсем отказавшись от всего остального. При этом общая ценность груза составит 2000, общая масса продуктов — 14,9 *кг*, а общий объем — 15,1 *л*.

Примечание

Заинтересованным читателям в качестве упражнения предлагается решить рассмотренную двухкритериальную задачу о рюкзаке методом уступок, изменив порядок важности критериев. Полученные результаты сравнить.

9.4.3. Решение двухкритериальной задачи о рюкзаке с помощью программы MS Excel методом минимального отклонения

Для решения задачи о рюкзаке с двумя целевыми функциями с помощью программы MS Excel методом минимального отклонения от идеальной точки будем использовать те же конкретные значения параметров рассматриваемой конкретной задачи о диете.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная оптимизация создадим новый рабочий лист с именем Задача о рюкзаке №2. Решение данной задачи многокритериальной оптимизации методом минимального отклонения от идеальной точки будет состоять из 2-х этапов. На первом этапе необходимо решить две обычных задачи оптимизации, используя в качестве критериальных функций, соответственно, целевые функции (9.4.1) и (9.4.2).

Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 9.4.2*, при решении двухкритериальной задачи о рюкзаке методом уступок. Чтобы исключить дублирование действий, скопируем данные из листа с именем Задача о рюкзаке в лист с именем Задача о рюкзаке №2.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными, необходимыми для решения задачи о рюкзаке методом минимального отклонения от идеальной точки, на первом этапе должен иметь вид, аналогичный изображенному на рис. 9.19.

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения и задать соответствующие параметры. Поскольку последовательность действий по заданию параметров поиска решения однокритериальной задачи о рюкзаке уже неоднократно рассматривались ранее, здесь она

не приводится. Общий вид окна мастера поиска решения с заданными параметрами для решения задачи о рюкзаке по первому критерию представлен на рис. 9.20.

После задания ограничений и первой целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет вид, изображенный ранее на рис. 9.21. Результатом решения однокритериальной задачи о рюкзаке на первом этапе являются найденное значение первой целевой функции: $f_1^{opt} = 12$.

Диалоговое окно задания параметров мастера поиска решения задачи о рюкзаке по второму критерию представлено на рис. 9.23.

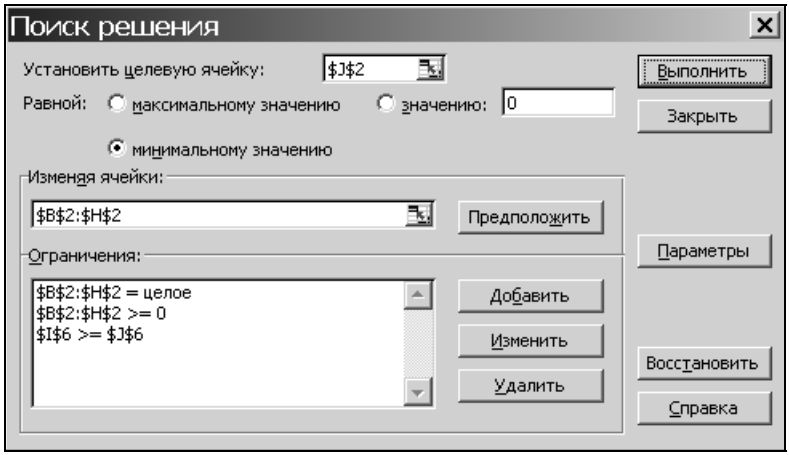


Рис. 9.23. Параметры мастера поиска решения и базовые ограничения для второго критерия двухкритериальной задачи о рюкзаке

После задания ограничений и второй целевой функции можно приступить к поиску численного решения по второму критерию, для этого следует предварительно удалить ранее найденные значения переменных и нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено соответствующее количественное решение. Результатом решения однокритериальной задачи о рюкзаке на первом этапе является найденное значение второй целевой функции: $f_2^{opt} = 14{,}5$.

Для окончательного решения двухкритериальной задачи о рюкзаке на втором этапе внесем следующие дополнительные данные в лист с именем Задача о рюкзаке №2:

1. Введем дополнительную надпись в ячейку **A7**.

2. В ячейку **I3** введем найденное на первом этапе оптимальное значение первой целевой функции $f_1^{opt} = 12$.

3. В ячейку **J3** введем найденное на первом этапе оптимальное значение второй целевой функции $f_2^{opt} = 14{,}5$.

4. В ячейку **I3** введем формулу: `=(I2-I3)^2+(J2-J3)^2`, которая представляет минимальное отклонение от найденной идеальной точки, полученной в результате решения задачи на первом этапе.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о рюкзаке методом минимального отклонения от идеальной точки на втором этапе имеет следующий вид (рис. 9.24).

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**								0,000	0,000
3	**Масса:**	0,8	0,4	0,3	0,2	0,5	0,4	0,6	**12**	**14,5**
4	**Объем:**	1,2	1,5	0,4	0,3	0,8	0,9	0,5		
5		**Коэффициенты ограничений:**							Значения огр-ний:	Общая ценность:
6	по ценности:	20	30	50	30	40	20	70	0,000	2000
7	**Итоговая целевая функция:**								**354,250**	
8										
9										
10										
11										
12										

Задача о рюкзаке №2

Рис. 9.24. Исходные данные для решения задачи о рюкзаке методом минимального отклонения от идеальной точки на втором этапе

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$I$7`.

2. Для группы **Равной:** оставить вариант поиска решения — **минимальному значению**.

3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$B$2:$H$2`.

4. В окне дополнительных параметров мастера поиска решения следует оставить выбранной отметку **Неотрицательные значения** и обязательно убрать отметку **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами представлен на рис. 9.25.

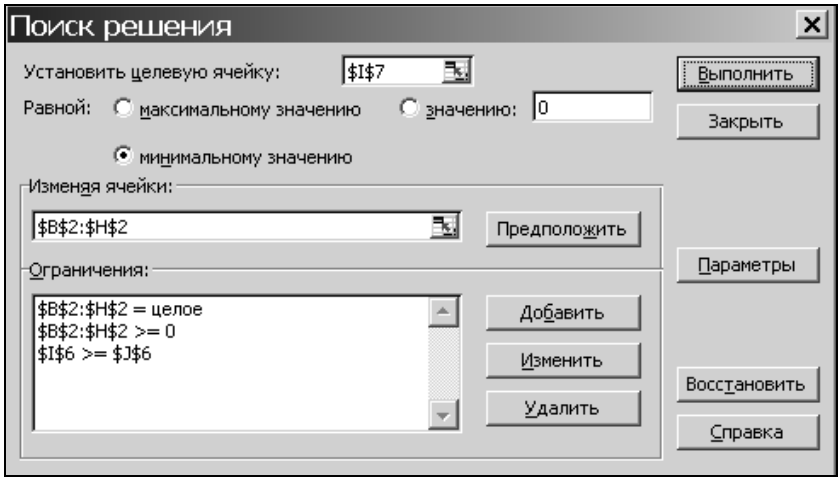


Рис. 9.25. Параметры мастера поиска решения и базовые ограничения для окончательного решения двухкритериальной задачи о рюкзаке методом минимального отклонения от идеальной точки

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**	**0**	**0**	**33**	**0**	**0**	**0**	**5**	**12,900**	**15,700**
3	**Масса:**	0,8	0,4	0,3	0,2	0,5	0,4	0,6	**12**	**14,5**
4	**Объем:**	1,2	1,5	0,4	0,3	0,8	0,9	0,5		
5		**Коэффициенты ограничений:**							Значения огр-ний:	Общая ценность:
6	по ценности:	20	30	50	30	40	20	70	2000,000	2000
7	**Итоговая целевая функция:**								**2,250**	

Задача о рюкзаке №2

Рис. 9.26. Результат окончательного решения двухкритериальной задачи о рюкзаке методом минимального отклонения от идеальной точки

После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи о рюкзаке, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 9.26).

Результатом решения двухкритериальной задачи о рюкзаке методом минимального отклонения от идеальной точки являются найденные оптимальные значения переменных: $x_1 = 0$, $x_2 = 0$, $x_3 = 33$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$, $x_7 = 5$, которым соответствуют значения целевых функций: $f_1^{opt} = 12{,}9$ и $f_2^{opt} = 15{,}7$.

Анализ найденного решения показывает, что с точки зрения заданных критерием массы и объема, туристу в поход следует взять 33 *банки* тушеной говядины и 5 *пачек* сахара рафинада. В этом случае содержательный анализ результатов решения двухкритериальной задачи о рюкзаке выявляет недостаток метода минимального отклонения от идеальной точки, связанный с формированием более однообразного состава продуктов, которые следует взять в поход. В этом контексте метод уступок дает более разнообразный набор, в котором дополнительно появляется 1 *банка* лосося в масле.

Несмотря на указанные ранее недостатки методов свертки как универсальных способов решения задач многокритериальной оптимизации, далее приводится описание метода аддитивной свертки на основе задания весов отдельных целевых функций применительно к решению двухкритериальной задачи о рюкзаке.

9.4.4. Решение двухкритериальной задачи о рюкзаке с помощью программы MS Excel методом аддитивной свертки

Для решения двухкритериальной задачи о рюкзаке с помощью программы MS Excel методом аддитивной свертки на основе задания весов отдельных целевых функций будем использовать те же значения параметров рассмотренной ранее конкретной задачи о рюкзаке. Дополнительно следует задать количественные значения *весов* исходным целевым функциям, которые удовлетворяют условиям, отмеченным в *разд. 9.1.2*. Если предположить, что с точки зрения субъективных предпочтений лица, принимающего решение, первый критерий менее важен, чем второй, то такими весами могут быть, например, $\alpha_1 = 0{,}2$ и $\alpha_2 = 0{,}8$.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная оптимизация создадим новый рабочий лист с именем Задача о рюкзаке №3. Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 9.4.2* при решении двухкритериальной задачи о рюкзаке методом уступок. В то же время, чтобы сократить дублирование действий, скопируем данные из листа с именем Задача о рюкзаке №2 (рис. 9.24) в лист с именем Задача о рюкзаке №3.

Для решения двухкритериальной задачи о рюкзаке методом аддитивной свертки на основе задания весов отдельных целевых функций внесем следующие изменения в лист с именем Задача о рюкзаке №3:

1. Удалим данные из ячеек **I3** и **J3**, необходимость использования которых отпала.

2. В ячейку **I3** введем формулу: `=0,2*I2+0,8*J2`, которая представляет аддитивную свертку исходных целевых функций с использованием заданных ранее весов критериев.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о рюкзаке методом аддитивной свертки имеет следующий вид (рис. 9.27).

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**								**=СУММПРОИЗВ($B$2:$H$2;$B$3:$H$3)**	**=СУММПРОИЗВ($B$2:$H$2;$B$4:$H$4)**
3	**Масса:**	0,8	0,4	0,3	0,2	0,5	0,4	0,6		
4	**Объем:**	1,2	1,5	0,4	0,3	0,8	0,9	0,5		
5		**Коэффициенты ограничений:**							Значения огр-ний:	Общая ценность:
6	по ценности:	20	30	50	30	40	20	70	=СУММПРОИЗВ($B$2:$H$2;B6:H6)	2000
7	**Итоговая целевая функция:**								**=0,2*I2+0,8*J2**	
8										
9										
10										
11										
12										

Задача о рюкзаке №3

Рис. 9.27. Исходные данные для решения двухкритериальной задачи о рюкзаке методом аддитивной свертки

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**. В диалоговом окне **Поиск решения** следует задать целевую ячейку, диапазон изменения ячеек и ограничения задачи таким же образом, как было рассмотрено в *разд. 9.4.3* при решении задачи методом минимального отклонения от идеальной точки. В окне дополнительных параметров мастера поиска решения следует выбрать отметки **Неотрицательные значения** и **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами аналогичен окну параметров мастера поиска, представленному на рис. 9.25.

После задания ограничений и целевой функции на втором этапе можно приступить к поиску решения двухкритериальной задачи о рюкзаке, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 9.28).

Результатом решения двухкритериальной задачи о рюкзаке методом аддитивной свертки на основе задания весов отдельных целевых функций являются найденные оптимальные значения переменных: $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$, $x_7 = 29$, которым соответствуют значения целевых функций: $f_1^{opt} = 17{,}4$ и $f_2^{opt} = 14{,}5$.

МногокритериальнаяОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	**Переменные:**	*x1*	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	**Значение ЦФ 1:**	**Значение ЦФ 2:**
2	**Значения:**	**0**	**0**	**0**	**0**	**0**	**0**	**29**	**17,400**	**14,500**
3	**Масса:**	0,8	0,4	0,3	0,2	0,5	0,4	0,6		
4	**Объем:**	1,2	1,5	0,4	0,3	0,8	0,9	0,5		
5		**Коэффициенты ограничений:**							Значения огр-ний:	Общая ценность:
6	по ценности:	20	30	50	30	40	20	70	2030,000	2000
7	**Итоговая целевая функция:**								**15,080**	
8										
9										
10										
11										
12										

Задача о рюкзаке №3

Рис. 9.28. Результат окончательного решения двухкритериальной задачи о рюкзаке методом аддитивной свертки

Анализ найденного решения показывает, что оно в точности соответствует оптимальному решению по второй целевой функции. В этом случае содержательный анализ результатов решения двухкритериальной задачи о рюкзаке выявляет недостаток метода аддитивной свертки, особенно явно проявляющийся при решении задач целочисленного линейного программирования.

Примечание

При применении метода аддитивной свертки к решению рассматриваемой задачи туристу гарантирована "сладкая жизнь" в течение всего похода, ибо набор из 29 пачек сахара рафинада вряд ли следует считать удачным выбором. Изменить ситуацию можно, попытавшись задать другие веса целевым функциям. В качестве упражнения предлагается также решить данную задачу методом аддитивной свертки на основе задания весов отдельных целевых функций для весов, например, $\alpha_1 = 0,8$ и $\alpha_2 = 0,2$, а полученные результаты сравнить. Впрочем, набор из 40 банок тушеной говядины признает лучшим лишь небольшая часть читателей. Радикальным изменением математической модели может служить включение в нее известного "универсального" продукта, который традиционно обменивается в походах на все остальное. Однако подобные аспекты прикладного системного анализа автор оставляет на усмотрение самих читателей.

9.5. Упражнения

В качестве упражнений для самостоятельного решения с помощью программы MS Excel предлагаются несколько типовых задач линейного и целочисленного программирования с двумя целевыми функциями. В качестве методов решения данных задач многокритериальной оптимизации рекомендуется использовать рассмотренные в настоящей главе методы уступок, минимального отклонения от идеальной точки и аддитивной свертки критериев. При этом важность целевых функций и конкретные значения уступок по одной

из них читателям предлагается задать самостоятельно в ходе решения соответствующих задач.

9.5.1. Двухкритериальная задача о производстве клея

Данная задача обобщает аналогичную однокритериальную задачу, сформулированную в *разд. 4.6.1*. Некоторое производственное предприятие выпускает три вида клея. Для производства клея используется 4 типа химических веществ: крахмал, желатин, квасцы и мел. Расход этих веществ в *кг* для получения 1 *кг* каждого вида клея и их запас на складе предприятия представлены в следующей таблице (табл. 9.2).

Таблица 9.2. *Расход химических веществ на изготовление клея, их запас на складе*

Вид клея/ Химические вещества	**Клей №1**	**Клей №2**	**Клей №3**	**Запас на складе**
Крахмал	0,4	0,3	0,2	20
Желатин	0,2	0,3	0,4	35
Квасцы	0,05	0,07	0,1	7
Мел	0,01	0,05	0,15	10

Стоимость каждого вида клея на внутреннем рынке составляет: $c_1 = 380$ *руб/кг*, $c_2 = 430$ *руб/кг*, $c_3 = 460$ *руб/кг*, стоимость каждого вида клея на внешнем рынке составляет: $d_1 = 12$ *$/кг*, $d_2 = 10$ *$/кг*, $d_3 = 8$ *$/кг*.

Требуется определить оптимальный объем выпуска клея каждого вида, обеспечивающий максимум общей стоимости всей партии клея как на внутреннем, так и на внешнем рынке.

9.5.2. Двухкритериальная задача о погрузке автомобиля

Данная задача обобщает аналогичную однокритериальную задачу, сформулированную в *разд. 5.7.1*. Для перевозки штучного товара используется грузовой автомобиль с ограниченными грузоподъемностью и объемом кузова. Груз представляет собой коробки с продуктами питания следующих видов: тушеная говядина, лосось в масле, шпроты, сгущенное молоко, томатный соус и зеленый горошек. Масса одной коробки и ее объем, а также стоимость

и субъективная ценность для заказчика одной коробки по каждому из видов продуктов питания представлены в следующей таблице (табл. 9.3).

Таблица 9.3. *Исходные данные для двухкритериальной задачи о погрузке автомобиля*

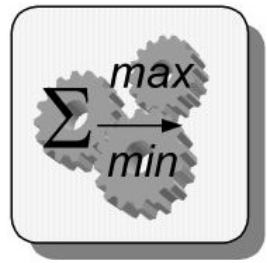
Характеристика груза	Виды продуктов питания					
	Тушеная говядина	**Лосось в масле**	**Шпроты**	**Сгущенное молоко**	**Томатный соус**	**Зеленый горошек**
Масса (*кг*)	8	8	6	5	5	4
Объем ($м^2$)	0,3	0,25	0,2	0,15	0,2	0,15
Стоимость (*руб.*)	500	400	350	380	320	300
Субъективная ценность (баллы)	60	65	70	80	85	75

При этом общая грузоподъемность автомобиля равна 5 *т*, а объем кузова — 120 $м^2$. Требуется определить оптимальную загрузку автомобиля коробками с различными продуктами питания, так чтобы обеспечить максимальную стоимость и субъективную ценность для заказчика всего груза, но при этом не допустить перегрузки автомобиля и превышения допустимого объема его кузова.

9.5.3. Двухкритериальная задача об изготовлении обуви

Данная задача обобщает аналогичную однокритериальную задачу, сформулированную в *разд. 5.7.2*. Обувная фабрика производит три вида обуви: мужскую, женскую и детскую. На каждую пару мужской, женской и детской обуви, соответственно, требуется кожи 4, 2 и 1 $дм^2$, клея 20, 20 и 10 *г*. Стоимость мужской, женской и детской обуви с учетом всех работ, соответственно, равна 1000, 1500, 500 *руб*, а предпочтения покупателей, согласно выполненному маркетинговому исследованию, соответственно, составляют: 60, 30 и 50 баллов для каждой пары обуви. Запасы кожи на складе фабрики составляют 4000 $м^2$, а клея — 3 *т*. Требуется определить оптимальный план изготовления мужской, женской и детской обуви, при котором стоимость и общая сумма баллов предпочтений покупателей всей выпущенной партии является максимальной.

Глава 10



Задачи многокритериальной булевой оптимизации

К классу задач многокритериальной оптимизации с булевыми переменными относятся такие задачи оптимизации с несколькими целевыми функциями, в которых переменные могут принимать только значения 0 или 1. При этом на характер целевых функций и ограничений в общем случае не накладывается никаких дополнительных требований. Более строгое определение данного класса задач можно получить на основе обобщения математической постановки задачи оптимизации с булевыми переменными, которая была рассмотрена в *главе 6*. Поскольку к задачам этого класса сводятся многие другие задачи многокритериальной оптимизации, в частности, многокритериальные обобщения задач оптимизации на графах и задач комбинаторной оптимизации, в настоящей главе будут рассмотрены только методы решения задач многокритериальной оптимизации с булевыми переменными.

10.1. Общая характеристика задач многокритериальной оптимизации с булевыми переменными

В общем случае под *задачей многокритериальной оптимизации с булевыми переменными* понимается такая задача оптимизации, в которой имеется несколько целевых функций, а переменные могут принимать только булевы значения. Поскольку решение многих прикладных задач многокритериальной оптимизации с помощью программы MS Excel предполагает постановку этих задач в форме обобщенной модели булева программирования, изучение данного класса задач приобретает самостоятельное значение. Более строгое определение класса задач многокритериальной оптимизации с булевыми

переменными можно получить на основе конкретизации общей математической постановки задачи многокритериальной оптимизации.

В общем случае математическая постановка задачи многокритериальной оптимизации с булевыми переменными может быть сформулирована в следующем виде:

$$f_i(x) \to \max_{x \in \Delta_\beta} \quad \text{или} \quad f_i(x) \to \min_{x \in \Delta_\beta} (\forall i \in L = \{1, 2, ..., l\}, \tag{10.1.1}$$

$$\text{где } \Delta_\beta = \{\Delta \mid g_k(x_1, x_2, ..., x_n) \le (=) b_k; (k \in \{1, 2, ..., m\})\} \tag{10.1.2}$$

$$\text{и } x_1, x_2, ..., x_n \in \{0, 1\}. \tag{10.1.3}$$

Математическую модель (10.1.1)—(10.1.3) называют также задачей *многокритериального булева программирования*. Если дополнительно в данную математическую модель вводятся предположения о линейном характере целевой функции и левых частей ограничений, то соответствующую задачу часто называют задачей *многокритериального булева линейного программирования*.

Свойства задач многокритериальной оптимизации с булевыми переменными аналогичны общим свойствам задач многокритериальной оптимизации, рассмотренным ранее в *разд. 9.1.1*. В частности, понятие доминирования по Парето определяется согласно выражению (9.1.3), а множество недоминируемых альтернатив или множество Парето (9.1.4) также служит основой многокритериального анализа задач данного класса.

Основные подходы и методы решения задач многокритериальной оптимизации с булевыми переменными аналогичны рассмотренным в *разд. 9.1.2*. В частности, метод уступок и метод минимального отклонения от идеальной точки являются базовыми для решения задач многокритериальной оптимизации с булевыми переменными. В то же время графический способ построения множества Парето для данного класса задач не может быть применен.

В настоящей главе рассматриваются особенности постановки и решения нескольких типовых задач многокритериальной оптимизации с булевыми переменными, которые являются обобщением ранее изученных задач булева программирования.

10.2. Задача водопроводчика с двумя целевыми функциями

Содержательная постановка задачи водопроводчика приводится в *разд. 1.2.11*, а способы ее решения — в *разд. 6.3*. В настоящей главе рассматривается вариант постановки двухкритериальной задачи водопроводчика в форме мате-

матической модели многокритериального булева программирования и особенности ее решения методом уступок, методом минимального отклонения от идеальной точки и методом аддитивной свертки целевых функций.

10.2.1. Математическая постановка двухкритериальной задачи водопроводчика

Для двухкритериальной задачи водопроводчика вводятся в рассмотрение две дополнительных характеристики, связанные со временем и стоимостью поднятия плит при установке вентильных перекрытий. В этом случае для разработки математической модели индивидуальной двухкритериальной задачи водопроводчика с конкретной схемой расположения труб (см. рис. 1.9) и после нумерации всех имеющихся плит (см. рис. 6.6) следует для каждой плиты задать конкретные значения соответствующих характеристик.

Для определенности предположим, что время подъема каждой плиты фиксировано и равно в *минутах*: $c_1 = 30$, $c_2 = 40$, $c_3 = 35$, $c_4 = 25$, $c_5 = 30$, $c_6 = 45$, $c_7 = 50$, $c_8 = 55$, $c_9 = 60$, $c_{10} = 65$, $c_{11} = 70$, $c_{12} = 60$, $c_{13} = 65$, $c_{14} = 75$, $c_{15} = 80$, $c_{16} = 90$. Стоимость работ после подъема каждой плиты также фиксирована и равна в *тыс. руб.*: $d_1 = 5$, $d_2 = 4{,}5$, $d_3 = 4$, $d_4 = 3{,}5$, $d_5 = 4{,}5$, $d_6 = 3{,}7$, $d_7 = 4{,}2$, $d_8 = 3$, $d_9 = 2{,}8$, $d_{10} = 2{,}5$, $d_{11} = 3{,}2$, $d_{12} = 2{,}2$, $d_{13} = 2$, $d_{14} = 1{,}8$, $d_{15} = 2{,}1$, $d_{16} = 1{,}5$.

Требуется определить номера плит, которые необходимо приподнять для установки вентильных перекрытий, чтобы общее время и стоимость выполнения работ были минимальными.

В качестве переменных математической модели двухкритериальной задачи водопроводчика рассмотрим переменные: x_i ($i \in \{1, 2, \ldots, 16\}$), которые интерпретируются следующим образом. Переменная $x_i = 1$, если для установки вентильных перекрытий принято решение приподнять i-ю плиту, $x_i = 0$ в противном случае, т. е. когда при выполнении работ i-я плита не приподнимается ($\forall i \in \{1, 2, \ldots, 16\}$).

Тогда математическая постановка рассматриваемой двухкритериальной задачи водопроводчика может быть записана в следующем виде:

$$30x_1 + 40x_2 + 35x_3 + 25x_4 + 30x_5 + 45x_6 + 50x_7 + 55x_8 + 60x_9 + \\ + 65x_{10} + 70x_{11} + 60x_{12} + 65x_{13} + 75x_{14} + 80x_{15} + 90x_{16} \to \min\limits_{x \in \Delta_\beta}, \tag{10.2.1}$$

$$5x_1 + 4{,}5x_2 + 4x_3 + 3{,}5x_4 + 4{,}5x_5 + 3{,}7x_6 + 4{,}2x_7 + 3x_8 + 2{,}8x_9 + \\ + 2{,}5x_{10} + 3{,}2x_{11} + 2{,}2x_{12} + 2x_{13} + 1{,}8x_{14} + 2{,}1x_{15} + 1{,}5x_{16} \to \min\limits_{x \in \Delta_\beta}, \tag{10.2.2}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} x_1 + x_2 + x_3 \geq 1; \\ x_2 + x_5 + x_6 + x_9 \geq 1; \\ x_3 + x_7 + x_8 + x_{12} + x_{16} \geq 1; \\ x_6 + x_7 + x_{10} + x_{11} \geq 1; \\ x_9 + x_{10} + x_{14} + x_{15} \geq 1; \\ x_i \in \{0, 1\} \ (\forall i \in \{1, 2, ..., 16\}). \end{cases} \quad (10.2.3)$$

Заметим, что каждой трубе на схеме по-прежнему соответствует ровно одно ограничение (10.2.3), которое означает, что для получения доступа к той или иной трубе необходимо приподнять по крайней мере одну из плит, под которыми проходит данная труба. Первая целевая функция (10.2.1) соответствует требованию минимизации общего времени, необходимого для поднятия плит, а вторая целевая функция (10.2.2) соответствует требованию минимизации общей стоимости производства данного вида работ.

Математическая модель (10.2.1)—(10.2.3) относится к классу задач многокритериального булева программирования, которая может быть решена с помощью рассмотренных ранее методов уступок и минимального отклонения от идеальной точки. При этом общая схема данных методов останется прежней, изменятся лишь особенности поиска решений соответствующих однокритериальных задач оптимизации с помощью программы MS Excel.

10.2.2. Решение двухкритериальной задачи водопроводчика с помощью программы MS Excel методом уступок

Для решения данной задачи с помощью программы MS Excel создадим новую книгу с именем Многокритериальная булева оптимизация и изменим имя ее первого рабочего листа на Задача водопроводчика. Решение данной задачи многокритериальной оптимизации методом уступок будет состоять из 2-х этапов. На первом этапе необходимо решить обычную задачу оптимизации, используя в качестве критериальной функции первую целевую функцию (10.2.1) в предположении, что она является более важной, чем вторая целевая функция. Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *главе 6* при решении однокритериальной задачи водопроводчика:

1. Внесем необходимые надписи в ячейки **A1:A17**, **C1:G1**. Следует отметить, что конкретное содержание этих надписей не оказывает влияния на

решение рассматриваемой задачи целочисленного линейного программирования.

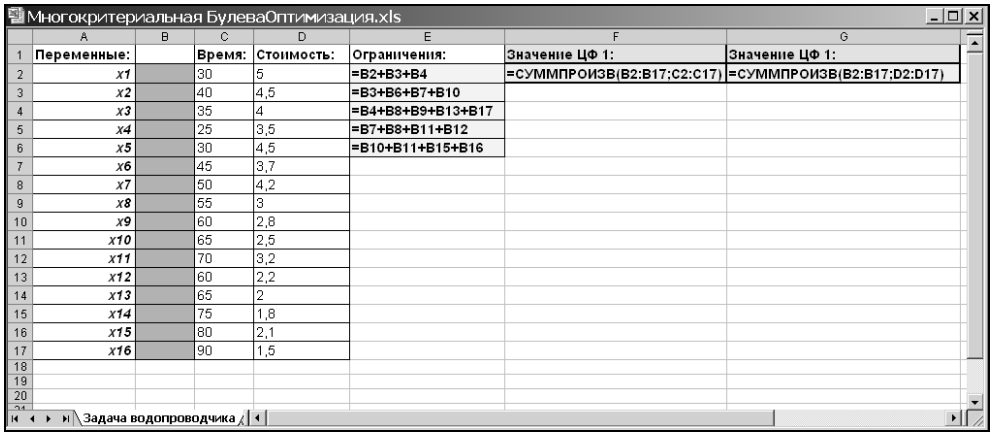
2. В ячейки **C2:C17** введем значения коэффициентов первой целевой функции: $c_1 = 30$, $c_2 = 40$, $c_3 = 35$, $c_4 = 25$, $c_5 = 30$, $c_6 = 45$, $c_7 = 50$, $c_8 = 55$, $c_9 = 60$, $c_{10} = 65$, $c_{11} = 70$, $c_{12} = 60$, $c_{13} = 65$, $c_{14} = 75$, $c_{15} = 80$, $c_{16} = 90$.
3. В ячейки **D2:D17** введем значения коэффициентов второй целевой функции: $d_1 = 5$, $d_2 = 4{,}5$, $d_3 = 4$, $d_4 = 3{,}5$, $d_5 = 4{,}5$, $d_6 = 3{,}7$, $d_7 = 4{,}2$, $d_8 = 3$, $d_9 = 2{,}8$, $d_{10} = 2{,}5$, $d_{11} = 3{,}2$, $d_{12} = 2{,}2$, $d_{13} = 2$, $d_{14} = 1{,}8$, $d_{15} = 2{,}1$, $d_{16} = 1{,}5$.
4. В ячейку **E2** введем формулу: `=B2:B4`, выражающую левую часть первого ограничения (10.2.3).
5. В ячейку **E3** введем формулу: `=B3+B6+B7+B10`, выражающую левую часть второго ограничения (10.2.3).
6. В ячейку **E4** введем формулу: `=B4+B8+B9+B13+B17`, выражающую левую часть третьего ограничения (10.2.3).
7. В ячейку **E5** введем формулу: `=B7+B8+B11+B12`, выражающую левую часть четвертого ограничения (10.2.3).
8. В ячейку **E6** введем формулу: `=B10+B11+B15+B16`, выражающую левую часть пятого ограничения (10.2.3).
9. В ячейку **F2** введем формулу: `=СУММПРОИЗВ(B2:B17;C2:C17)`, которая представляет первую целевую функцию (10.2.1).
10. В ячейку **G2** введем формулу: `=СУММПРОИЗВ(B2:B17;D2:D17)`, которая представляет вторую целевую функцию (10.2.2).

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения двухкритериальной задачи водопроводчика методом уступок на первом этапе имеет следующий вид (рис. 10.1).

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$F$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес ячеек `$B$2:$B$17`.



Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G
1	Переменные:		Время:	Стоимость:	Ограничения:	Значение ЦФ 1:	Значение ЦФ 1:
2	x1		30	5	=B2+B3+B4	=СУММПРОИЗВ(B2:B17;C2:C17)	=СУММПРОИЗВ(B2:B17;D2:D17)
3	x2		40	4,5	=B3+B6+B7+B10		
4	x3		35	4	=B4+B8+B9+B13+B17		
5	x4		25	3,5	=B7+B8+B11+B12		
6	x5		30	4,5	=B10+B11+B15+B16		
7	x6		45	3,7			
8	x7		50	4,2			
9	x8		55	3			
10	x9		60	2,8			
11	x10		65	2,5			
12	x11		70	3,2			
13	x12		60	2,2			
14	x13		65	2			
15	x14		75	1,8			
16	x15		80	2,1			
17	x16		90	1,5			
18							
19							
20							

Задача водопроводчика

Рис. 10.1. Исходные данные для решения двухкритериальной задачи водопроводчика методом уступок на первом этапе

4. Задать 5 первых ограничений для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - нажать кнопку с надписью **Добавить** в исходном диалоговом окне **Поиск решения**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$E$2:$E$6**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство ">=";
   - в качестве значения правой части ограничения выбрать ячейку ввести с клавиатуры число 1;
   - для добавления этой группы ограничений в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Задать ограничение на булевы значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$2:$B$17**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";

- в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**двоичное**";
- для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

6. В дополнительном окне **Параметры поиска решения** выбрать отметку **Линейная модель**.

Общий вид окна мастера поиска решения на первом этапе с заданными параметрами представлен на рис. 10.2.

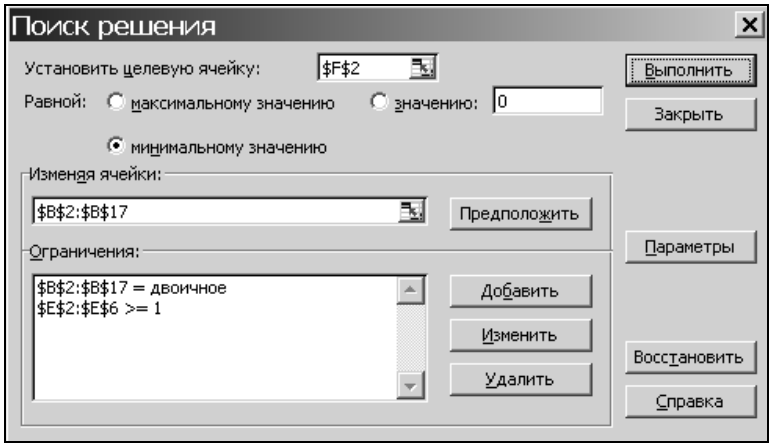


Рис. 10.2. Параметры мастера поиска решения и базовые ограничения первого этапа для двухкритериальной задачи водопроводчика

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K
1	Переменные:		Время:	Стоимость:	Ограничения:	Значение ЦФ 1:	Значение ЦФ 1:				
2	x1	0	30	5,0	1	130	11				
3	x2	0	40	4,5	1						
4	x3	1	35	4,0	1						
5	x4	0	25	3,5	1						
6	x5	1	30	4,5	1						
7	x6	0	45	3,7							
8	x7	0	50	4,2							
9	x8	0	55	3,0							
10	x9	0	60	2,8							
11	x10	1	65	2,5							
12	x11	0	70	3,2							
13	x12	0	60	2,2							
14	x13	0	65	2,0							
15	x14	0	75	1,8							
16	x15	0	80	2,1							
17	x16	0	90	1,5							
18											

Задача водопроводчика

Рис. 10.3. Результат количественного решения двухкритериальной задачи водопроводчика методом уступок на первом этапе

После задания ограничений и целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.3).

Результатом решения рассматриваемой задачи водопроводчика являются найденные оптимальные значения переменных: $x_3 = 1$, $x_5 = 1$, $x_{10} = 1$, остальные переменные равны 0, которым соответствует значение первой целевой функции: $f_1^{opt} = 130$. Анализ найденного решения показывает, что общее время поднятия плит составит 130 *мин*. При этом стоимость выполнения работ при поднятии плит с номерами 3, 5 и 10 составит 11 *тыс. руб*.

Предположим, что, исходя из технических требований, оказывается возможной уступка по первой целевой функции, равная 30 *мин*. Тем самым можно перейти ко второму этапу решения данной задачи оптимизации методом уступок.

С этой целью следует рассмотреть дополнительное ограничение следующего вида:

$$30x_1 + 40x_2 + 35x_3 + 25x_4 + 30x_5 + 45x_6 + 50x_7 + 55x_8 + 60x_9 + \\ +65x_{10} + 70x_{11} + 60x_{12} + 65x_{13} + 75x_{14} + 80x_{15} + 90x_{16} \le 160. \qquad (10.2.4)$$

в котором правая часть равна значению суммы: $f_1^{opt} + 30$, а знак ограничения выбран таким образом, чтобы соответствовать решенной задаче минимизации по первой целевой функции.

Для окончательного решения двухкритериальной задачи водопроводчика внесем следующие дополнительные данные в лист с именем Задача водопроводчика:

1. Внесем дополнительную надпись в ячейку **E7**.
2. Скопируем формулу из ячейки **F2** в ячейку **E8**, которая представляет левую часть дополнительного ограничения (10.2.4).
3. В ячейку **F8** введем значение 160, которое представляет правую часть дополнительного ограничения (10.2.4).
4. Удалим значения переменных из ячеек **B2:B17**, полученные в результате решения задачи на первом этапе.

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**. После появления диалогового окна **Поиск решения** следует выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$G$2`.

2. Для группы **Равной:** оставить вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$B$2:$B$17`.
4. Задать дополнительное ограничение (10.2.4). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$E$8**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения выбрать ячейку **$F$8**;
   - для добавления нового ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
5. В дополнительном окне **Параметры поиска решения** следует оставить выбранной отметку **Линейная модель**.

После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи водопроводчика, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.4).

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K
1	Переменные:		Время:	Стоимость:	Ограничения:	Значение ЦФ 1:	Значение ЦФ 1:				
2	x1	0	30	5,0	1	160	9,3				
3	x2	0	40	4,5	1						
4	x3	1	35	4,0	1						
5	x4	0	25	3,5	1						
6	x5	0	30	4,5	2						
7	x6	0	45	3,7	Дополнительное ограничение:						
8	x7	0	50	4,2	160	160					
9	x8	0	55	3,0							
10	x9	1	60	2,8							
11	x10	1	65	2,5							
12	x11	0	70	3,2							
13	x12	0	60	2,2							
14	x13	0	65	2,0							
15	x14	0	75	1,8							
16	x15	0	80	2,1							
17	x16	0	90	1,5							
18											

Задача водопроводчика

Рис. 10.4. Результат окончательного решения двухкритериальной задачи водопроводчика методом уступок

Результатом решения двухкритериальной задачи водопроводчика являются найденные оптимальные значения переменных: $x_3 = 1$, $x_9 = 1$, $x_{10} = 1$, остальные переменные равны 0, им соответствуют значения целевых функций: $f_1^{opt} = 160$ и $f_2^{opt} = 9{,}3$.

Анализ найденного решения показывает, что для установки вентильных перекрытий следует поднять плиты с номерами 3, 9 и 10. При этом общее время поднятия плит составит160 *мин*, а общая стоимость выполнения работ составит 9,3 *тыс. рублей*.

Примечание

Заинтересованным читателям в качестве упражнения предлагается решить рассмотренную двухкритериальную задачу водопроводчика методом уступок, изменив порядок важности критериев. Полученные результаты сравнить.

10.2.3. Решение двухкритериальной задачи водопроводчика с помощью программы MS Excel методом минимального отклонения

Для решения двухкритериальной задачи водопроводчика с помощью программы MS Excel методом минимального отклонения от идеальной точки будем использовать те же конкретные значения параметров рассматриваемой индивидуальной задачи водопроводчика.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная булева оптимизация создадим новый рабочий лист с именем: Задача водопроводчика №2. Решение данной задачи многокритериальной оптимизации методом минимального отклонения от идеальной точки будет состоять из 2-х этапов. На первом этапе необходимо решить две обычных задачи оптимизации, используя в качестве критериальных функций, соответственно, целевые функции (10.2.1) и (10.2.2).

Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 10.2.2* при решении двухкритериальной задачи водопроводчика методом уступок. Чтобы сократить дублирование действий, скопируем данные из листа с именем Задача водопроводчика в лист с именем Задача водопроводчика №2.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи водопроводчика методом минимального отклонения от идеальной точки на первом этапе должен иметь вид, аналогичный изображенному на рис. 10.1.

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения и задать соответствующие параметры. Поскольку последовательность действий по заданию параметров мастера поиска решения однокритериальной задачи водопроводчика уже неоднократно рассматривалась ранее, здесь она не приводится. Общий вид окна мастера поиска решения с заданными параметрами для решения задачи водопроводчика по первому критерию представлен на рис. 10.2.

После задания ограничений и первой целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет вид, изображенный на рис. 10.3. Результатом решения однокритериальной задачи водопроводчика на первом этапе является найденное значение первой целевой функции: $f_1^{opt} = 130$.

Диалоговое окно задания параметров для мастера поиска решения задачи водопроводчика по второй целевой функции представлено на рис. 10.5.

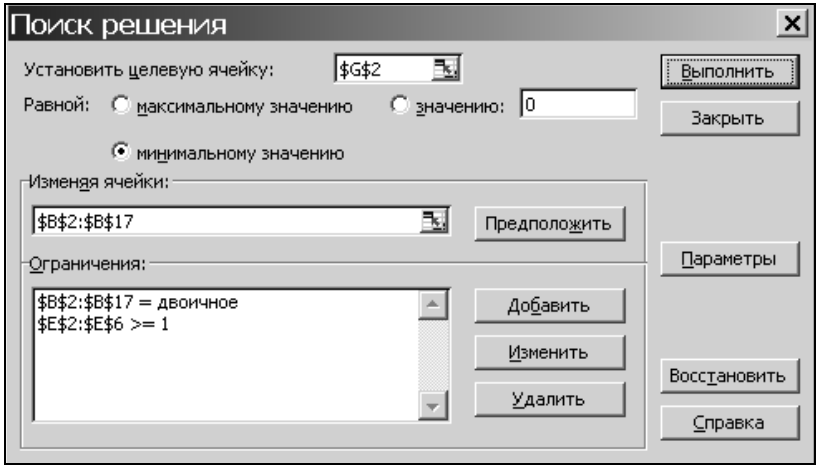


Рис. 10.5. Параметры мастера поиска решения и базовые ограничения для второй целевой функции двухкритериальной задачи водопроводчика

После задания ограничений и второй целевой функции можно приступить к поиску численного решения по второму критерию, для этого следует предварительно удалить ранее найденные значения переменных и нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено соответствующее количественное решение. Результатом решения однокритериальной задачи водопроводчика на первом этапе является найденное значение второй целевой функции: $f_2^{opt} = 8{,}5$.

Для окончательного решения двухкритериальной задачи водопроводчика на втором этапе внесем следующие дополнительные данные в лист с именем Задача водопроводчика №2:

1. Введем дополнительную надпись в ячейку **H1**.
2. В ячейку **F3** введем найденное на первом этапе оптимальное значение первой целевой функции $f_1^{opt} = 130$.
3. В ячейку **G3** введем найденное на первом этапе оптимальное значение второй целевой функции $f_2^{opt} = 8{,}5$.
4. В ячейку **H2** введем формулу: `=(F2-F3)^2+(G2-G3)^2`, которая представляет минимальное отклонение от найденной идеальной точки, полученной в результате решения задачи на первом этапе.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи водопроводчика методом минимального отклонения от идеальной точки на втором этапе имеет следующий вид (рис. 10.6).

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I
1	Переменные:		Время:	Стоимость:	Ограничения:	Значение ЦФ 1:	Значение ЦФ 1:	Новая ЦФ:	
2	x1		30	5,0	0	0	0,0	16972,250	
3	x2		40	4,5	0	130	8,5		
4	x3		35	4,0	0				
5	x4		25	3,5	0				
6	x5		30	4,5	0				
7	x6		45	3,7					
8	x7		50	4,2					
9	x8		55	3,0					
10	x9		60	2,8					
11	x10		65	2,5					
12	x11		70	3,2					
13	x12		60	2,2					
14	x13		65	2,0					
15	x14		75	1,8					
16	x15		80	2,1					
17	x16		90	1,5					
18									

Задача водопроводчика №2

Рис. 10.6. Исходные данные для решения задачи водопроводчика методом минимального отклонения от идеальной точки на втором этапе

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**.

После появления диалогового окна **Поиск решения** выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$I$7`.

2. Для группы **Равной:** оставить вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$B$2:$H$2`.

В дополнительном окне **Параметры поиска решения** следует обязательно убрать отметку **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами представлен на рис. 10.7.

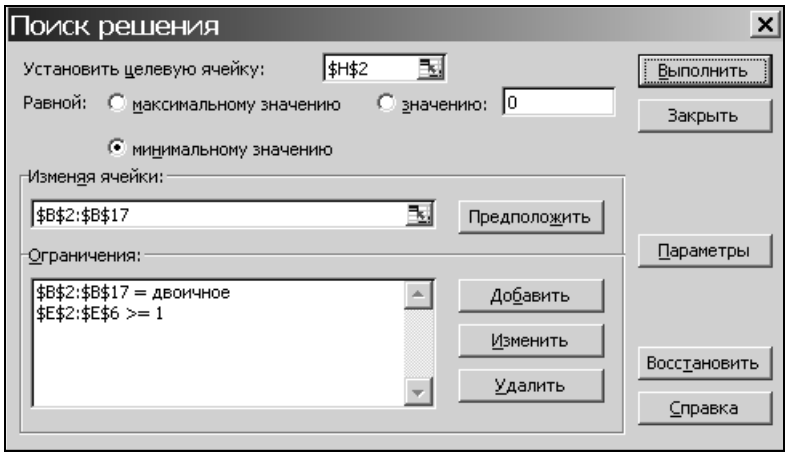
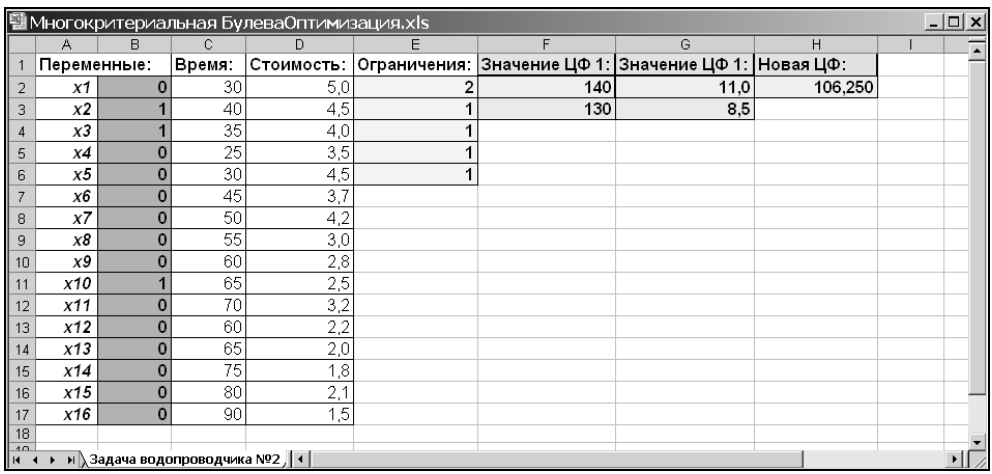


Рис. 10.7. Параметры мастера поиска решения и базовые ограничения для окончательного решения двухкритериальной задачи водопроводчика методом минимального отклонения от идеальной точки


Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I
1	Переменные:		Время:	Стоимость:	Ограничения:	Значение ЦФ 1:	Значение ЦФ 1:	Новая ЦФ:	
2	x1	0	30	5,0	2	140	11,0	106,250	
3	x2	1	40	4,5	1	130	8,5		
4	x3	1	35	4,0	1				
5	x4	0	25	3,5	1				
6	x5	0	30	4,5	1				
7	x6	0	45	3,7					
8	x7	0	50	4,2					
9	x8	0	55	3,0					
10	x9	0	60	2,8					
11	x10	1	65	2,5					
12	x11	0	70	3,2					
13	x12	0	60	2,2					
14	x13	0	65	2,0					
15	x14	0	75	1,8					
16	x15	0	80	2,1					
17	x16	0	90	1,5					
18									

Задача водопроводчика N92

Рис. 10.8. Результат окончательного решения двухкритериальной задачи водопроводчика методом минимального отклонения от идеальной точки

После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи водопроводчика, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.8).

Результатом решения двухкритериальной задачи водопроводчика методом минимального отклонения от идеальной точки являются найденные оптимальные значения переменных: $x_2 = 1$, $x_3 = 1$, $x_{10} = 1$, остальные переменные равны 0, им соответствуют значения целевых функций: $f_1^{opt} = 140$ и $f_2^{opt} = 11$. Анализ найденного решения показывает, что для установки вентильных перекрытий следует поднять плиты с номерами 2, 3 и 10. При этом общее время поднятия плит составит 140 *мин*, а общая стоимость выполнения работ составит 11 *тыс. рублей*.

Далее приводится описание метода аддитивной свертки на основе задания весов отдельных целевых функций. Этот метод также может быть использован для решения двухкритериальной задачи водопроводчика.

10.2.4. Решение двухкритериальной задачи водопроводчика с помощью программы MS Excel методом аддитивной свертки

Для решения задачи водопроводчика с двумя целевыми функциями с помощью программы MS Excel методом аддитивной свертки на основе задания весов отдельных целевых функций будем использовать те же значения параметров рассматриваемой ранее конкретной задачи водопроводчика. Дополнительно следует задать количественные значения *весов* исходным целевым функциям, которые удовлетворяют условиям, отмеченным в *разд. 9.1.2*. Если предположить, что с точки зрения предпочтений руководителя работ, принимающего решение о выборе плит, оба критерия равноценны, то такими весами являются значения: $\alpha_1 = 0{,}5$ и $\alpha_2 = 0{,}5$.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная булева оптимизация создадим новый рабочий лист с именем Задача водопроводчика №3. Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 10.2.2* при решении двухкритериальной задачи водопроводчика методом уступок. В то же время, чтобы сократить дублирование действий, скопируем данные из листа с именем Задача водопроводчика №2 (см. рис. 10.6) в лист с именем Задача водопроводчика №3.

Для решения двухкритериальной задачи водопроводчика методом аддитивной свертки на основе задания весов отдельных целевых функций внесем следующие изменения в лист с именем Задача водопроводчика №3:

1. Удалим данные из ячеек **F3** и **G3**, необходимость использования которых отпала.
2. В ячейку **H2** введем формулу: `=0,5*F2/195+0,5*G2/11`, которая представляет аддитивную свертку исходных целевых функций с использованием заданных ранее весов критериев и шкалированием критериев относительно максимальных значений соответствующих целевых функций.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи водопроводчика методом аддитивной свертки имеет следующий вид (рис. 10.9).

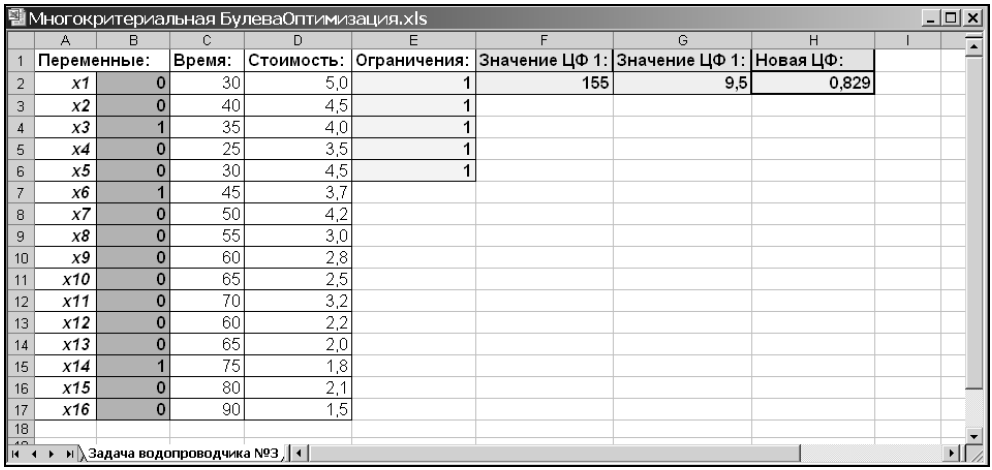
Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I
1	Переменные:		Время:	Стоимость:	Ограничения:	Значение ЦФ 1:	Значение ЦФ 1:	Новая ЦФ:	
2	x1		30	5,0	0	0	0,0	0,000	
3	x2		40	4,5	0				
4	x3		35	4,0	0				
5	x4		25	3,5	0				
6	x5		30	4,5	0				
7	x6		45	3,7					
8	x7		50	4,2					
9	x8		55	3,0					
10	x9		60	2,8					
11	x10		65	2,5					
12	x11		70	3,2					
13	x12		60	2,2					
14	x13		65	2,0					
15	x14		75	1,8					
16	x15		80	2,1					
17	x16		90	1,5					
18									

Задача водопроводчика №3

Рис. 10.9. Исходные данные для решения двухкритериальной задачи водопроводчика методом аддитивной свертки

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**. В диалоговом окне **Поиск решения** следует задать целевую ячейку, диапазон изменения ячеек и ограничения задачи таким же образом, как было рассмотрено в *разд. 9.4.3* при решении задачи методом минимального отклонения от идеальной точки. В дополнительном окне **Параметры поиска решения** следует выбрать отметку **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами аналогичен окну параметров мастера поиска, представленному на рис. 10.7.



Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I
1	Переменные:		Время:	Стоимость:	Ограничения:	Значение ЦФ 1:	Значение ЦФ 1:	Новая ЦФ:	
2	x1	0	30	5,0	1	155	9,5	0,829	
3	x2	0	40	4,5	1				
4	x3	1	35	4,0	1				
5	x4	0	25	3,5	1				
6	x5	0	30	4,5	1				
7	x6	1	45	3,7					
8	x7	0	50	4,2					
9	x8	0	55	3,0					
10	x9	0	60	2,8					
11	x10	0	65	2,5					
12	x11	0	70	3,2					
13	x12	0	60	2,2					
14	x13	0	65	2,0					
15	x14	1	75	1,8					
16	x15	0	80	2,1					
17	x16	0	90	1,5					
18									

Задача водопроводчика №3

Рис. 10.10. Результат окончательного решения двухкритериальной задачи водопроводчика методом аддитивной свертки

После задания ограничений и целевой функции на втором этапе можно приступить к поиску решения двухкритериальной задачи водопроводчика, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.10).

Результатом решения двухкритериальной задачи водопроводчика методом аддитивной свертки на основе задания весов отдельных целевых функций являются найденные оптимальные значения переменных: $x_3 = 1$, $x_6 = 1$, $x_{14} = 1$, которым соответствуют значения целевых функций: $f_1^{opt} = 155$ и $f_2^{opt} = 9{,}5$.

Содержательный анализ найденного решения показывает, что при выполнении работ по установлению перекрытий на водопроводные трубы для заданной схемы их расположения (см. рис. 6.10) следует поднять три плиты, а именно плиты с номерами 3, 6 и 14. При этом будет достигнут некоторый компромисс относительно общего времени и стоимости выполнения работ.

Анализ найденного решения также показывает, что шкалирование критериев позволяет избавиться от недостатка метода аддитивной свертки, связанного с неравноценным характером влияния абсолютных значений отдельных целевых функций на итоговый результат, особенно явно проявляющийся при решении задач булева программирования.

10.3. Двухкритериальная задача о назначении

Содержательная постановка задачи о назначении приводится в *разд. 1.2.8*, а способы ее решения — в *разд. 6.4*. В настоящей главе рассматривается вариант постановки двухкритериальной задачи о назначении в форме математической модели многокритериального булева программирования и особенности ее решения методом уступок, методом минимального отклонения от идеальной точки и методом аддитивной свертки целевых функций.

10.3.1. Математическая постановка двухкритериальной задачи о назначении

Для двухкритериальной задачи о назначении вводятся в рассмотрение две характеристики, одна из которых по-прежнему связана с затратами на подготовку кандидатов, а вторая — с эффективностью в форме прибыли, которая связана с работой кандидата в соответствующей должности. Для разработки математической модели индивидуальной двухкритериальной задачи о назначении следует задать конкретные значения соответствующих характеристик.

Для определенности рассмотрим двухкритериальной вариант индивидуальной задачи о назначении в форме минимизации общих затрат на подготовку кандидатов и максимизации общей прибыли при выполнении ими работ. В этом случае в качестве кандидатов рассмотрим 5 сотрудников некоторой фирмы: это Андреев, Бубнов, Васильев, Григорьев и Дмитриев, а в качестве работ — 5 вакантных должностей в этой фирме: менеджер, программист, бизнес-аналитик, маркетолог и руководитель проектов. При этом затраты c_{ij} ($\forall i, j \in \{1, 2, 3, 4, 5\}$) на замещение должностей кандидатами, связанные с необходимостью их предварительного обучения и стажировки, аналогичны рассмотренным в *разд. 6.4* и заданы в форме следующей таблицы (см. табл. 6.1).

Отдельные значения ожидаемой прибыли кандидатов при работе в должностях d_{ij} ($\forall i, j \in \{1, 2, 3, 4, 5\}$) заданы в форме следующей таблицы (табл. 10.1).

Требуется определить такое назначение кандидатов на вакантные должности, при котором общие затраты на их подготовку будут минимальными, а общая прибыль при их работе в соответствующих должностях будет максимальной.

Таблица 10.1. *Ожидаемая прибыль от работы кандидатов в должностях (в тыс. рублей)*

Кандидаты/ Должности	Андреев	Бубнов	Васильев	Григорьев	Дмитриев
Менеджер	150	95	90	210	100
Программист	185	200	170	140	175
Бизнес-аналитик	125	110	140	115	105
Маркетолог	80	180	85	100	135
Руководитель проектов	110	145	230	125	220

В качестве переменных математической модели двухкритериальной задачи о назначении рассмотрим следующие булевы переменные: $x_{ij} \in \{0, 1\}$, которые будут соответствовать назначению кандидатов на выполнение работ. В этом случае $x_{ij} = 1$ означает, что i-й кандидат назначается на выполнение j-й работы, и $x_i = 0$, если i-й кандидат не назначается на выполнение j-й работы. Тогда математическая постановка *двухкритериальной задачи о назначении* может быть записана в следующем виде:

$$\begin{aligned} &5x_{11} + 10x_{12} + 9x_{13} + 14x_{14} + 6x_{15} + \\ &+ 13x_{21} + 15x_{22} + 11x_{23} + 19x_{24} + 17x_{25} + \\ &+ 7x_{31} + 14x_{32} + 12x_{33} + 8x_{34} + 10x_{35} + \\ &+ 8x_{41} + 11x_{42} + 6x_{43} + 7x_{44} + 9x_{45} + \\ &+ 15x_{51} + 12x_{52} + 17x_{53} + 13x_{54} + 16x_{55} \to \min\limits_{x \in \Delta_\beta}, \end{aligned} \tag{10.3.1}$$

$$\begin{aligned} &150x_{11} + 95x_{12} + 90x_{13} + 210x_{14} + 100x_{15} + \\ &+ 185x_{21} + 200x_{22} + 170x_{23} + 140x_{24} + 175x_{25} + \\ &+ 125x_{31} + 110x_{32} + 140x_{33} + 115x_{34} + 105x_{35} + \\ &+ 80x_{41} + 180x_{42} + 85x_{43} + 100x_{44} + 135x_{45} + \\ &+ 110x_{51} + 145x_{52} + 230x_{53} + 125x_{54} + 220x_{55} \to \max\limits_{x \in \Delta_\beta}, \end{aligned} \tag{10.3.2}$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$\begin{cases} x_{11}+x_{12}+x_{13}+x_{14}+x_{15}=1; \\ x_{21}+x_{22}+x_{23}+x_{24}+x_{25}=1; \\ x_{31}+x_{32}+x_{33}+x_{34}+x_{35}=1; \\ x_{41}+x_{42}+x_{43}+x_{44}+x_{45}=1; \\ x_{51}+x_{52}+x_{53}+x_{54}+x_{55}=1; \\ x_{11}+x_{21}+x_{31}+x_{41}+x_{51}=1; \\ x_{12}+x_{22}+x_{32}+x_{42}+x_{52}=1; \\ x_{13}+x_{23}+x_{33}+x_{43}+x_{53}=1; \\ x_{14}+x_{24}+x_{34}+x_{44}+x_{54}=1; \\ x_{15}+x_{25}+x_{35}+x_{45}+x_{55}=1; \\ x_{ij} \in \{0,1\}\ (\forall i, j \in \{1,2,3,4,5\}). \end{cases} \qquad (10.3.3)$$

Математическая модель (10.3.1)—(10.3.3) относится к классу задач многокритериального булева программирования и может быть решена с помощью рассмотренных ранее методов решения задач этого класса. При этом общая схема данных методов останется прежней, изменятся лишь особенности поиска решений соответствующих однокритериальных задач оптимизации с помощью программы MS Excel.

10.3.2. Решение двухкритериальной задачи о назначении с помощью программы MS Excel методом уступок

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная булева оптимизация создадим рабочий лист с именем Задача о назначении. Решение данной задачи многокритериальной оптимизации методом уступок будет состоять из 2-х этапов. На первом этапе необходимо решить обычную задачу оптимизации, используя в качестве критериальной функции первую целевую функцию (10.3.1) в предположении, что она является более важной, чем вторая целевая функция. Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *главе 6* при решении однокритериальной задачи о назначении:

1. Внесем необходимые надписи в ячейки **A1**, **A7**, **A13:A19**, **B1**, **G1**, **H1**, **B7:G7**, **H7**, **B13:G13**, как это изображено на рис. 10.11. Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой двухкритериальной задачи о назначении.

2. В ячейки **B2:F6** введем значения коэффициентов первой целевой функции (табл. 6.1).
3. В ячейки **B8:F12** введем значения коэффициентов второй целевой функции (табл. 10.1).
4. В ячейку **G2** введем формулу: `=СУММПРОИЗВ($B$2:$F$6;$B$14:$F$18)`, которая представляет первую целевую функцию (10.3.1).
5. В ячейку **H2** введем формулу: `=СУММПРОИЗВ($B$8:$F$12;$B$14:$F$18)`, которая представляет вторую целевую функцию (10.3.2).
6. В ячейку **G14** введем формулу: `=СУММ(B14:F14)`, которая представляет первое ограничение (10.3.3).
7. Скопируем формулу, введенную в ячейку **G14**, в ячейки **G15:G18**.
8. В ячейку **B19** введем формулу: `=СУММ(B14:B18)`, которая представляет шестое ограничение (10.3.3).
9. Скопируем формулу, введенную в ячейку **B19**, в ячейки **C19:F19**.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения двухкритериальной задачи о назначении методом уступок на первом этапе имеет следующий вид (рис. 10.11).

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H
1	Затраты -	Коэффициенты первой целевой функции:					Значение ЦФ 1:	
2		5	10	9	14	6	=СУММПРОИЗВ($B$2:$F$6;$B$14:$F$18)	=СУММПРОИЗ
3		13	15	11	19	17		
4		7	14	12	8	10		
5		8	11	6	7	9		
6		15	12	17	13	16		
7	Прибыль -	Коэффициенты второй целевой функции:						
8		150	95	90	210	100		
9		185	200	170	140	175		
10		125	110	140	115	105		
11		80	180	85	100	135		
12		110	145	230	125	220		
13	Переменные:	*Xi1*	*Xi2*	*Xi3*	*Xi4*	*Xi5*	Ограничения:	
14	*X1j*						=СУММ(B14:F14)	
15	*X2j*						=СУММ(B15:F15)	
16	*X3j*						=СУММ(B16:F16)	
17	*X4j*						=СУММ(B17:F17)	
18	*X5j*						=СУММ(B18:F18)	
19	Ограничения:	=СУММ(B14:B18)	=СУММ(C14:C18)	=СУММ(D14:D18)	=СУММ(E14:E18)	=СУММ(F14:F18)		
20								
21								
22								
23								
24								
25								

Задача о назначении

Рис. 10.11. Исходные данные для решения двухкритериальной задачи о назначении методом уступок на первом этапе

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$G$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес диапазона ячеек `$B$14:$F$18`.
4. Задать первую группу ограничений, соответствующих первым 5 базовым ограничениям исходной постановки решаемой задачи о назначении (10.3.3). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$G$14:$G$18**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры число 1;
   - для добавления первой группы ограничений в дополнительном окне нажать кнопку с надписью **Добавить**.
5. Задать вторую группу ограничений, соответствующих оставшимся 5 базовым ограничениям исходной постановки решаемой задачи о назначении (10.3.3). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$19:$F$19**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры число 1;
   - для добавления второй группы ограничений в дополнительном окне нажать кнопку с надписью **Добавить**.

6. Добавить последнее ограничение на булевы значения переменных задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$B$14:$F$18**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "двоичное";
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
7. В дополнительном окне **Параметры** поиска решения выбрать отметку **Линейная модель**.

Общий вид окна мастера поиска решения на первом этапе с заданными параметрами представлен на рис. 10.12.

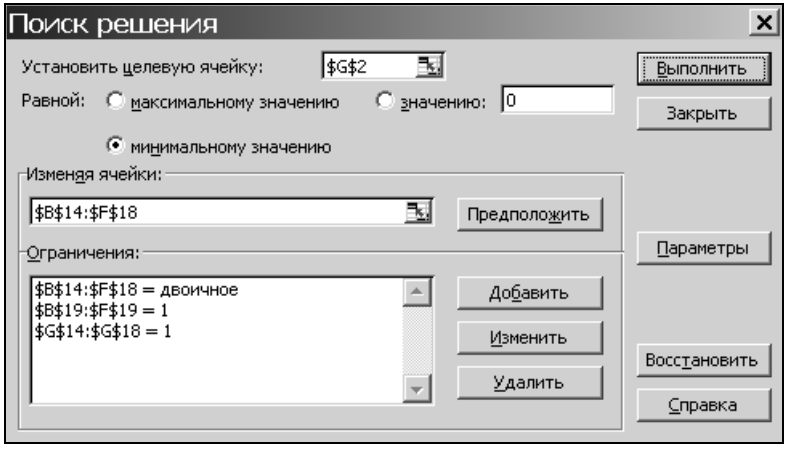
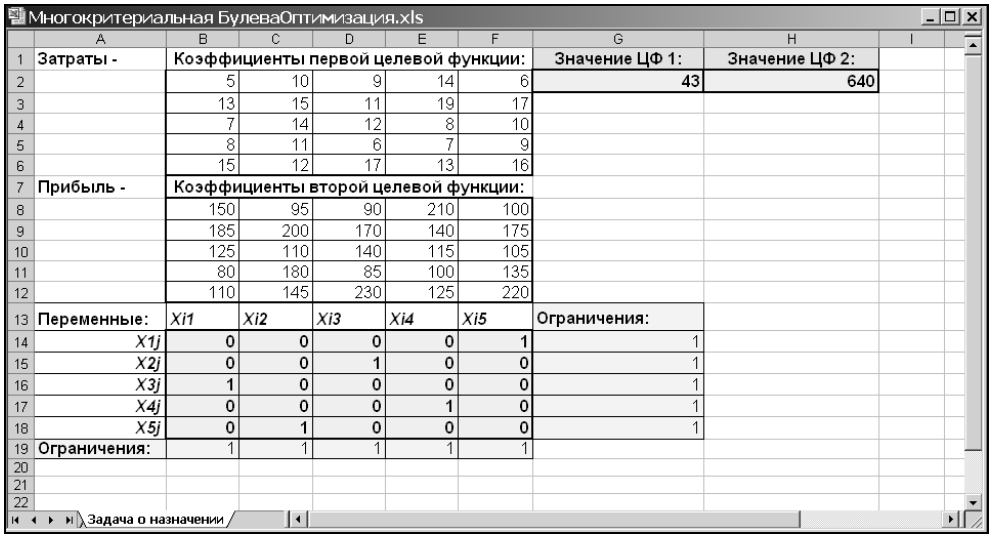


Рис. 10.12. Параметры мастера поиска решения и базовые ограничения первого этапа для двухкритериальной задачи о назначении

После задания ограничений и целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.13).



Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I
1	Затраты -	Коэффициенты первой целевой функции:					Значение ЦФ 1:	Значение ЦФ 2:	
2		5	10	9	14	6	43	640	
3		13	15	11	19	17			
4		7	14	12	8	10			
5		8	11	6	7	9			
6		15	12	17	13	16			
7	Прибыль -	Коэффициенты второй целевой функции:							
8		150	95	90	210	100			
9		185	200	170	140	175			
10		125	110	140	115	105			
11		80	180	85	100	135			
12		110	145	230	125	220			
13	Переменные:	Xi1	Xi2	Xi3	Xi4	Xi5	Ограничения:		
14	X1j	0	0	0	0	1	1		
15	X2j	0	0	1	0	0	1		
16	X3j	1	0	0	0	0	1		
17	X4j	0	0	0	1	0	1		
18	X5j	0	1	0	0	0	1		
19	Ограничения:	1	1	1	1	1			
20									
21									
22									

Задача о назначении

Рис. 10.13. Результат количественного решения двухкритериальной задачи о назначении методом уступок на первом этапе

Результатом решения рассматриваемой задачи о назначении являются найденные оптимальные значения переменных: $x_{15} = 1$, $x_{23} = 1$, $x_{31} = 1$, $x_{44} = 1$, $x_{52} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствует минимальное значение целевой функции: $f_{\text{opt}} = 43$ *тыс. руб*.

Предположим, что, исходя из экономических требований, оказывается возможной уступка по первой целевой функции, равная 10 *тыс. руб*. Тем самым, можно перейти ко второму этапу решения данной задачи оптимизации методом уступок.

С этой целью следует рассмотреть дополнительное ограничение следующего вида:

$$\begin{aligned} &5x_{11} + 10x_{12} + 9x_{13} + 14x_{14} + 6x_{15} + \\ &+ 13x_{21} + 15x_{22} + 11x_{23} + 19x_{24} + 17x_{25} + \\ &+ 7x_{31} + 14x_{32} + 12x_{33} + 8x_{34} + 10x_{35} + \\ &+ 8x_{41} + 11x_{42} + 6x_{43} + 7x_{44} + 9x_{45} + \\ &+ 15x_{51} + 12x_{52} + 17x_{53} + 13x_{54} + 16x_{55} \le 53, \end{aligned} \qquad (10.3.4)$$

в котором правая часть равна значению суммы: $f_1^{opt} + 10$, а знак ограничения выбран таким образом, чтобы соответствовать решенной задаче минимизации по первой целевой функции.

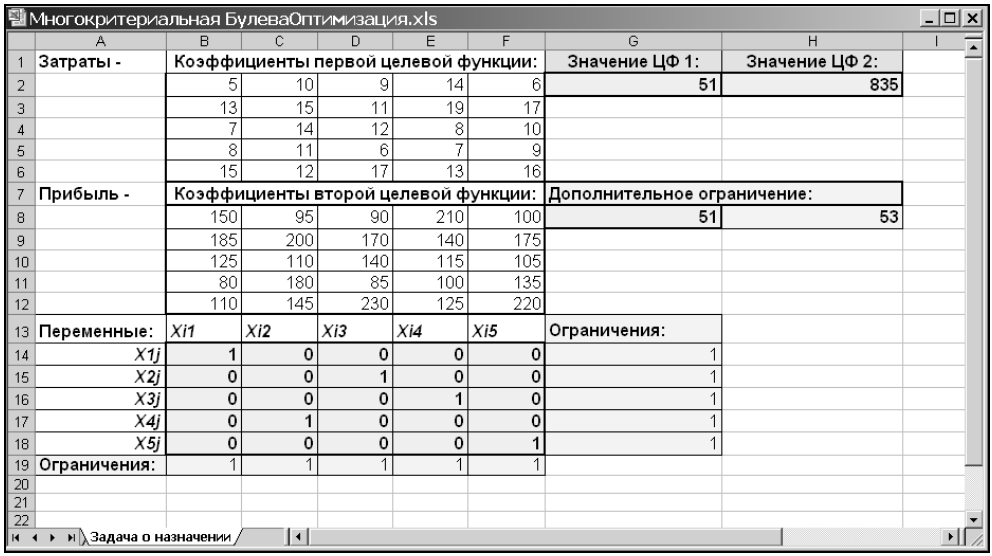
Для окончательного решения двухкритериальной задачи о назначении внесем следующие дополнительные данные в лист с именем Задача о назначении:

1. Внесем дополнительную надпись в ячейку **G7**.
2. Скопируем формулу из ячейки **G2** в ячейку **G8**, которая представляет левую часть дополнительного ограничения (10.3.4).
3. В ячейку **H8** введем значение 53, которое представляет правую часть дополнительного ограничения (10.3.4).
4. Удалим значения переменных из ячеек **B14:F18**, полученные в результате решения задачи на первом этапе.

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**. После появления диалогового окна **Поиск решения** следует выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$H$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **максимальному значению**.
3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$B$14:$B$18`.
4. Задать дополнительное ограничение (10.3.4). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$G$8**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения выбрать ячейку **$H$8**;
   - для добавления нового ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

В дополнительном окне **Параметры поиска решения** следует оставить выбранной отметку **Линейная модель**. После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи о назначении, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.14).



Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I
1	Затраты -	Коэффициенты первой целевой функции:					Значение ЦФ 1:	Значение ЦФ 2:	
2		5	10	9	14	6	51	835	
3		13	15	11	19	17			
4		7	14	12	8	10			
5		8	11	6	7	9			
6		15	12	17	13	16			
7	Прибыль -	Коэффициенты второй целевой функции:					Дополнительное ограничение:		
8		150	95	90	210	100	51	53	
9		185	200	170	140	175			
10		125	110	140	115	105			
11		80	180	85	100	135			
12		110	145	230	125	220			
13	Переменные:	Xi1	Xi2	Xi3	Xi4	Xi5	Ограничения:		
14	X1j	1	0	0	0	0	1		
15	X2j	0	0	1	0	0	1		
16	X3j	0	0	0	1	0	1		
17	X4j	0	1	0	0	0	1		
18	X5j	0	0	0	0	1	1		
19	Ограничения:	1	1	1	1	1			
20									
21									
22									

Задача о назначении

Рис. 10.14. Результат окончательного решения двухкритериальной задачи о назначении методом уступок

Результатом решения рассматриваемой задачи о назначении являются найденные оптимальные значения переменных: $x_{11} = 1$, $x_{23} = 1$, $x_{34} = 1$, $x_{42} = 1$, $x_{55} = 1$, остальные переменные равны 0. Найденному решению соответствуют значения целевых функций: $f_1^{opt} = 51$ и $f_2^{opt} = 835$.

Анализ найденного решения показывает, что при замещении вакантных должностей в рассматриваемой фирме следует на должность менеджера назначить сотрудника Андреева, на должность программиста — Васильева, на должность бизнес-аналитика — Григорьева, на должность маркетолога — Бубнова и, наконец, на должность руководителя проектов — Дмитриева. При этом общие затраты на обучение и стажировку сотрудников составят 51 *тыс. рублей*, а общая прибыль от работы этой группы составит 835 *тыс. рублей*.

Примечание

Заинтересованным читателям в качестве упражнения предлагается решить рассмотренную двухкритериальную задачу о назначении методом уступок, изменив порядок важности критериев. Полученные результаты сравнить.

10.3.3. Решение двухкритериальной задачи о назначении с помощью программы MS Excel методом минимального отклонения

Для решения задачи о назначении с двумя целевыми функциями с помощью программы MS Excel методом минимального отклонения от идеальной точки будем использовать те же конкретные значения параметров рассматриваемой индивидуальной задачи о назначении.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная булева оптимизация создадим новый рабочий лист с именем: Задача о назначении №2. Решение данной задачи многокритериальной оптимизации методом минимального отклонения от идеальной точки будет состоять из 2-х этапов. На первом этапе необходимо решить две обычных задачи оптимизации, используя в качестве критериальных функций, соответственно, целевые функции (10.3.1) и (10.3.2).

Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 10.3.2* при решении двухкритериальной задачи о назначении методом уступок. Чтобы сократить дублирование действий, скопируем данные из листа с именем Задача о назначении в лист с именем Задача о назначении №2.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о назначении методом минимального отклонения от идеальной точки на первом этапе должен иметь вид, аналогичный изображенному на рис. 10.11.

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения и задать соответствующие параметры. Поскольку последовательность действий по заданию параметров мастера поиска решения однокритериальной задачи о назначении уже неоднократно рассматривались ранее, здесь она не приводится. Общий вид окна мастера поиска решения с заданными параметрами для решения задачи о назначении по первому критерию представлен на рис. 10.12.

После задания ограничений и первой целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет вид, изображенный на рис. 10.13. Результатом решения однокритериальной задачи о назначении на первом этапе является найденное значение первой целевой функции: $f_1^{opt} = 43$.

Диалоговое окно задания параметров для мастера поиска решения задачи о назначении по второй целевой функции представлено на рис. 10.15.

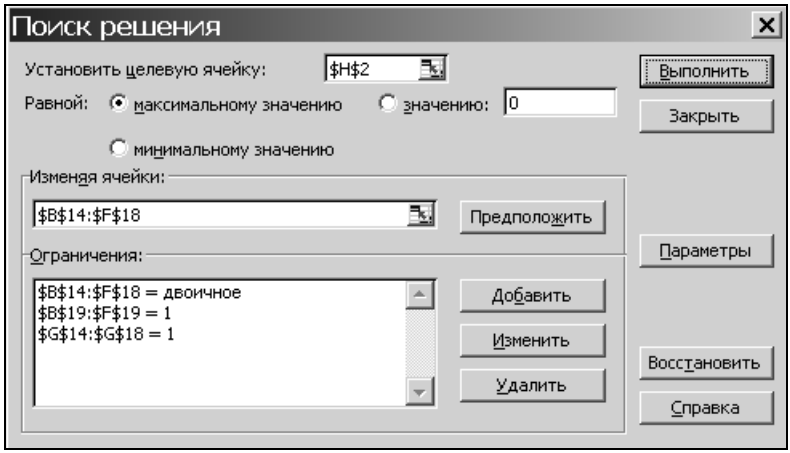


Рис. 10.15. Параметры мастера поиска решения и базовые ограничения для второй целевой функции двухкритериальной задачи о назначении

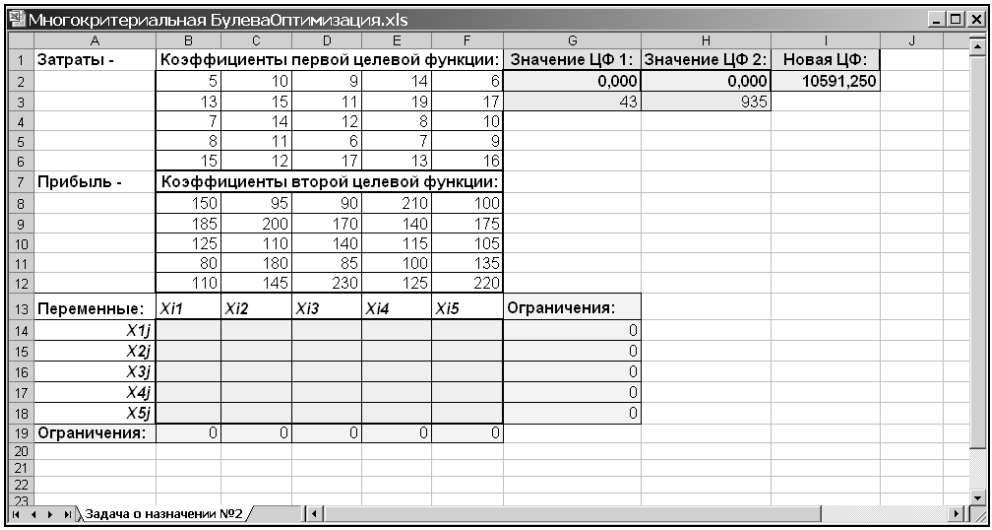
После задания ограничений и второй целевой функции можно приступить к поиску численного решения по второму критерию, для этого следует предварительно удалить ранее найденные значения переменных и нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено соответствующее количественное решение. Результатом решения однокритериальной задачи о назначении на первом этапе является найденное значение второй целевой функции: $f_2^{opt} = 935$.

Для окончательного решения двухкритериальной задачи о назначении на втором этапе внесем следующие дополнительные данные в лист с именем Задача о назначении №2:

1. Введем дополнительную надпись в ячейку **I1**.
2. В ячейку **G3** введем найденное на первом этапе оптимальное значение первой целевой функции $f_1^{opt} = 43$.
3. В ячейку **H3** введем найденное на первом этапе оптимальное значение второй целевой функции $f_2^{opt} = 935$.
4. В ячейку **H2** введем формулу: `=(G2-G3)^2+(0,1*(H2-H3))^2`, которая представляет минимальное отклонение от найденной идеальной точки, полученной в результате решения задачи на первом этапе.

Примечание

Следует отметить, что с целью выравнивания порядка целевых функций в выражении для новой ЦФ был использован по второй целевой функции масштабирующий коэффициент 0,1.



Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	Затраты -	Коэффициенты первой целевой функции:					Значение ЦФ 1:	Значение ЦФ 2:	Новая ЦФ:	
2		5	10	9	14	6	0,000	0,000	10591,250	
3		13	15	11	19	17	43	935		
4		7	14	12	8	10				
5		8	11	6	7	9				
6		15	12	17	13	16				
7	Прибыль -	Коэффициенты второй целевой функции:								
8		150	95	90	210	100				
9		185	200	170	140	175				
10		125	110	140	115	105				
11		80	180	85	100	135				
12		110	145	230	125	220				
13	Переменные:	*Xi1*	*Xi2*	*Xi3*	*Xi4*	*Xi5*	Ограничения:			
14	*X1j*						0			
15	*X2j*						0			
16	*X3j*						0			
17	*X4j*						0			
18	*X5j*						0			
19	Ограничения:	0	0	0	0	0				
20										
21										
22										
23										

Задача о назначении №2

Рис. 10.16. Исходные данные для решения задачи о назначении методом минимального отклонения от идеальной точки на втором этапе

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о назначении методом минимального отклонения от идеальной точки на втором этапе имеет следующий вид (рис. 10.16).

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$I$7`.
2. Для группы **Равной:** оставить вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$B$14:$F$18`.

В дополнительном окне **Параметры поиска решения** следует обязательно убрать отметку **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами представлен на рис. 10.17.

После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи о назначении, для этого следует нажать кнопку **Выполнить**. После выполнения

расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.18).

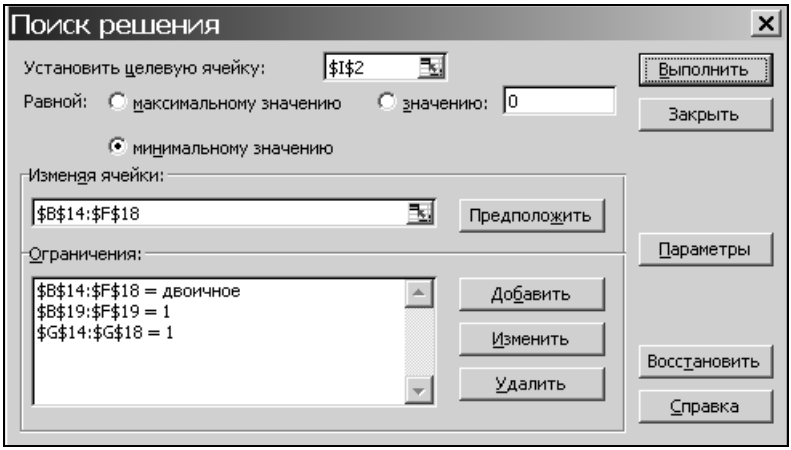


Рис. 10.17. Параметры мастера поиска решения и базовые ограничения для окончательного решения двухкритериальной задачи о назначении методом минимального отклонения от идеальной точки

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I	J
1	Затраты -	Коэффициенты первой целевой функции:					Значение ЦФ 1:	Значение ЦФ 2:	Новая ЦФ:	
2		5	10	9	14	6	51,000	835,000	164,000	
3		13	15	11	19	17	43	935		
4		7	14	12	8	10				
5		8	11	6	7	9				
6		15	12	17	13	16				
7	Прибыль -	Коэффициенты второй целевой функции:								
8		150	95	90	210	100				
9		185	200	170	140	175				
10		125	110	140	115	105				
11		80	180	85	100	135				
12		110	145	230	125	220				
13	Переменные:	*Xi1*	*Xi2*	*Xi3*	*Xi4*	*Xi5*	Ограничения:			
14	*X1j*	1	0	0	0	0	1			
15	*X2j*	0	0	1	0	0	1			
16	*X3j*	0	0	0	1	0	1			
17	*X4j*	0	1	0	0	0	1			
18	*X5j*	0	0	0	0	1	1			
19	Ограничения:	1	1	1	1	1				
20										
21										
22										
23										

Задача о назначении №2

Рис. 10.18. Результат окончательного решения двухкритериальной задачи о назначении методом минимального отклонения от идеальной точки

Результатом решения рассматриваемой задачи о назначении являются найденные оптимальные значения переменных: $x_{11} = 1$, $x_{23} = 1$, $x_{34} = 1$, $x_{42} = 1$, $x_{55} = 1$, остальные переменные равны 0. Найденному решению соответствуют

значения целевых функций: f_1^{opt} =51 и f_2^{opt} = 835. Анализ найденного решения показывает, что оно полностью совпадает с решением, найденным ранее с помощью метода уступок.

Далее приводится описание метода аддитивной свертки на основе задания весов отдельных целевых функций. Данный метод также может быть использован для решения двухкритериальной задачи о назначении.

10.3.4. Решение двухкритериальной задачи о назначении с помощью программы MS Excel методом аддитивной свертки

Для решения задачи о назначении с двумя целевыми функциями с помощью программы MS Excel методом аддитивной свертки на основе задания весов отдельных целевых функций будем использовать те же значения параметров рассматриваемой ранее конкретной задачи о назначении. Дополнительно следует задать количественные значения *весов* исходным целевым функциям, которые удовлетворяют условиям, отмеченным в *разд. 9.1.2*. Если предположить, что с точки зрения предпочтений директора фирмы, принимающего решение о назначении на должности, оба критерия равноценны, то такими весами являются значения: $\alpha_1 = 0{,}5$ и $\alpha_2 = 0{,}5$.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная булева оптимизация создадим новый рабочий лист с именем: Задача о назначении №3. Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 10.3.2* при решении двухкритериальной задачи о назначении методом уступок. В то же время, чтобы сократить дублирование действий, скопируем данные из листа с именем Задача о назначении №2 (рис. 10.16) в лист с именем Задача о назначении №3.

Для решения двухкритериальной задачи о назначении методом аддитивной свертки на основе задания весов отдельных целевых функций внесем следующие изменения в лист с именем Задача о назначении №3:

1. Удалим данные из ячеек **G3** и **H3**, необходимость использования которых отпала.
2. Переместим данные из ячеек **H1:I2** в ячейки **G3:G5** (рис. 10.19).
3. В ячейку **G6** введем формулу: `=0,5*G2/43-0,5*H2/935`, которая представляет аддитивную свертку исходных целевых функций (ЦФ) с использованием заданных ранее весов критериев и шкалированием критериев относительно минимального значения первой ЦФ и максимального значения второй ЦФ.

Примечание

В выражении для аддитивной свертки исходных целевых функций вторая ЦФ должна быть взята со знаком "–", поскольку в математической постановке задачи она подлежит максимизации.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о назначении методом аддитивной свертки имеет следующий вид (рис. 10.19).

МногокритериальнаяБулеваОптимизация.xls

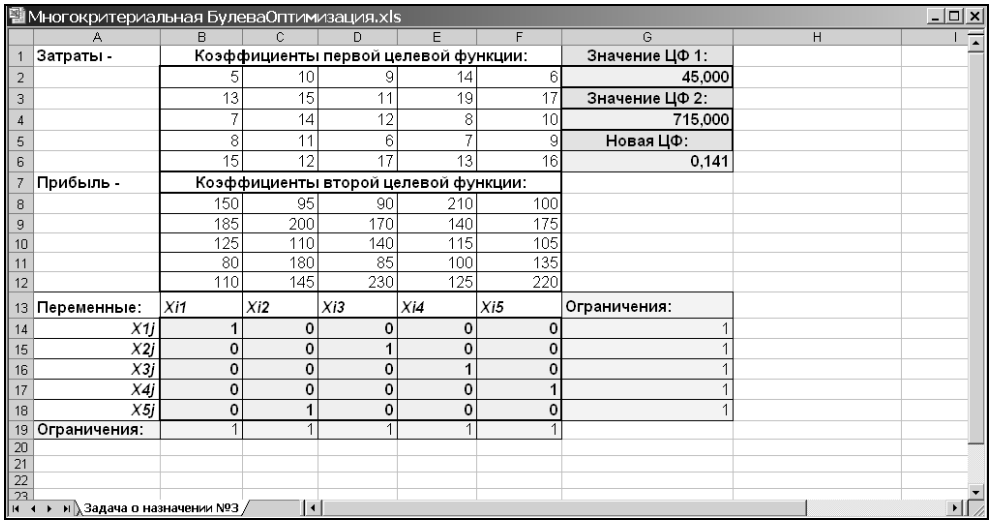
	A	B	C	D	E	F	G
1	Затраты -	Коэффициенты первой целевой функции:					Значение ЦФ 1:
2		5	10	9	14	6	=СУММПРОИЗВ($B$2:$F$6;$B$14:$F$18)
3		13	15	11	19	17	Значение ЦФ 2:
4		7	14	12	8	10	=СУММПРОИЗВ($B$8:$F$12;$B$14:$F$18)
5		8	11	6	7	9	Новая ЦФ:
6		15	12	17	13	16	=0,5*G2/43+0,5*G4/935
7	Прибыль -	Коэффициенты второй целевой функции:					
8		150	95	90	210	100	
9		185	200	170	140	175	
10		125	110	140	115	105	
11		80	180	85	100	135	
12		110	145	230	125	220	
13	Переменные:	*Xi1*	*Xi2*	*Xi3*	*Xi4*	*Xi5*	Ограничения:
14	*X1j*						=СУММ(B14:F14)
15	*X2j*						=СУММ(B15:F15)
16	*X3j*						=СУММ(B16:F16)
17	*X4j*						=СУММ(B17:F17)
18	*X5j*						=СУММ(B18:F18)
19	Ограничения	=СУММ(B14:B18	=СУММ(C14:C18	=СУММ(D14:D18	=СУММ(E14:E18	=СУММ(F14:F18	
20							
21							
22							
23							
24							

Задача о назначении №3

Рис. 10.19. Исходные данные для решения двухкритериальной задачи о назначении методом аддитивной свертки

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**. В диалоговом окне **Поиск решения** следует задать целевую ячейку, диапазон изменения ячеек и ограничения задачи таким же образом, как было рассмотрено в *разд. 10.3.3* при решении задачи методом минимального отклонения от идеальной точки. В дополнительном окне **Параметры поиска решения** следует выбрать отметку **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами аналогичен окну параметров мастера поиска, представленному на рис. 10.17.

После задания ограничений и целевой функции на втором этапе можно приступить к поиску решения двухкритериальной задачи о назначении, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.20).



МногокритериальнаяБулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I
1	Затраты -	Коэффициенты первой целевой функции:					Значение ЦФ 1:		
2		5	10	9	14	6	45,000		
3		13	15	11	19	17	Значение ЦФ 2:		
4		7	14	12	8	10	715,000		
5		8	11	6	7	9	Новая ЦФ:		
6		15	12	17	13	16	0,141		
7	Прибыль -	Коэффициенты второй целевой функции:							
8		150	95	90	210	100			
9		185	200	170	140	175			
10		125	110	140	115	105			
11		80	180	85	100	135			
12		110	145	230	125	220			
13	Переменные:	Xi1	Xi2	Xi3	Xi4	Xi5	Ограничения:		
14	X1j	1	0	0	0	0	1		
15	X2j	0	0	1	0	0	1		
16	X3j	0	0	0	1	0	1		
17	X4j	0	0	0	0	1	1		
18	X5j	0	1	0	0	0	1		
19	Ограничения:	1	1	1	1	1			
20									
21									
22									
23									

Задача о назначении №3

Рис. 10.20. Результат окончательного решения двухкритериальной задачи о назначении методом аддитивной свертки

Результатом решения рассматриваемой задачи о назначении являются найденные оптимальные значения переменных: $x_{11} = 1$, $x_{23} = 1$, $x_{34} = 1$, $x_{45} = 1$, $x_{52} = 1$, остальные переменные равны 0. Найденному решению соответствуют значения целевых функций: $f_1^{opt} = 45$ и $f_2^{opt} = 715$. Анализ найденного решения показывает, что оно отличается от решений, найденных ранее с помощью метода уступок и метода минимального отклонения. При этом значение первой ЦФ у данного решения лучше, а значение второй ЦФ — хуже, чем у найденных ранее решений.

Примечание

В данном случае можно также заметить, что шкалирование критериев не позволяет избавиться от недостатка метода аддитивной свертки, связанного с неравноценным характером влияния абсолютных значений отдельных целевых функций на итоговый результат, особенно явно проявляющийся при решении задач булева программирования. В этом случае можно порекомендовать дополнительно использовать масштабирующий коэффициент 0,1 для второй целевой функции, что позволит выровнять порядок ее значений относительно первой целевой функции. Выполнить соответствующие расчеты и сравнить полученные результаты предлагается читателям самостоятельно в качестве упражнения.

10.4. Двухкритериальная задача о наборе высоты и скорости

В настоящей главе рассматриваются методы решения двухкритериальной задачи о наборе высоты и скорости некоторым летательным аппаратом, например, самолетом или вертолетом. Поскольку однокритериальный вариант данной задачи ранее не рассматривался, в данной главе приводится содержательная ее постановка. Хотя особенности постановки данной задачи позволяют отнести ее к классу задач оптимизации на графах, ее решение с помощью программы MS Excel основано на математической постановке в форме модели многокритериального булева программирования.

10.4.1. Содержательная постановка индивидуальной задачи о наборе высоты и скорости летательным аппаратом

Некоторый летательный аппарат, находящийся на высоте h_0 и имеющий скорость v_0, должен подняться на заданную высоту h_f, а его скорость должна быть доведена до значения v_f. Предполагается, что по техническим условиям летательный аппарат одновременно может либо набирать высоту, либо увеличивать скорость. При этом имеется 3 последовательных режима набора высоты и 3 последовательных режима увеличения скорости. Известен расход топлива и время, необходимые для выполнения маневров в различных режимах. Схема маневров летательного аппарата в различных режимах может быть представлена в виде ориентированного графа следующего вида (рис. 10.21).

Рядом с дугами данного графа указаны значения расхода топлива в *тоннах* и времени в *минутах*, необходимые для выполнения маневра в различных режимах. При этом вершина графа с номером 1 соответствует начальной высоте h_0 и скорости v_0 летательного аппарата, а вершина графа с номером 16 соответствует заданной высоте h_f и скорости v_f летательного аппарата.

Требуется определить последовательность режимов набора высоты и скорости, так чтобы достичь заданной высоты и заданной скорости, обеспечив при этом минимальный расход топлива и минимальное время выполнения всего маневра. Другими словами, применительно к схеме выполнения маневров летательным аппаратом в различных режимах требуется определить минимальный по расходу топлива и времени путь, соединяющий вершину графа с номером 1 с вершиной 16.

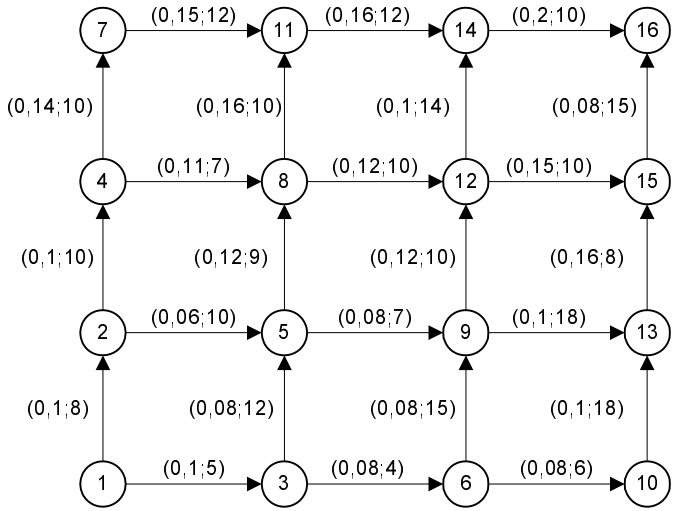


Рис. 10.21. Схема выполнения маневров летательным аппаратом в различных режимах

10.4.2. Математическая постановка двухкритериальной задачи о наборе высоты и скорости

В качестве переменных математической модели двухкритериальной задачи о наборе высоты и скорости рассмотрим следующие булевы переменные: $x_{ij} \in \{0, 1\}$, которые будут соответствовать выбору отдельного варианта выполнения маневра. Применительно к схеме выполнения маневров в этом случае $x_{ij} = 1$ означает, что дуга (i, j) входит в минимальный по расходу топлива и времени путь, и $x_{ij} = 0$ в противном случае. Тогда математическая постановка *двухкритериальной задачи о наборе высоты и скорости* может быть записана в следующем виде:

$$\begin{aligned} &0{,}1x_{12} + 0{,}1x_{13} + 0{,}1x_{24} + 0{,}06x_{25} + 0{,}08x_{35} + \\ &+ 0{,}08x_{36} + 0{,}14x_{47} + 0{,}11x_{48} + 0{,}12x_{58} + 0{,}08x_{59} + \\ &+ 0{,}08x_{69} + 0{,}08x_{6,10} + 0{,}15x_{7,11} + 0{,}16x_{8,11} + 0{,}12x_{8,12} + \\ &+ 0{,}12x_{9,12} + 0{,}1x_{9,13} + 0{,}1x_{10,13} + 0{,}16x_{11,14} + 0{,}1x_{12,14} + \\ &+ 0{,}15x_{12,15} + 0{,}16x_{13,15} + 0{,}2x_{14,16} + 0{,}08x_{15,16} \to \min\limits_{x \in \Delta_\beta}, \end{aligned} \qquad (10.4.1)$$

$$
\begin{aligned}
&8x_{12} + 5x_{13} + 10x_{24} + 10x_{25} + 12x_{35} + \\
&+ 4x_{36} + 10x_{47} + 7x_{48} + 9x_{58} + 7x_{59} + \\
&+ 15x_{69} + 6x_{6,10} + 12x_{7,11} + 10x_{8,11} + 10x_{8,12} + \\
&+ 10x_{9,12} + 18x_{9,13} + 18x_{10,13} + 12x_{11,14} + 14x_{12,14} + \\
&+ 10x_{12,15} + 8x_{13,15} + 10x_{14,16} + 15x_{15,16} \to \min\limits_{x \in \Delta_\beta},
\end{aligned}
\tag{10.4.2}
$$

где множество допустимых альтернатив Δ_β формируется следующей системой ограничений типа неравенств:

$$
\begin{cases}
x_{12} + x_{13} = 1; \\
x_{12} - x_{24} - x_{25} = 0; \\
x_{13} - x_{35} - x_{36} = 0; \\
x_{24} - x_{47} - x_{48} = 0; \\
x_{25} + x_{35} - x_{58} - x_{59} = 0; \\
x_{36} - x_{69} - x_{6,10} = 0; \\
x_{47} - x_{7,11} = 0; \\
x_{48} + x_{58} - x_{8,11} - x_{8,12} = 0; \\
x_{59} + x_{69} - x_{9,12} - x_{9,13} = 0; \\
x_{6,10} - x_{10,13} = 0; \\
x_{7,11} + x_{8,11} - x_{11,14} = 0; \\
x_{8,12} + x_{9,12} - x_{12,14} - x_{12,15} = 0; \\
x_{9,13} + x_{10,13} - x_{13,15} = 0; \\
x_{11,14} + x_{12,14} - x_{14,16} = 0; \\
x_{12,15} + x_{13,15} - x_{15,16} = 0; \\
x_{14,16} + x_{15,16} = 1; \\
x_{ij} \in \{0, 1\} \; (\forall i, j \in \{1, 2, ..., 16\}).
\end{cases}
\tag{10.4.3}
$$

Заметим, что те переменные x_{ij}, для которых весовая функция дуг для исходного графа (рис. 10.21) не определена или равна 0, не входят в математическую постановку рассматриваемой задачи (10.4.1)—(10.4.3). Математическая модель (10.4.1)—(10.4.3) относится к классу задач многокритериального булева программирования, которая может быть решена с помощью рассмотренных ранее методов решения задач этого класса. При этом общая схема

данных методов останется прежней, изменятся лишь особенности поиска решений соответствующих однокритериальных задач оптимизации с помощью программы MS Excel.

10.4.3. Решение двухкритериальной задач и о наборе высоты и скорости с помощью программы MS Excel методом уступок

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная булева оптимизация и создадим рабочий лист с именем Набор высоты и скорости. Решение данной задачи многокритериальной оптимизации методом уступок будет состоять из 2-х этапов. На первом этапе необходимо решить обычную задачу оптимизации, используя в качестве критериальной функции первую целевую функцию (10.4.1) в предположении, что она является более важной, чем вторая целевая функция. Для решения этой задачи выполним следующие подготовительные действия:

1. Внесем необходимые надписи в ячейки **A1:H1**, **F2:F17**, **H3** (рис. 10.22). Следует отметить, что конкретное содержание этих надписей не оказывает влияния на решение рассматриваемой задачи.
2. В ячейки **A2:A25** введем индексы начальных вершин, а в ячейки **B2:B25** — индексы конечных вершин всех имеющихся дуг исходного графа.
3. В ячейки **C2:C25** введем значения коэффициентов первой целевой функции (10.4.1).
4. В ячейки **D2:D25** введем значения коэффициентов второй целевой функции (10.4.2).
5. В ячейку **H2** введем формулу: `=СУММПРОИЗВ($C$2:$C$25;$E$2:$E$25)`, которая представляет собой выражение для первой целевой функции (10.4.1).
6. В ячейку **H4** введем формулу: `=СУММПРОИЗВ($D$2:$D$25;$E$2:$E$25)`, которая представляет собой выражение для второй целевой функции (10.4.2).
7. В ячейку **G2** введем формулу: `=E2+E3`, которая представляет собой левую часть первого ограничения (10.4.3).
8. В ячейку **G3** введем выражение для левой части второго ограничения: `=E2-(E4+E5)`.

9. В ячейку **G4** введем выражение для левой части третьего ограничения: `=E3-(E6+E7)`.
10. В ячейку **G5** введем выражение для левой части четвертого ограничения: `=E4-(E8+E9)`.
11. В ячейку **G6** введем выражение для левой части пятого ограничения: `=E5+E6-(E10+E11)`.
12. В ячейку **G7** введем выражение для левой части шестого ограничения: `=E7-(E12+E13)`.
13. В ячейку **G8** введем выражение для левой части седьмого ограничения: `=E8-E14`.
14. В ячейку **G9** введем выражение для левой части восьмого ограничения: `=E9+E10-(E15+E16)`.
15. В ячейку **G10** введем выражение для левой части девятого ограничения: `=E11+E12-(E17+E18)`.
16. В ячейку **G11** введем выражение для левой части десятого ограничения: `=E13-E19`.
17. В ячейку **G12** введем выражение для левой части 11-го ограничения: `=E14+E15-E20`.
18. В ячейку **G13** введем выражение для левой части 12-го ограничения: `=E16+E17-(E21+E22)`.
19. В ячейку **G14** введем выражение для левой части 13-го ограничения: `=E18+E19-E23`.
20. В ячейку **G15** введем выражение для левой части 14-го ограничения: `=E20+E21-E24`.
21. В ячейку **G16** введем выражение для левой части 15-го ограничения: `=E22+E23-E25`.
22. И, наконец, в ячейку **G17** введем выражение для левой части 16-го ограничения: `=E24+E25`.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения двухкритериальной задачи о наборе высоты и скорости методом уступок на первом этапе имеет следующий вид (рис. 10.22).

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения, для чего необходимо выполнить операцию главного меню: **Сервис** | **Поиск решения**.

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H
1	*vi*	*vj*	*cij*	*dij*	Переменные:	№ *V*	Ограничения:	Значение ЦФ 1:
2	1	2	0,1	8		1	=E2+E3	=СУММПРОИЗВ(C$2:C$25;$E$2:$E$25)
3	1	3	0,1	5		2	=E2-(E4+E5)	Значение ЦФ 2:
4	2	4	0,1	10		3	=E3-(E6+E7)	=СУММПРОИЗВ(D$2:D$25;$E$2:$E$25)
5	2	5	0,06	10		4	=E4-(E8+E9)	
6	3	5	0,08	12		5	=E5+E6-(E10+E11)	
7	3	6	0,08	4		6	=E7-(E12+E13)	
8	4	7	0,14	10		7	=E8-E14	
9	4	8	0,11	7		8	=E9+E10-(E15+E16)	
10	5	8	0,12	9		9	=E11+E12-(E17+E18)	
11	5	9	0,08	7		10	=E13-E19	
12	6	9	0,08	15		11	=E14+E15-E20	
13	6	10	0,08	6		12	=E16+E17-(E21+E22)	
14	7	11	0,15	12		13	=E18+E19-E23	
15	8	11	0,16	10		14	=E20+E21-E24	
16	8	12	0,12	10		15	=E22+E23-E25	
17	9	12	0,12	10		16	=E24+E25	
18	9	13	0,1	18				
19	10	13	0,1	18				
20	11	14	0,16	12				
21	12	14	0,1	14				
22	12	15	0,15	10				
23	13	15	0,16	8				
24	14	16	0,2	10				
25	15	16	0,08	15				

Набор высоты и скорости

Рис. 10.22. Исходные данные для решения двухкритериальной задачи о наборе высоты и скорости методом уступок на первом этапе

После появления диалогового окна **Поиск решения** следует выполнить следующие действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$H$2`.
2. Для группы **Равной:** выбрать вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** ввести абсолютный адрес диапазона ячеек `$E$2:$E$25`.
4. Задать первое ограничение для рассматриваемой задачи. С этой целью выполнить следующие действия:
   - для задания первого ограничения в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$G$2**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры значение 1;

- для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

5. Аналогичным образом добавить 16-е ограничение (10.4.3), используя в качестве исходной ячейки **$G$17**.
6. Задать группу ограничений для промежуточных вершин рассматриваемой задачи. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$G$3:$G$16**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать равенство "=";
   - в качестве значения правой части ограничения ввести с клавиатуры значение 0;
   - для добавления первого ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.
7. Задать последнее ограничение на булевы значения переменных. С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать диапазон ячеек **$E$2:$E$25**, который должен отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать строку "**двоичн**";
   - в качестве значения правой части ограничения в поле с именем **Ограничение:** оставить без изменения вставленное программой значение "**двоичное**";
   - для добавления ограничения в дополнительном окне нажать кнопку с надписью **Добавить**;
   - в дополнительном окне **Параметры поиска решения** выбрать отметку Линейная модель.

Общий вид окна мастера поиска решения на первом этапе с заданными параметрами представлен на рис. 10.23.

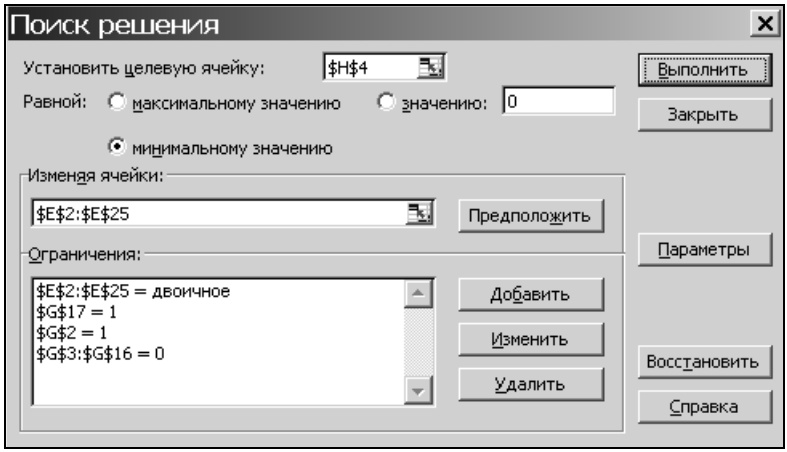


Рис. 10.23. Параметры мастера поиска решения и базовые ограничения первого этапа для двухкритериальной задачи о наборе высоты и скорости

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	*vi*	*vj*	*cij*	*dij*	Переменные:	№ *V*	Ограничения:	Значение ЦФ 1:					
2	1	2	0,1	8	1	1	1	0,58					
3	1	3	0,1	5	0	2	0	Значение ЦФ 2:					
4	2	4	0,1	10	0	3	0	66					
5	2	5	0,06	10	1	4	0						
6	3	5	0,08	12	0	5	0						
7	3	6	0,08	4	0	6	0						
8	4	7	0,14	10	0	7	0						
9	4	8	0,11	7	0	8	0						
10	5	8	0,12	9	0	9	0						
11	5	9	0,08	7	1	10	0						
12	6	9	0,08	15	0	11	0						
13	6	10	0,08	6	0	12	0						
14	7	11	0,15	12	0	13	0						
15	8	11	0,16	10	0	14	0						
16	8	12	0,12	10	0	15	0						
17	9	12	0,12	10	0	16	1						
18	9	13	0,1	18	1								
19	10	13	0,1	18	0								
20	11	14	0,16	12	0								
21	12	14	0,1	14	0								
22	12	15	0,15	10	0								
23	13	15	0,16	8	1								
24	14	16	0,2	10	0								
25	15	16	0,08	15	1								
26													

Набор высоты и скорости

Рис. 10.24. Результат количественного решения двухкритериальной задачи о наборе высоты и скорости методом уступок на первом этапе

После задания ограничений и целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.24).

Результатом решения рассматриваемой задачи о наборе высоты и скорости являются найденные оптимальные значения переменных: $x_{12} = 1$, $x_{25} = 1$, $x_{59} = 1$, $x_{9,13} = 1$, $x_{13,15} = 1$, $x_{15,16} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствует минимальное значение первой целевой функции: $f_{opt} = 0{,}58$ *тонн* горючего.

Предположим, что, исходя из поставленной целевой задачи, оказывается возможной уступка по первой целевой функции, равная 0,1 *тонн* горючего. Тем самым можно перейти ко второму этапу решения данной задачи оптимизации методом уступок.

С этой целью следует рассмотреть дополнительное ограничение следующего вида:

$$\begin{aligned} &0{,}1x_{12} + 0{,}1x_{13} + 0{,}1x_{24} + 0{,}06x_{25} + 0{,}08x_{35} + \\ &+ 0{,}08x_{36} + 0{,}14x_{47} + 0{,}11x_{48} + 0{,}12x_{58} + 0{,}08x_{59} + \\ &+ 0{,}08x_{69} + 0{,}08x_{6,10} + 0{,}15x_{7,11} + 0{,}16x_{8,11} + 0{,}12x_{8,12} + \\ &+ 0{,}12x_{9,12} + 0{,}1x_{9,13} + 0{,}1x_{10,13} + 0{,}16x_{11,14} + 0{,}1x_{12,14} + \\ &+ 0{,}15x_{12,15} + 0{,}16x_{13,15} + 0{,}2x_{14,16} + 0{,}08x_{15,16} \le 0{,}59, \end{aligned} \qquad (10.4.4)$$

в котором правая часть равна значению суммы: $f_1^{opt} + 0{,}1$, а знак ограничения выбран таким образом, чтобы соответствовать решенной задаче минимизации по первой целевой функции.

Для окончательного решения двухкритериальной задачи о наборе высоты и скорости внесем следующие дополнительные данные в лист с именем Задача о наборе высоты и скорости:

1. Введем дополнительную надпись в ячейку **G18**.
2. Скопируем формулу из ячейки **H2** в ячейку **G19**, которая представляет левую часть дополнительного ограничения (10.4.4).
3. В ячейку **H19** введем значение 0,59, которое представляет правую часть дополнительного ограничения (10.4.4).
4. Удалим значения переменных из ячеек **E2:E25**, полученные в результате решения задачи на первом этапе.

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**. После появления диалогового окна **Поиск решения** следует выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$H$4`.

2. Для группы **Равной:** оставить вариант поиска решения — **минимальному значению**.
3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$E$2:$E$25`.
4. Задать дополнительное ограничение (10.4.4). С этой целью выполнить следующие действия:
   - в исходном диалоговом окне **Поиск решения** нажать кнопку с надписью **Добавить**;
   - в появившемся дополнительном окне выбрать ячейку **$G$19**, которая должна отобразиться в поле с именем **Ссылка на ячейку**;
   - в качестве знака ограничения из выпадающего списка выбрать нестрогое неравенство "<=";
   - в качестве значения правой части ограничения выбрать ячейку **$H$19**;
   - для добавления нового ограничения в дополнительном окне нажать кнопку с надписью **Добавить**.

В дополнительном окне **Параметры поиска решения** следует оставить выбранной отметку **Линейная модель**. После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи о наборе высоты и скорости, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.25).

Результатом решения рассматриваемой задачи о наборе высоты и скорости являются найденные оптимальные значения переменных: $x_{12} = 1$, $x_{25} = 1$, $x_{59} = 1$, $x_{9,12} = 1$, $x_{12,15} = 1$, $x_{15,16} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствуют значения целевых функций: $f_1^{opt} = 0{,}59$ и $f_2^{opt} = 60$.

Анализ найденного решения показывает, что при выборе схемы выполнения маневра по набору высоты и скорости летательным аппаратом следует принять решение о следующей последовательности смены режимов: (1, 2), (2, 5), (5, 9), (9,12), (12, 15), (15, 16). При этом общие затраты топлива составят 0,59 *тонн*, а общее время выполнения маневра составит 1 *час*.

Примечание

Заинтересованным читателям в качестве упражнения предлагается решить рассмотренную двухкритериальную задачу о наборе высоты и скорости методом уступок, изменив порядок важности критериев. Полученные результаты сравнить.

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	*vi*	*vj*	*cij*	*dij*	Переменные:	№ *V*	Ограничения:	Значение ЦФ 1:					
2	1	2	0,1	8	1	1	1	0,59					
3	1	3	0,1	5	0	2	0	Значение ЦФ 2:					
4	2	4	0,1	10	0	3	0	60					
5	2	5	0,06	10	1	4	0						
6	3	5	0,08	12	0	5	0						
7	3	6	0,08	4	0	6	0						
8	4	7	0,14	10	0	7	0						
9	4	8	0,11	7	0	8	0						
10	5	8	0,12	9	0	9	0						
11	5	9	0,08	7	1	10	0						
12	6	9	0,08	15	0	11	0						
13	6	10	0,08	6	0	12	0						
14	7	11	0,15	12	0	13	0						
15	8	11	0,16	10	0	14	0						
16	8	12	0,12	10	0	15	0						
17	9	12	0,12	10	1	16	1						
18	9	13	0,1	18	0		Дополнительное ограничение:						
19	10	13	0,1	18	0		0,59	0,59					
20	11	14	0,16	12	0								
21	12	14	0,1	14	0								
22	12	15	0,15	10	1								
23	13	15	0,16	8	0								
24	14	16	0,2	10	0								
25	15	16	0,08	15	1								
26													

Набор высоты и скорости

Рис. 10.25. Результат окончательного решения двухкритериальной задачи о наборе высоты и скорости методом уступок

10.4.4. Решение двухкритериальной задачи о наборе высоты и скорости с помощью программы MS Excel методом минимального отклонения

Для решения задачи о наборе высоты и скорости с двумя целевыми функциями с помощью программы MS Excel методом минимального отклонения от идеальной точки будем использовать те же конкретные значения параметров рассматриваемой индивидуальной задачи о наборе высоты и скорости.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная булева оптимизация создадим новый рабочий лист с именем: Набор высоты и скорости №2. Решение данной задачи многокритериальной оптимизации методом минимального отклонения от идеальной точки будет состоять из 2-х этапов. На первом этапе необходимо решить две обычных задачи оптимизации, используя в качестве критериальных функций соответственно целевые функции (10.4.1) и (10.4.2).

Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 10.3.3* при

решении двухкритериальной задачи о наборе высоты и скорости методом уступок. Чтобы сократить дублирование действий, скопируем данные из листа с именем Набор высоты и скорости в лист с именем Набор высоты и скорости №2.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о наборе высоты и скорости методом минимального отклонения от идеальной точки на первом этапе должен иметь вид, аналогичный изображенному на рис. 10.22.

Для дальнейшего решения задачи на первом этапе следует вызвать мастер поиска решения и задать соответствующие параметры. Общий вид окна мастера поиска решения с заданными параметрами для решения задачи о наборе высоты и скорости по первому критерию представлен на рис. 10.23.

После задания ограничений и первой целевой функции можно приступить к поиску численного решения, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет вид, изображенный ранее на рис. 10.24. Результатом решения однокритериальной задачи о наборе высоты и скорости на первом этапе является найденное значение первой целевой функции: $f_1^{opt} = 0{,}58$.

Диалоговое окно задания параметров для мастера поиска решения задачи о наборе высоты и скорости по второй целевой функции представлено на рис. 10.26.

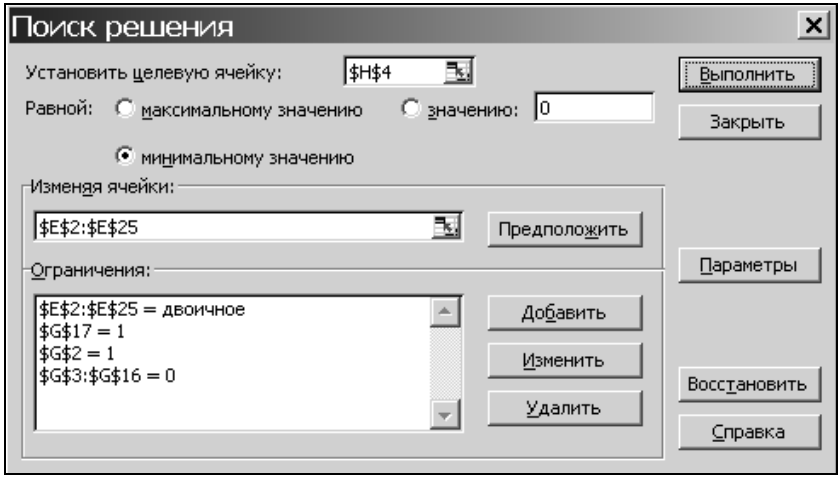


Рис. 10.26. Параметры мастера поиска решения и базовые ограничения для второй целевой функции двухкритериальной задачи о наборе высоты и скорости

После задания ограничений и второй целевой функции можно приступить к поиску численного решения по второму критерию, для этого следует предварительно удалить ранее найденные значения переменных и нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено соответствующее количественное решение. Результатом решения однокритериальной задачи о наборе высоты и скорости на первом этапе является найденное минимальное значение второй целевой функции: $f_2^{opt} = 56$.

Для окончательного решения двухкритериальной задачи о наборе высоты и скорости на втором этапе внесем следующие дополнительные данные в лист с именем Задача о наборе высоты и скорости №2:

1. Введем дополнительную надпись в ячейку **H7**.
2. В ячейку **H5** введем найденное на первом этапе оптимальное значение первой целевой функции $f_1^{opt} = 0{,}58$.
3. В ячейку **H6** введем найденное на первом этапе оптимальное значение второй целевой функции $f_2^{opt} = 56$.
4. В ячейку **H8** введем формулу: `=(100* (H2-H5))^2+(H4-H6))^2`, которая представляет минимальное отклонение от найденной идеальной точки, полученной в результате решения задачи на первом этапе.

Примечание

Следует отметить, что с целью выравнивания порядка целевых функций в выражении для новой ЦФ используется по первой целевой функции масштабирующий коэффициент 100.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о наборе высоты и скорости методом минимального отклонения от идеальной точки на втором этапе имеет следующий вид (рис. 10.27).

Для дальнейшего решения задачи на втором этапе следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**.

После появления диалогового окна **Поиск решения** следует выполнить следующие дополнительные действия:

1. В поле с именем **Установить целевую ячейку:** ввести абсолютный адрес ячейки `$H$8`.
2. Для группы **Равной:** оставить вариант поиска решения — **минимальному значению**.

3. В поле с именем **Изменяя ячейки:** оставить абсолютный адрес ячеек `$E$2:$E$25`.

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H
1	*vi*	*vj*	*cij*	*dij*	Переменные:	№ *V*	Ограничения:	Значение ЦФ 1:
2	1	2	0,1	8	0	1	=E2+E3	=СУММПРОИЗВ($C$2:$C$25;$E$2:$E$25)
3	1	3	0,1	5	1	2	=E2-(E4+E5)	Значение ЦФ 2:
4	2	4	0,1	10	0	3	=E3-(E6+E7)	=СУММПРОИЗВ($D$2:$D$25;$E$2:$E$25)
5	2	5	0,06	10	0	4	=E4-(E8+E9)	0,58
6	3	5	0,08	12	0	5	=E5+E6-(E10+E11)	56
7	3	6	0,08	4	1	6	=E7-(E12+E13)	Новая ЦФ:
8	4	7	0,14	10	0	7	=E8-E14	=(100* (H2-H5))^2+(H4-H6)^2
9	4	8	0,11	7	0	8	=E9+E10-(E15+E16)	
10	5	8	0,12	9	0	9	=E11+E12-(E17+E18)	
11	5	9	0,08	7	0	10	=E13-E19	
12	6	9	0,08	15	0	11	=E14+E15-E20	
13	6	10	0,08	6	1	12	=E16+E17-(E21+E22)	
14	7	11	0,15	12	0	13	=E18+E19-E23	
15	8	11	0,16	10	0	14	=E20+E21-E24	
16	8	12	0,12	10	0	15	=E22+E23-E25	
17	9	12	0,12	10	0	16	=E24+E25	
18	9	13	0,1	18	0			
19	10	13	0,1	18	1			
20	11	14	0,16	12	0			
21	12	14	0,1	14	0			
22	12	15	0,15	10	0			
23	13	15	0,16	8	1			
24	14	16	0,2	10	0			
25	15	16	0,08	15	1			
26								

Набор высоты и скорости | Набор высоты и скорости №2

Рис. 10.27. Исходные данные для решения задачи о наборе высоты и скорости методом минимального отклонения от идеальной точки на втором этапе

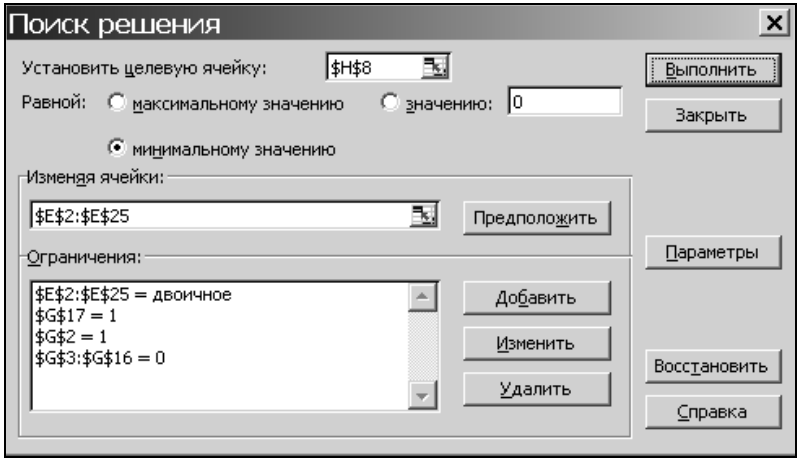


Рис. 10.28. Параметры мастера поиска решения и базовые ограничения для окончательного решения двухкритериальной задачи о наборе высоты и скорости методом минимального отклонения от идеальной точки

В дополнительном окне **Параметры поиска решения** следует обязательно убрать отметку **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами представлен на рис. 10.28.

После задания ограничений и целевой функции на втором этапе можно приступить к поиску окончательного решения двухкритериальной задачи о наборе высоты и скорости, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.29).

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	*vi*	*vj*	*cij*	*dij*	Переменные:	№ *V*	Ограничения:	Значение ЦФ 1:					
2	1	2	0,1	8	1	1	1	0,59					
3	1	3	0,1	5	0	2	0	Значение ЦФ 2:					
4	2	4	0,1	10	0	3	0	60					
5	2	5	0,06	10	1	4	0	0,58					
6	3	5	0,08	12	0	5	0	56					
7	3	6	0,08	4	0	6	0	Новая ЦФ:					
8	4	7	0,14	10	0	7	0	17					
9	4	8	0,11	7	0	8	0						
10	5	8	0,12	9	0	9	0						
11	5	9	0,08	7	1	10	0						
12	6	9	0,08	15	0	11	0						
13	6	10	0,08	6	0	12	0						
14	7	11	0,15	12	0	13	0						
15	8	11	0,16	10	0	14	0						
16	8	12	0,12	10	0	15	0						
17	9	12	0,12	10	1	16	1						
18	9	13	0,1	18	0								
19	10	13	0,1	18	0								
20	11	14	0,16	12	0								
21	12	14	0,1	14	0								
22	12	15	0,15	10	1								
23	13	15	0,16	8	0								
24	14	16	0,2	10	0								
25	15	16	0,08	15	1								
26													

Набор высоты и скорости | Набор высоты и скорости №2

Рис. 10.29. Результат окончательного решения двухкритериальной задачи о наборе высоты и скорости методом минимального отклонения от идеальной точки

Результатом решения рассматриваемой задачи о наборе высоты и скорости являются найденные оптимальные значения переменных: $x_{12} = 1$, $x_{25} = 1$, $x_{59} = 1$, $x_{9,\,12} = 1$, $x_{12,\,15} = 1$, $x_{15,\,16} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствуют значения целевых функций: $f_1^{opt} = 0{,}59$ и $f_2^{opt} = 60$. Анализ найденного решения показывает, что оно полностью совпадает с решением, найденным ранее с помощью метода уступок.

Далее приводится описание метода аддитивной свертки на основе задания весов отдельных целевых функций. Данный метод также может быть использован для решения двухкритериальной задачи о наборе высоты и скорости.

10.4.5. Решение двухкритериальной задачи о наборе высоты и скорости с помощью программы MS Excel методом аддитивной свертки

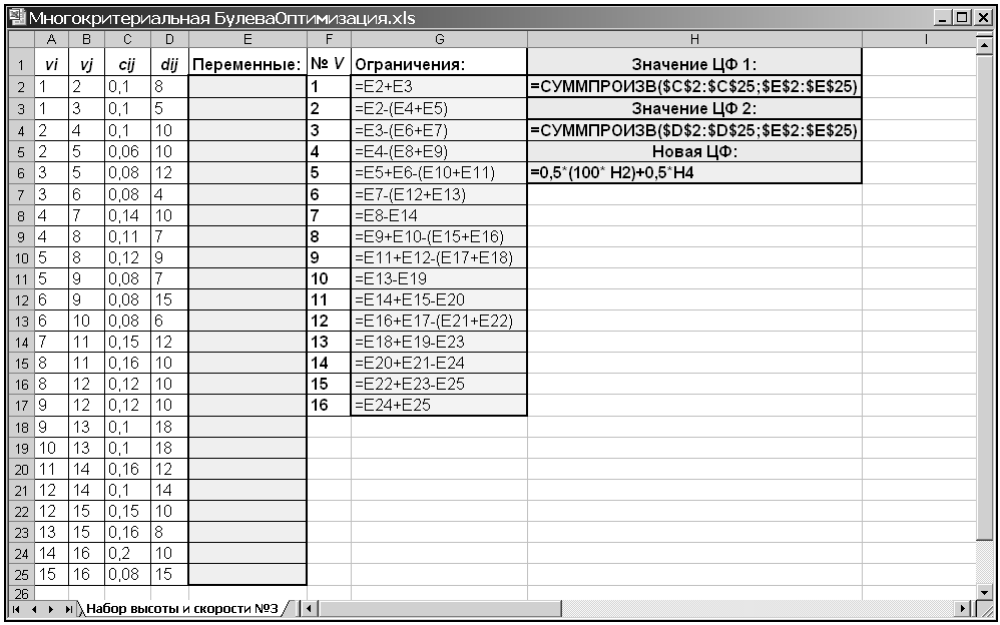
Для решения задачи о наборе высоты и скорости с двумя целевыми функциями с помощью программы MS Excel методом аддитивной свертки на основе задания весов отдельных целевых функций будем использовать те же значения параметров рассматриваемой ранее конкретной задачи о наборе высоты и скорости. Дополнительно следует задать количественные значения *весов* исходным целевым функциям, которые удовлетворяют условиям, отмеченным в *разд. 9.1.2*. Если предположить, что, с точки зрения выполнения целевой задачи летательным аппаратом, оба критерия равноценны, то такими весами являются значения: $\alpha_1 = 0{,}5$ и $\alpha_2 = 0{,}5$.

Для решения данной задачи с помощью программы MS Excel в книге с именем Многокритериальная булева оптимизация создадим новый рабочий лист с именем: Задача о наборе высоты и скорости №3. Для решения этой задачи выполним следующие подготовительные действия, которые во многом аналогичны действиям, рассмотренным в *разд. 10.4.3* при решении двухкритериальной задачи о наборе высоты и скорости методом уступок. В то же время, чтобы сократить дублирование действий, скопируем данные из листа с именем Задача о наборе высоты и скорости №2 (рис. 10.27) в лист с именем Задача о наборе высоты и скорости №3.

Для решения двухкритериальной задачи о наборе высоты и скорости методом аддитивной свертки на основе задания весов отдельных целевых функций внесем следующие изменения в лист с именем Задача о наборе высоты и скорости №3:

1. Удалим данные из ячеек **H5** и **H6**, необходимость использования которых отпала.
2. Переместим данные из ячеек **H7:H8** в ячейки **H5:H6** (рис. 10.30).
3. В ячейку **H6** введем формулу: `=0,5*(100* H2)+0,5*H4`, которая представляет аддитивную свертку исходных целевых функций с использованием заданных ранее весов критериев и масштабируемого коэффициента по первой целевой функции.

Внешний вид рабочего листа MS Office Excel 2003 с исходными данными для решения задачи о наборе высоты и скорости методом аддитивной свертки имеет следующий вид (рис. 10.30).



Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I
1	*vi*	*vj*	*cij*	*dij*	Переменные:	№ *V*	Ограничения:	Значение ЦФ 1:	
2	1	2	0,1	8		1	=E2+E3	=СУММПРОИЗВ($C$2:$C$25;$E$2:$E$25)	
3	1	3	0,1	5		2	=E2-(E4+E5)	Значение ЦФ 2:	
4	2	4	0,1	10		3	=E3-(E6+E7)	=СУММПРОИЗВ($D$2:$D$25;$E$2:$E$25)	
5	2	5	0,06	10		4	=E4-(E8+E9)	Новая ЦФ:	
6	3	5	0,08	12		5	=E5+E6-(E10+E11)	=0,5*(100* H2)+0,5*H4	
7	3	6	0,08	4		6	=E7-(E12+E13)		
8	4	7	0,14	10		7	=E8-E14		
9	4	8	0,11	7		8	=E9+E10-(E15+E16)		
10	5	8	0,12	9		9	=E11+E12-(E17+E18)		
11	5	9	0,08	7		10	=E13-E19		
12	6	9	0,08	15		11	=E14+E15-E20		
13	6	10	0,08	6		12	=E16+E17-(E21+E22)		
14	7	11	0,15	12		13	=E18+E19-E23		
15	8	11	0,16	10		14	=E20+E21-E24		
16	8	12	0,12	10		15	=E22+E23-E25		
17	9	12	0,12	10		16	=E24+E25		
18	9	13	0,1	18					
19	10	13	0,1	18					
20	11	14	0,16	12					
21	12	14	0,1	14					
22	12	15	0,15	10					
23	13	15	0,16	8					
24	14	16	0,2	10					
25	15	16	0,08	15					
26									

Набор высоты и скорости №3

Рис. 10.30. Исходные данные для решения двухкритериальной задачи о наборе высоты и скорости методом аддитивной свертки

Для дальнейшего решения задачи следует вызвать мастер поиска решения, для этого необходимо выполнить операцию главного меню: **Сервис | Поиск решения**. В диалоговом окне **Поиск решения** следует задать целевую ячейку, диапазон изменения ячеек и ограничения задачи таким же образом, как было рассмотрено в *разд. 10.4.4* при решении задачи методом минимального отклонения от идеальной точки. В дополнительном окне **Параметры поиска решения** следует выбрать отметку **Линейная модель**. Общий вид окна мастера поиска решения с заданными параметрами аналогичен окну параметров мастера поиска, представленному на рис. 10.28.

После задания ограничений и целевой функции на втором этапе можно приступить к поиску решения двухкритериальной задачи о наборе высоты и скорости, для этого следует нажать кнопку **Выполнить**. После выполнения расчетов программой MS Excel будет получено количественное решение, которое имеет следующий вид (рис. 10.31).

Результатом решения рассматриваемой задачи о наборе высоты и скорости являются найденные оптимальные значения переменных: $x_{13} = 1$, $x_{36} = 1$, $x_{6,\,10} = 1$, $x_{10,\,13} = 1$, $x_{13,\,15} = 1$, $x_{15,\,16} = 1$, остальные переменные равны 0. Найденному оптимальному решению соответствуют значения целевых функций: $f_1^{opt} = 0{,}6$ и $f_2^{opt} = 56$. Анализ найденного решения показывает, что оно отли-

чается от решений, найденных ранее с помощью метода уступок и метода минимального отклонения. При этом значение первой ЦФ у данного решения хуже, а значение второй ЦФ — наименьшее из всех возможных допустимых решений.

Многокритериальная БулеваОптимизация.xls

	A	B	C	D	E	F	G	H	I	J	K	L
1	*vi*	*vj*	*cij*	*dij*	Переменные:	№ *V*	Ограничения:	Значение ЦФ 1:				
2	1	2	0,1	8	0	1	1	0,60				
3	1	3	0,1	5	1	2	0	Значение ЦФ 2:				
4	2	4	0,1	10	0	3	0	56				
5	2	5	0,06	10	0	4	0	Новая ЦФ:				
6	3	5	0,08	12	0	5	0	58				
7	3	6	0,08	4	1	6	0					
8	4	7	0,14	10	0	7	0					
9	4	8	0,11	7	0	8	0					
10	5	8	0,12	9	0	9	0					
11	5	9	0,08	7	0	10	0					
12	6	9	0,08	15	0	11	0					
13	6	10	0,08	6	1	12	0					
14	7	11	0,15	12	0	13	0					
15	8	11	0,16	10	0	14	0					
16	8	12	0,12	10	0	15	0					
17	9	12	0,12	10	0	16	1					
18	9	13	0,1	18	0							
19	10	13	0,1	18	1							
20	11	14	0,16	12	0							
21	12	14	0,1	14	0							
22	12	15	0,15	10	0							
23	13	15	0,16	8	1							
24	14	16	0,2	10	0							
25	15	16	0,08	15	1							
26												

Набор высоты и скорости №3

Рис. 10.31. Результат окончательного решения двухкритериальной задачи о наборе высоты и скорости методом аддитивной свертки

Примечание

Можно показать, что множество недоминируемых альтернатив или множество Парето для рассматриваемой задачи о наборе высоты и скорости состоит из 3-х допустимых решений. Первое из них было получено при решении задачи нахождения минимального значения первой ЦФ. Второе решение было получено методами уступок и минимального отклонения от идеальной точки. И, наконец, третье решение соответствует решению однокритериальной задачи оптимизации по второй ЦФ. Выполнить соответствующие расчеты предлагается читателям самостоятельно в качестве упражнения.

Таким образом, анализ решения данной задачи различными методами позволяет сделать вывод о том, что в качестве ее окончательного решения каждый из методов предлагает некоторую недоминируемую по Парето альтернативу. Для окончательного выбора решения необходимы дополнительные предположения относительно предпочтений лиц, принимающих решение об этом

выборе. Фиксация подобных предпочтений в форме величин уступок, метрики отклонения от идеальной точки или весов аддитивной свертки целевых функций позволяет получить окончательное решение, принадлежащее множеству Парето.

10.5. Упражнения

В качестве упражнений для самостоятельного решения предлагаются задачи, аналогичные типовым задачам многокритериальной оптимизации, рассмотренным в данной главе. Предлагаемые в качестве упражнений задачи оптимизации содержат конкретные значения исходных данных, что позволяет получить их количественное решение с помощью программы MS Excel. Те из читателей, кто сочтет для себя интересным найти решение данных задач несколькими способами, получат возможность убедиться в правильности своих решений.

10.5.1. Двухкритериальная задача о рюкзаке

Имеются 7 грузовых контейнеров: красный, оранжевый, желтый, зеленый, голубой, синий и фиолетовый, которые предполагается транспортировать в трюме морского судна. Каждый из контейнеров имеет 3 характеристики: масса контейнера в *тоннах*, объем в $м^3$ и стоимость каждого из контейнеров (например, в *тыс. рублей*). Значения этих характеристик для рассматриваемых контейнеров представлены в следующей таблице (табл. 10.2).

Таблица 10.2. *Значения отдельных характеристик для контейнеров*

Контейнер/ Характеристика	Красный	Оранжевый	Желтый	Зеленый	Голубой	Синий	Фиолетовый
Масса	10	12	15	18	16	11	20
Объем	26	45	25	35	24	28	32
Стоимость	20	45	40	50	30	25	35

Общая грузоподъемность судна равна 60 *т*, а максимальный объем трюма не ограничен. Требуется определить те контейнеры, которые следует погрузить в трюм морского судна, так чтобы обеспечить максимальную стоимость и минимальный объем общего груза и при этом не допустить перегрузки судна.

10.5.2. Двухкритериальная задача водопроводчика

Для конкретного плана прокладки труб под землей в пределах района производства ремонтных работ (рис. 10.32) определить минимальное число плит, которые требуется приподнять, чтобы установить по одному вентильному перекрытию на каждой трубе.

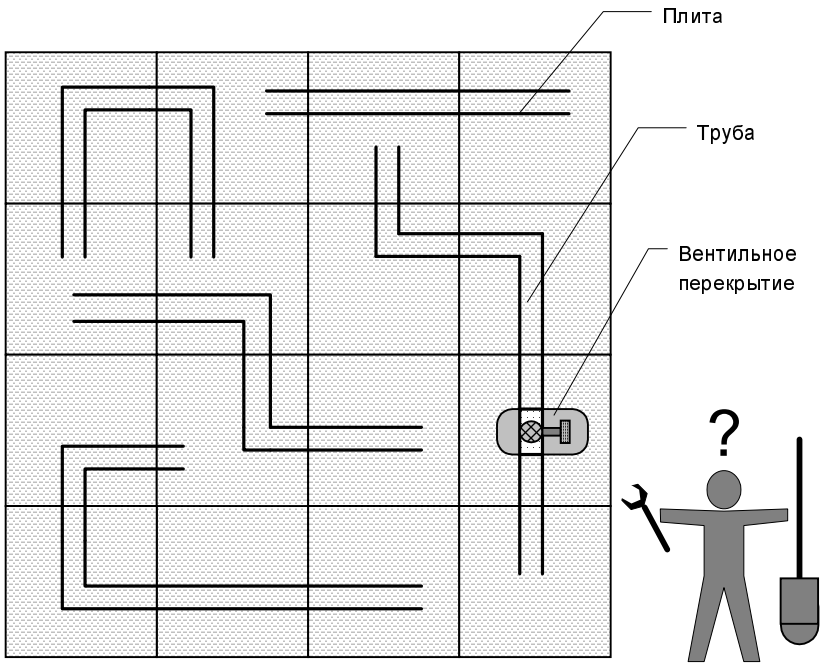


Рис. 10.32. План прокладки труб в задаче водопроводчика

При этом время подъема каждой плиты фиксировано и равно в *минутах*: $c_1 = 60$, $c_2 = 45$, $c_3 = 30$, $c_4 = 25$, $c_5 = 40$, $c_6 = 55$, $c_7 = 50$, $c_8 = 55$, $c_9 = 60$, $c_{10} = 65$, $c_{11} = 70$, $c_{12} = 75$, $c_{13} = 55$, $c_{14} = 85$, $c_{15} = 80$, $c_{16} = 90$. Стоимость работ после подъема каждой плиты также фиксирована и равна в *тыс. руб.*: $d_1 = 6$, $d_2 = 5{,}5$, $d_3 = 7$, $d_4 = 3{,}5$, $d_5 = 6{,}5$, $d_6 = 3{,}8$, $d_7 = 4{,}5$, $d_8 = 3$, $d_9 = 3{,}9$, $d_{10} = 5{,}6$, $d_{11} = 3{,}2$, $d_{12} = 2{,}1$, $d_{13} = 2$, $d_{14} = 1{,}9$, $d_{15} = 2{,}2$, $d_{16} = 1{,}4$.

Требуется определить номера плит, которые необходимо приподнять для установки вентильных перекрытий, чтобы общее время и стоимость выполнения работ были минимальными.

10.5.3. Двухкритериальная задача о назначении

Рассмотрим несимметричную задачу о назначении, когда количество кандидатов превышает количество вакантных должностей. В этом случае в качестве кандидатов выступают рекруты под отдельными номерами, а в качестве работ — вакантные должности в некоторой фирме, например, менеджер, программист, бизнес-аналитик. Затраты на замещение этих должностей рекрутами, связанные с необходимостью их предварительного обучения и стажировки, заданы в следующей таблице (табл. 10.3).

Таблица 10.3. *Затраты на замещение должностей рекрутами (в тыс. рублей)*

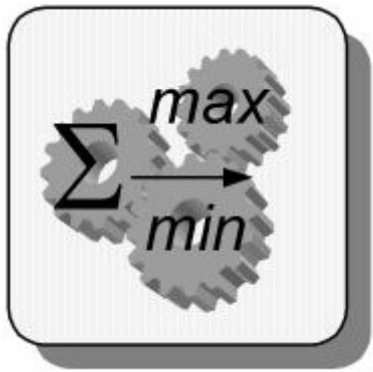
Рекруты/ Должности	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Менеджер	9	10	7	14	6	12	8	6
Программист	11	15	13	11	17	15	12	20
Бизнес-аналитик	7	14	12	8	10	13	19	7

Значения ожидаемой прибыли при работе этих рекрутов на должностях заданы в следующей таблице (табл. 10.4).

Таблица 10.4. *Ожидаемая прибыль от работы рекрутов в должностях (в тыс. рублей)*

Рекруты/ Должности	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Менеджер	80	100	170	140	160	120	180	155
Программист	210	115	130	110	175	150	125	200
Бизнес-аналитик	75	145	200	185	160	135	195	105

Требуется определить такое назначение рекрутов на вакантные должности, при котором общие затраты на их подготовку будут минимальными, а общая прибыль от их работы на соответствующих должностях будет максимальной.



Часть V

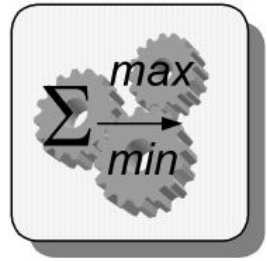
Программирование задач оптимизации в среде Excel

Глава 11. Алгоритмы и программы решения задач оптимизации на графах

Глава 12. Алгоритмы и программы решения задач комбинаторной оптимизации

Наличие в программе MS Excel большого количества встроенных функций создает у пользователей невольную уверенность в том, что любая задача оптимизации может быть решена с помощью специального инструмента поиска решений. Однако, с одной стороны, отдельные практические задачи оптимизации могут потребовать выполнения множества рутинных операций при подготовке соответствующих исходных данных. С другой стороны, инструмент поиска их решения вовсе не гарантирует получение оптимального результата за приемлемое время. В этих случаях неоценимую помощь пользователям окажет технология Visual Basic For Applications (VBA), которая обеспечивает совместное использование одного из наиболее мощных языков офисного программирования и всех вычислительных возможностей MS Excel.

С помощью VBA пользователи, не являясь профессиональными программистами, могут легко и быстро создавать различные приложения, включая собственные новые функции, отсутствующие в среде MS Excel. Являясь встроенной средой программирования для всех программ, входящих в состав пакета MS Office System 2003, Visual Basic For Applications дает возможность достаточно просто решать многие задачи оптимизации в среде MS Excel, позволяя пользователям автоматизировать целый ряд ручных операций при решении практических задач оптимизации.



Глава 11

Алгоритмы и программы решения задач оптимизации на графах

Материал предыдущих глав может создать у пользователей уверенность в том, что любая практическая задача оптимизации может быть решена с помощью программы MS Excel. Однако уже при подготовке отдельных исходных данных для решения рассмотренных задач оптимизации требуется выполнение значительного числа действий по заданию исходных данных и логических усилий по формированию ограничений. Более того, мастер поиска решений программы MS Excel вовсе не гарантирует нахождение решения для всех задач оптимизации, даже если удалось их сформулировать в форме той или иной типовой задачи оптимизации. Наконец, в отдельных случаях возникает желание изобразить исходные данные или результаты решения задач оптимизации в графической форме, отличающейся от стандартных диаграмм программы MS Excel.

Наиболее естественным выходом во всех подобных ситуациях является использование Visual Basic For Applications (VBA). Термином VBA одновременно называют язык программирования высокого уровня и встроенную среду программирования пакета MS Office System 2003, которая содержит все необходимые средства быстрой разработки программ, включая редактор кода, конструктор графических форм и управляющих элементов, а также средства отладки и тестирования программ.

Далее в настоящей главе описываются основные конструкции языка VBA и программное решение некоторых задач оптимизации. В качестве простейших задач, которые могут быть решены с помощью VBA, рассматриваются задача подготовки исходных данных для построения графика двумерной экспоненциальной функции и задача изображения структуры неориентированного графа.

11.1. Особенности разработки пользовательских программ в среде MS Excel

Для разработки и написания программ на языке VBA необходимо не только знать его синтаксис, но и владеть основными приемами работы со встроенным редактором VBA. Поскольку языку программирования VBA посвящены отдельные руководства, в настоящей книге те или иные конструкции этого языка рассматриваются по мере необходимости в случае их использования в рассматриваемых программах.

11.1.1. Среда и язык программирования Visual Basic For Applications

Текст программы на языке VBA размещается в отдельном модуле, который является составной частью рабочей книги MS Excel. Первоначально рабочие книги MS Excel не содержат модулей с текстами пользовательских программ VBA. При необходимости эти модули могут быть созданы пользователями самостоятельно для решения тех или иных задач. При этом каждая книга MS Excel может содержать несколько модулей, соответственно этому программа VBA может размещаться в одном или нескольких модулях.

Модуль представляет собой отдельный лист с текстом программы, являющийся составной частью рабочей книги MS Excel. Для написания текста программы на языке VBA необходимо в рабочей книге создать новый модуль, в который будет помещен текст программы.

Для этой цели необходимо вызвать встроенный редактор Visual Basic, для чего следует выполнить операцию главного меню MS Excel: **Сервис** | **Макрос** | **Редактор Visual Basic** или нажать клавиши <Alt>+<F11>. В результате этих действий откроется дополнительное окно редактора Visual Basic. Рабочее окно редактора Visual Basic имеет главное меню, которое позволяет пользователю выполнять весь набор операций, обеспечивающих реализацию широкого диапазона функциональных возможностей программы. Назначение отдельных пунктов главного меню редактора Visual Basic, входящего в состав офисного пакета MS Office System 2003, рассматривается в *приложении 3*.

Примечание

Следует заметить, что главное меню редактора Visual Basic для локализованной версии пакета MS Office System 2003 не является локализованным. Это может вызвать некоторые трудности у начинающих пользователей. Именно по этой причине приводятся в скобках названия соответствующих операций главного меню на русском языке.

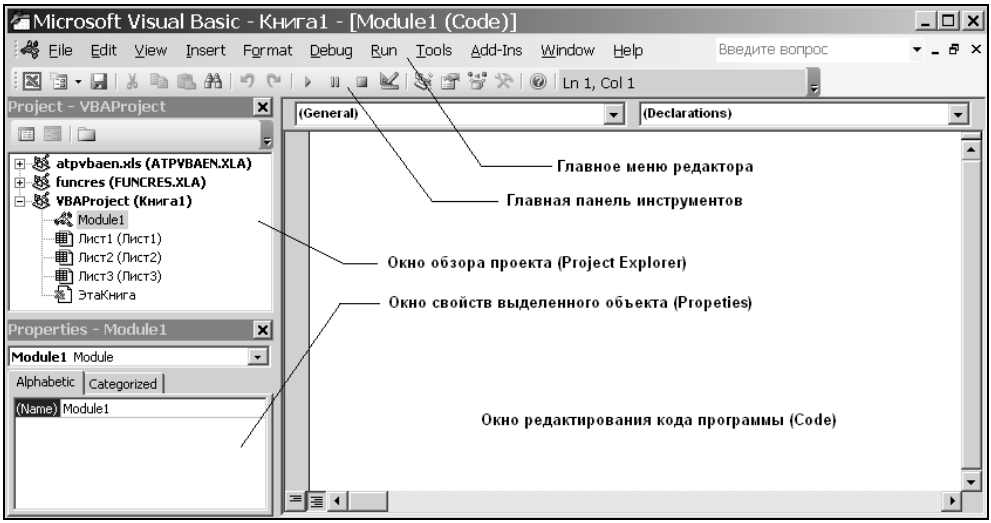


Рис. 11.1. Внешний вид окна рабочего интерфейса редактора Visual Basic

Далее, находясь в редакторе Visual Basic, следует выполнить операцию главного меню редактора: **Insert (Вставка)** | **Module (Модуль)**. В результате выполнения этой операции в редакторе Visual Basic появится рабочее окно с пустым содержимым созданного модуля, которому по умолчанию будет присвоено имя Module1 (рис. 11.1).

Базовым средством расширения функциональных возможностей программы MS Excel является создание дополнительных функций пользователя с помощью языка и среды программирования VBA. Этот способ имеет неоспоримое достоинство, которое заключается в том, что созданные пользователем дополнительные функции можно поместить в отдельные ячейки рабочего листа аналогично любой встроенной функции MS Excel. При этом не требуется ни отдельной компиляции программы VBA, ни создания дополнительных управляющих элементов или экранных форм.

В общем случае функция пользователя имеет следующий синтаксис:

```
Function ИМЯ_ФУНКЦИИ (СПИСОК_ПАРАМЕТРОВ)
    ТЕЛО_ФУНКЦИИ
End Function
```

Именем функции или переменной в языке VBA может быть любой идентификатор, определенный пользователем. *Идентификатор* — это произвольная последовательность букв, цифр и символа подчеркивания, начинающаяся с буквы. Каждый идентификатор в программе должен быть уникальным и может содержать символы как английского языка, так и символы кириллицы. При этом буквы верхнего и нижнего регистров не различаются.

Следует заметить, что пробелы внутри идентификаторов недопустимы. В случае необходимости их можно заменить символом подчеркивания. Для упрощения чтения и понимания значения на практике рекомендуется при написании сложных идентификаторов использовать строчные и прописные буквы, а также символ подчеркивания.

Носителем возвращаемого значения функции пользователя является имя функции. Поэтому в теле функции пользователя, вычисляющей некоторое значение, должен присутствовать, по крайней мере, один оператор, присваивающий имени функции значение какого-либо выражения.

`ТЕЛО_ФУНКЦИИ` состоит из описательной части и блока операторов, которые выполняются последовательно в порядке их записи в программе. Если необходимо прекратить выполнение функции в некотором конкретном месте программы, то это можно сделать с помощью оператора `Exit Function`. При определении функции иногда бывает удобно описать типы параметров и вычисляемого значения функции.

Кроме функций пользователя лист модуля может содержать текст одной или нескольких подпрограмм, которые в языке VBA называются *процедурами*. Процедура имеет стандартное оформление:

```
Sub ИМЯ_ПРОЦЕДУРЫ(СПИСОК_ПАРАМЕТРОВ)
    ТЕЛО_ПРОЦЕДУРЫ
End Sub
```

Следует отметить, что кроме выполнения определенных действий процедура так же, как и функция, может возвращать значения, которые присваиваются параметрам внутри процедуры. В отличие от функций пользователя, процедуры нельзя вызывать из ячеек рабочего листа. Для их вызова необходимо создание специальных форм или панелей инструментов пользователя.

Все переменные в VBA имеют тип, однако язык VBA не является строго типизированным. Тип указывает, что может хранить переменная: целое число, строку, дату и т. д. Базовые типы переменных VBA и их основные характеристики приведены в табл. 11.1.

Таблица 11.1. *Типы переменных языка VBA и их характеристики*

Тип данных	Диапазон значений	Занимаемый размер памяти (байт)
Byte (Байт)	От 0 до 255	1
Boolean (Логический)	True (Истина) или False (Ложь)	2
Integer (Целое)	От –32 768 до 32 767	2

Таблица 11.1 *(окончание)*

Тип данных	Диапазон значений	Занимаемый размер памяти (байт)
Long (Длинное целое)	От –2 147 483 648 до 2 147 483 647	4
Single (Число с плавающей точкой)	По абсолютной величине от 1,401298E-45 до 3,402823E38	4
Date (Дата)	От 1 января 100 г. до 31 декабря 9999 г.	8
Double (Число с плавающей точкой двойной точности)	По абсолютной величине от 4,94065645841247E-324 до 1,79769313486232E308	8
Object (Объект)	Любой указатель объекта	4
String (Строка переменной длины)	От 0 до приблизительно 2 миллиардов	10 + длина строки
String (Строка постоянной длины)	От 0 до приблизительно 65 400	Длина строки
Currency (Денежный)	От –922 337 203 685 477,5808 до 922 337 203 685 477,5807	8
Decimal (Масштабируемое целое)	±79 228 162 514 264 337 593 543 950 335 без дробной части	14
Variant (Числовые подтипы)	Любое числовое значение (до границ диапазона типа Decimal)	16
Variant (Строковые подтипы)	Как для типа String переменной длины	22 + длина строки

Если для некоторой переменной в тексте программы по какой-либо причине не указан ее тип, то по умолчанию ей присваивается тип `Variant`. Переменные этого типа могут хранить данные любого типа, при этом тип переменной изменяется в зависимости от типа данных при последнем присваивании. В этом смысле тип `Variant` довольно удобен для использования в программах VBA. Однако следует помнить о том, что этот тип требует больше памяти для своего хранения. В то же время указание для каждой переменной в программе конкретного типа позволяет более быстро и надежно выполнять операции с этими переменными.

Для определения или объявления переменных в VBA используется следующая синтаксическая конструкция:

```
Dim ИМЯ_ПЕРЕМЕННОЙ As ТИП_ПЕРЕМЕННОЙ
```

Например, возможны следующие объявления переменных:

```
Dim A As Integer
Dim C, D As Integer, E As Single
```

Объявлению типа переменной, функции и процедуры может предшествовать одно из ключевых слов: `Public`, `Private` или `Static`, которое обозначает область видимости соответствующего объекта. При этом объекты с *локальной* областью видимости только в пределах процедуры или функции объявляются как `Private` или `Dim`. К соответствующим переменным и массивам можно обращаться только в той процедуре или функции, где они объявлены. Объекты с *глобальной* областью видимости в пределах всех программ модуля объявляются как `Public`. К соответствующим переменным и массивам можно обращаться из всех процедур или функций модуля, в котором они объявлены.

Локальная переменная или массив, объявленная с ключевым словом `Static`, не уничтожается при окончании выполнения процедуры или функции, в которой объявлен соответствующий объект, а сохраняет свое последнее значение. Однако значения такой переменной или массива не могут быть изменены вне процедуры или функции, в которой они объявлены.

Строковая или текстовая переменная (`String`) по умолчанию является массивом переменной длины, который содержит символы. При необходимости строковая переменная может быть объявлена с фиксированной длиной. В следующем примере объявляется символьный массив размером в 25 символов:

```
Dim S As String*25
```

Если в программе переменной `S` будет присвоена строка длиной более 25 символов, то она будет усечена.

Как и в других языках программирования, в VBA можно использовать массивы переменных различного типа. Признаком массива переменных является наличие скобок, внутри которых указываются размеры этого массива, например:

```
Dim A(3, 3) As Single
   Dim B(12) As Integer
```

Первая строка объявляет двумерный массив или матрицу 3×3, состоящую из действительных чисел. Вторая строка объявляет одномерный массив (вектор), состоящий из 12 целых чисел, причем по умолчанию первый элемент массива будет `B(0)`, а последний — `B(11)`. В этом случае говорят, что 0 — *базовый индекс массива*. При необходимости базовый индекс массива можно

изменить, написав в начале текста программы модуля оператор `option Base 1`. После этого индексы массивов `A` и `B` будут начинаться с единицы. Другим способом изменения базового индекса является использование ключевого слова `To` при объявлении массива, например:

```
Dim A(1 To 3, 1 To 3) As Single
Dim B(1 To 12) As Integer
```

Иногда до начала выполнения программы VBA неизвестен размер объявленного массива. В этом случае его предварительно следует объявить как *динамический* массив без указания его размерности, например:

```
A( ) As Single
```

Затем в программе следует установить необходимый размер массива, присвоив его некоторой переменной, например `N`, и указать размер динамического массива с помощью оператора `ReDim`, например:

```
N = 10
ReDim A(N, N)
```

После использования динамических массивов их рекомендуется уничтожать явным образом. Этой цели служит специальный оператор `Erase`, который освобождает память, занимаемую этим массивом, и делает ее доступной для дальнейшего использования в вычислительных программах. Пример использования данного оператора: `Erase A`.

Если значение некоторой переменной не должно изменяться в программе, то она объявляется как *константа*. Такой переменной должно предшествовать ключевое слово `Const`, после которого может быть указан тип константы и задается ее значение. Например, известная в математике постоянная может быть задана следующим образом:

```
Const Pi = 3.141592653589
```

При написании программ на VBA, расширяющих возможности MS Excel, довольно часто приходится обращаться к стандартным объектам модели рабочей книги. Соответствующие объекты после объявления своего типа должны быть созданы, для чего используется ключевое слово `Set`. Так, например, группа операторов:

```
Dim NewSheet As Worksheet
Set NewSheet = Worksheets.Add
NewSheet.Name = "Новый лист"
```

вначале объявляет переменную с именем `NewSheet` как рабочий лист MS Excel, затем создает соответствующий объект этого типа и присваивает ему имя `"Новый лист"`. Обращение к свойствам и методам объектов происходит с помощью оператора точка ".", слева от которого указывается имя объекта,

а справа — его свойство или метод. Если у объекта отсутствует указанное свойство или метод, то такая ситуация является ошибкой программы. В этом случае при компиляции такой программы эта ошибка будет выявлена в редакторе Visual Basic, а программа, содержащая ошибки, не будет выполнена.

Оперативную информацию по свойствам и методам объектов VBA можно получить с помощью справки редактора Visual Basic. В этом случае достаточно выделить некоторое слово или символ в окне редактора кода и нажать функциональную клавишу <F1>. В результате будет открыто специальное окно со справкой по VBA и отображена информация по выбранному элементу программы, если соответствующий раздел справки существует.

Примечание

Редактор Visual Basic содержит встроенные средства отладки программ, позволяющие идентифицировать и устранить синтаксические и логические ошибки программ. Поскольку данные вопросы выходят за пределы тематики книги, заинтересованные читатели могут обратиться за более детальной информацией к специальным руководствам по программированию. Впрочем, во многих таких руководствах зачастую приводятся целые разделы встроенной справки VBA с соответствующими примерами.

При написании программ в ее текст целесообразно помещать *комментарии*, которые представляют собой любой пояснительный текст, содержащий произвольные символы. Каждая строка комментариев начинается со знака апострофа. Комментарии игнорируются компилятором, и поэтому никакого влияния на программу не оказывают. Комментарии удобно использовать также при отладке операторов для их временного отключения.

Если необходимо разместить несколько операторов в одной строке программы, то следует использовать в качестве разделителя отдельных операторов символ двоеточия ":". Этот символ также используется для группировки операторов в отдельный блок. Если необходимо перенести некоторый длинный оператор на другую строку, то следует использовать символ подчеркивания "_". Этот символ может помещаться в любом месте длинного оператора, однако ему должен предшествовать пробел.

Для удобства визуального восприятия программ принято сдвигать вправо операторы, входящие в отдельные логические блоки. Этот прием является общепринятым во многих языках программирования, поэтому практически не требует пояснения.

В общем случае порядок выполнения операторов программы VBA является линейным, при котором операторы выполняются последовательно, начиная с ранее расположенных до конца текста модуля. Изменить порядок следования операторов можно с помощью оператора условного и безусловного перехода.

Оператор *условного перехода* представляет собой конструкцию вида `If-Then-End If` или `If-Then-Else-End If`, которая имеет следующий синтаксис:

```
If ЛОГИЧЕСКОЕ_УСЛОВИЕ Then БЛОК_ОПЕРАТОРОВ_1 End If
If ЛОГИЧЕСКОЕ_УСЛОВИЕ Then БЛОК_ОПЕРАТОРОВ_1
    Else БЛОК_ОПЕРАТОРОВ_2 End If
```

где `ЛОГИЧЕСКОЕ_УСЛОВИЕ` — любое выражение VBA, которое возвращает значение `ИСТИНА` или `ЛОЖЬ`, `БЛОК_ОПЕРАТОРОВ_1` — произвольный набор операторов, который должен быть выполнен в случае, когда `ЛОГИЧЕСКОЕ_УСЛОВИЕ` принимает значение `ИСТИНА`, а `БЛОК_ОПЕРАТОРОВ_2` — произвольный набор операторов, который должен быть выполнен в случае, когда `ЛОГИЧЕСКОЕ_УСЛОВИЕ` принимает значение `ЛОЖЬ`.

Оператор *безусловного перехода* представляет собой конструкцию вида `GoTo`, которая имеет следующий синтаксис:

```
GoTo МЕТКА
```

где `МЕТКА` — произвольный идентификатор, который обозначает метку или номер строки текста программы, к которой выполняется переход после исполнения данной инструкции.

Для повторения выполнения отдельной группы операторов программы следует использовать циклы. В языке VBA имеется несколько типов циклов. Наиболее распространенными являются цикл с параметрами и цикл с предусловием.

Цикл с параметрами представляет собой конструкцию вида `For-To-Next`, которая имеет следующий синтаксис:

```
For СЧЕТЧИК=НАЧАЛО To КОНЕЦ
    БЛОК_ОПЕРАТОРОВ
Next СЧЕТЧИК
```

где `СЧЕТЧИК` — произвольный идентификатор, который используется для подсчета числа повторения отдельных итераций цикла, `НАЧАЛО` — число, указывающее начальное значение для идентификатора подсчета итераций, `КОНЕЦ` — число, указывающее конечное значение для идентификатора подсчета итераций, `БЛОК_ОПЕРАТОРОВ` — последовательность операторов, которые выполняются за одну итерацию цикла.

В данной конструкции по умолчанию предполагается, что подсчет количества итераций происходит с шагом 1. В случае необходимости это значение может быть изменено пользователем. Соответствующий цикл с параметрами должен иметь следующий синтаксис:

```
For СЧЕТЧИК=НАЧАЛО To КОНЕЦ Step ШАГ
    БЛОК_ОПЕРАТОРОВ
Next СЧЕТЧИК
```

где ШАГ — число, указывающее шаг подсчета количества итераций данного цикла.

Цикл с предусловием представляет собой конструкцию вида While-Wend, которая имеет следующий синтаксис:

```
While ЛОГИЧЕСКОЕ_УСЛОВИЕ
     БЛОК_ОПЕРАТОРОВ
Wend
```

где ЛОГИЧЕСКОЕ_УСЛОВИЕ — любое выражение VBA, которое возвращает значение ИСТИНА или ЛОЖЬ, БЛОК_ОПЕРАТОРОВ — последовательность операторов, которые выполняются за одну итерацию цикла, когда ЛОГИЧЕСКОЕ_УСЛОВИЕ принимает значение ИСТИНА. В случае, когда ЛОГИЧЕСКОЕ_УСЛОВИЕ принимает значение ЛОЖЬ, операторы БЛОК_ОПЕРАТОРОВ не выполняются, и управление передается к оператору, следующему за служебным словом Wend.

Примечание

Следует заметить, что цикл с параметрами представляет более надежное решение, чем цикл с предусловием, поскольку последний довольно часто служит источником ошибок, результатом которых является зацикливание и, как следствие, зависание программы. Это происходит в том случае, когда пользователь не предусмотрел изменение значения ЛОГИЧЕСКОЕ_УСЛОВИЕ в операторах БЛОК_ОПЕРАТОРОВ либо выход из данного цикла при выполнении некоторого дополнительного условия. В любом случае операторы цикла требуют повышенного внимания от пользователей при их использовании в тексте программы.

Модуль с текстом программы VBA может начинаться со служебных операторов, которые, например, устанавливают правила задания переменных, способ сравнения строк. После них следует объявление глобальных переменных и констант для модуля, которые используются во всех функциях и процедурах данного модуля. Далее располагается текст функций или процедур пользователя, составляющих собственно программу модуля и расширяющих функциональные возможности MS Excel в контексте выполнения дополнительных операций.

Примечание

Рассмотреть все конструкции языка VBA в контексте материала данной книги не представляется возможным. Систематическому изложению синтаксических конструкций языка VBA посвящены специальные руководства, к которым могут обратиться заинтересованные читатели. Однако рассмотренных в данной главе базовых возможностей может оказаться вполне достаточно для написания простейших программ, реализующих пользовательские функции. При этом

отдельные конструкции языка VBA будут рассмотрены далее по мере необходимости при их включении в текст рассматриваемых программ, которые расширяют функциональные возможности MS Excel в области решения задач оптимизации.

11.1.2. Создание пользовательской функции для вычисления двумерной экспоненциальной функции

Простейшей задачей, которая может быть решена с помощью языка программирования VBA, является создание дополнительных функций пользователя. Хотя программа MS Excel содержит большое количество встроенных функций, в целом ряде случаев возникает необходимость в вычислениях, для которых подходящая функция в MS Excel либо отсутствует, либо требует для своего вычисления ручного заполнения ячеек рабочего листа при подготовке исходных данных. Неоспоримым достоинством программы MS Excel и среды VBA является тот факт, что разработанные функции пользователя на языке VBA можно вызывать с помощью мастера функций так же, как и обычные встроенные функции программы MS Excel.

В качестве примера создания дополнительной функции рассмотрим двумерную экспоненциальную функцию, которая отсутствует в числе встроенных функций программы MS Excel. Напомним, что в *разд. 3.5.3* было рассмотрено решение с помощью программы MS Excel задачи нелинейной оптимизации с двумерной экспоненциальной целевой функцией, которая задается выражением (3.5.3).

При построении графика данной функции с целью визуального анализа найденного решения возникает техническая трудность, связанная с трудоемкостью записи формулы вычисления значений этой функции для некоторого диапазона значений независимых переменных x_1 и x_2. Действительно, при вычислении отдельного значения двумерной экспоненциальной целевой функции для фиксированной пары значений переменных (x_1, x_2) согласно формуле (3.5.3) необходимо выполнить последовательное суммирование для возрастающих значений параметра a, начиная от значения 0,1 и заканчивая значением 1 с интервалом изменения данного параметра, равным 0,1. Эта задача может быть эффективно решена с помощью специальной программы на VBA.

Для решения данной задачи с помощью программы MS Excel создадим новую книгу с именем Программирование Задач и изменим имя ее первого листа на Экспоненциальная функция. Для написания текста пользовательской функции, в данном случае — для написания текста программы вычисления

двумерной экспоненциальной целевой функции, следует выполнить следующие действия:

1. Выполнить операцию главного меню: **Сервис** | **Макрос** | **Редактор Visual Basic** или нажать клавиши <Alt>+<F11>, в результате чего откроется дополнительное окно редактора Visual Basic.

Примечание

Следует заметить, что при первоначальном обращении к редактору Visual Basic можно не обнаружить соответствующего пункта в главном меню **Сервис**. Это означает, что компонент программирования на VBA в программе MS Office Excel 2003 не установлен. Хотя компонент Microsoft Visual Basic для приложений, как правило, устанавливается по умолчанию, в отдельных случаях этого может не произойти. Поэтому для продолжения работы необходимо убедиться, что данный компонент установлен. С этой целью можно воспользоваться инструментом **Установка и удаление программ** ОС Windows и проверить состав установленных компонентов пакета MS Office System 2003. В случае отсутствия компонента MS Visual Basic для приложений следует его установить, для чего потребуется установочный диск пакета MS Office System 2003.

2. Первоначально рабочая книга программы MS Excel не содержит никаких модулей с текстами программ VBA. Поэтому необходимо создать новый модуль, для чего необходимо выполнить в редакторе Visual Basic операцию главного меню: **Insert (Вставка)** | **Module (Модуль)**. В результате выполнения этой операции в редакторе Visual Basic появится рабочее окно с пустым содержимым созданного модуля, которому по умолчанию будет присвоено имя Module1.
3. В созданный модуль введем следующий текст программы вычисления двумерной экспоненциальной функции (листинг 11.1).

Листинг 11.1. Программа вычисления двумерной экспоненциальной функции

```
Function ExpFunc2(x1, x2)
   S = 0
   For I = 0.1 To 1 Step 0.1
      S = S + ((Exp(-I*x1)-Exp(-I*x2))-(Exp(-I)-Exp(-10 * I)))^2
   Next I
   ExpFunc2 = S '<-- возвращаемое функцией значение
End Function
```

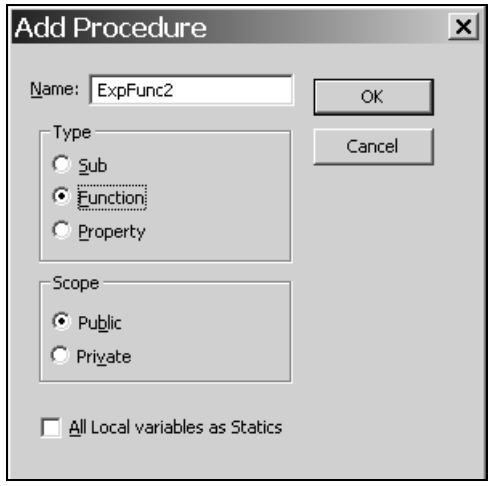


Рис. 11.2. Диалоговое окно задания новой функции или подпрограммы

Хотя данный текст программы можно непосредственно ввести в окне программного кода редактора, для задания новых функций или процедур модуля целесообразно использовать специальное окно добавления процедуры. Это окно можно вызвать посредством операции главного меню: **Insert (Вставка)** | **Procedure (Процедура)**, после чего ввести имя функции и ее область видимости (рис. 11.2). После нажатия кнопки **ОК** будет создан текст с объявлением данной функции в выбранном модуле. В этом случае пользователю останется лишь написать текст ее реализации.

Данная программа задает пользовательскую функцию `ExpFunc2`, которая имеет два формальных параметра: `x1` и `x2`. Текст программы пользовательской функции начинается с помощью стандартного оператора объявления функции: `Function`, после которого указывается имя функции `ExpFunc2` и ее формальные параметры в скобках. Ввиду простоты данной функции типы переменных не объявляются, т. е. используется по умолчанию тип `Variant`.

Собственно вычисление значения двумерной экспоненциальной функции для двух значений независимых переменных `x1` и `x2` реализуется с помощью цикла `For-To-Step`, который выполняется последовательно с помощью переменной `I`. Для вычисления обычной экспоненциальной функции применяется встроенная функция `Exp` с единственным аргументом. Оператор VBA `Next` указывает на конец этого цикла, а оператор `ExpFunc2 = S` задает возвращаемое данной функцией значение. Наконец, последний оператор `End Function` служит для указания конца текста функции.

Внешний вид редактора Visual Basic с текстом программы в модуле с именем Модуль1 изображен на рис. 11.3.

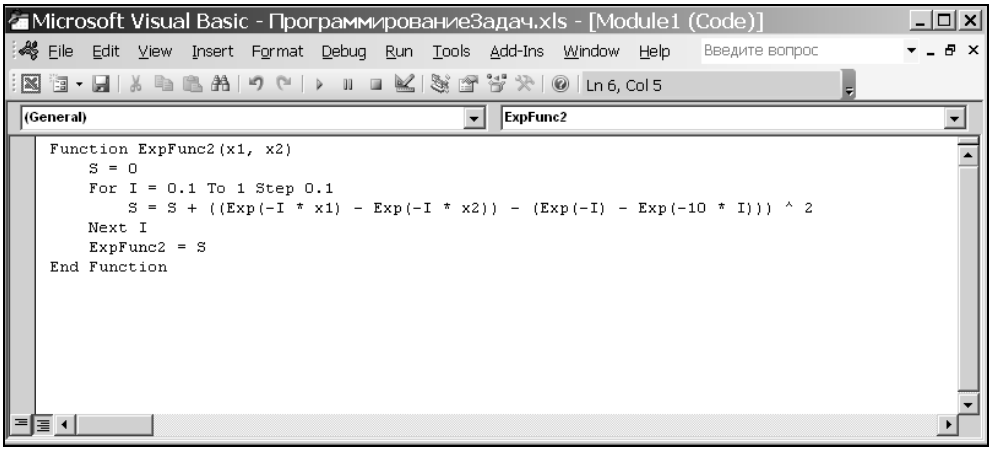


Рис. 11.3. Текст модуля с программой вычисления двумерной экспоненциальной функции

После записи текста программы в редакторе Visual Basic пользовательская функция с именем `ExpFunc2` становится доступной для использования в качестве формулы в ячейках рабочего листа программы MS Excel. Никаких дополнительных действий по компиляции данной программы не требуется. Если программа имеет значительный размер или при наличии в ней ошибок, может оказаться необходимой ее предварительная отладка.

Если текст данной программы не содержит ошибок, то соответствующая функция `ExpFunc2` может быть непосредственно вызвана из ячеек рабочего листа данной книги. При этом особенности вызова данной формулы из ячеек полностью аналогичны встроенным формулам программы MS Excel, которые были рассмотрены в *разд. 2.3.3*.

11.1.3. Построение графика функции двух переменных

Общая последовательность выполнения действий в программе MS Excel, необходимых для построения графика функции двух переменных, была описана в *разд. 2.4.2*. В данном разделе рассматривается конкретизация этих действий при построении графика поверхности двумерной экспоненциальной функции.

Напомним, что для построения графика поверхности, которая представляет двумерную экспоненциальную функцию (3.5.3), предварительно необходимо определить диапазон значений независимых переменных и соответствующее множество значений зависимой (функциональной) переменной. После этого также следует воспользоваться мастером построения диаграмм программы MS Excel для изображения графика заданной функциональной зависимости.

Поскольку при создании функции в первой строке ее кода были указаны два аргумента, то при ее вызове необходимо указать значения обоих этих аргументов. Поэтому значения аргументов должны быть заданы предварительно в соответствующих ячейках рабочего листа.

Для построения графика двумерной экспоненциальной функции двух переменных следует выбрать интервал их изменения, например, $x_1 \in [0, 2]$ и $x_2 \in [0, 15]$ и выполнить следующие практические действия:

1. В ячейку **A1** рабочего листа с именем Экспоненциальная функция книги Программирование задач введем текст `Значения переменных X1 и X2:`.
2. С помощью операции автозаполнения зададим значения первой переменной x_1 в ячейках **B1:L1**, а значения второй переменной x_2 — в ячейках **A2:A17**. Заметим, что для построения графика данной функции с учетом результатов решения соответствующей задачи нелинейной оптимизации целесообразно рассмотреть последовательный диапазон значений первой переменной от 0 до 2 с интервалом их изменения, равным 0,2, и диапазон значений второй переменной от 0 до 15 с интервалом их изменения, равным 1.
3. Далее в ячейку **B2** введем формулу: `=ExpFunc2(B$1;$A2)`, которую с помощью автозаполнения скопируем в ячейки **C2:L2**. При задании данной формулы необходимо выбрать функцию `ExpFunc2` в группе **Определенные пользователем** мастера задания функций, после чего указать ячейки, содержащие значения независимых переменных.
4. Предварительно выделив диапазон ячеек **B2:L2**, также с помощью автозаполнения скопируем соответствующие данные в ячейки **B3:L17**. Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 11.4).

Для построения графика функции двух переменных также следует воспользоваться мастером диаграмм, который может быть вызван с помощью кнопки стандартной панели инструментов (см. табл. 2.1) или операции главного меню: **Вставка** | **Диаграмма**. При этом последовательность выполнения действий в программе MS Excel, необходимых для построения трехмерного графика функции двух переменных, аналогична описанной в *разд. 2.4.2* и поэтому здесь не приводится. В результате выполнения этих действий с использованием мастера диаграмм будет получен график двумерной экспоненциальной функции (рис. 11.5).

Изображенный на рис. 11.5 график функции получен после редактирования его свойств, а именно — изменения цвета фона диаграммы и ее незначительного поворота для лучшего восприятия характера поверхности. Визуальный анализ данного графика служит дополнительным средством контроля

правильности полученного оптимального решения задачи оптимизации с соответствующей целевой функцией.

ПрограммированиеЗадач

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Значения переменных X1 и X2:	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2	
2	0	3,064	4,2189	5,4424	6,6697	7,8609	8,9926	10,053	11,036	11,943	12,775	13,536	
3	1	0,9045	1,0645	1,4332	1,9245	2,4809	3,064	3,6491	4,2204	4,7684	5,2879	5,7763	
4	2	0,9623	0,6472	0,6035	0,7364	0,9803	1,2905	1,6367	1,9984	2,3621	2,7191	3,064	
5	3	1,2312	0,6772	0,424	0,3725	0,4539	0,6206	0,8397	1,0885	1,3518	1,6192	1,8839	
6	4	1,4384	0,7572	0,3911	0,2392	0,231	0,3176	0,4648	0,6492	0,8543	1,0693	1,2866	
7	5	1,5791	0,8262	0,3956	0,1855	0,1249	0,164	0,2683	0,4135	0,5831	0,7656	0,9532	
8	6	1,6765	0,8806	0,4109	0,1653	0,0722	0,0816	0,1586	0,2789	0,4255	0,5868	0,7548	
9	7	1,7477	0,9245	0,4298	0,1612	0,0469	0,0368	0,0957	0,1992	0,3302	0,4771	0,6316	
10	8	1,8027	0,9614	0,45	0,1658	0,037	0,0134	0,0598	0,1515	0,2716	0,4082	0,5531	
11	9	1,8472	0,9934	0,4702	0,1751	0,036	0,0028	0,0401	0,1234	0,2355	0,3645	0,5023	
12	10	1,8846	1,0217	0,49	0,1869	0,0403	0	0,0306	0,1075	0,2136	0,337	0,4694	
13	11	1,9167	1,0472	0,5091	0,2	0,0478	0,002	0,0275	0,0996	0,2011	0,3201	0,4483	
14	12	1,9449	1,0702	0,5273	0,2136	0,057	0,0071	0,0286	0,0969	0,1947	0,3103	0,4352	
15	13	1,9698	1,0912	0,5445	0,2271	0,0671	0,0139	0,0323	0,0975	0,1925	0,3053	0,4276	
16	14	1,9921	1,1103	0,5606	0,2404	0,0776	0,0217	0,0376	0,1004	0,193	0,3036	0,4238	
17	15	2,0121	1,1278	0,5757	0,2531	0,0881	0,0301	0,0438	0,1047	0,1954	0,3041	0,4225	
18													
19													

Экспоненциальная функция / Лист2 / Лист3

Рис. 11.4. Исходные данные для построения графика двумерной экспоненциальной функции

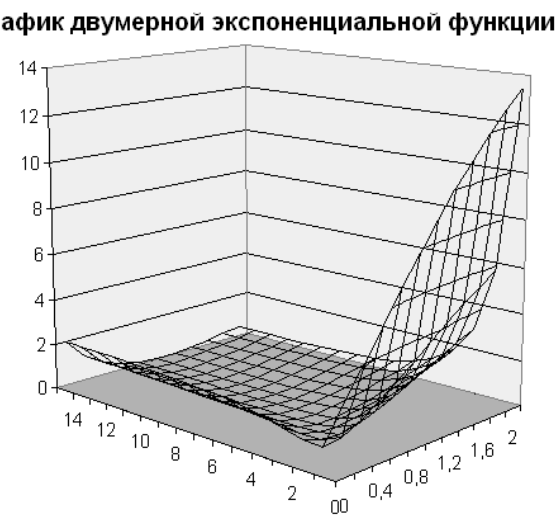


Рис. 11.5. Результат построения графика двумерной экспоненциальной функции

11.1.4. Программа изображения структуры неориентированного графа

Хотя программа MS Excel содержит большой набор типов диаграмм для графического представления данных, их использование для изображения структуры неориентированного графа сопряжено с известными трудностями. В то же время визуализация графа может оказаться необходимой для дополнительного контроля решений задач оптимизации на графах, которые рассматриваются в настоящей книге. В данном разделе описывается специальная программа на языке VBA, которая позволяет изображать структуру произвольного неориентированного графа. В последующем модификация данной программы используется для изображения специальных графов, таких как покрывающие деревья и оптимальные пути в графе.

Основное неудобство при изображении графа связано с тем обстоятельством, что пользователю предварительно необходимо задать некоторые координаты на плоскости каждой из вершин исходного графа. Дело в том, что программа MS Excel не в состоянии самостоятельно определить размещение вершин графа на плоскости. Идея расположить все вершины графа в вершинах соответствующего правильного многоугольника также не подходит для общего случая, поскольку этот способ является менее наглядным для большинства практических задач.

Задача изображения структуры произвольного неориентированного графа может быть решена с помощью специальной подпрограммы, которая использует в качестве исходных данных матрицу смежности исходного графа и относительные координаты вершин на плоскости. При этом в качестве матрицы смежности может выступать матрица весов ребер, а координаты вершин зависят от размеров графической области, в которой изображается соответствующий граф. В книге предлагается способ задания этих значений, основанный на практическом опыте визуализации графов в программе MS Excel. После небольшого опыта пользователям не составит труда самостоятельно задавать координаты вершин для произвольного графа, следуя рекомендации, чтобы расположение вершин исключало или минимизировало пересечения ребер графа.

Для изображения структуры неориентированного графа предварительно следует задать исходные данные для конкретного графа. С этой целью в рабочей книге Программирование задач следует создать новый рабочий лист с именем Изображение графа и выполнить следующие практические действия:

1. В ячейки **A1**, **I1**, **A3:A9**, **B2:J2** данного рабочего листа ввести соответствующий текст, конкретное содержание которого не влияет на результат решаемой задачи.

2. В ячейки **B3:H9** следует ввести значения матрицы весов ребер исходного неориентированного графа, структуру которого следует изобразить на плоскости. В качестве такого графа используется граф, который изображен на рис. 7.1 и использовался в качестве исходного для решения рассмотренных задач оптимизации на графах.
3. В ячейки **I3:J9** следует ввести значения координат вершин изображаемого графа. При этом в ячейки **I3:J9** записываются горизонтальные координаты вершин, а в ячейки **J3:J9** — вертикальные координаты вершин. Следует отметить, что координаты вершин отсчитываются от левого верхнего угла данного рабочего листа. Для выполнения этих действий желательно предварительно сделать набросок графа на листе бумаги.

Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 11.6).

ПрограммированиеЗадач.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Матрица весов ребер исходного графа								Координаты вершин					
2		v1	v2	v3	v4	v5	v6	v7	X	Y				
3	v1	0	5	8	7	0	0	0	100	200				
4	v2	5	0	5	0	10	0	0	200	100				
5	v3	8	5	0	4	7	6	0	300	200				
6	v4	7	0	4	0	0	7	0	200	300				
7	v5	0	10	7	0	0	4	8	400	100				
8	v6	0	0	6	7	4	0	10	400	300				
9	v7	0	0	0	0	8	10	0	500	200				
10														
11														
12														
13														
14														
15														
16														

Изображение графа

Рис. 11.6. Исходные данные для изображения неориентированного графа

Для записи текста программы изображения структуры неориентированного графа удобно создать новый модуль, для чего необходимо выполнить в редакторе Visual Basic операцию главного меню: **Insert (Вставка)** | **Module (Модуль)**. В результате выполнения этой операции в редакторе Visual Basic появится рабочее окно с пустым содержимым созданного модуля, которому по умолчанию будет присвоено имя Module2. В созданный модуль введем следующий текст программы, оформленной в виде отдельной подпрограммы с именем `GraphPainter` без входных параметров или аргументов (листинг 11.2).

Листинг 11.2. Программа изображения структуры неориентированного графа

```
Public Sub GraphPainter()
' -- Спецификация переменных и массивов
 Const N = 7 ' <-- количество вершин графа -->
 Dim I, J As Integer
 Dim Vert(N), Sh As Shape
 Dim Dis(N, N), Xvert(N), Yvert(N) As Integer
' -- Ввод и формирование матрицы смежности
 For I = 1 To N
    Xvert(I) = ActiveSheet.Cells(I + 2, N + 2)
    Yvert(I) = ActiveSheet.Cells(I + 2, N + 3)
        For J = 1 To N
            Dis(I, J) = ActiveSheet.Cells(I + 2, J + 1)
        Next J
 Next I
' -- Изображение вершин графа
 For I = 1 To N
    Set Vert(I) = ActiveSheet.Shapes.AddShape(msoShapeOval, Xvert(I),
Yvert(I) + 60, 16, 16)
    Vert(I).TextFrame.Characters.Text = CStr(I)
    Vert(I).TextFrame.Characters.Font.Size = 10
    Vert(I).TextFrame.Characters.Font.Bold = True
 Next I
' -- Изображение ребер графа
 For I = 1 To N - 1
   For J = I + 1 To N
     If Dis(I, J) <> 0 Then ' -- изображаем только те ребра, для которых
' связь не равна 0
   Set Sh = ActiveSheet.Shapes.AddConnector(msoConnectorStraight,0,0,0,0)
        With Sh.ConnectorFormat
            .BeginConnect ConnectedShape:=Vert(I), ConnectionSite:=1
            .EndConnect ConnectedShape:=Vert(J), ConnectionSite:=1
            Sh.RerouteConnections
        End With
' -- Изображение веса рядом с соответствующим ребром
        If Xvert(I) = Xvert(J) Then ' -- две вершины расположены на одной
' вертикали
     Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
```

```
      (Xvert(I)+Xvert(J))/2-5, (Yvert(I)+Yvert(J))/2+60,60,150)
            Else ' -- две вершины не расположены на одной вертикали
      Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
      (Xvert(I) + Xvert(J)) / 2, (Yvert(I) + Yvert(J)) / 2 + 50, 60, 150)
         End If
         Sh.TextFrame.Characters.Text = CStr(Dis(I, J))
         Sh.TextFrame.Characters.Font.Size = 12
      End If
    Next J
  Next I
' -- Изображение подписи под рисунком
  Set Sh = ActiveSheet.Shapes.AddShape(msoShapeRectangle, 200, 380, 200, 20)
  Sh.TextFrame.Characters.Text = "Изображение исходного графа"
  Sh.TextFrame.Characters.Font.Size = 11
  Sh.TextFrame.Characters.Font.Bold = True
  Sh.TextFrame.HorizontalAlignment = xlHAlignCenter
  Sh.TextFrame.VerticalAlignment = xlVAlignCenter
  End Sub
```

Хотя текст данной программы содержит комментарии, поясняющие назначение некоторых операторов, следует отметить особенности использования специальных методов и свойств языка VBA для работы с изображениями.

Во-первых, при спецификации переменных и массивов следует для данной программы задавать количество вершин графа, указывая его после знака равенства константы `N`, которая служит в последующем для управления циклами. Все массивы задаются явно, при этом указывается также их тип данных. Массив `Vert(N)` предназначен для задания и хранения графической информации о вершинах графа, а переменная `Sh` — для создания изображения ребер графа. Тип данных `Shape` указывает на способ изображения графа не в форме диаграммы MS Excel, а в форме рисунка формата MS Office System. Массив `Dis(N, N)` служит для хранения значений матрицы смежности или весов ребер, а массивы `Xvert(N)` и `Yvert(N)` — для хранения значений горизонтальных и вертикальных координат вершин исходного графа.

Далее следует блок операторов, которые выполняют чтение данных из рабочего листа `Изображение графа` (рис. 11.6), который должен быть активным для работы данной программы. Обращение к ячейкам осуществляется с помощью свойства `Cells` объекта `ActiveSheet`, при этом в языке VBA принадлежность свойств и методов объекту указывается через точку в форме: `ActiveSheet.Cells`. Соответствующее свойство `Cells` является массивом, что позволяет обращаться к его значениям поэлементно.

После ввода и формирования матрицы смежности следуют операторы, которые изображают вершины исходного графа в форме кружков, внутри которых помещаются номера этих вершин. Поскольку общее число вершин равно `N`, данную операцию удобно реализовать в цикле, который выполняется `N` раз. Для изображения вершины в форме небольшого круга предназначен оператор: `Set Vert(I) = ActiveSheet.Shapes.AddShape(msoShapeOval, Xvert(I), Yvert(I) + 60, 16, 16)`, который добавляет на рабочий лист объект `Vert(I)` типа графической формы, а сама форма круга указывается константой `msoShapeOval`. Размещение вершины происходит в точке с координатами `Xvert(I)` и `Yvert(I)+60`, при этом значение второй координаты подбирается опытным путем посредством изменения некоторого целочисленного значения, в данном случае — это число 60. Величина круга также подбирается пользователем и устанавливается с помощью 2-х чисел, каждое из которых равно 16.

Следующий оператор отображает номер вершины, при этом используется встроенная функция преобразования числа в строку `CStr(I)`. Далее устанавливается размер шрифта и полужирный способ его изображения.

После изображения вершин графа следует блок операторов, отображающих ребра или связи между вершинами. Эта группа операторов организована в двойной цикл, при этом связь проводится только в том случае, когда соответствующее паре вершин значение матрицы смежности не равно 0. Этой цели служит условный оператор внутри цикла. Собственно связь создается оператором: `Sh = ActiveSheet.Shapes.AddConnector(msoConnectorStraight, 0, 0, 0, 0)`, в котором специальная константа `msoConnectorStraight` специфицирует связь в форме отрезка прямой линии. После создания линии связи устанавливаются ее начало `Vert(I)` и конец `Vert(J)`. Метод `RerouteConnections` определяет способ изображения связи как кратчайшую линию от начальной вершины к конечной.

Специальный оператор `With` обеспечивает доступ к методам и свойствам объектов, позволяя избежать повторения длинных цепочек вложенных объектов. Для создания изображения веса ребра рядом с соответствующей связью служит оператор:

```
Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal,
(Xvert(I) + Xvert(J)) / 2 - 5, (Yvert(I) + Yvert(J))/2+60,60,150),
```

если две вершины расположены на одной вертикали, и оператор:

```
Set Sh = ActiveSheet.Shapes. AddLabel(msoTextOrientationHorizontal,
(Xvert(I) + Xvert(J)) / 2, (Yvert(I) + Yvert(J)) / 2 + 50, 60, 150)
```

в противном случае. Для изображения значения веса ребра рядом с соответствующей связью служит оператор:

```
Sh.TextFrame. Characters.Text = CStr(Dis(I, J)),
```

а размер шрифта устанавливается оператором:

```
Sh.TextFrame.Characters.Font.Size = 12.
```

После изображения ребер графа и их весов следуют операторы, которые отображают подпись под рисунком. Собственно прямоугольник для надписи создается с помощью оператора

```
Sh = ActiveSheet.Shapes.AddShape(msoShapeRectangle, 200, 380, 200, 20),
```

в котором метод `AddShape` использует в качестве параметров константу `msoShapeRectangle`, специфицирующую форму области как прямоугольник. Следующие далее в качестве параметров 2 числа задают координаты левой верхней вершины прямоугольника, а последние 2 числа — длину и ширину этой области.

Далее следуют операторы с текстом самой надписи и параметрами шрифта. Завершается программа операторами, устанавливающими способ выравнивания этой надписи, в данном случае — по центру горизонтали и вертикали.

После написания текста программы она может быть запущена непосредственно из редактора Visual Basic, для чего следует выполнить операцию главного меню: **Run** | **Run**, нажать функциональную клавишу <F5> или соответствующую кнопку на главной панели инструментов редактора.

Запуск подпрограмм из рабочего листа MS Excel, так же, как запуск макросов в общем случае, может быть выполнен с помощью специального диалогового окна программы MS Excel, которое вызывается посредством выполнения операции главного меню: **Сервис** | **Макрос** | **Макросы** или нажатия комбинацию клавиш: <Alt>+<F8> (рис. 11.7).

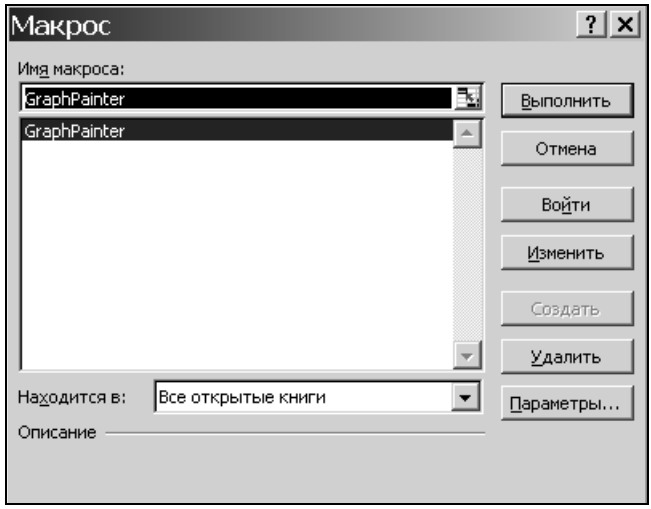


Рис. 11.7. Диалоговое окно запуска подпрограммы или макроса

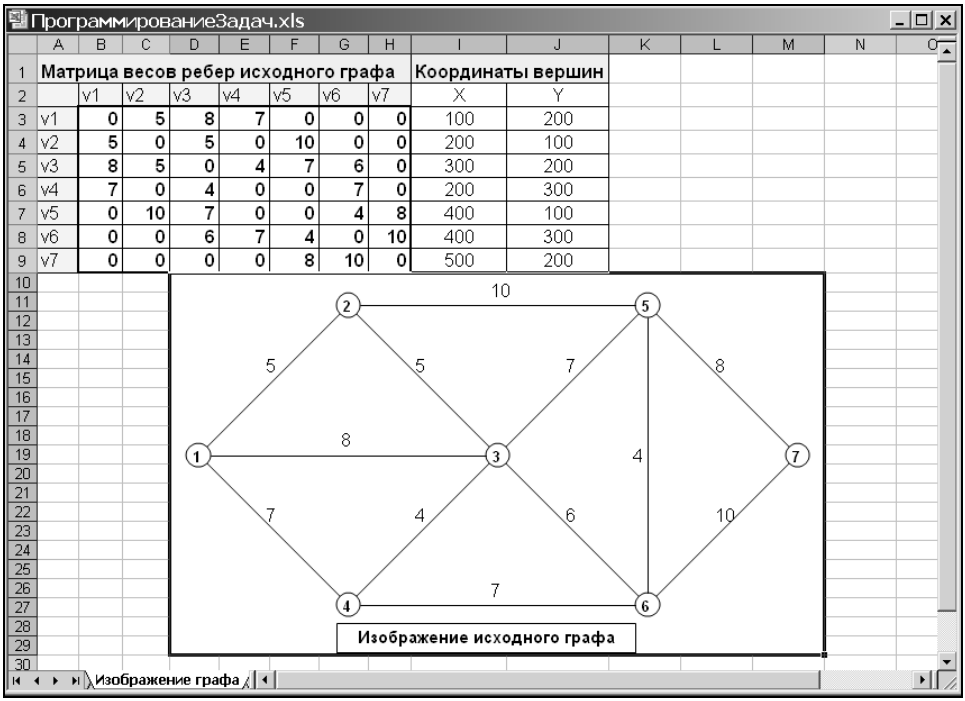


Рис. 11.8. Результат выполнения программы изображением структуры неориентированного графа

В этом окне следует выбрать подпрограмму `GraphPainter` и нажать кнопку **ОК**. Если текст программы не содержит ошибок, то она будет выполнена, и на данном рабочем листе появится рисунок с изображением структуры исходного графа (рис. 11.8).

Для удобства графического представления данного рисунка ячейки **D10:M30** объединены, что может быть выполнено с помощью операции форматирования выделенного диапазона ячеек, доступной из контекстного меню. В дальнейшем рассмотренная программа изображения структуры неориентированного графа после небольших изменений может быть использована для отображения структуры оптимальных деревьев и путей в графах.

11.2. Минимальное покрывающее дерево графа и его графическое изображение

Содержательная постановка задачи о минимальном покрывающем дереве графа приводится в *разд. 1.2.9*, математическая постановка задачи приводится в *разд. 7.2.1*. Решение данной задачи с помощью программы MS Excel

приводится в *разд. 7.2.2*. Особенностью процесса ее решения является тот факт, что предварительно следует сформировать m переменных, где m — число ребер исходного графа, и ввести в ячейки рабочего листа $n + 1$ ограничений вида (7.2.8), где n — число вершин исходного графа. Данный процесс для значений m, $n > 20$ представляется весьма рутинной процедурой, поэтому возникает естественное желание разработать специальную программу, которая на основе заданной матрицы весов ребер произвольного неориентированного графа находила бы для него минимальное покрывающее дерево.

11.2.1. Программа нахождения минимального покрывающего дерева графа

Для нахождения минимального покрывающего дерева графа целесообразно реализовать рассмотренный в *разд. 7.2.4* алгоритм, что позволит избежать задания исходных переменных и ограничений для любой индивидуальной задачи этого типа. При этом реализацию данного алгоритма в виде пользовательской функции следует выполнить таким образом, чтобы она позволяла находить оптимальное дерево для произвольного количества вершин и ребер исходного графа.

Для записи текста программы нахождения минимального покрывающего дерева графа воспользуемся модулем Module1, который был ранее создан и имеется в книге с именем Программирование задач, либо следует создать новый модуль описанным ранее способом. В данный модуль введем следующий текст программы, оформленной в виде отдельной функции с именем `MinTree`, которая использует единственный входной параметр для спецификации матрицы весов ребер исходного графа (листинг 11.3).

Листинг 11.3. Программа нахождения минимального покрывающего дерева графа

```
Function MinTree(C As Variant) As Variant
 Dim I, J, K, Im, Jm As Integer
 Dim Dis(), Tree(), Vlink() As Integer
 Dim S, S1 As String
 N = C.Columns.Count
 ReDim Dis(N, N), Tree(N, N), Vlink(N)
' -- Подготовка исходных массивов
 For I = 1 To N
    Vlink(I) = 0  '<-- этот массив хранит номера вершин, добавляемых к ' дереву
    For J = 1 To N
       Dis(I, J) = C(I, J)
```

```
      Tree(I, J) = 0 '<-- этот массив хранит структуру оптимального
' дерева
    Next J
 Next I
 K = 1
 Vlink(1) = 1
' -- Нахождение ребра с минимальным весом
 While K <= N
    Rec = 1000
    For I = 1 To N
       For J = 1 To N
   If Dis(I, J)<>0 And Rec>Dis(I, J) And Vlink(I)=1 And Vlink(J)=0 Then
            Rec = Dis(I, J)
            Im = I
            Jm = J
         End If
       Next J
    Next I
' -- Добавление найденной вершины к покрывающему дереву
    Vlink(Jm) = 1
    Tree(Im, Jm) = 1
    Tree(Jm, Im) = 1
    K = K + 1
 Wend
' -- Расчет значения суммы весов ребер оптимального дерева и структуры
' дерева
 Rec = 0
 K = 0
 S = ""
 For I = 1 To N - 1
    For J = I + 1 To N
       If Tree(I, J) = 1 Then
          K = K + 1
          S = S + " (" + CStr(I) + "," + CStr(J) + "),"
          Rec = Rec + Dis(I, J)
       End If
    Next J
 Next I
 S1 = "Минимальное покрывающее дерево:"
```

```
 S = S1 + S + "  Fmin= " + CStr(Rec)
 MinTree = S '<-- возвращаемое функцией значение
 Erase Dis, Tree, Vlink '-- уничтожаем все динамические массивы
End Function
```

Приведем некоторые пояснения по тексту программы. Во-первых, программа оформлена в виде функции `MinTree` с единственным аргументом, в качестве которого выступает матрица весов ребер исходного графа. Основные массивы программы объявляются как динамические, что дает возможность предварительно не указывать их размерность и позволяет не задавать в программе в качестве аргумента функции количество вершин графа. Это унифицирует применение данной функции к графам с произвольным количеством вершин.

Собственно количество вершин исходного графа вычисляется оператором: `N = C.Columns.Count`, который присваивает переменной `N` значение количества столбцов матрицы весов ребер, которые используются в качестве исходных данных. В последующем после получения аргумента данной функцией все массивы переопределяются с целью точного указания их размеров: `ReDim Dis(N, N), Tree(N, N), Vlink(N)`.

Массив `Dis(N, N)` служит для хранения значений матрицы смежности или весов ребер, массив `Tree(N, N)` — для хранения структуры оптимального дерева, а массив `Vlink(N)` — для хранения номеров вершин, которые последовательно добавляются к оптимальному дереву.

Первоначально согласно данному алгоритму оптимальному дереву принадлежит вершина с номером 1, что фиксируется оператором: `Vlink(1) = 1`. Далее к этой вершине последовательно добавляются остальные, причем на каждом шаге цикла с предусловием `While` добавляется ровно одна вершина с минимальным ребром, соединяющим уже принадлежащие оптимальному дереву вершины с одной из не принадлежащих ему вершин.

После завершения этого цикла массив `Tree(N, N)` будет содержать ребра, которые входят в минимальное покрывающее дерево графа. Для их получения на выходе функции служит последний двойной цикл, на каждом шаге которого к строковой переменной `S` добавляется обозначение соответствующего ребра оптимального дерева. В качестве результата данная функция возвращает в выбранную ячейку строку текста, содержащего список ребер, образующих минимальное покрывающее дерево исходного графа, и оптимальное значение соответствующей целевой функции.

После написания текста программы соответствующая функция может быть вызвана из любого рабочего листа этой книги. Для удобства тестирования в книге Программирование задач создадим новый рабочий лист с именем

Минимальное дерево, в который скопируем ячейки **A1:H9** с рабочего листа Изображение графа (рис. 11.6).

Далее из любой свободной ячейки следует вызвать мастер добавления функции и в списке функций, определенных пользователем, выбрать созданную функцию: `MinTree()`, как это изображено на рис. 11.9.

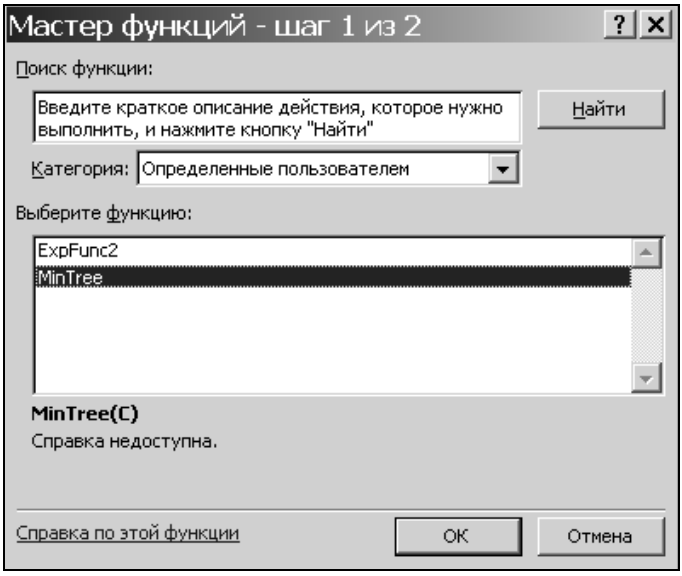


Рис. 11.9. Окно мастера выбора определенной пользователем функции

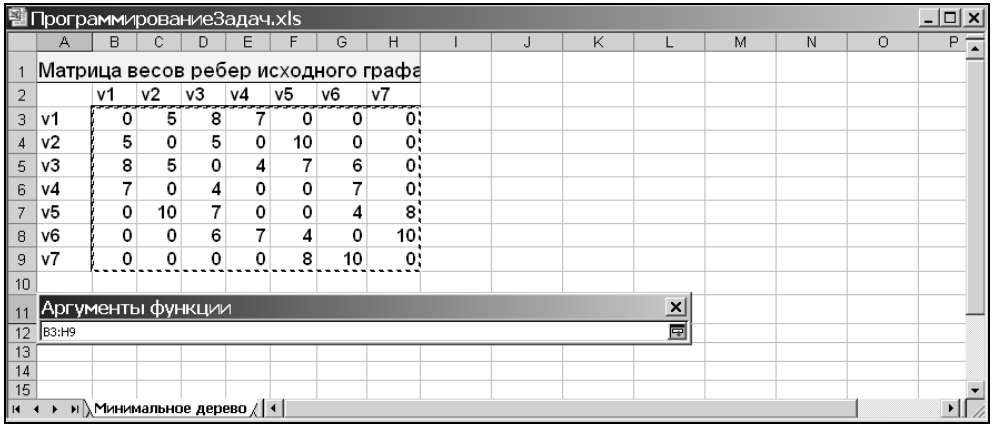
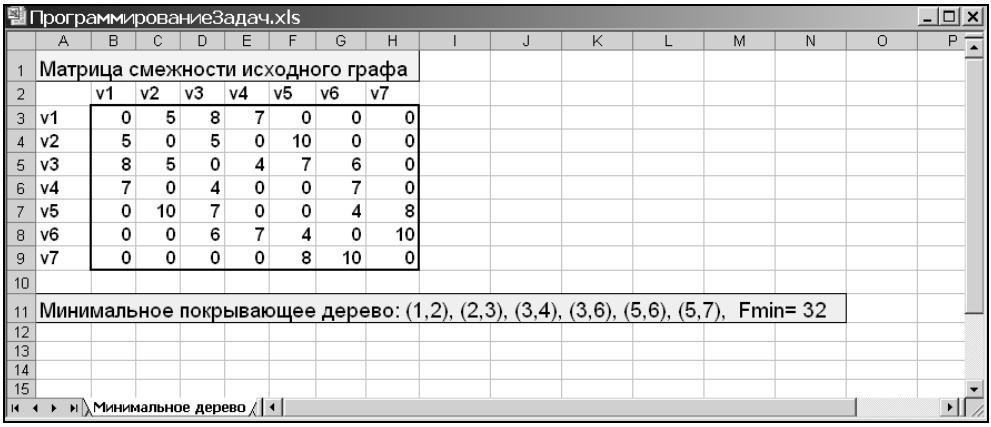


Рис. 11.10. Окно мастера задания аргументов пользовательской функции

После выбора данной функции мастер предлагает специфицировать значения ее аргументов. С этой целью следует выделить диапазон ячеек **B3:H9**, в которых содержатся значения матрицы весов ребер исходного графа (рис. 11.10).

После задания аргументов и нажатия кнопки **ОК**, если текст программы не содержит ошибок, она будет выполнена, а в ячейке с этой функцией на данном рабочем листе появится результат решения задачи нахождения минимального покрывающего дерева графа (рис. 11.11).



	v1	v2	v3	v4	v5	v6	v7
v1	0	5	8	7	0	0	0
v2	5	0	5	0	10	0	0
v3	8	5	0	4	7	6	0
v4	7	0	4	0	0	7	0
v5	0	10	7	0	0	4	8
v6	0	0	6	7	4	0	10
v7	0	0	0	0	8	10	0

Рис. 11.11. Результат решения задачи нахождения минимального покрывающего дерева графа

Как показывает анализ найденного решения, оно полностью совпадает с решением данной задачи, полученным с помощью программы MS Excel в *разд. 7.2.2*. Данный факт может свидетельствовать в пользу правильности разработанной программы. Далее можно отобразить структуру найденного решения, для чего целесообразно модифицировать ранее написанную программу для отображения структуры неориентированного графа.

11.2.2. Программа изображения минимального покрывающего дерева графа

Для изображения структуры минимального покрывающего дерева графа следует дополнить матрицу весов ребер исходного графа значениями координат вершин. Поскольку структура рассматриваемого графа уже изображалась ранее, можно скопировать ячейки **I1:J9** с координатами вершин из рабочего листа Изображение графа (рис. 11.6) в соответствующие ячейки рабочего листа с именем Минимальное дерево либо ввести соответствующие данные заново.

Для записи текста программы изображения структуры минимального покрывающего дерева графа в Module2 введем следующий текст программы, оформленной в виде отдельной подпрограммы с именем `MinTreePainter` без входных параметров (листинг 11.4).

Листинг 11.4. Программа изображения структуры минимального покрывающего дерева графа

```
Public Sub MinTreePainter()
' -- Спецификация переменных и массивов
 Const N = 7 ' <-- это количество вершин графа
 Dim I, J, K, Im, Jm, YSdvig, Rec As Integer
 Dim S As String
 Dim Vert(N), Sh As Shape
 Dim Dis(N, N), Tree(N, N), Xvert(N), Yvert(N), Vlink(N) As Integer
 YSdvig = 60 '<-- это сдвиг рисунка по вертикали
' -- Ввод и формирование матрицы весов ребер и координат вершин
 For I = 1 To N
    Xvert(I) = ActiveSheet.Cells(I + 2, N + 2)
    Yvert(I) = ActiveSheet.Cells(I + 2, N + 3)
        For J = 1 To N
            Dis(I, J) = ActiveSheet.Cells(I + 2, J + 1)
        Next J
 Next I
' -- Подготовка исходных массивов
 For I = 1 To N
    Vlink(I) = 0  '<-- этот массив хранит номера вершин, добавляемых к
' дереву
    For J = 1 To N
       Tree(I, J) = 0 '<-- этот массив хранит структуру оптимального
' дерева
    Next J
 Next I
 K = 1
 Vlink(1) = 1
' -- Нахождение ребра с минимальным весом
 While K <= N
    Rec = 1000
    For I = 1 To N
```

```
      For J = 1 To N
      If Dis(I,J)<>0 And Rec>Dis(I,J) And Vlink(I)=1 And Vlink(J)=0 Then
            Rec = Dis(I, J)
            Im = I
            Jm = J
         End If
      Next J
   Next I
' -- Добавление найденной вершины к покрывающему дереву
   Vlink(Jm) = 1
   Tree(Im, Jm) = 1
   Tree(Jm, Im) = 1
   K = K + 1
 Wend
' -- Расчет значения суммы весов ребер оптимального дерева и структуры
' дерева
 Rec = 0
 K = 0
 For I = 1 To N - 1
   For J = I + 1 To N
      If Tree(I, J) = 1 Then
         K = K + 1
         Rec = Rec + Dis(I, J)
      End If
   Next J
 Next I
 S = "Минимальное покрывающее дерево:" + "  Fmin= " + CStr(Rec)
' -- Изображение вершин графа
 For I = 1 To N
   Set Vert(I) = ActiveSheet.Shapes.AddShape(msoShapeOval, Xvert(I), _
Yvert(I) + YSdvig, 16, 16)
   Vert(I).TextFrame.Characters.Text = CStr(I)
   Vert(I).TextFrame.Characters.Font.Size = 10
   Vert(I).TextFrame.Characters.Font.Bold = True
 Next I
' -- Изображение ребер графа
 For I = 1 To N - 1
   For J = I + 1 To N
     If Tree(I, J) <> 0 Then ' -- изображаем только те ребра, для которых
```

```
' связь не равна 0
  Set Sh = ActiveSheet.Shapes.AddConnector(msoConnectorStraight, 0,0,0,0)
        With Sh.ConnectorFormat
             .BeginConnect ConnectedShape:=Vert(I), ConnectionSite:=1
             .EndConnect ConnectedShape:=Vert(J), ConnectionSite:=1
             Sh.RerouteConnections
        End With
' -- Изображение веса рядом с соответствующим ребром
        If Xvert(I) = Xvert(J) Then ' -- две вершины расположены на одной
' вертикали
   Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
   (Xvert(I)+Xvert(J))/2 - 5, (Yvert(I)+Yvert(J))/2 + YSdvig, 60, 150)
           Else ' -- две вершины не расположены на одной вертикали
    Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
   (Xvert(I)+Xvert(J))/2, (Yvert(I)+Yvert(J))/2 + YSdvig - 10, 60, 150)
        End If
        Sh.TextFrame.Characters.Text = CStr(Dis(I, J))
        Sh.TextFrame.Characters.Font.Size = 12
     End If
   Next J
 Next I
' -- Изображение подписи под рисунком
 Set Sh = ActiveSheet.Shapes.AddShape(msoShapeRectangle, 150,380,300,20)
 Sh.TextFrame.Characters.Text = S
 Sh.TextFrame.Characters.Font.Size = 11
 Sh.TextFrame.Characters.Font.Bold = True
 Sh.TextFrame.HorizontalAlignment = xlHAlignCenter
 Sh.TextFrame.VerticalAlignment = xlVAlignCenter
End Sub
```

Нетрудно заметить, что данная программа является комбинацией 2-х ранее рассмотренных программ: изображения структуры неориентированного графа и функции нахождения минимального покрывающего дерева. Подобное решение принято на том основании, чтобы обеспечить независимость использования соответствующей пользовательской функции и подпрограммы рисования графической фигуры. В этом случае сам пользователь решает, какую из них ему удобнее использовать для своих целей.

Программа изображения структуры минимального покрывающего дерева графа использует в качестве исходных данных матрицу весов ребер графа

и координаты его вершин. При этом координаты вершин задаются внутри подпрограммы с помощью операторов: `Xvert(I) = ActiveSheet.Cells(I + 2, N + 2)` и `Yvert(I) = ActiveSheet.Cells(I + 2, N + 3)`, а матрица весов ребер — с помощью оператора: `Dis(I, J) = ActiveSheet.Cells(I + 2, J + 1)`. При этом адреса ячеек с соответствующими данными строго фиксированы и должны корректироваться в программе для каждой конкретной задачи. Предварительно должно быть явно задано количество вершин исходного графа.

Далее следует реализация рассмотренного ранее (см. листинг 11.3) алгоритма нахождения минимального покрывающего дерева графа, структура которого определяется матрицей смежности Tree(N, N). Именно эта матрица используется в качестве исходной для изображения структуры минимального покрывающего дерева последующими операторами, назначение которых уже было рассмотрено ранее (листинг 11.2). Изменить положение рисунка по вертикали можно с помощью задания значения переменной `Ysdvig`.

После написания текста программы она может быть запущена из данного рабочего листа MS Excel с помощью специального диалогового окна программы MS Excel, которое вызывается посредством выполнения операции главного меню: **Сервис** | **Макрос** | **Макросы** или нажатия комбинации клавиш: <Alt>+<F8> (рис. 11.12).

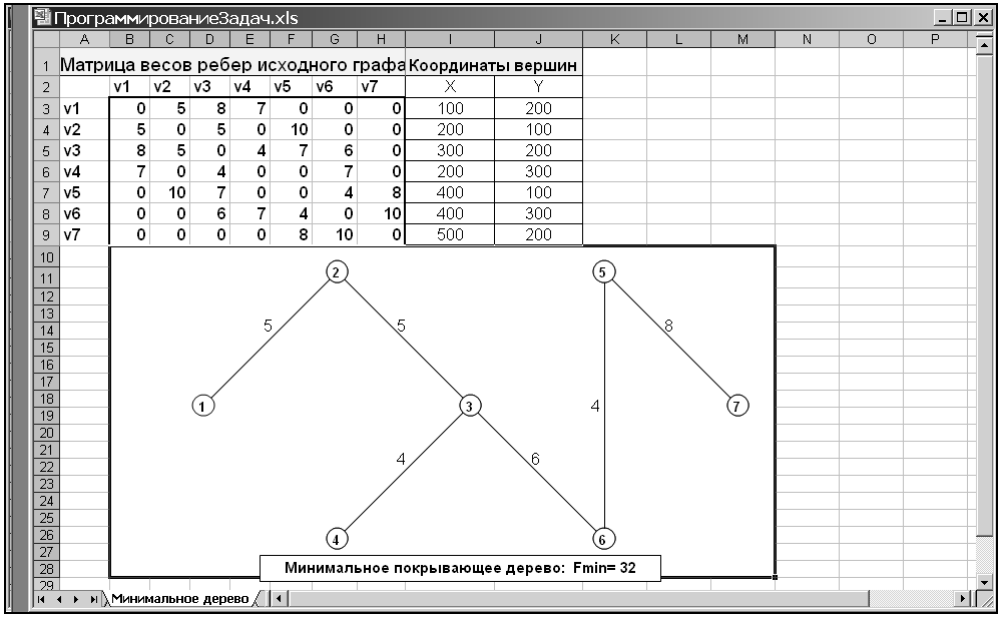


Рис. 11.12. Результат изображения минимального покрывающего дерева графа подпрограммой `MinTreePainter`

Как можно заметить, изображение минимального покрывающего дерева графа, полученное с помощью подпрограммы `MinTreePainter`, полностью совпадает со структурой дерева, найденного ранее с помощью программы MS Excel (см. рис. 7.5). Данная подпрограмма может быть использована для нахождения и изображения минимального покрывающего дерева произвольного графа. В этом случае необходимо скорректировать ее текст, задав необходимое количество вершин графа N, матрицу весов ребер и координаты вершин.

11.3. Максимальное покрывающее дерево графа и его графическое изображение

Для нахождения максимального покрывающего дерева графа также целесообразно реализовать рассмотренный в *разд. 7.2.4* алгоритм, что позволяет избежать задания исходных переменных и ограничений для любой индивидуальной задачи этого типа. При этом реализацию данного алгоритма следует выполнить в виде отдельной пользовательской функции, чтобы избежать использования дополнительных параметров.

11.3.1. Программа нахождения максимального покрывающего дерева графа

Для записи текста программы нахождения максимального покрывающего дерева графа воспользуемся модулем Module1, который был ранее создан и имеется в книге с именем Программирование задач. В данный модуль введем следующий текст программы, оформленной в виде отдельной функции с именем `MaxTree`, которая использует единственный входной параметр для спецификации матрицы весов ребер исходного графа (листинг 11.5).

Листинг 11.5. Программа нахождения максимального покрывающего дерева графа

```
Function MaxTree(C As Variant) As Variant
 Dim I, J, K, Im, Jm As Integer
 Dim Dis(), Tree(), Vlink() As Integer
 Dim S As String
 N = C.Columns.Count
 ReDim Dis(N, N), Tree(N, N), Vlink(N)
' -- Подготовка исходных массивов
 For I = 1 To N
```

```
    Vlink(I) = 0  '<-- этот массив хранит номера вершин, добавляемых к
' дереву
    For J = 1 To N
       Dis(I, J) = C(I, J)
       Tree(I, J) = 0 '<-- этот массив хранит структуру оптимального
' дерева
    Next J
 Next I
 K = 1
 Vlink(1) = 1
' -- Нахождение ребра с максимальным весом
 While K <= N
    Rec = 0
    For I = 1 To N
       For J = 1 To N
    If Dis(I,J)<>0 And Rec<Dis(I,J) And Vlink(I)=1 And Vlink(J)=0 Then
             Rec = Dis(I, J)
             Im = I
             Jm = J
          End If
       Next J
    Next I
' -- Добавление найденной вершины к покрывающему дереву
    Vlink(Jm) = 1
    Tree(Im, Jm) = 1
    Tree(Jm, Im) = 1
    K = K + 1
 Wend
' -- Расчет значения суммы весов ребер оптимального дерева и структуры
' дерева
 Rec = 0
 K = 0
 S = ""
 For I = 1 To N - 1
    For J = I + 1 To N
       If Tree(I, J) = 1 Then
          K = K + 1
          S = S + " (" + CStr(I) + "," + CStr(J) + "),"
```

```
            Rec = Rec + Dis(I, J)
        End If
    Next J
 Next I
 S = "Максимальное покрывающее дерево:" + S + "  Fmax= " + CStr(Rec)
 MaxTree = S '<-- возвращаемое функцией значение
 Erase Dis, Tree, Vlink '-- уничтожаем все динамические массивы
End Function
```

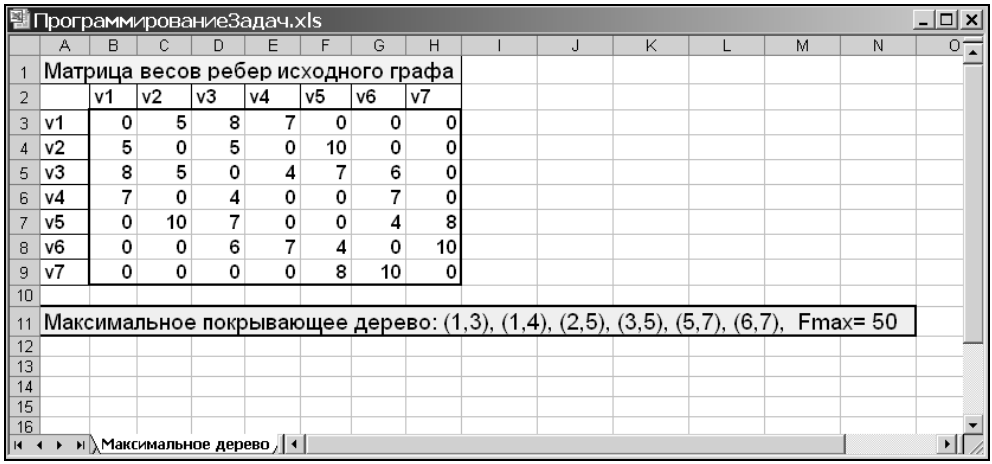
Данная программа практически не требует пояснений, поскольку отличается от программы функции `MinTree()` буквально парой операторов. После выполнения необходимых действий алгоритма массив `Tree(N, N)` будет содержать ребра, которые входят в максимальное покрывающее дерево графа. В качестве результата данная функция возвращает в выбранную ячейку строку текста `S`, содержащего список ребер, образующих максимальное покрывающее дерево исходного графа, и оптимальное значение соответствующей целевой функции.

После написания текста программы соответствующая функция может быть вызвана из любого рабочего листа этой книги. Для удобства тестирования в книге Программирование задач создадим новый рабочий лист с именем Максимальное дерево, в который скопируем ячейки **A1:H9** с рабочего листа Изображение графа (рис. 11.6).

Далее из любой свободной ячейки следует вызвать мастер добавления функции и в списке функций, определенных пользователем, выбрать созданную функцию: `MaxTree()`. Выбор данной функции и спецификация значений ее аргументов выполняется аналогично ранее рассмотренной функции `MinTree()`. А именно в качестве единственного аргумента этой функции следует выделить диапазон ячеек **B3:H9**, в которых содержатся значения матрицы весов ребер исходного графа.

После задания аргументов следует нажать кнопку **ОК**. Если текст программы не содержит ошибок, то она будет выполнена, а в ячейке с этой функцией на данном рабочем листе появится результат решения задачи нахождения максимального покрывающего дерева графа (рис. 11.13).

Как показывает анализ найденного решения, оно полностью совпадает с решением данной задачи, полученным с помощью программы MS Excel в *разд. 7.2.3*. Данный факт может свидетельствовать в пользу правильности разработанной программы. Далее можно отобразить структуру найденного решения, для чего целесообразно использовать отдельную программу.



	v1	v2	v3	v4	v5	v6	v7
v1	0	5	8	7	0	0	0
v2	5	0	5	0	10	0	0
v3	8	5	0	4	7	6	0
v4	7	0	4	0	0	7	0
v5	0	10	7	0	0	4	8
v6	0	0	6	7	4	0	10
v7	0	0	0	0	8	10	0

Рис. 11.13. Результат решения задачи нахождения максимального покрывающего дерева графа

11.3.2. Программа изображения максимального покрывающего дерева графа

Для изображения структуры максимального покрывающего дерева графа также следует дополнить матрицу весов ребер исходного графа значениями координат вершин. Поскольку структура рассматриваемого графа уже изображалась ранее, можно скопировать ячейки **I1:J9** с координатами вершин из рабочего листа Изображение графа (см. рис. 11.6) в соответствующие ячейки рабочего листа с именем Максимальное дерево либо ввести соответствующие данные заново.

Для записи текста программы изображения структуры максимального покрывающего дерева графа в Module2 введем следующий текст программы, оформленной в виде отдельной подпрограммы с именем `MaxTreePainter` без входных параметров (листинг 11.6).

Листинг 11.6. Программа изображения структуры максимального покрывающего дерева графа

```
Public Sub MaxTreePainter()
' -- Спецификация переменных и массивов
 Const N = 7 ' <-- это количество вершин графа
 Dim I, J, K, Im, Jm, YSdvig, Rec As Integer
 Dim S As String
 Dim Vert(N), Sh As Shape
```

```
 Dim Dis(N, N), Tree(N, N), Xvert(N), Yvert(N), Vlink(N) As Integer
 YSdvig = 60 '<-- это сдвиг рисунка по вертикали
' -- Ввод и формирование матрицы смежности
 For I = 1 To N
    Xvert(I) = ActiveSheet.Cells(I + 2, N + 2)
    Yvert(I) = ActiveSheet.Cells(I + 2, N + 3)
        For J = 1 To N
            Dis(I, J) = ActiveSheet.Cells(I + 2, J + 1)
        Next J
 Next I
' -- Подготовка исходных массивов
 For I = 1 To N
    Vlink(I) = 0  '<-- этот массив хранит номера вершин, добавляемых к
' дереву
    For J = 1 To N
       Tree(I, J) = 0 '<-- этот массив хранит структуру оптимального
' дерева
    Next J
 Next I
 K = 1
 Vlink(1) = 1
' -- Нахождение ребра с минимальным весом
 While K <= N
    Rec = 0
    For I = 1 To N
       For J = 1 To N
   If Dis(I, J)<>0 And Rec<Dis(I, J) And Vlink(I)=1 And Vlink(J)=0 Then
             Rec = Dis(I, J)
             Im = I
             Jm = J
          End If
       Next J
    Next I
' -- Добавление найденной вершины к покрывающему дереву
    Vlink(Jm) = 1
    Tree(Im, Jm) = 1
    Tree(Jm, Im) = 1
    K = K + 1
 Wend
```

```
' -- Расчет значения суммы весов ребер оптимального дерева и структуры
' дерева
 Rec = 0
 K = 0
 For I = 1 To N - 1
    For J = I + 1 To N
       If Tree(I, J) = 1 Then
          K = K + 1
          Rec = Rec + Dis(I, J)
       End If
    Next J
 Next I
 S = "Максимальное покрывающее дерево:" + "  Fmin= " + CStr(Rec)
' -- Изображение вершин графа
 For I = 1 To N
   Set Vert(I) = ActiveSheet.Shapes.AddShape(msoShapeOval, Xvert(I), _
Yvert(I) + YSdvig, 16, 16)
    Vert(I).TextFrame.Characters.Text = CStr(I)
    Vert(I).TextFrame.Characters.Font.Size = 10
    Vert(I).TextFrame.Characters.Font.Bold = True
 Next I
' -- Изображение ребер графа
 For I = 1 To N - 1
   For J = I + 1 To N
     If Tree(I, J) <> 0 Then ' -- изображаем только те ребра, для которых
' связь не равна 0
  Set Sh = ActiveSheet.Shapes.AddConnector(msoConnectorStraight, 0,0,0,0)
       With Sh.ConnectorFormat
            .BeginConnect ConnectedShape:=Vert(I), ConnectionSite:=1
            .EndConnect ConnectedShape:=Vert(J), ConnectionSite:=1
            Sh.RerouteConnections
       End With
' -- Изображение веса рядом с соответствующим ребром
       If Xvert(I) = Xvert(J) Then ' -- две вершины расположены на одной
' вертикали
  Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
(Xvert(I)+Xvert(J))/2 - 5, (Yvert(I)+Yvert(J))/2 + YSdvig, 60,150)
          Else ' -- две вершины не расположены на одной вертикали
Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
```

```
(Xvert(I)+Xvert(J))/2, (Yvert(I)+Yvert(J))/2 + YSdvig - 10, 60, 150)
        End If
        Sh.TextFrame.Characters.Text = CStr(Dis(I, J))
        Sh.TextFrame.Characters.Font.Size = 12
     End If
   Next J
 Next I
' -- Изображение подписи под рисунком
 Set Sh = ActiveSheet.Shapes.AddShape(msoShapeRectangle, 150,380,300,20)
 Sh.TextFrame.Characters.Text = S
 Sh.TextFrame.Characters.Font.Size = 11
 Sh.TextFrame.Characters.Font.Bold = True
 Sh.TextFrame.HorizontalAlignment = xlHAlignCenter
 Sh.TextFrame.VerticalAlignment = xlVAlignCenter
End Sub
```

Нетрудно заметить, что данная программа также является комбинацией 2-х ранее рассмотренных программ: изображения структуры неориентированного графа и функции нахождения максимального покрывающего дерева.

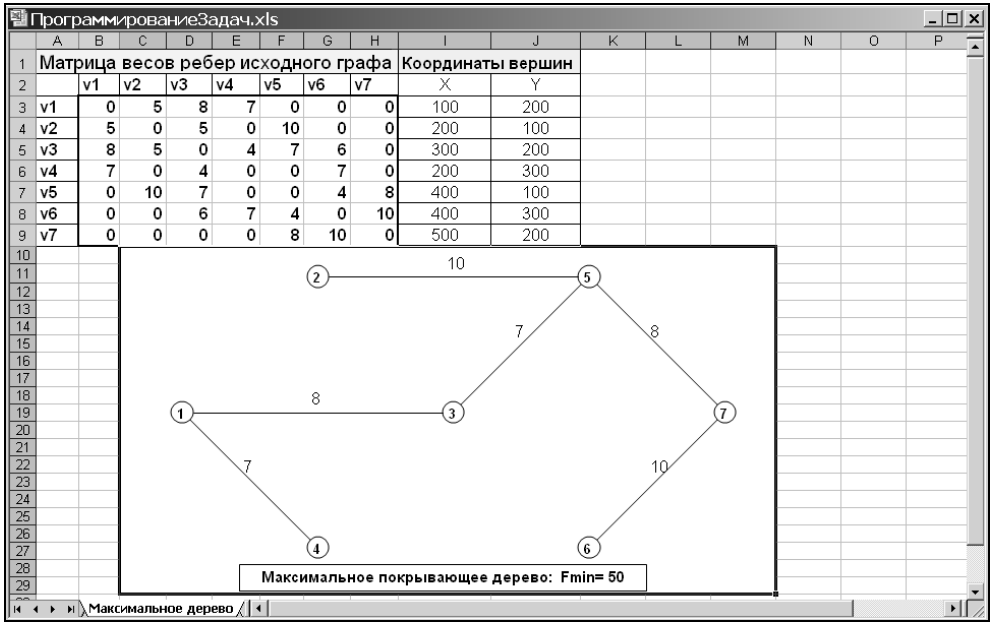


Рис. 11.14. Результат изображения максимального покрывающего дерева графа подпрограммой MaxTreePainter

Программа изображения структуры максимального покрывающего дерева графа использует в качестве исходных данных матрицу весов ребер графа и координаты его вершин. После написания текста программы она может быть запущена из данного рабочего листа MS Excel с помощью специального диалогового окна программы MS Excel, которое вызывается посредством выполнения операции главного меню: **Сервис** | **Макрос** | **Макросы** или нажатия комбинации клавиш: <Alt>+<F8> (рис. 11.14).

Как можно заметить, изображение максимального покрывающего дерева графа, полученное с помощью подпрограммы `MaxTreePainter`, полностью совпадает со структурой дерева, найденного ранее с помощью программы MS Excel (см. рис. 7.7). Данная подпрограмма может быть использована для нахождения и изображения максимального покрывающего дерева произвольного графа. В этом случае необходимо скорректировать ее текст, задав необходимое количество вершин графа N, матрицу весов ребер и координаты вершин.

11.4. Путь минимальной длины и его графическое изображение

Содержательная постановка задачи о минимальном пути в ориентированном графе приводится в *разд. 1.2.5*, математическая постановка задачи приводится в *разд. 7.3.1*. Решение данной задачи с помощью программы MS Excel приводится в *разд. 7.3.2*. Особенностью процесса ее решения является тот факт, что предварительно следует определить m переменных, где m — число дуг исходного графа, а также сформировать и ввести в ячейки рабочего листа n ограничений вида (7.3.2)—(7.3.4), где n — число вершин исходного графа. Данный процесс для значений $m, n > 20$ представляется весьма рутинной процедурой, поэтому возникает естественное желание разработать специальную программу, которая на основе заданной матрицы весов дуг произвольного ориентированного графа находила бы путь минимальной длины для 2-х заданных вершин.

11.4.1. Программа нахождения минимального пути в ориентированном графе

Для нахождения минимального пути в ориентированном графе целесообразно реализовать рассмотренный в *разд. 7.3.3* алгоритм пометок Дейкстры, что позволит избежать задания исходных переменных и ограничений для любой индивидуальной задачи этого типа. При этом реализацию данного алгоритма в виде пользовательской функции следует выполнить таким образом, чтобы

она позволяла находить минимальный путь в ориентированном графе для произвольного количества вершин и дуг.

Для записи текста программы минимального пути в ориентированном графе воспользуемся модулем Module1, который был ранее создан и имеется в книге с именем Программирование задач. В данный модуль введем следующий текст программы, оформленной в виде отдельной функции с именем `MinPath`, которая использует единственный входной параметр для спецификации матрицы весов дуг исходного графа (листинг 11.7).

Листинг 11.7. Программа нахождения минимального пути в ориентированном графе

```
Function MinPath(C As Variant) As Variant
 Dim I, J, K, Im, Jm As Integer
 Dim Dis(), DisMin(), Vpom(), VertMin() As Integer
 Dim S, S1 As String
 N = C.Columns.Count
 ReDim Dis(N, N), DisMin(N), Vpom(N), VertMin(N)
' -- Подготовка исходных массивов
 For I = 1 To N
    DisMin(I) = 1000 '<-- вектор минимальных расстояний из начальной
' вершины
    Vpom(I) = 0 '<-- здесь: 1 - постоянная пометка, 0 - временная пометка
    VertMin(I) = 0 '<-- здесь номер вершины, куда ведет минимальный путь
       For J = 1 To N
          Dis(I, J) = C(I, J)
       Next J
 Next I
 DisMin(1) = 0
 Vpom(1) = 1
 VertMin(1) = 0
 K = 1
 Im = 0
' -- Нахождение временных пометок вершин
 While K <= N
    For I = 1 To N - 1
       For J = I + 1 To N
    If Dis(I, J)<>0 And Vpom(J)=0 And DisMin(J)>Dis(I, J)+DisMin(I) Then
             DisMin(J) = Dis(I, J) + DisMin(I)
             VertMin(J) = I
```

```
          End If
       Next J
    Next I
' -- Нахождение постоянной пометки для вершины
    Rec = 1000
    For I = 2 To N
       If Vpom(I) = 0 And Rec > DisMin(I) Then
          Im = I
       End If
    Next I
    Vpom(Im) = 1
    K = K + 1
 Wend
' -- Формирование структуры минимального пути
 Rec = 0
 K = 0
 For I = 1 To N
    Vpom(I) = 0
 Next I
 Jm = N
 For I = 0 To N - 2
    J = N - I
    If J = Jm Then
        Jm = VertMin(J)
        Vpom(Jm) = J
    End If
 Next I
 S = ""
 For I = 1 To N - 1
    If Vpom(I) <> 0 Then
       K = K + 1
       S = S + " (" + CStr(I) + "," + CStr(Vpom(I)) + "),"
       Rec = Rec + Dis(I, Vpom(I))
    End If
 Next I
 S1 = "Минимальный путь из вершины 1 в вершину " + CStr(N) + ": "
 S = S1 + S + "  Fmin= " + CStr(Rec)
 MinPath = S '<-- возвращаемое функцией значение
 Erase Dis, DisMin, Vpom, VertMin '-- уничтожаем все динамические массивы
End Function
```

Данная программа оформлена в виде функции `MinPath` с единственным аргументом, в качестве которого выступает матрица весов дуг исходного графа. Основные массивы программы также объявляются как динамические, что дает возможность предварительно не указывать их размерность и позволяет не задавать в программе в качестве аргумента функции количество вершин графа. Это унифицирует применение данной функции к графам с произвольным количеством вершин.

Массив `Dis(N, N)` служит для хранения значений матрицы весов дуг, массив `DisMin(N)` — для хранения временных пометок вершин, а массив `Vpom(N)` — для хранения постоянных пометок вершин, массив `VertMin(N)` — для хранения номеров вершин, которые входят в минимальный путь.

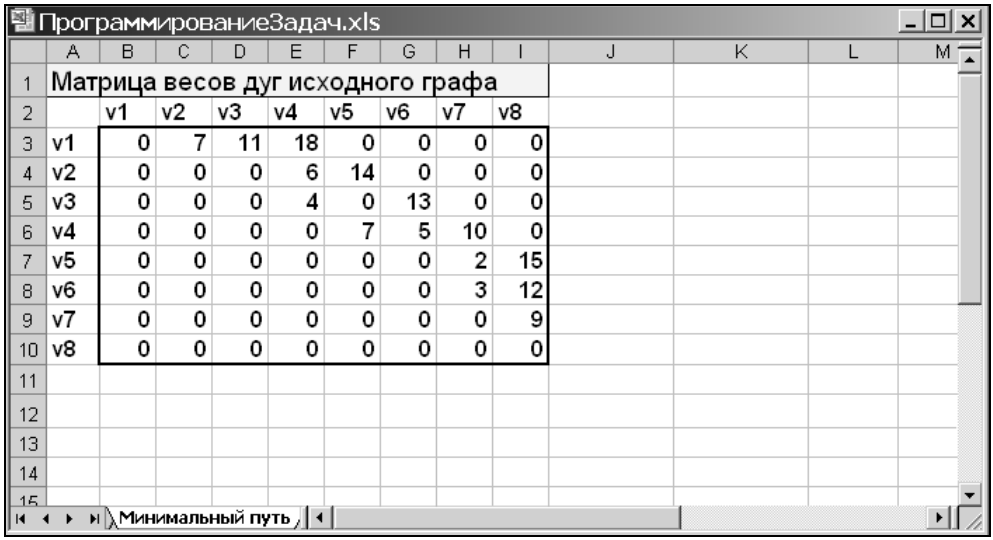
Согласно реализации алгоритма пометок Дейкстры (см. рис. 7.14) всем вершинам исходного графа присваиваются временные пометки. В последующем из них выбирается минимальное значение, которое становится постоянной меткой соответствующей вершины. Эти действия в программе выполняются с помощью циклов со счетчиком, которые прокомментированы в тексте.

После завершения этих циклов необходимо сформировать структуру пути и записать ее в массив `VertMin(N)`. Для их получения на выходе функции служит последний двойной цикл. В качестве результата данная функция возвращает в выбранную ячейку строку текста `S`, содержащего список дуг, образующих путь минимальной длины из вершины с номером 1 в вершину с номером N, где N — количество вершин исходного графа, и оптимальное значение соответствующей целевой функции.

После написания текста программы соответствующая функция может быть вызвана из любого рабочего листа этой книги. С этой целью в книге Программирование задач создадим новый рабочий лист с именем Минимальный путь, в который введем в качестве исходных данных матрицу весов дуг рассмотренного ранее ориентированного графа (см. рис. 7.9).

Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 11.15).

Далее из любой свободной ячейки следует вызвать мастер добавления функции и в списке функций, определенных пользователем, выбрать созданную функцию: `MinPath()`. После выбора данной функции в качестве значений ее аргументов следует выделить диапазон ячеек **B3:I10**, в которых содержатся значения матрицы весов дуг исходного графа (рис. 11.15). После задания аргументов и нажатия кнопки **ОК**, если текст программы не содержит ошибок, она будет выполнена, а в ячейке с этой функцией на данном рабочем листе появится результат решения задачи нахождения минимального пути в графе (рис. 11.16).



ПрограммированиеЗадач.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Матрица весов дуг исходного графа												
2		v1	v2	v3	v4	v5	v6	v7	v8				
3	v1	0	7	11	18	0	0	0	0				
4	v2	0	0	0	6	14	0	0	0				
5	v3	0	0	0	4	0	13	0	0				
6	v4	0	0	0	0	7	5	10	0				
7	v5	0	0	0	0	0	0	2	15				
8	v6	0	0	0	0	0	0	3	12				
9	v7	0	0	0	0	0	0	0	9				
10	v8	0	0	0	0	0	0	0	0				
11													
12													
13													
14													
15													

Минимальный путь

Рис. 11.15. Исходные данные для решения задачи нахождения минимального пути в графе



ПрограммированиеЗадач.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Матрица весов дуг исходного графа															
2		v1	v2	v3	v4	v5	v6	v7	v8							
3	v1	0	7	11	18	0	0	0	0							
4	v2	0	0	0	6	14	0	0	0							
5	v3	0	0	0	4	0	13	0	0							
6	v4	0	0	0	0	7	5	10	0							
7	v5	0	0	0	0	0	0	2	15							
8	v6	0	0	0	0	0	0	3	12							
9	v7	0	0	0	0	0	0	0	9							
10	v8	0	0	0	0	0	0	0	0							
11																
12	Минимальный путь из вершины 1 в вершину 8: (1,2), (2,4), (4,6), (6,8), Fmin= 30															
13																
14																

Минимальный путь

Рис. 11.16. Результат решения задачи нахождения минимального пути в графе

Как показывает анализ найденного решения, оно полностью совпадает со вторым оптимальным решением данной задачи, полученным с помощью ручного расчета в *разд. 7.3.3*. Данный факт может свидетельствовать в пользу правильности разработанной программы. Далее можно отобразить структуру найденного решения, для чего целесообразно модифицировать ранее написанную программу для отображения структуры неориентированного графа.

11.4.2. Программа изображения минимального пути в ориентированном графе

Для изображения структуры минимального пути в ориентированном графе следует дополнить матрицу весов дуг исходного графа значениями координат вершин. Поскольку структура рассматриваемого графа не изображалась ранее, соответствующие значения координат следует ввести в ячейки **J3:K10** рабочего листа Минимальный путь.

Для записи текста программы изображения структуры минимального покрывающего дерева графа в Module2 введем следующий текст программы, оформленной в виде отдельной подпрограммы с именем `MinPathPainter` без входных параметров (листинг 11.8).

Листинг 11.8. Программа изображения структуры минимального пути в ориентированном графе

```
Public Sub MinPathPainter()
' -- Спецификация переменных и массивов
 Const N = 8 ' <-- это количество вершин графа
 Dim I, J, K, Im, Jm, YSdvig, Rec As Integer
 Dim Dis(N, N), Path(N, N), DisMin(N), Vpom(N), VertMin(N), Xvert(N), _
Yvert(N) As Integer
 Dim S, S1 As String
 Dim Vert(N), Sh As Shape
 YSdvig = 80 '<-- это сдвиг рисунка по вертикали
' -- Ввод и формирование матрицы смежности
 For I = 1 To N
    Xvert(I) = ActiveSheet.Cells(I + 2, N + 2)
    Yvert(I) = ActiveSheet.Cells(I + 2, N + 3)
        For J = 1 To N
            Dis(I, J) = ActiveSheet.Cells(I + 2, J + 1)
        Next J
 Next I
' -- Подготовка исходных массивов
 For I = 1 To N
    DisMin(I) = 1000 '<-- вектор минимальных расстояний из начальной
' вершины
    Vpom(I) = 0 '<-- здесь: 1 - постоянная пометка, 0 - временная пометка
    VertMin(I) = 0 '<-- здесь номер вершины, куда ведет минимальный путь
       For J = 1 To N
```

```
         Path(I, J) = 0
      Next J
 Next I
 DisMin(1) = 0
 Vpom(1) = 1
 VertMin(1) = 0
 K = 1
 Im = 0
' -- Нахождение временных пометок вершин
 While K <= N
    For I = 1 To N - 1
       For J = I + 1 To N
    If Dis(I, J)<>0 And Vpom(J)=0 And DisMin(J)>Dis(I, J)+DisMin(I) Then
             DisMin(J) = Dis(I, J) + DisMin(I)
             VertMin(J) = I
          End If
       Next J
    Next I
' -- Нахождение постоянной пометки для вершины
    Rec = 1000
    For I = 2 To N
       If Vpom(I) = 0 And Rec > DisMin(I) Then
          Im = I
       End If
    Next I
    Vpom(Im) = 1
    K = K + 1
 Wend
' -- Формирование структуры минимального пути
 Rec = 0
 K = 0
 For I = 1 To N
    Vpom(I) = 0
 Next I
 Jm = N
 For I = 0 To N - 2
    J = N - I
    If J = Jm Then
        Jm = VertMin(J)
```

```
        Vpom(Jm) = J
    End If
 Next I
 S = ""
 For I = 1 To N - 1
    If Vpom(I) <> 0 Then
       K = K + 1
       Path(I, Vpom(I)) = 1
       Rec = Rec + Dis(I, Vpom(I))
    End If
 Next I
S = "Минимальный путь из вершины 1 в вершину "+CStr(N)+":  Fmin= " + _
CStr(Rec)
' -- Изображение вершин графа
 For I = 1 To N
  Set Vert(I) = ActiveSheet.Shapes.AddShape(msoShapeOval, Xvert(I), _
    Yvert(I) + YSdvig, 16, 16)
    Vert(I).TextFrame.Characters.Text = CStr(I)
    Vert(I).TextFrame.Characters.Font.Size = 10
    Vert(I).TextFrame.Characters.Font.Bold = True
 Next I
' -- Изображение ребер графа
 For I = 1 To N - 1
   For J = I + 1 To N
     If Path(I, J) <> 0 Then ' -- изображаем только те ребра, для которых
' связь не равна 0
  Set Sh = ActiveSheet.Shapes.AddConnector(msoConnectorStraight, 0,0,0,0)
       With Sh.ConnectorFormat
            .BeginConnect ConnectedShape:=Vert(I), ConnectionSite:=1
            .EndConnect ConnectedShape:=Vert(J), ConnectionSite:=1
       End With
       Sh.RerouteConnections
       Sh.Line.EndArrowheadLength = msoArrowheadLong '-- здесь нужна
' стрелка
       Sh.Line.EndArrowheadStyle = msoArrowheadTriangle '-- форма
' стрелки - треугольник
       Sh.Line.EndArrowheadWidth = msoArrowheadWide '-- стрелка будет
' широкая
' -- Изображение веса рядом с соответствующей дугой
```

```
        If Xvert(I) = Xvert(J) Then ' -- две вершины расположены на одной
' вертикали
  Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
     (Xvert(I)+Xvert(J))/2 - 5, (Yvert(I)+Yvert(J))/2 + YSdvig, 60,150)
           Else ' -- две вершины не расположены на одной вертикали
  Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
     (Xvert(I)+Xvert(J))/2, (Yvert(I)+Yvert(J))/2 + YSdvig - 10, 60, 150)
        End If
        Sh.TextFrame.Characters.Text = CStr(Dis(I, J))
        Sh.TextFrame.Characters.Font.Size = 12
     End If
   Next J
 Next I
' -- Изображение подписи под рисунком
 Set Sh = ActiveSheet.Shapes.AddShape(msoShapeRectangle, 150, 330 + _
YSdvig, 350, 20)
 Sh.TextFrame.Characters.Text = S
 Sh.TextFrame.Characters.Font.Size = 11
 Sh.TextFrame.Characters.Font.Bold = True
 Sh.TextFrame.HorizontalAlignment = xlHAlignCenter
 Sh.TextFrame.VerticalAlignment = xlVAlignCenter
End Sub
```

Данная программа является комбинацией 2-х ранее рассмотренных программ: функции нахождения минимального пути в графе и программы изображения структуры ориентированного графа.

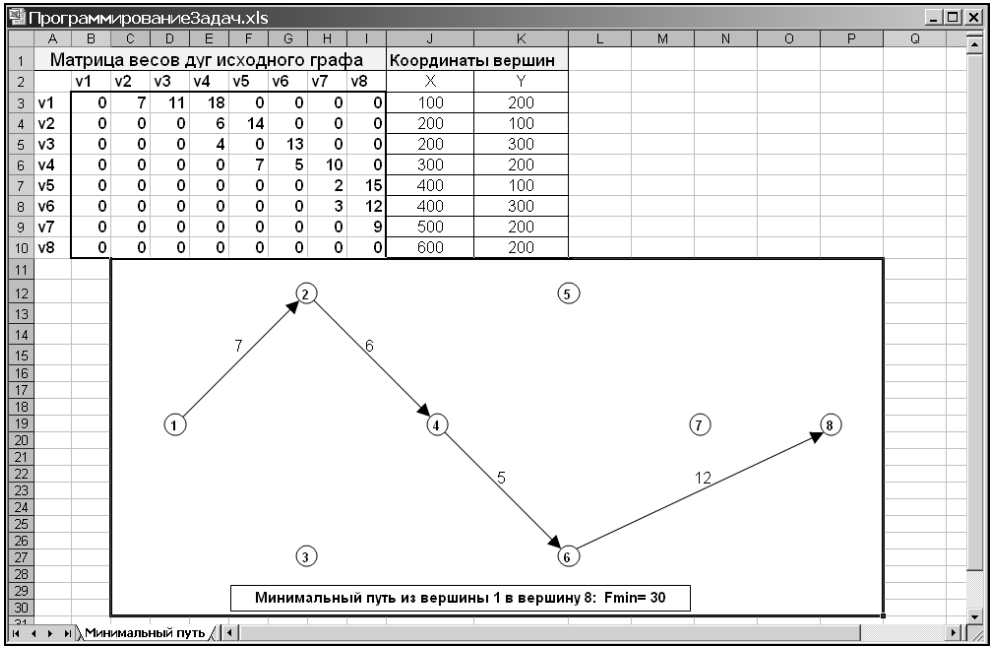
Программа изображения структуры минимального покрывающего дерева графа использует в качестве исходных данных матрицу весов дуг графа и координаты его вершин. При этом координаты вершин задаются внутри подпрограммы с помощью операторов, аналогично ранее рассмотренным подпрограммам отображения структуры графов. При этом адреса ячеек с соответствующими данными также строго фиксированы и должны корректироваться в программе для каждой конкретной задачи. Предварительно должно быть явно задано количество вершин исходного графа.

Далее следует реализация рассмотренного ранее (листинг 11.7) алгоритма нахождения минимального пути в графе, структура которого определяется двумерным массивом `Path(N, N)`. Именно этот массив используется для изображения структуры минимального пути последующими операторами, назначение некоторых из которых уже было рассмотрено ранее (листинг 11.2).

При этом переменная `Ysdvig` служит для изменения положения рисунка по вертикали.

Основная особенность данной подпрограммы заключается в дополнительных операторах, позволяющих изображать стрелку на изображении дуг ориентированного графа. Первый из них — оператор: `Sh.Line.EndArrowheadLength = msoArrowheadLong` специфицирует данную стрелку на концевой вершине связи. Второй — оператор: `Sh.Line.EndArrowheadStyle = msoArrowheadTriangle` задает форму стрелки в виде треугольника. И, наконец, последний оператор этой группы: `Sh.Line.EndArrowheadWidth = msoArrowheadWide` задает ширину стрелки.

После написания текста программы она может быть запущена из данного рабочего листа MS Excel с помощью специального диалогового окна программы MS Excel, которое вызывается посредством выполнения операции главного меню: **Сервис** | **Макрос** | **Макросы** или нажатия комбинации клавиш: <Alt>+<F8> (рис. 11.17).



	v1	v2	v3	v4	v5	v6	v7	v8	X	Y
v1	0	7	11	18	0	0	0	0	100	200
v2	0	0	0	6	14	0	0	0	200	100
v3	0	0	0	4	0	13	0	0	200	300
v4	0	0	0	0	7	5	10	0	300	200
v5	0	0	0	0	0	0	2	15	400	100
v6	0	0	0	0	0	0	3	12	400	300
v7	0	0	0	0	0	0	0	9	500	200
v8	0	0	0	0	0	0	0	0	600	200

Рис. 11.17. Результат изображения минимального пути в ориентированном графе подпрограммой MinPathPainter

Как можно заметить, изображение минимального пути в ориентированном графе, полученное с помощью подпрограммы `MinPathPainter`, полностью

совпадает со вторым оптимальным решением данной задачи, полученным с помощью ручного расчета в *разд. 7.3.3* (см. рис. 7.15). Данная подпрограмма может быть использована для нахождения и изображения минимального покрывающего дерева произвольного графа. В этом случае необходимо скорректировать ее текст, задав необходимое количество вершин графа N, матрицу весов дуг и координаты вершин.

11.5. Путь максимальной длины и его графическое изображение

Содержательная постановка задачи о пути максимальной длины или критическом пути в сетевом графе приводится в *разд. 7.4.1*, математическая постановка задачи приводится в *разд. 7.4.2*. Решение данной задачи с помощью программы MS Excel приводится в *разд. 7.4.3*. Особенностью процесса ее решения является тот факт, что предварительно следует определить m переменных, где m — число дуг исходного графа, а также сформировать и ввести в ячейки рабочего листа n ограничений вида (7.3.2)—(7.3.4), где n — число вершин исходного графа. Данный процесс для значений $m, n > 20$ представляется весьма рутинной процедурой, поэтому возникает естественное желание разработать специальную программу, которая на основе заданной матрицы весов дуг произвольного сетевого графа находила бы путь максимальной длины между начальной и конечной вершинами.

11.5.1. Программа нахождения крит ического пути в сетевом графе

Для нахождения критического пути в сетевом графе целесообразно реализовать рассмотренный в *разд. 7.4.4* алгоритм расстановки постоянных пометок, что позволит избежать задания исходных переменных и ограничений для любой индивидуальной задачи этого типа. При этом реализацию данного алгоритма в виде пользовательской функции следует выполнить таким образом, чтобы она позволяла находить критический путь в сетевом графе для произвольного количества вершин и дуг.

Для записи текста программы критического пути в сетевом графе воспользуемся модулем Module1, который был ранее создан и имеется в книге с именем Программирование задач. В данный модуль введем следующий текст программы, оформленной в виде отдельной функции с именем `MaxPath`, которая использует единственный входной параметр для спецификации матрицы весов дуг исходного графа (листинг 11.9).

Листинг 11.9. Программа нахождения критического пути в сетевом графе

```
Function MaxPath(C As Variant) As Variant
 Dim I, J, K, N As Integer
 Dim Dis(), DisMax(), VertMax() As Integer
 Dim S, S1 As String
 N = C.Columns.Count
 ReDim Dis(N, N), DisMax(N), VertMax(N)
' -- Подготовка исходных массивов
 For I = 1 To N
    DisMax(I) = 0 '<-- вектор максимальных расстояний из начальной
' вершины
    VertMax(I) = 0 '<-- здесь номер вершины, куда ведет максимальный путь
       For J = 1 To N
          Dis(I, J) = C(I, J)
       Next J
 Next I
' -- Нахождение критического пути
 For K = 2 To N
    For I = 1 To K - 1
       If DisMax(I) + Dis(I, K) > DisMax(K) And Dis(I, K) <> 0 Then
          DisMax(K) = DisMax(I) + Dis(I, K)
       End If
    Next I
 Next K
 VertMax(N) = N
 For J = 1 To N - 1
    K = N - J
    For I = K + 1 To N
       If DisMax(K) + Dis(K, I) = DisMax(I) And Dis(K, I) <> 0 And Vert-
Max(I) <> 0 Then
          VertMax(K) = K
       End If
    Next I
 Next J
' -- Формирование структуры критического пути
 Rec = 0
 S = ""
 For I = 1 To N - 1
```

```
    For J = I + 1 To N
       If VertMax(I) <> 0 And VertMax(J) <> 0 And Dis(I, J) <> 0 And _
       DisMax(J) - DisMax(I) = Dis(I, J) Then
          S = S + " (" + CStr(VertMax(I)) + "," + CStr(J) + "),"
          Rec = Rec + Dis(I, J)
       End If
    Next J
 Next I
 S1 = "Критический путь из вершины 1 в вершину " + CStr(N) + ": "
 S = S1 + S + "  Fmax= " + CStr(Rec)
 MaxPath = S '<-- возвращаемое функцией значение
 Erase Dis, DisMax, VertMax '-- уничтожаем все динамические массивы
End Function
```

Данная программа оформлена в виде функции `MaxPath` с единственным аргументом, в качестве которого выступает матрица весов дуг исходного графа. Массив `Dis(N, N)` служит для хранения значений матрицы весов дуг, массив `DisMax(N)` — для хранения значений максимальных путей из начальной вершины, массив `VertMax(N)` — для хранения номеров вершин, которые входят в критический путь.

Согласно реализации алгоритма расстановки постоянных пометок (см. рис. 7.19) для всех вершин исходного графа циклически рассчитываются постоянные пометки. Эти действия в программе выполняются с помощью циклов со счетчиком, которые прокомментированы в тексте.

После завершения этих циклов необходимо сформировать структуру критического пути и записать ее в массив `VertMax(N)`. Для получения соответствующих значений этого массива служит последний двойной цикл. В качестве результата данная функция возвращает в выбранную ячейку строку текста `S`, содержащего список дуг, образующих критический путь из вершины с номером 1 в вершину с номером N, где N — количество вершин исходного сетевого графа, и оптимальное значение соответствующей целевой функции.

После написания текста программы соответствующая функция может быть вызвана из любого рабочего листа этой книги. С этой целью в книге Программирование задач создадим новый рабочий лист с именем Критический путь, в который введем в качестве исходных данных матрицу весов дуг рассмотренного ранее ориентированного графа (см. рис. 7.9). Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь вид аналогичный данным на рабочем листе с именем Минимальный путь (см. рис. 11.15).

Далее из любой свободной ячейки следует вызвать мастер добавления функции и в списке функций, определенных пользователем, выбрать созданную функцию: `MaxPath()`. После выбора данной функции в качестве значений ее аргументов следует выделить диапазон ячеек **B3:I10**, в которых содержатся значения матрицы весов дуг исходного графа. После задания аргументов и нажатия кнопки **ОК**, если текст программы не содержит ошибок, то она будет выполнена, а в ячейке с этой функцией на данном рабочем листе появится результат решения задачи нахождения критического пути в графе (рис. 11.18).

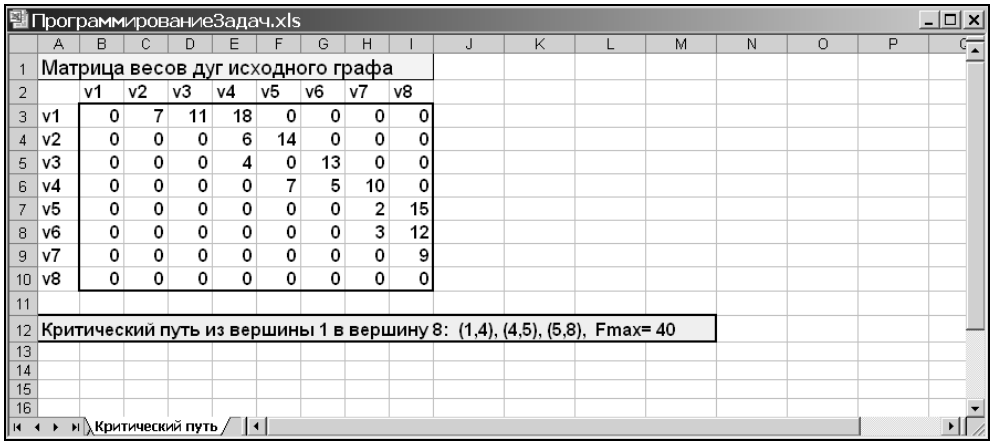


Рис. 11.18. Результат решения задачи нахождения критического пути в графе программой `MaxPath`

Как показывает анализ найденного решения, оно полностью совпадает с оптимальным решением данной задачи, полученным с помощью программы MS Excel в *разд. 7.4.3*. Данный факт может свидетельствовать в пользу правильности разработанной программы. Далее можно отобразить структуру найденного решения, для чего целесообразно использовать ранее написанную программу для отображения структуры ориентированного графа.

11.5.2. Программа изображения критического пути в сетевом графе

Для изображения структуры критического пути в сетевом графе следует дополнить матрицу весов дуг исходного графа значениями координат вершин. Поскольку структура минимального пути рассматриваемого графа уже изображалась ранее, соответствующие значения координат можно скопировать из листа Минимальный путь в ячейки **J3:K10** рабочего листа Критический путь.

Для записи текста программы изображения структуры критического пути графа в Module2 введем следующий текст программы, оформленной в виде отдельной подпрограммы с именем `MaxPathPainter` без входных параметров (листинг 11.10).

Листинг 11.10. Программа изображения структуры критического пути в сетевом графе

```
Public Sub MaxPathPainter()
' -- Спецификация переменных и массивов
 Const N = 7 ' <-- это количество вершин графа
 Dim I, J, K, Im, Jm, YSdvig, Rec As Integer
 Dim Dis(N, N), Path(N, N), DisMax(N), VertMax(N), Xvert(N), _
     Yvert(N) As Integer
 Dim S, S1 As String
 Dim Vert(N), Sh As Shape
 YSdvig = 80 '<-- это сдвиг рисунка по вертикали
' -- Ввод и формирование матрицы смежности
 For I = 1 To N
    Xvert(I) = ActiveSheet.Cells(I + 2, N + 2)
    Yvert(I) = ActiveSheet.Cells(I + 2, N + 3)
        For J = 1 To N
            Dis(I, J) = ActiveSheet.Cells(I + 2, J + 1)
        Next J
 Next I
' -- Подготовка исходных массивов
 For I = 1 To N
    DisMax(I) = 0 '<-- вектор максимальных расстояний из начальной
' вершины
    VertMax(I) = 0 '<-- здесь номер вершины, куда ведет критический путь
       For J = 1 To N
          Path(I, J) = 0
       Next J
 Next I
' -- Нахождение критического пути
 For K = 2 To N
    For I = 1 To K - 1
       If DisMax(I) + Dis(I, K) > DisMax(K) And Dis(I, K) <> 0 Then
          DisMax(K) = DisMax(I) + Dis(I, K)
```

```
       End If
    Next I
 Next K
 VertMax(N) = N
 For J = 1 To N - 1
    K = N - J
    For I = K + 1 To N
If DisMax(K)+Dis(K, I)=DisMax(I) And Dis(K, I)<>0 And VertMax(I)<>0 Then
          VertMax(K) = K
       End If
    Next I
 Next J
' -- Формирование структуры критического пути
 Rec = 0
 For I = 1 To N - 1
    For J = I + 1 To N
       If VertMax(I) <> 0 And VertMax(J) <> 0 And Dis(I, J) <> 0 And _
       DisMax(J) - DisMax(I) = Dis(I, J) Then
          Path(VertMax(I), J) = 1
          Rec = Rec + Dis(I, J)
       End If
    Next J
 Next I
S = "Критический путь из вершины 1 в вершину "+CStr(N) + ":  Fmax= " + _
     CStr(Rec)
' -- Изображение вершин графа
 For I = 1 To N
  Set Vert(I) = ActiveSheet.Shapes.AddShape(msoShapeOval, Xvert(I), _
    Yvert(I) + YSdvig, 16, 16)
    Vert(I).TextFrame.Characters.Text = CStr(I)
    Vert(I).TextFrame.Characters.Font.Size = 10
    Vert(I).TextFrame.Characters.Font.Bold = True
 Next I
' -- Изображение ребер графа
 For I = 1 To N - 1
   For J = I + 1 To N
     If Path(I, J) <> 0 Then ' -- изображаем только те ребра, для которых
' связь не равна 0
  Set Sh = ActiveSheet.Shapes.AddConnector(msoConnectorStraight, 0,0,0,0)
```

```
        With Sh.ConnectorFormat
            .BeginConnect ConnectedShape:=Vert(I), ConnectionSite:=1
            .EndConnect ConnectedShape:=Vert(J), ConnectionSite:=1
        End With
        Sh.RerouteConnections
        Sh.Line.EndArrowheadLength = msoArrowheadLong '-- здесь нужна
' стрелка
        Sh.Line.EndArrowheadStyle = msoArrowheadTriangle '-- форма
' стрелки - треугольник
        Sh.Line.EndArrowheadWidth = msoArrowheadWide '-- стрелка будет
' широкая
' -- Изображение веса рядом с соответствующим ребром
        If Xvert(I) = Xvert(J) Then ' -- две вершины расположены на одной
' вертикали
    Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
   (Xvert(I)+Xvert(J))/2 - 5, (Yvert(I)+Yvert(J))/2 + YSdvig, 60, 150)
          Else ' -- две вершины не расположены на одной вертикали
    Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
   (Xvert(I)+Xvert(J))/2, (Yvert(I)+Yvert(J))/2 + YSdvig - 10, 60, 150)
        End If
        Sh.TextFrame.Characters.Text = CStr(Dis(I, J))
        Sh.TextFrame.Characters.Font.Size = 12
     End If
   Next J
 Next I
' -- Изображение подписи под рисунком
Set Sh = ActiveSheet.Shapes.AddShape(msoShapeRectangle, 150, 330 + _
YSdvig, 400, 20)
 Sh.TextFrame.Characters.Text = S
 Sh.TextFrame.Characters.Font.Size = 11
 Sh.TextFrame.Characters.Font.Bold = True
 Sh.TextFrame.HorizontalAlignment = xlHAlignCenter
 Sh.TextFrame.VerticalAlignment = xlVAlignCenter
End Sub
```

Данная программа является комбинацией 2-х ранее рассмотренных программ: функции нахождения критического пути в графе и модификации программы изображения структуры ориентированного графа.

Программа изображения структуры критического пути в сетевом графе использует в качестве исходных данных матрицу весов дуг графа и координаты

его вершин. Адреса ячеек с соответствующими данными строго фиксированы и должны корректироваться в программе для каждой конкретной задачи. Предварительно должно быть явно задано количество вершин исходного графа.

Далее следует реализация рассмотренного ранее (листинг 11.9) алгоритма нахождения критического пути в сетевом графе, структура которого записывается в матрицу `Path(N, N)`. Именно эта матрица используется в качестве исходной для изображения структуры критического пути последующими операторами, назначение которых уже было рассмотрено ранее (листинг 11.8). Изменить положение рисунка по вертикали можно с помощью задания значения переменной `Ysdvig`.

После написания текста программы она может быть запущена из данного рабочего листа MS Excel с помощью специального диалогового окна программы MS Excel, которое вызывается посредством выполнения операции главного меню: **Сервис** | **Макрос** | **Макросы** или нажатия комбинации клавиш: <Alt>+<F8> (рис. 11.19).

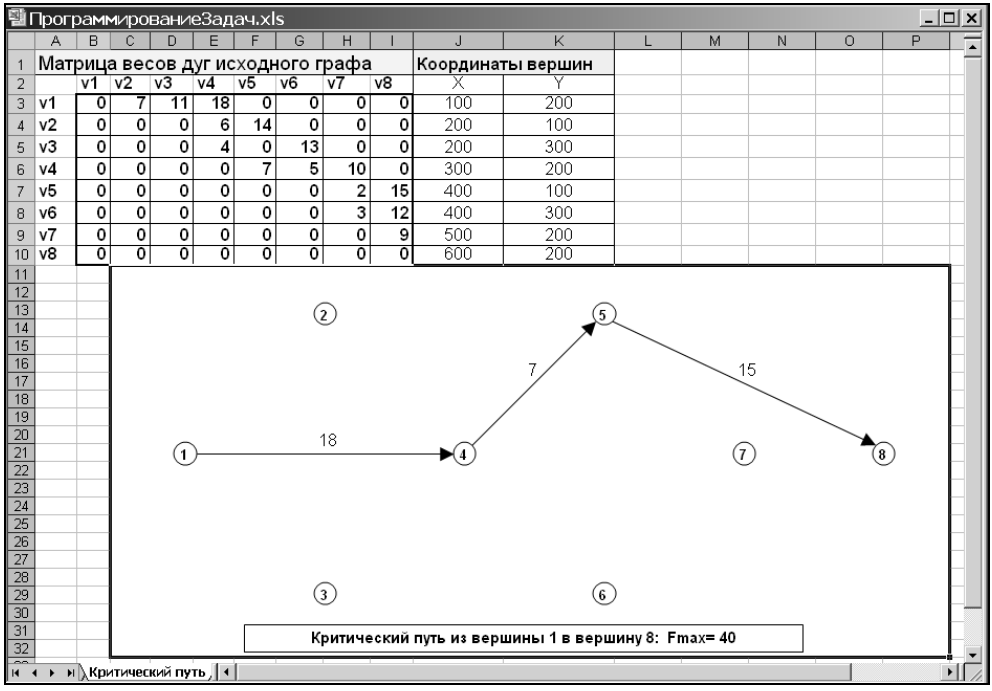


Рис. 11.19. Результат изображения критического пути в графе подпрограммой MaxPathPainter

Как можно заметить, изображение критического покрывающего дерева графа, полученное с помощью подпрограммы `MaxPathPainter`, полностью совпадает со структурой дерева, найденного ранее с помощью программы MS Excel (см. рис. 7.18).

Достоинством данной программы является то обстоятельство, что она позволяет находить и изображать графически несколько оптимальных значений критического пути в случае многоэкстремального характера соответствующей индивидуальной задачи оптимизации. Вариант нахождения 2-х критических путей с одинаковым оптимальным значением целевой функции для сетевого графа с 7 вершинами изображен на рис. 11.20.

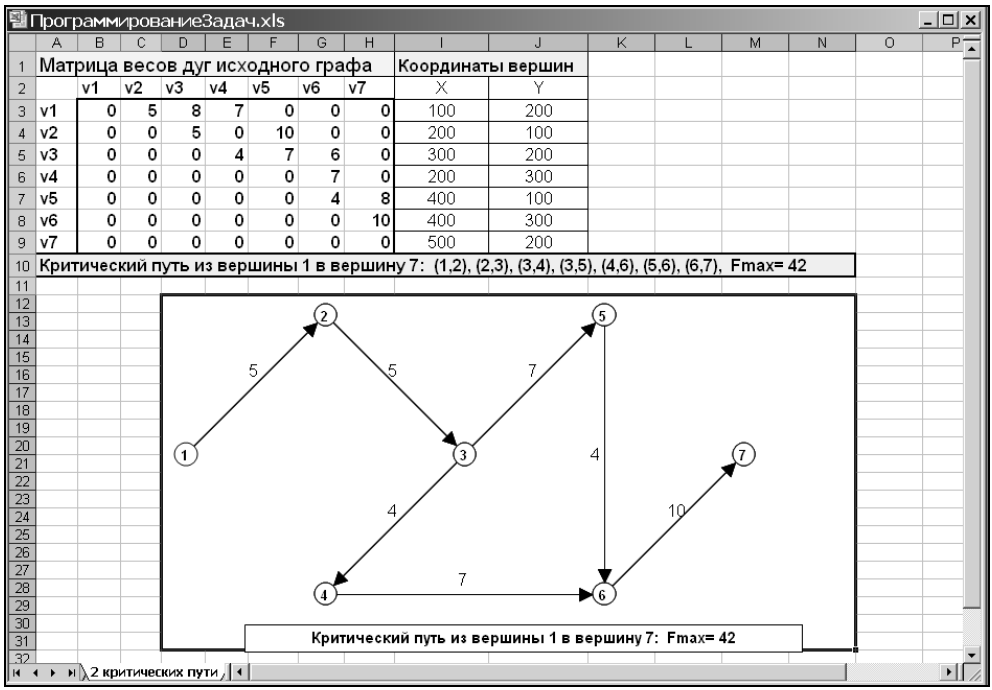


Рис. 11.20. Результат изображения двух критических путей одинаковой длины подпрограммой MaxPathPainter

Данная подпрограмма может быть использована для нахождения и изображения критического пути произвольного сетевого графа. В этом случае необходимо скорректировать ее текст, задав необходимое количество вершин графа N, матрицу весов дуг и координаты вершин.

11.6. Упражнения

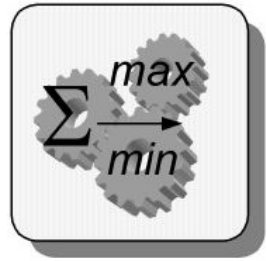
В качестве упражнений для самостоятельного решения с помощью программирования на VBA предлагаются несколько задач, расширяющих функциональность программы MS Excel.

11.6.1. Максимальный поток в сети

Для задачи о максимальном потоке в сети написать программу на VBA, которая реализует алгоритм пометок Форда — Фалкерсона для нахождения максимального потока, диаграмма деятельности которого представлена на рис. 7.25.

11.6.2. Графическое изображение максимального потока в сети

Для задачи о максимальном потоке в сети написать программу на VBA, которая позволяет изобразить максимальный поток для произвольного сетевого графа.



Глава 12

Алгоритмы и программы решения задач комбинаторной оптимизации

В настоящей главе описываются программы на VBA, предназначенные для нахождения решения некоторых задач комбинаторной оптимизации. В качестве таких задач рассматриваются задача коммивояжера и задача оптимального разбиения взвешенного графа. Использование данных программ избавляет пользователя от необходимости формирования ограничений в программе MS Excel, а результаты решения указанных задач с помощью разработанных программ могут быть представлены в графическом виде, что существенно повышает наглядность их анализа и содержательной интерпретации. В качестве дополнительной возможности программы MS Excel рассматриваются особенности использования внешних функций, разработанных с помощью интегрированных сред программирования и реализованных в форме библиотек динамической компоновки.

12.1. Задача коммивояжера и ее решение с помощью VBA

Содержательная постановка задачи коммивояжера приводится в *разд. 1.2.6*, математическая постановка задачи рассматривается в *разд. 8.2.1*. Решение данной задачи с помощью программы MS Excel приводится в *разд. 8.2.2*. Особенностью процесса ее решения является тот факт, что предварительно следует определить $n(n-1)$ основных переменных и $n-1$ вспомогательных переменных, а также сформировать и ввести в ячейки рабочего листа n^2-n+2 ограничений вида (8.2.2)—(8.2.4), где n — число вершин исходного графа. Данный процесс уже для значений $n > 10$ представляется весьма рутинной процедурой. Именно по этой причине возникает естественное желание разработать специальную программу, которая на основе заданной матрицы весов

дуг произвольного ориентированного графа находила бы полный замкнутый путь минимальной длины.

Далее в этом разделе рассматриваются алгоритм и программа приближенного решения задачи коммивояжера, которая может быть использована для практического нахождения решений задач данного класса с достаточно большим числом городов ($n > 40$). В последнем случае пользователь также избавляется от необходимости выполнения рутинных операций по записи системы ограничений в программе MS Excel.

12.1.1. Алгоритм приближенного решения задачи коммивояжера

В качестве алгоритма приближенного решения задачи коммивояжера может быть использована общая схема метода локальной оптимизации, конкретизированная применительно к решению данной типовой задачи комбинаторного программирования. При этом спецификация данного метода должна учитывать специфику и комбинаторный характер задачи коммивояжера.

Основная идея алгоритма приближенного решения задачи коммивояжера в теоретико-графовой постановке состоит во введении в рассмотрение так называемой локальной окрестности допустимого решения. С этой целью рассмотрим некоторый полный замкнутый путь $\{v_{i1}, v_{i2}, \ldots, v_{in}, v_{i1}\}$, которому соответствует допустимое решение $x_i \in \Delta_\beta$ данной задачи со значением целевой функции $f(x_i)$. Не уменьшая общности рассмотрения, можно утверждать, что этому решению соответствует также перестановка вида $(1; i_2, i_3, \ldots, i_{n-1}, i_n; 1)$ из элементов числового множества $\{1, 2, \ldots, n\}$, в которых первый и последний элементы всегда равны 1. При этом предполагается, что коммивояжер начинает и заканчивает свой путь в вершине с номером 1.

Тогда под *локальной окрестностью* $\delta_2(x_i)$ *ранга 2* допустимого решения $x_i \in \Delta_\beta$ будем понимать совокупность произвольных перестановок $(1; j_2, j_3, \ldots, j_{n-1}, j_n; 1)$, которые отличаются порядком 2-х чисел по сравнению с исходной перестановкой $(1; i_2, i_3, \ldots, i_{n-1}, i_n; 1)$, соответствующей рассматриваемому решению. Другими словами, в этих перестановках $i_k = j_k$ для всех значений k, кроме некоторых k_1 и k_2, для которых $i_{k1} = j_{k2}$ и $i_{k2} = j_{k1}$.

Например, для перестановки (1; 2, 3, 4, 5, 6; 1) к локальной окрестности ранга 2 будут принадлежать следующие перестановки: (1; 3, 2, 4, 5, 6; 1), (1; 4, 3, 2, 5, 6; 1), (1; 5, 3, 4, 2, 6; 1), (1; 6, 3, 4, 5, 2; 1), (1; 2, 4, 3, 5, 6; 1) и т. д. Очевидно, каждой из этих перестановок соответствует некоторое допустимое решение задачи коммивояжера $x_j \in \Delta_\beta$ данной задачи со значением целевой функции $f(x_j)$.

Для приближенного решения задачи коммивояжера с помощью метода локальной оптимизации необходимо предварительно задать некоторое допустимое начальное решение задачи $x_0 \in \Delta_\beta$ данной задачи, которому соответствует значение целевой функции $f(x_0)$. Далее следует рассмотреть все допустимые решения в локальной окрестности $x_j \in \delta_2(x_0)$ этого решения и выбрать из них такое, которому соответствует минимальное значение целевой функции $f(x_j)$. Это решение следует считать новым начальным решением, для которого рассматривается новая локальная окрестность решений. Подобный процесс завершается, когда в локальной окрестности некоторого решения будут отсутствовать решения со значением целевой функции меньшим, чем значение $f(x_0)$.

Метод локальной оптимизации, ориентированный на решение задачи коммивояжера в комбинаторной постановке, основывается на рекуррентном характере процедуры построения последовательности локальных окрестностей. Основанный на ней алгоритм приближенного решения задачи коммивояжера имеет итеративный характер и заключается в выполнении следующих действий:

1. *Предварительное задание начального решения* x_0. До начала выполнения основных итераций алгоритма задается некоторая допустимая перестановка $(1; i_2, i_3, \ldots, i_{n-1}, i_n; 1)$ из элементов числового множества $\{1, 2, \ldots, n\}$, которой соответствует допустимое решение $x_0 \in \Delta_\beta$ и значение целевой функции $f(x_0)$. После чего следует перейти к выполнению действий шага 2.
2. *Нахождение локально-оптимального решения*. Для решения x_0 рассматриваются все допустимые решения в локальной окрестности $\delta_2(x_0)$ и среди них выбирается решение с минимальным значением целевой функции: $x_j = \boldsymbol{argmin}\{f(x_i)\}$ для всех $x_i \in \delta_2(x_0)$. После чего следует перейти к выполнению действий шага 3.
3. *Нахождение нового начального решения*. Если выполняется условие: $f(x_0) \le f(x_j)$, то в локальной окрестности $\delta_2(x_0)$ отсутствуют допустимые решения со значением целевой функции меньшим, чем значение целевой функции начального решения. На этом выполнение алгоритма заканчивается, и в качестве приближенного решения задачи принимается решение x_0. В противном случае допустимое решение x_j следует принять за новое начальное решение x_0 и перейти к выполнению действий шага 3.

Общая схема рассмотренного алгоритма приближенного решения задачи коммивояжера может быть изображена графически в форме следующей диаграммы деятельности языка UML (рис. 12.1).

В силу конечности общего количества вершин исходного графа можно заключить, что рассмотренный алгоритм приближенного решения задачи коммивояжера также является конечным. Однако в практических случаях общее количество итераций для алгоритмов локальной оптимизации дополнительно

ограничивают некоторым натуральным числом. Далее следует описание программы, которая реализует данный алгоритм приближенного решения задачи коммивояжера.

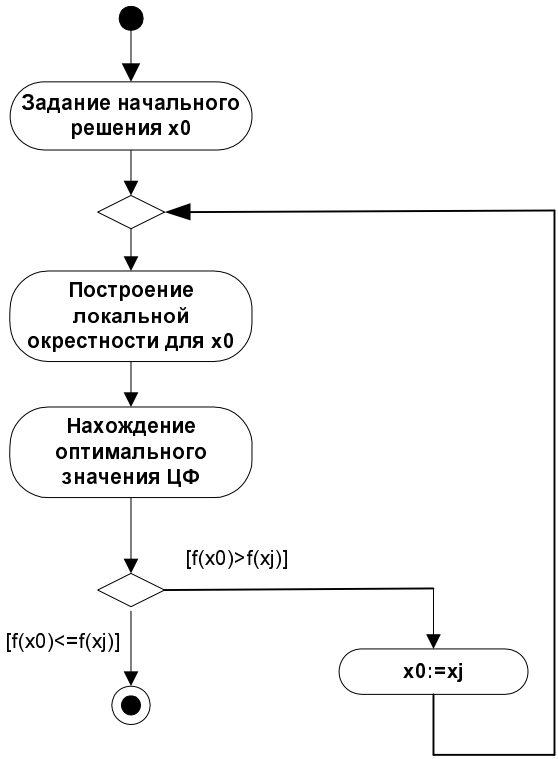


Рис. 12.1. Диаграмма деятельности алгоритма приближенного решения задачи коммивояжера

12.1.2. Программа приближенного решения задачи коммивояжера

Для приближенного решения задачи коммивояжера целесообразно реализовать рассмотренный ранее алгоритм локальной оптимизации, что позволит избежать задания исходных переменных и ограничений для любой индивидуальной задачи этого типа. При этом реализацию данного алгоритма в виде пользовательской функции следует выполнить таким образом, чтобы она позволяла находить минимальный полный замкнутый путь в ориентированном графе для произвольного количества вершин и дуг.

Для записи текста программы приближенного решения задачи коммивояжера воспользуемся модулем Module1, который был ранее создан и имеется в книге с именем Программирование задач. В данный модуль введем следующий текст программы, оформленной в виде отдельной функции с именем LocOptKom, которая использует единственный входной параметр для спецификации матрицы весов дуг исходного графа (листинг 12.1).

Листинг 12.1. Программа приближенного решения задачи коммивояжера

```
Function LocOptKom(C As Variant) As Variant
' Алгоритм локальной оптимизации для нахождения полного замкнутого пути
 Dim Rec, LocOptRec, FuncLocVar, I, J, K, L, Im As Integer
 Dim Dis(), VRec(), VLocOpt(), VLocVar() As Integer
 Dim S, S1 As String
 N = C.Columns.Count
 ReDim Dis(N, N), VRec(N), VLocOpt(N), VLocVar(N)
' -- Подготовка исходных массивов
 For I = 1 To N
    For J = 1 To N
       Dis(I, J) = C(I, J)
    Next J
 Next I
' -- Задание начального решения - рекорда
 For I = 1 To N - 1
    VRec(I) = I + 1
 Next I
 VRec(N) = 1
 Rec = SumPath(VRec, Dis, N)
 LocOptRec = Rec
 L = 0
 For I = 1 To N
 VLocOpt(I) = VRec(I)
 Next I
 While Rec <= LocOptRec And L = 0
    LocOptRec = Rec
    L = 1
    For I = 1 To N - 2 ' -- генерация новой перестановки - решения
       For J = I + 1 To N - 1
          For K = 1 To N
```

```
            VLocVar(K) = VRec(K)
         Next K
         Im = VLocVar(I)
         VLocVar(I) = VLocVar(J)
         VLocVar(J) = Im
         FuncLocVar = SumPath(VLocVar, Dis, N)
         If FuncLocVar < LocOptRec And FuncLocVar <> 0 Then
' Новое решение лучшее в локальной окрестности
            LocOptRec = FuncLocVar
            For K = 1 To N
               VLocOpt(K) = VLocVar(K)
            Next K
         End If
      Next J
   Next I
   If LocOptRec < Rec And LocOptRec <> 0 Then
' Лучшее в локальной окрестности решение становится новым рекордом
      Rec = LocOptRec
      L = 0
      For K = 1 To N
         VRec(K) = VLocOpt(K)
      Next K
   End If
 Wend
 S = " (" & CStr(1) & "," & CStr(VRec(1)) & ")"
 For I = 1 To N - 1
   S = S + " (" & CStr(VRec(I)) & "," & CStr(VRec(I + 1)) & "),"
 Next I
 S1 = "Минимальный полный замкнутый путь  "
 S = S1 + S + "  Fmin= " + CStr(Rec)
 LocOptKom = S '<-- возвращаемое функцией значение
 Erase Dis, VRec, VLocOpt, VLocVar '-- уничтожаем все динамические
' массивы
End Function
```

Данная программа оформлена в виде функции `LocOptKom` с единственным аргументом, в качестве которого выступает матрица весов дуг исходного графа. Основные массивы программы объявляются как динамические, что дает возможность предварительно не указывать их размерность и позволяет

не задавать в качестве аргумента количество вершин графа. Это унифицирует применение данной функции к графам с произвольным количеством вершин.

Массив `Dis(N, N)` служит для хранения значений матрицы весов дуг, массив `VRec(N)` — для хранения начальной и рекордной перестановки вершин, массив `VLocOpt(N)` — для хранения локально-оптимальной перестановки вершин, а массив `VLocVar(N)` — для хранения перестановки вершин, которая входит в локальную окрестность рекордной перестановки.

Данная реализация алгоритма приближенного решения задачи коммивояжера использует в качестве начального рекордного решения перестановку (1, 2,…, *n* ,1), где *n* — общее количество вершин исходного графа. Соответствующая перестановка генерируется в цикле после комментария: "Задание начального решения — рекорда". В последующем эта перестановка является исходной для формирования перестановок в ее локальной окрестности. Соответствующие перестановки формируются последовательным изменением порядка следования пары вершин. При этом для каждой новой перестановки рассчитывается значение целевой функции.

При этом переменная `Rec` используется для начального и рекордного значения ЦФ, переменная `LocOptRec` — для локально-оптимального значения ЦФ, переменная `FuncLocVar` — для значения ЦФ произвольной перестановки в локальной окрестности начальной.

Основной цикл с предусловием завершится в том случае, когда в локальной окрестности очередного начального и рекордного решения будут отсутствовать решения со значением ЦФ меньшим, чем значение ЦФ начального и рекордного значения. Тем самым будут закончены основные итерации алгоритма приближенного решения.

После завершения этого цикла структура локально-оптимального полного замкнутого пути будет содержаться в массиве `VRec(N)`. Для получения окончательного решения служит последний двойной цикл. В качестве результата данная функция возвращает в выбранную ячейку строку текста `S`, содержащего список дуг, образующих полный замкнутый путь минимальной длины, начинающийся и заканчивающийся в вершине с номером 1, и оптимальное значение соответствующей целевой функции.

Для удобства расчетов функция расчета длины полного замкнутого пути оформлена в виде отдельной программы. При этом данная функция использует 3 входных параметра для спецификации матрицы весов дуг исходного графа (листинг 12.2).

Листинг 12.2. Программа расчета длины полного замкнутого пути

```
Function SumPath(V, D, N As Variant) As Integer
' Функция расчета длины полного замкнутого пути в графе
 Dim I, L1, Fun As Integer
 Fun = 0
 L1 = 1
 If D(1, V(1)) <> 0 Then
       Fun = D(1, V(1))
    Else
       L1 = 0
 End If
 For I = 1 To N - 1
    If D(V(I), V(I + 1)) <> 0 Then
' для пары вершин расстояние не равно 0
       Fun = Fun + D(V(I), V(I + 1))
    Else
' для пары вершин расстояние равно 0
       L1 = 0
    End If
    If L1 = 1 Then
' если для пути нет пар вершин с расстоянием 0
       SumPath = Fun
    Else
' если для пути есть пара вершин с расстоянием 0
       SumPath = 0
    End If
 Next I
End Function
```

Данная функция использует 3 входных параметра: `V(N)` — значение перестановки, для которой рассчитывается длина полного замкнутого пути, `D(N, N)` — матрица весов дуг исходного графа, `N` — общее количество вершин исходного графа. Поскольку основные операторы данной функции прокомментированы, они не требуют дополнительных пояснений. Заметим, что эта функция также может быть вызвана из рабочего листа программы MS Excel.

После написания текста программы соответствующая функция может быть вызвана из любого рабочего листа этой книги. С этой целью в книге Программирование задач создадим новый рабочий лист с именем Задача комми-

вояжера, в который введем в качестве исходных данных матрицу весов дуг рассмотренного ранее ориентированного графа (рис. 8.1). Результат подготовки исходных данных для решения задачи коммивояжера будет иметь следующий вид (рис. 12.2).

Далее из любой свободной ячейки следует вызвать мастер добавления функции и в списке функций, определенных пользователем, выбрать созданную функцию: `LocOptKom()`. После выбора данной функции в качестве значений ее аргументов следует выделить диапазон ячеек **B3:G8**, в которых содержатся значения матрицы весов дуг исходного графа (рис. 12.2). После задания аргументов и нажатия кнопки **ОК**, если текст программы не содержит ошибок, то она будет выполнена, а в ячейке с этой функцией на данном рабочем листе появится результат приближенного решения задачи коммивояжера (рис. 12.3).

ПрограммированиеЗадач.xls

	A	B	C	D	E	F	G
1	Матрица весов дуг исходного графа						
2		v1	v2	v3	v4	v5	v6
3	v1	0	4	10	13	4	8
4	v2	2	0	9	7	6	7
5	v3	8	5	0	5	5	9
6	v4	5	8	5	0	7	10
7	v5	6	4	4	9	0	4
8	v6	5	1	4	8	3	0

Задача коммивояжера

Рис. 12.2. Исходные данные для решения задачи коммивояжера

ПрограммированиеЗадач.xls

	A	B	C	D	E	F	G
1	Матрица весов дуг исходного графа						
2		v1	v2	v3	v4	v5	v6
3	v1	0	4	10	13	4	8
4	v2	2	0	9	7	6	7
5	v3	8	5	0	5	5	9
6	v4	5	8	5	0	7	10
7	v5	6	4	4	9	0	4
8	v6	5	1	4	8	3	0
9							
10	Локально-оптимальный полный замкнутый путь: (1,5) (5,3), (3,4), (4,6), (6,2), (2,1), Fmin= 26						

Задача коммивояжера

Рис. 12.3. Результат приближенного решения задачи коммивояжера

Как показывает анализ найденного решения, оно полностью совпадает с точным решением данной задачи, полученным с помощью программы MS Excel в *разд. 8.2.2*. Данный факт может свидетельствовать в пользу не только правильности разработанной программы, но и возможности нахождения с ее помощью точных решений соответствующей задачи комбинаторной оптимизации.

Дополнительно для сравнения результата с начальным решением можно вычислить значение длины полного замкнутого пути для начальной перестановки с помощью ранее описанной функции. С этой целью следует из любой ячейки рабочего листа вызвать функцию пользователя `SumPath` и задать соответствующие входные параметры. При этом в качестве вектора начальной перестановки следует ввести в любые ячейки следующие значения: V(6) = (2, 3, 4, 5, 6, 1). Функция `SumPath` в качестве результата возвратит значение 34, что действительно равно длине соответствующего полного замкнутого пути в исходном графе (см. рис. 8.1).

Далее можно отобразить структуру найденного решения, для чего целесообразно использовать ранее написанную программу для отображения структуры ориентированного графа.

12.1.3. Программа изображения полного замкнутого пути в ориентированном графе

Для изображения структуры минимального пути в ориентированном графе следует дополнить матрицу весов дуг исходного графа значениями координат вершин. Поскольку структура рассматриваемого графа не изображалась ранее, соответствующие значения координат следует ввести в ячейки **J3:K10** рабочего листа Минимальный путь.

Для программы изображения структуры полного замкнутого пути в ориентированном графе в Module2 введем следующий текст, оформленный в виде отдельной подпрограммы с именем `KomPathPainter` без входных параметров (листинг 12.3).

Листинг 12.3. Программа изображения структуры полного замкнутого пути в ориентированном графе

```
Public Sub KomPathPainter()
' -- Спецификация переменных и массивов
 Const N = 6 ' <-- это количество вершин графа
 Dim Rec, LocOptRec, FuncLocVar, I, J, K, L, Im, YSdvig As Integer
 Dim S, S1 As String
 Dim Dis(N, N), VRec(N), VLocOpt(N), VLocVar(N), Path(N, N), Xvert(N), _
```

```
 Yvert(N) As Integer
 Dim Vert(N), Sh As Shape
 YSdvig = 80 '<-- это сдвиг рисунка по вертикали
' -- Ввод и формирование координат вершин и матрицы весов дуг
 For I = 1 To N
    Xvert(I) = ActiveSheet.Cells(I + 2, N + 2)
    Yvert(I) = ActiveSheet.Cells(I + 2, N + 3)
        For J = 1 To N
            Dis(I, J) = ActiveSheet.Cells(I + 2, J + 1)
        Next J
 Next I
' -- Подготовка исходных массивов
 For I = 1 To N
    For J = 1 To N
       Path(I, J) = 0
    Next J
 Next I
' -- Задание начального решения - рекорда
 For I = 1 To N - 1
    VRec(I) = I + 1
 Next I
 VRec(N) = 1
 Rec = SumPath(VRec, Dis, N)
 LocOptRec = Rec
 L = 0
 For I = 1 To N
 VLocOpt(I) = VRec(I)
 Next I
 While Rec <= LocOptRec And L = 0
    LocOptRec = Rec
    L = 1
    For I = 1 To N - 2 ' -- генерация новой перестановки - решения
       For J = I + 1 To N - 1
          For K = 1 To N
             VLocVar(K) = VRec(K)
          Next K
          Im = VLocVar(I)
          VLocVar(I) = VLocVar(J)
          VLocVar(J) = Im
```

```
          FuncLocVar = SumPath(VLocVar, Dis, N)
          If FuncLocVar < LocOptRec And FuncLocVar <> 0 Then
' Новое решение лучшее в локальной окрестности
             LocOptRec = FuncLocVar
             For K = 1 To N
                VLocOpt(K) = VLocVar(K)
             Next K
          End If
       Next J
    Next I
    If LocOptRec < Rec And LocOptRec <> 0 Then
' Лучшее в локальной окрестности решение становится новым рекордом
       Rec = LocOptRec
       L = 0
       For K = 1 To N
          VRec(K) = VLocOpt(K)
       Next K
    End If
 Wend
' -- Формирование структуры полного минимального пути
 Path(1, VRec(1)) = 1
 For I = 1 To N - 1
    Path(VRec(I), VRec(I + 1)) = 1
 Next I
 S1 = "Локально-оптимальный полный замкнутый путь: "
 S = S1 + S + "  Fmin= " + CStr(Rec)
' -- Изображение вершин графа
 For I = 1 To N
    Set Vert(I) = ActiveSheet.Shapes.AddShape(msoShapeOval, Xvert(I), _
    Yvert(I) + YSdvig, 16, 16)
    Vert(I).TextFrame.Characters.Text = CStr(I)
    Vert(I).TextFrame.Characters.Font.Size = 10
    Vert(I).TextFrame.Characters.Font.Bold = True
 Next I
' -- Изображение ребер графа
 For I = 1 To N
   For J = 1 To N
     If Path(I, J) <> 0 Then ' -- изображаем только те ребра, для которых
' связь не равна 0
```

```
  Set Sh = ActiveSheet.Shapes.AddConnector(msoConnectorStraight, 0,0,0,0)
        With Sh.ConnectorFormat
             .BeginConnect ConnectedShape:=Vert(I), ConnectionSite:=1
             .EndConnect ConnectedShape:=Vert(J), ConnectionSite:=1
        End With
        Sh.RerouteConnections
        Sh.Line.EndArrowheadLength = msoArrowheadLong '-- здесь нужна
' стрелка
        Sh.Line.EndArrowheadStyle = msoArrowheadTriangle '-- форма
' стрелки - треугольник
        Sh.Line.EndArrowheadWidth = msoArrowheadWide '-- стрелка будет
' широкая
' -- Изображение веса рядом с соответствующим ребром
        If Xvert(I) = Xvert(J) Then ' -- две вершины расположены на одной
' вертикали
     Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
    (Xvert(I)+Xvert(J))/2 - 5, (Yvert(I)+Yvert(J))/2 + YSdvig, 60, 150)
         Else ' -- две вершины не расположены на одной вертикали
     Set Sh = ActiveSheet.Shapes.AddLabel(msoTextOrientationHorizontal, _
    (Xvert(I)+Xvert(J))/2, (Yvert(I)+Yvert(J))/2 + YSdvig - 10, 60, 150)
        End If
        Sh.TextFrame.Characters.Text = CStr(Dis(I, J))
        Sh.TextFrame.Characters.Font.Size = 12
     End If
   Next J
 Next I
' -- Изображение подписи под рисунком
 Set Sh = ActiveSheet.Shapes.AddShape(msoShapeRectangle, 150, 330 + _
          YSdvig, 400, 20)
 Sh.TextFrame.Characters.Text = S
 Sh.TextFrame.Characters.Font.Size = 11
 Sh.TextFrame.Characters.Font.Bold = True
 Sh.TextFrame.HorizontalAlignment = xlHAlignCenter
 Sh.TextFrame.VerticalAlignment = xlVAlignCenter
End Sub
```

Данная программа является комбинацией 2-х ранее рассмотренных программ: функции нахождения приближенного решения задачи коммивояжера и программы изображения структуры ориентированного графа, которая была детально рассмотрена в *главе 11*.

Программа изображения структуры полного замкнутого пути в ориентированном графе использует в качестве исходных данных матрицу весов дуг графа и координаты его вершин. При этом координаты вершин задаются внутри подпрограммы с помощью операторов, аналогично ранее рассмотренным подпрограммам отображения структуры графов. При этом адреса ячеек с соответствующими данными также строго фиксированы и должны корректироваться в программе для каждой конкретной задачи. Предварительно должно быть явно задано количество вершин исходного графа.

Далее следует реализация рассмотренного ранее (листинг 12.1) алгоритма приближенного решения задачи коммивояжера, окончательная структура которого задается двумерным массивом `Path(N, N)`. Именно этот массив используется для изображения структуры полного замкнутого пути в ориентированном графе последующими операторами, назначение некоторых из них уже было рассмотрено ранее (листинг 11.10). При этом переменная `Ysdvig` служит для изменения положения рисунка по вертикали.

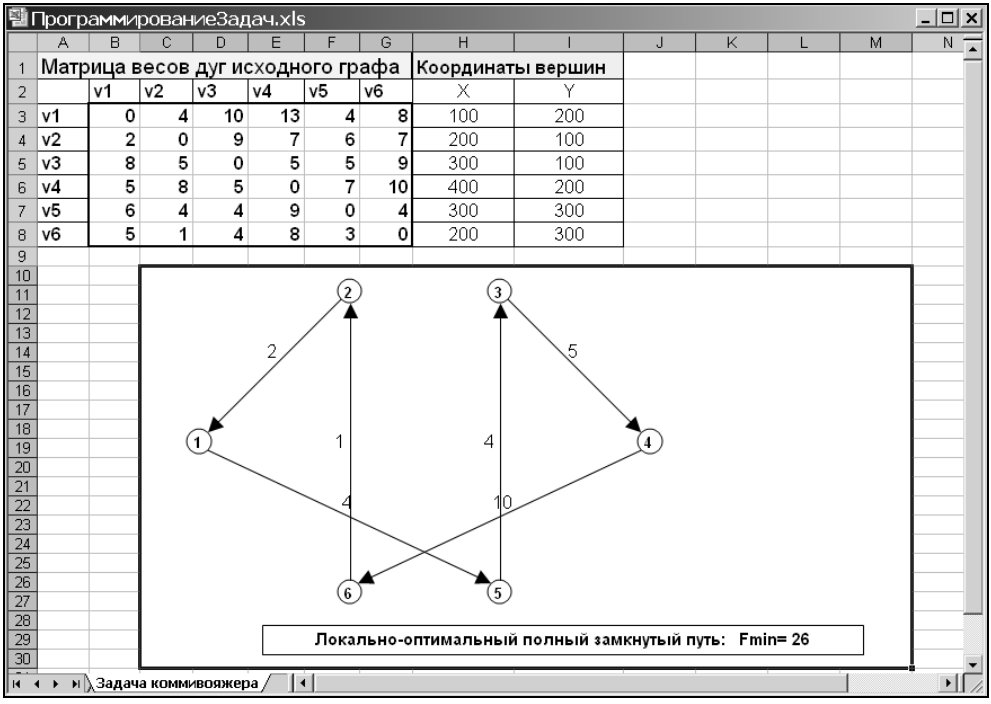


Рис. 12.4. Результат изображения полного замкнутого минимального пути подпрограммой KomPathPainter

После написания текста программы она может быть запущена из данного рабочего листа MS Excel с помощью специального диалогового окна программы MS Excel, которое вызывается посредством выполнения операции главного меню: **Сервис** | **Макрос** | **Макросы** или нажатия комбинации клавиш: <Alt>+<F8> (рис. 12.4).

Как можно заметить, изображение структуры локально-оптимального полного замкнутого пути в ориентированном графе, полученное с помощью подпрограммы `KomPathPainter`, полностью совпадает с точным решением данной задачи, полученным с помощью программы MS Excel в *разд. 8.2.2* (см. рис. 8.5). Данная подпрограмма может быть использована для нахождения и изображения минимального полного замкнутого пути произвольного графа. В этом случае необходимо скорректировать ее текст, задав необходимое количество вершин графа N, матрицу весов дуг и координаты вершин.

12.2. Задача о разбиении и ее решение с помощью VBA

Содержательная постановка задачи о разбиении приводится в *разд. 8.3.1*, математическая постановка задачи приводится в *разд. 8.3.2*. Решение данной задачи с помощью программы MS Excel приводится в *разд. 8.3.3*. Особенностью процесса ее решения является тот факт, что предварительно следует определить n^2 основных переменных, а также сформировать и ввести в ячейки рабочего листа $2m$ ограничений вида (8.3.2) и (8.3.3) и m произведений переменных для вычисления целевой функции, где n — число вершин исходного графа, m — число ребер или дуг исходного графа. Данный процесс уже для значений $n, m > 20$ представляется весьма рутинной процедурой. Поэтому возникает естественное желание разработать специальную программу, которая на основе заданной матрицы весов дуг произвольного ориентированного графа находила бы разбиение вершин с максимальным значением целевой функции (8.3.1).

Далее в этом разделе рассматриваются алгоритм и программа приближенного решения задачи о разбиении, которая может быть использована для практического нахождения решений задач данного класса с достаточно большим числом вершин ($n > 40$). В последнем случае пользователь также избавляется от необходимости выполнения рутинных операций по записи системы ограничений в программе MS Excel.

12.2.1. Алгоритм приближенного решения задачи о разбиении

В качестве алгоритма приближенного решения задачи о разбиении может быть использована общая схема метода локальной оптимизации, конкретизи-

рованная применительно к решению данной типовой задачи комбинаторного программирования. При этом спецификация данного метода должна учитывать специфику и комбинаторный характер задачи о разбиении.

Основная идея алгоритма приближенного решения задачи о разбиении в теоретико-графовой постановке состоит во введении в рассмотрение так называемой локальной окрестности допустимого решения. С этой целью рассмотрим некоторое допустимое разбиение множества вершин исходного графа: $R_i = \{A_1, A_2, \ldots, A_r\}$, где каждый класс разбиения A_k ($\forall k \in \{1, 2, \ldots, r\}$) состоит из вершин исходного графа $G = (V, E, d, h)$ или, что эквивалентно, из элементов числового множества $A = \{1, 2, \ldots, n\}$. Этому разбиению соответствует допустимое решение $x_i \in \Delta_\beta$ данной задачи со значением целевой функции $f(x_i)$.

В дальнейшем произвольное разбиение множества чисел $A = \{1, 2, \ldots, n\}$ удобно представить в виде вектора: $x = (i_1, i_2, i_3, \ldots, i_{n-1}, i_n)$, каждый элемент которого равен минимальному числу, с которым данный элемент попадает в один класс разбиения. Так, например, разбиение: $R = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\}$ может быть представлено в виде вектора: $x = (1, 1, 3, 3, 5, 5, 7, 7)$. С другой стороны, тривиальное разбиение: $R_0 = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}\}$ может быть представлено в виде вектора: $x_0 = (1, 2, 3, 4, 5, 6, 7, 8)$.

Тогда под *локальной окрестностью* $\delta_2(x_i)$ *ранга 2* допустимого решения $x_i \in \Delta_\beta$ будем понимать совокупность произвольных разбиений R_j, которые отличаются вхождением 2-х вершин или чисел в различные классы по сравнению с исходным разбиением: R_i. Другими словами, в векторах, соответствующих этим разбиениям, элементы $i_k = j_k$ для всех значений k, кроме некоторого k_l, для которого $i_{kl} \neq j_{kl}$.

Например, для разбиения, вектор которого равен (1, 2, 3, 4, 5, 6, 7, 8), к локальной окрестности ранга 2 будут принадлежать следующие разбиения, которым соответствуют вектора: (1, 1, 3, 4, 5, 6, 7, 8), (1, 2, 2, 4, 5, 6, 7, 8), (1, 2, 3, 3, 5, 6, 7, 8), (1, 2, 3, 4, 4, 6, 7, 8), (1, 2, 3, 4, 5, 5, 7, 8) и т. д. Очевидно, каждому из этих разбиений соответствует некоторое значение целевой функции $f(x_j)$. Однако не каждое из них может быть допустимым решением задачи о разбиении, поскольку существует ограничение на общий вес вершин, попадающих в один класс разбиения.

Для приближенного решения задачи о разбиении с помощью метода локальной оптимизации необходимо предварительно задать некоторое допустимое начальное разбиение $x_0 \in \Delta_\beta$ данной задачи, которому соответствует значение целевой функции $f(x_0)$. Далее следует рассмотреть все допустимые разбиения в локальной окрестности $x_j \in \delta_2(x_0)$ этого решения и выбрать из них такое, которому соответствует максимальное значение целевой функции $f(x_j)$. Это разбиение следует считать новым начальным разбиением, для которого рас-

сматривается новая локальная окрестность решений. Подобный процесс завершается, когда в локальной окрестности некоторого разбиения будут отсутствовать допустимые разбиения со значением целевой функции большим, чем значение $f(x_0)$.

Метод локальной оптимизации, ориентированный на решение задачи о разбиении в комбинаторной постановке, основывается на рекуррентном характере процедуры построения последовательности локальных окрестностей для допустимых разбиений. Основанный на ней алгоритм приближенного решения задачи о разбиении имеет итеративный характер и заключается в выполнении следующих действий:

1. *Предварительное задание начального разбиения* x_0. До начала выполнения основных итераций алгоритма задается некоторое допустимое разбиение, которому соответствует вектор $(i_1, i_2, i_3, \ldots, i_{n-1}, i_n)$ из элементов числового множества $\{1, 2, \ldots, n\}$. Этому разбиению соответствует допустимое решение $x_0 \in \Delta_\beta$ и значение целевой функции $f(x_0)$. После чего следует перейти к выполнению действий шага 2.
2. *Нахождение локально-оптимального допустимого разбиения*. Для разбиения x_0 рассматриваются все допустимые разбиения в локальной окрестности $\delta_2(x_0)$, и среди них выбирается разбиение с максимальным значением целевой функции: $x_j = \boldsymbol{argmax}\{f(x_i)\}$ для всех $x_i \in \delta_2(x_0)$. После чего следует перейти к выполнению действий шага 3.
3. *Нахождение нового начального разбиения*. Если выполняется условие: $f(x_0) \geq f(x_j)$, то в локальной окрестности $\delta_2(x_0)$ отсутствуют допустимые разбиения со значением целевой функции большим, чем значение целевой функции начального разбиения. На этом выполнение алгоритма заканчивается, и в качестве приближенного решения задачи принимается разбиение, которому соответствует вектор x_0. В противном случае допустимое разбиение x_j следует принять за новое начальное (рекордное) разбиение x_0 и перейти к выполнению действий шага 3.

Общая схема рассмотренного алгоритма приближенного решения задачи о разбиении может быть изображена графически в форме следующей диаграммы деятельности языка UML (рис. 12.5).

В силу конечности общего количества вершин исходного графа можно заключить, что рассмотренный алгоритм приближенного решения задачи о разбиении также является конечным. Однако в практических случаях общее количество итераций для алгоритмов локальной оптимизации иногда ограничивают некоторым натуральным числом. Далее следует описание программы, которая реализует данный алгоритм приближенного решения задачи о разбиении.

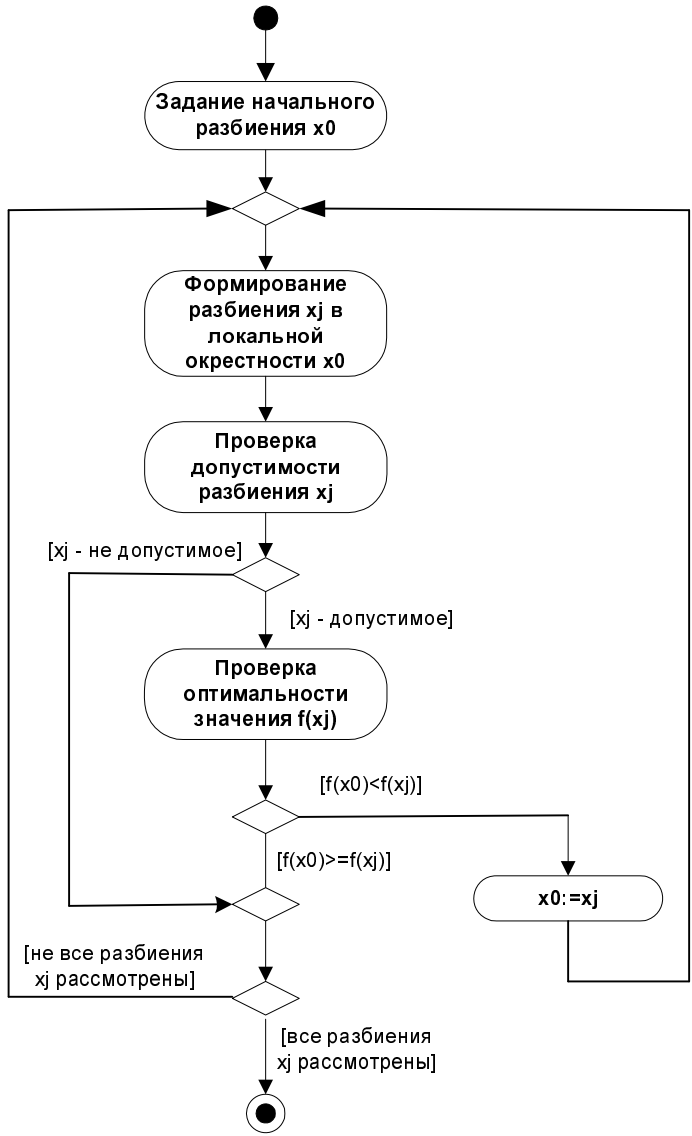


Рис. 12.5. Диаграмма деятельности алгоритма приближенного решения задачи о разбиении

12.2.2. Программа приближенного решения задачи о разбиении

Для приближенного решения задачи о разбиении целесообразно реализовать рассмотренный ранее алгоритм локальной оптимизации, что позволит избежать задания исходных переменных и ограничений для любой индивидуальной задачи этого типа. При этом реализацию данного алгоритма в виде пользовательской функции следует выполнить таким образом, чтобы она позволяла находить допустимое разбиение вершин графа для произвольного количества вершин и дуг.

Для записи текста программы приближенного решения задачи о разбиении воспользуемся модулем Module1, который был ранее создан и имеется в книге с именем Программирование задач. В данный модуль введем следующий текст программы, оформленной в виде отдельной функции с именем `LocOptRaz`, которая использует единственный входной параметр для спецификации матрицы весов дуг исходного графа (листинг 12.4).

Листинг 12.4. Программа приближенного решения задачи о разбиении

```
Public Function LocOptRaz(C, Wt, Wmax As Variant) As String
 Dim Rec, RecLocOpt, FuncLocVar, I, J, K, L, M As Integer
 Dim Sim(), RazRec(), RazLocOpt(), RazLocVar() As Integer
 Dim S, S1 As String
 N = C.Columns.Count
 ReDim Sim(N, N), RazRec(N), RazLocOpt(N), RazLocVar(N)
' -- Подготовка исходных массивов
 For I = 1 To N
    For J = 1 To N
       Sim(I, J) = C(I, J)
    Next J
 Next I
' -- Задание начального разбиения - рекорда
 For I = 1 To N
    RazRec(I) = I
 Next I
 Rec = FuncRaz(RazRec, Sim, N)
 M = 0
 For I = 1 To N
    RazLocOpt(I) = RazRec(I)
```

```
Next I
RecLocOpt = Rec
While Rec >= RecLocOpt And M = 0
   RecLocOpt = Rec
   M = 1
   ' Генерация нового разбиения в локальной окрестности
   For K = 1 To N - 1
      For J = K + 1 To N
         For L = 1 To N
            RazLocVar(L) = RazLocOpt(L)
         Next L
         RazLocVar(J) = K
         For L = 1 To N
            If RazLocVar(L) <> L And _
            RazLocVar(RazLocVar(L)) <> RazLocVar(L) Then
               RazLocVar(L) = L
            End If
         Next L
         FuncLocVar = FuncRaz(RazLocVar, Sim, N)
         If ProvRaz(RazLocVar, Wt, Wmax, N) = True And _
         FuncLocVar > RecLocOpt Then
         ' Новое разбиение лучшее в локальной окрестности
            RecLocOpt = FuncLocVar
            For I = 1 To N
               RazLocOpt(I) = RazLocVar(I)
            Next I
         End If
      Next J
   Next K
   If RecLocOpt > Rec Then
   ' Лучшее в локальной окрестности разбиение становится новым рекордом
      M = 0
      Rec = RecLocOpt
      For I = 1 To N
         RazRec(I) = RazLocOpt(I)
         Next I
   End If
Wend
' Подготовка строки с результатом решения задачи
```

```
 S1 = "Локально-оптимальное разбиение:"
 S = ""
 For I = 1 To N
    M = 0 ' <-- это значение нужно для печати фигурной скобки
    For J = I To N
       If RazRec(J) = I And M = 1 Then
       ' в классе разбиения уже есть элементы
          S = S + CStr(J) + ", "
       End If
       If RazRec(J) = I And M = 0 Then
       ' в классе разбиения еще нет элементов
          S = S + " {" + CStr(J) + ", "
          M = 1
       End If
       If J = N And M = 1 Then
       ' элементы в классе разбиения закончились
          S = Mid(S, 1, Len(S) - 2) + "},"
       End If
    Next J
 Next I
 S = S1 + S + "  Fmin= " + CStr(Rec)
 LocOptRaz = S '<-- возвращаемое функцией значение
 Erase Sim, RazRec, RazLocOpt, RazLocVar '-- уничтожаем все динамические
' массивы
End Function
```

Данная программа оформлена в виде функции `LocOptRaz` с тремя аргументами. Первый из них — матрица весов дуг исходного графа. Второй аргумент — вектор весов вершин исходного графа. Третий аргумент — максимально допустимое значение весов вершин в одном классе разбиения. Основные массивы программы объявляются как динамические, что дает возможность предварительно не указывать их размерность и позволяет не задавать в качестве аргумента количество вершин графа. Это унифицирует применение данной функции к графам с произвольным количеством вершин.

Массив `Sim(N, N)` служит для хранения значений матрицы весов дуг, массив `RazRec(N)` — для хранения номеров вершин начального и рекордного разбиения, массив `RazLocOpt(N)` — для хранения номеров вершин локально-оптимального разбиения, а массив `RazLocVar(N)` — для хранения номеров вершин разбиения, которое входит в локальную окрестность рекордного разбиения.

Данная реализация алгоритма приближенного решения задачи о разбиении использует в качестве начального разбиения вектор $x_0 = (1, 2, \ldots, n)$, где n — общее количество вершин исходного графа. Соответствующая перестановка генерируется в цикле после комментария: "Задание начального разбиения — рекорда". В последующем это разбиение является исходным для формирования всех возможных разбиений в ее локальной окрестности. Соответствующие разбиения формируются последовательным изменением значения одного из элементов исходного вектора $\boldsymbol{x_0}$. При этом для каждого нового разбиения рассчитывается значение целевой функции и проверяется его допустимость ограничению на общий вес вершин в одном классе разбиения.

При этом переменная `Rec` используется для начального и рекордного значения ЦФ, переменная `RecLocOpt` — для локально-оптимального значения ЦФ, переменная `FuncLocVar` — для значения ЦФ произвольного разбиения в локальной окрестности начального разбиения.

Основной цикл с предусловием завершится в том случае, когда в локальной окрестности очередного начального и рекордного разбиения будут отсутствовать допустимые разбиения со значением ЦФ большим, чем значение ЦФ начального и рекордного значения. Тем самым будут закончены основные итерации алгоритма приближенного решения рассматриваемой задачи.

После завершения этого цикла структура локально-оптимального допустимого разбиения будет содержаться в массиве `RazRec(N)`. Для получения удобного для интерпретации окончательного решения служат последующие циклы, в которых выполняются последовательные приращения с выходной строкой текста. В качестве результата данная функция возвращает в выбранную ячейку строку текста `S`, содержащего списки вершин, образующих отдельные классы локально-оптимального разбиения, и значение соответствующей целевой функции.

Для удобства расчетов функция расчета целевой функции, которая представляет собой сумму весов дуг, соединяющих вершины во всех классах разбиения, и функция проверки допустимости разбиения оформлены в виде отдельных программ. Первая из них — функция `FuncRaz` использует 3 входных параметра (листинг 12.5).

Листинг 12.5. Программа расчета целевой функции — суммы весов дуг для вершин в одном классе разбиения

```
Function FuncRaz(Raz, Sim, N As Variant) As Integer
 Dim I, J, Im, V As Integer
 V = 0
 For I = 1 To N - 1
```

```
    Im = Raz(I)
    For J = I + 1 To N
       If Raz(J) = Im Then
          V = V + Sim(I, J)
       End If
    Next J
 Next I
 FuncRaz = V
End Function
```

Данная функция использует следующие входные параметры: `Raz` — вектор разбиения, для которого рассчитывается значение целевой функции, `Sim(N, N)` — матрица весов дуг исходного графа, `N` — общее количество вершин исходного графа. В качестве результата функция `FuncRaz` возвращает значение целевой функции разбиения, которому соответствует вектор `Raz`. При этом для удобства реализации допустимость разбиения `Raz` не проверяется. Заметим, что эта функция также была вызвана из рабочего листа программы MS Excel.

Вторая из них — функция `ProvRaz` проверяет допустимость разбиения `Raz` и использует 4 входных параметра (листинг 12.6).

Листинг 12.6. Программа проверки допустимости разбиения

```
Function ProvRaz(Raz, Wt, Wmax, N As Variant) As Boolean
 Dim Bol As Boolean
 Dim I, K, Wvar As Integer
 Bol = True
 For K = 1 To N
    Wvar = 0
    For I = 1 To N
       If Raz(I) = K Then
          Wvar = Wvar + Wt(I)
       End If
    Next I
    If Wvar > Wmax Then
       Bol = False
    End If
 Next K
 ProvRaz = Bol
End Function
```

Данная функция использует следующие входные параметры: `Raz` — вектор разбиения, которое проверяется на допустимость ограничению по общему весу вершин, `Wt(N)` — вектор весов вершин исходного графа, `Wmax` — значение максимального веса вершин в одном классе разбиения, `N` — общее количество вершин исходного графа. В качестве результата функция `ProvRaz` возвращает значение ИСТИНА, если разбиение, которому соответствует вектор `Raz`, является допустимым; и значение ЛОЖЬ — в противном случае. Заметим, что эта функция также может быть вызвана из рабочего листа программы MS Excel.

После написания текстов этих программ функция нахождения приближенного решения задачи о разбиении `LocOptRaz` может быть вызвана из любого рабочего листа этой книги. С этой целью в книге Программирование задач создадим новый рабочий лист с именем Задача о разбиении, в который введем в качестве исходных данных матрицу весов дуг рассмотренного ранее ориентированного графа (см. рис. 8.7). Результат подготовки исходных данных для решения задачи о разбиении будет иметь следующий вид (рис. 12.6).

ПрограммированиеЗадач.xls

	A	B	C	D	E	F	G	H	I	J	K	L
1	Матрица весов дуг исходного графа									Веса		
2		v1	v2	v3	v4	v5	v6	v7	v8	вершин:		
3	v1	0	9	0	7	0	1	0	0	4		
4	v2	0	0	4	0	0	0	0	0	2		
5	v3	0	0	0	5	0	0	0	1	3		
6	v4	0	0	0	0	3	0	0	0	2		
7	v5	0	0	0	0	0	2	0	9	3		
8	v6	0	0	0	0	0	0	6	0	1		
9	v7	0	0	0	0	0	0	0	2	3		
10	v8	0	0	0	0	0	0	0	0	5		
11	Максимальный вес вершин в одном классе разбиения:									12		
12												
13												
14												
15												
16												

Задача о разбиении

Рис. 12.6. Исходные данные для решения задачи о разбиении

Далее из любой свободной ячейки следует вызвать мастер добавления функции и в списке функций, определенных пользователем, выбрать созданную функцию: `LocOptRaz()`. После выбора данной функции в качестве значений первого ее аргумента следует выделить диапазон ячеек **B3:I10**, в которых содержатся значения матрицы весов дуг исходного графа. В качестве значений второго ее аргумента следует выделить диапазон ячеек **J3:J10**, в которых

содержатся значения вектора весов вершин исходного графа. В качестве значений третьего ее аргумента следует выделить ячейку **J11**, в которой содержится значение максимального веса вершин, попадающих в один класс разбиения (рис. 12.7).

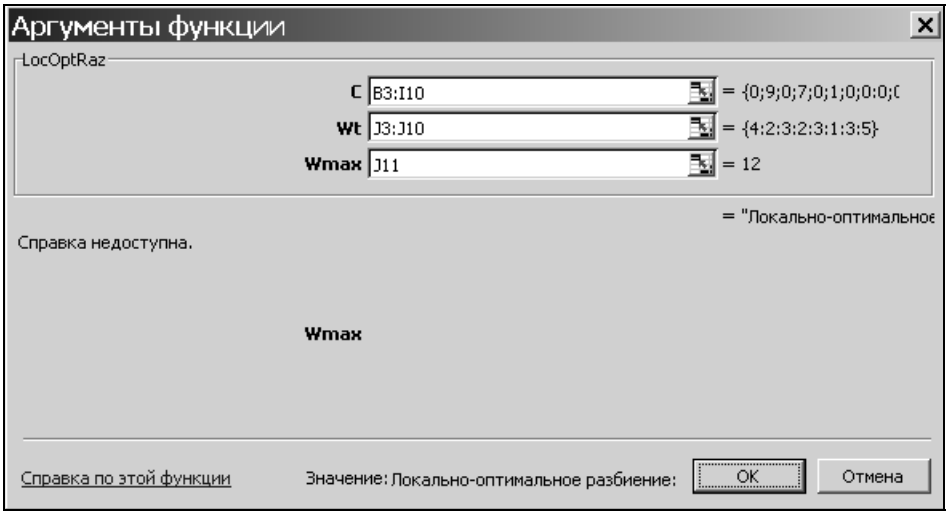


Рис. 12.7. Результат приближенного решения задачи о разбиении

ПрограммированиеЗадач.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Матрица весов дуг исходного графа									Веса			
2		v1	v2	v3	v4	v5	v6	v7	v8	вершин:			
3	v1	0	9	0	7	0	1	0	0	4			
4	v2	0	0	4	0	0	0	0	0	2			
5	v3	0	0	0	5	0	0	0	1	3			
6	v4	0	0	0	0	3	0	0	0	2			
7	v5	0	0	0	0	0	2	0	9	3			
8	v6	0	0	0	0	0	0	6	0	1			
9	v7	0	0	0	0	0	0	0	2	3			
10	v8	0	0	0	0	0	0	0	0	5			
11	Максимальный вес вершин в одном классе разбиения:									12			
12													
13	Локально-оптимальное разбиение: {1, 2, 3, 4}, {5, 6, 7, 8}, Fmin= 44												
14													
15													
16													
17													
18													

Задача о разбиении

Рис. 12.8. Результат приближенного решения задачи о разбиении

После задания аргументов и нажатия кнопки **ОК**, если текст программы не содержит ошибок, то она будет выполнена, а в ячейке с этой функцией на данном рабочем листе появится результат приближенного решения задачи о разбиении (рис. 12.8). При интерпретации полученного решения каждый класс разбиения заключается в фигурные скобки, внутри которых указаны номера вершин, образующих соответствующий класс разбиения.

Как показывает анализ найденного решения, оно полностью совпадает с точным решением данной задачи, полученным с помощью программы MS Excel в *разд. 8.3.3*. Данный факт может свидетельствовать в пользу не только правильности разработанной программы, но и возможности нахождения с ее помощью точных решений соответствующей задачи комбинаторной оптимизации.

12.3. Использование программ на языке VBA в книгах MS Excel

Разработанные программы на языке VBA, расширяющие функциональность программы MS Excel, оказываются доступными для использования только в тех рабочих книгах, в которых содержатся модули с текстом соответствующих программ. Вполне очевидно, что повторный набор этих текстов для других книг в случае использования разработанных функций нельзя признать удобным с практической точки зрения. Существует несколько способов того, как сделать доступными разработанные программы на языке VBA в других книгах программы MS Excel. Эти способы рассматриваются далее в данном разделе.

12.3.1. Экспорт и импорт модулей с текстами программ на VBA

Первый способ использования разработанных программ на языке VBA в среде Excel связан с экспортом выбранного модуля, содержащего разработанные тексты программ функций, во внешний файл с расширением bas, и последующим импортом этого файла в выбранную рабочую книгу программы MS Excel. Для экспорта модулей во внешний файл следует предварительно в проводнике проекта редактора Visual Basic выделить модуль с текстом программ, после чего выполнить операцию главного меню редактора Visual Basic: **File** (Файл) | **Export File** (Экспортировать в файл) или нажать комбинацию клавиш <Ctrl>+<E>.

В результате этих действий будет открыто стандартное окно сохранения файла для задания имени файла, в который будет скопирован текст программы выбранного модуля.

После выполнения этих действий соответствующий модуль может быть импортирован в любую книгу программы MS Excel. Для импорта модуля из внешнего файла следует предварительно в проводнике проекта редактора Visual Basic создать проект рабочей книги, куда следует поместить импортируемый модуль с текстами программ, после чего выполнить операцию главного меню редактора Visual Basic: **File** (Файл) | **Import File** (Импортировать из файла) или нажать комбинацию клавиш <Ctrl>+<M>.

В результате этих действий будет открыто стандартное окно открытия файла для задания имени файла, из которого следует вставить текст программы в качестве модуля редактируемого проекта. После этого все функции и подпрограммы, которые содержатся в импортируемом модуле, становятся доступными на рабочих листах той книги, в которую был импортирован соответствующий модуль.

12.3.2. Использование шаблонов с текстами программ на VBA

Второй способ использования разработанных программ на языке VBA в среде Excel связан с сохранением книги MS Excel с модулями программ в форме *шаблона* во внешнем файле с расширением XLT. С этой целью достаточно выполнить операцию главного меню программы MS Excel: **Файл** | **Сохранить как** и в стандартном диалоговом окне сохранения файла задать тип файла — **Шаблон (*.xlt)** и имя шаблона.

В последующем при создании новой книги следует выбрать сохраненный шаблон, на основании которого будет создана новая книга. В результате этого все функции и подпрограммы, которые содержатся в книге, на основании которой был создан шаблон пользователя, становятся доступными на рабочих листах новой книги.

Неудобство данного способа заключается в том, что в созданной на основе шаблона новой книге будут содержаться все рабочие листы, которые имеются в книге, которая была сохранена в форме шаблона. При этом в новую книгу будут скопированы не только модули с текстами программ на VBA, но и данные всех рабочих листов, форматирование ячеек этих листов, а также построенные диаграммы и другие графические изображения. При этом любой пользователь имеет доступ к исходным текстам программ, что может привести к их неработоспособности в случае неквалифицированного редактирования.

Для того чтобы исключить изменение текстов разработанных программ на VBA в новых книгах, созданных на основе сохраненного шаблона, можно

воспользоваться возможностью защиты проектов в редакторе Visual Basic. С этой целью следует выполнить следующие действия:

1. Выделить в проводнике проекта тот проект, к которому необходимо ограничить доступ.
2. Выполнить операцию контекстного меню: **VBAProject Properties** (Свойства проекта VBA) и в открывшемся диалоговом окне свойств проекта на вкладке **Protection** (Защита) выставить отметку в позиции: **Lock project for viewing** (Блокировать просмотр проекта).
3. Ввести пароль для доступа к просмотру и редактированию проекта и подтвердить его в соответствующих полях ввода (рис. 12.9).



Рис. 12.9. Внешний вид вкладки **Protection** для блокирования просмотра проекта

Если в последующем создать шаблон на основе книги с блокированным для просмотра проектом, то все функции и подпрограммы соответствующей книги будут доступны в новой книге, однако открыть проект в редакторе Visual Basic для просмотра текстов программ без ввода пароля не удастся.

12.3.3. Создание и использование надстроек пользователя с текстами программ на VBA

Данный способ использования разработанных программ на языке VBA в среде Excel связан с сохранением книги MS Excel с модулями программ в форме *надстройки пользователя* (Add-Ins) во внешнем файле с расширением XLA. В последующем созданная пользователем надстройка может быть подключена к программе MS Excel, что сделает доступными все содержащиеся в ней функции и подпрограммы.

Для создания надстройки пользователя следует выполнить следующие действия:

1. Задать имя надстройки и другие свойства файла проекта, который будет сохранен в форме надстройки. По умолчанию имя надстройки будет совпадать с именем файла, на основе которого создается надстройка. Диалоговое окно свойств файла проекта открывается при выполнении операции главного меню программы MS Excel: **Файл** | **Свойства**.
2. При необходимости защитить текст программ проекта от просмотра описанным ранее способом.
3. Сохранить книгу с разработанными программами на VBA в форме надстройки. С этой целью выполнить операцию главного меню программы MS Excel: **Файл** | **Сохранить как** и в стандартном диалоговом окне сохранения файла задать тип файла — **Надстройка Microsoft Office Excel (*.xla)** и его имя.

Созданную надстройку пользователя в последующем можно подключить к программе MS Excel. С этой целью следует выполнить следующие действия:

1. Открыть диалоговое окно надстроек с помощью операции главного меню программы MS Excel: **Сервис** | **Надстройки**.
2. В открывшемся окне выбрать ранее созданную надстройку пользователя.
3. Если имя надстройки пользователя отсутствует в перечне доступных надстроек, то нажать кнопку **Обзор** и в стандартном окне открытия файла выбрать папку и имя файла надстройки пользователя.

Например, если выполнить указанные действия и сохранить книгу с именем Программирование задач в виде файла надстройки, то в последующем можно подключить эту надстройку ко всем книгам программы MS Excel. В этом случае диалоговое окно доступных надстроек программы MS Excel имеет следующий вид (рис. 12.10).

При добавлении этой надстройки к программе MS Excel все разработанные ранее функции на языке VBA становятся доступными в любой книге до тех пор, пока данная надстройка не будет удалена из программы электронных

таблиц. Как нетрудно заметить, этот способ имеет неоспоримое преимущество по сравнению с ранее рассмотренными способами, поскольку рабочие листы исходной книги, на основе которой была создана надстройка пользователя, не копируются в новые книги программы MS Excel.

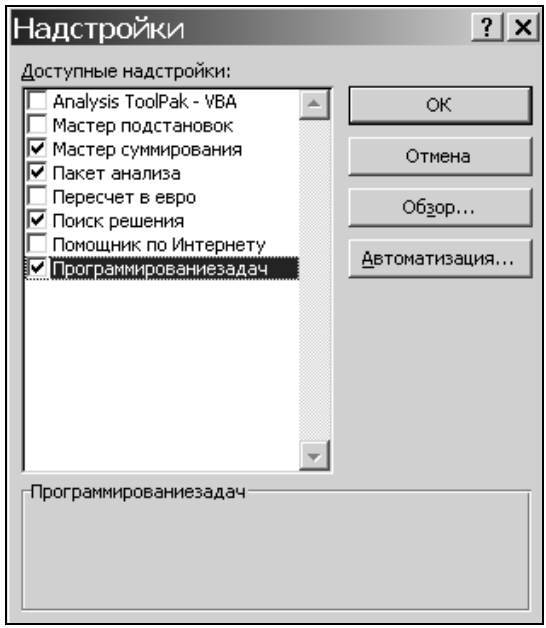


Рис. 12.10. Внешний вид окна доступных надстроек программы MS Excel

Однако для запуска содержащихся в надстройке пользователя подпрограмм и макросов необходимо заранее предусмотреть разработку форм пользователя или специальных программ добавления операций в главное меню программы MS Excel. Данное обстоятельство в какой-то степени ограничивает область применения данного способа использования разработанных программ на языке VBA в среде Excel.

12.4. Внешние программы и их использование в среде MS Excel

Разработанные функции и подпрограммы на языке VBA, расширяющие функциональность программы MS Excel, можно сделать доступными для использования в тех книгах, которые содержат модули с текстами этих программ. Однако программа MS Excel обладает и другой возможностью — использования функций, написанных на других языках программирования

и скомпилированных в виде отдельной библиотеки динамической компоновки (DLL). В настоящем разделе рассматриваются варианты использования в среде MS Excel функций и подпрограмм, разработанных на разных языках программирования.

12.4.1. Разработка внешней функции в среде Borland Delphi и ее использование в среде MS Excel

Рассмотренные ранее способы использования внешних программ VBA в среде MS Excel обладают тем недостатком, что приходится иметь в рабочей книге MS Excel дополнительные модули с текстами соответствующих программ. Этого неудобства можно избежать на основе использования функций, написанных на других языках программирования и скомпилированных в виде отдельной библиотеки динамической компоновки (DLL).

Для создания библиотек DLL необходимо дополнительно использовать некоторую интегрированную среду программирования, поскольку язык VBA не позволяет разрабатывать данные библиотеки. В качестве такой среды можно использовать, например, среду Borland Delphi или среду MS Visual Studio .NET/6. В настоящем разделе рассматриваются особенности процесса разработки пользовательской функции в среде Borland Delphi 7 и ее последующее использование в среде MS Excel.

В качестве примера создания функции, которая может быть разработана с помощью среды Borland Delphi 7 и в последующем использована в среде MS Excel, рассмотрим двумерную экспоненциальную функцию. Напомним, что соответствующая функция отсутствует в числе встроенных функций программы MS Excel. Именно поэтому в *разд. 11.1.2* был рассмотрен процесс создания пользовательской функции на языке VBA для вычисления двумерной экспоненциальной функции, которая в общем случае задается выражением (3.5.3).

Примечание

Следует заметить, что поскольку рассматриваемый материал ориентирован, главным образом, на пользователей, имеющих опыт программирования в интегрированных средах, общие особенности этих сред, а также синтаксис соответствующих языков программирования в книге не описываются. Читателям, которые заинтересованы в более подробной информации о назначении окон рабочего интерфейса и операций главного меню соответствующих сред, следует обратиться к специальным руководствам. В то же время приводимого в книге материала вполне достаточно для решения всех рассмотренных в ней задач.

Для вычисления отдельного значения двумерной экспоненциальной целевой функции для фиксированной пары значений переменных (x_1, x_2) согласно формуле (3.5.3) необходимо выполнить последовательное суммирование для возрастающих значений параметра a, начиная от значения 0,1 и заканчивая значением 1 с интервалом изменения данного параметра, равным 0,1. Именно эта задача будет решена с помощью специальной программы на языке Object Pascal в среде Borland Delphi 7.

Для разработки соответствующей программы в среде Borland Delphi 7 необходимо, прежде всего, установить эту среду на компьютер пользователя. Далее следует создать новый проект, для чего следует выполнить операцию главного меню редактора Delphi: **File** (Файл) | **New** (Новый) | **Other** (Другой), в результате чего откроется диалоговое окно мастера создания нового проекта в среде Delphi (рис. 12.11).

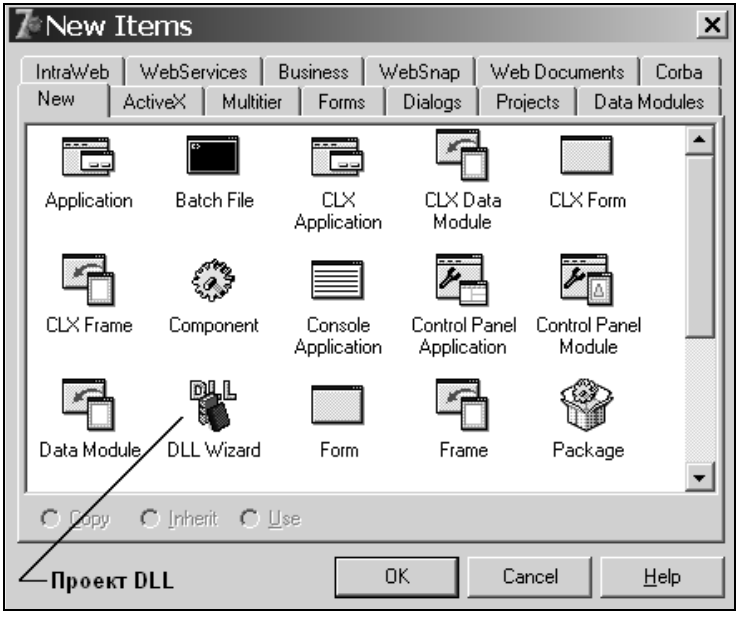


Рис. 12.11. Диалоговое окно мастера создания нового проекта в среде Delphi

В окне мастера создания нового проекта в среде Delphi следует выбрать пиктограмму DLL Wizard (Мастер DLL) и нажать кнопку **OK**. В результате выполнения этой операции будет создан новый проект, для которого следует задать имя: MyDLLDelphi. В редакторе Delphi появится окно кода проекта создания DLL, которое содержит необходимые служебные операторы (рис. 12.12).

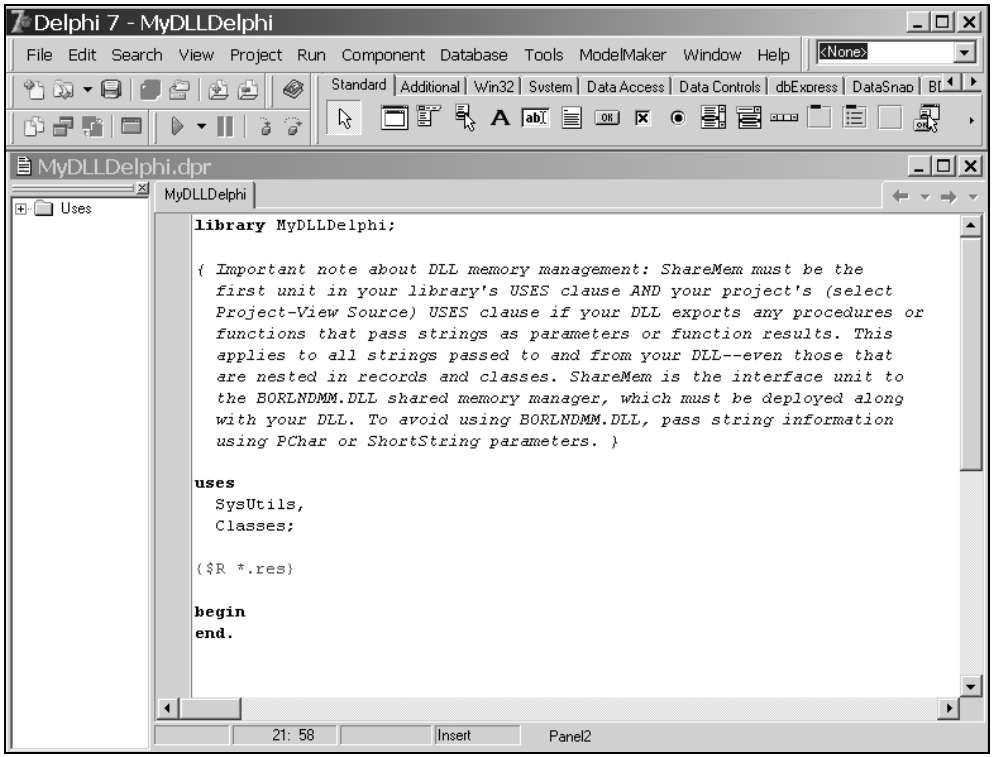


Рис. 12.12. Окно нового проекта по созданию DLL в среде Delphi

В редакторе кода Delphi перед завершающим блоком `begin-end` введем следующий текст программы на языке Pascal для вычисления двумерной экспоненциальной функции (листинг 12.7).

Листинг 12.7. Программа вычисления двумерной экспоненциальной функции на языке Pascal

```
function ExpFunc2(x1, x2: Real): Real; export; stdcall;
 var
   Sum, Func : Real;
   i : Byte;
 begin
   Sum := 0.0;
   For i := 1 To 10 do
      begin
         Func:=((Exp(-0.1*i*x1)-Exp(-0.1*i*x2))-(Exp(-0.1*i)-Exp(-i)));
         Sum := Sum + Func*Func;
```

```
      end;
   Result := Sum;
 end;
exports ExpFunc2;
```

Данная программа задает функцию `ExpFunc2`, которая имеет два формальных параметра: `x1` и `x2`. Текст программы этой функции начинается с помощью стандартного оператора языка Pascal — `Function`, предназначенного для объявления функции, после которого указывается имя функции `ExpFunc2` и ее формальные параметры в скобках. После имени функции следуют два ключевых слова: `export` — для указания того факта, что данная функция может быть использована другими приложениями, и ключевое слово: `stdcall` — для указания стандартного способа ее вызова.

Текст функции начинается с объявления типов переменных, при этом функция возвращает действительное значение, что соответствует типу переменной `Real`. Собственно вычисление значения двумерной экспоненциальной функции для двух значений независимых переменных `x1` и `x2` реализуется с помощью цикла `For-To-Do`, который выполняется последовательно с помощью переменной `i`. Для вычисления обычной экспоненциальной функции применяется встроенная функция `Exp` с единственным аргументом. Оператор `Result: = S` задает возвращаемое данной функцией значение. Наконец, последний оператор `Export ExpFunc2` служит для указания имени для обращения к функции из других приложений.

Внешний вид редактора Delphi с текстом программы в файле с именем MyDLLDelphi.dpr изображен на рис. 12.13.

После записи текста программы функции в редакторе Delphi необходимо выполнить компиляцию и сборку проекта. С целью выявления ошибок в тексте программы целесообразно предварительно выполнить компиляцию проекта, для чего следует выполнить следующую операцию главного меню редактора Delphi: **Project** (Проект) | **Compile MyDLLDelphi** (Компилировать MyDLLDelphi). При этом в случае наличия ошибок, они будут выявлены, а информация о них будет отображена в специальном окне редактора Delphi.

После устранения возможных ошибок следует выполнить сборку проекта, для чего следует выполнить следующую операцию главного меню редактора Delphi: **Project** (Проект) | **Build MyDLLDelphi** (Создать MyDLLDelphi). В результате этого в папке с именами файлов данного проекта будет создан файл с именем MyDLLDelphi.dll, который представляет собой бинарный файл с исполнимым кодом вычисления значений двумерной экспоненциальной функции.

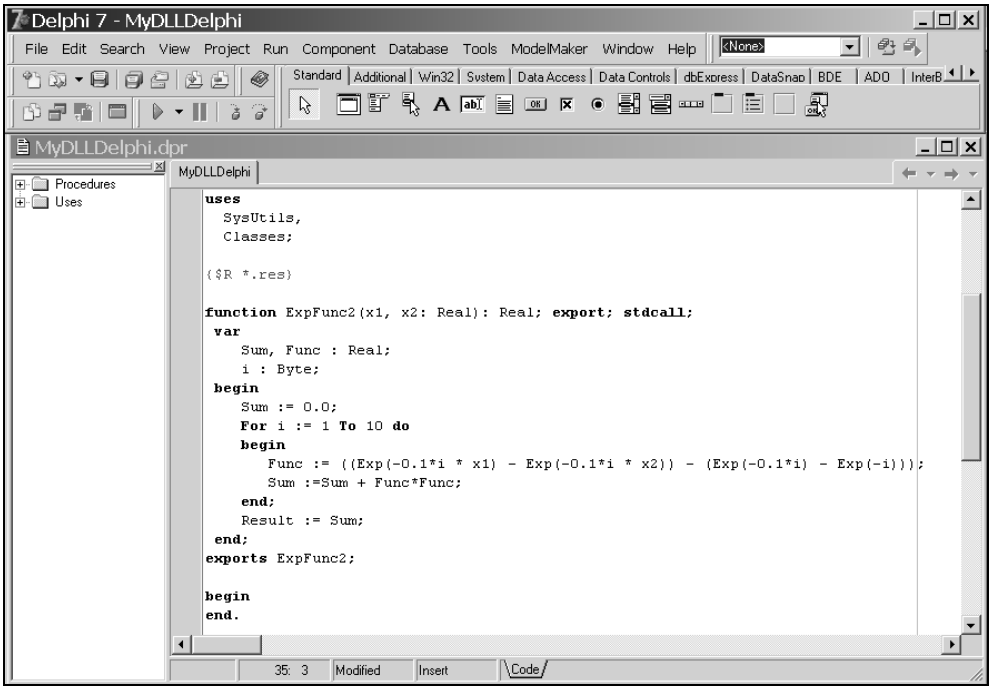


Рис. 12.13. Редактор Delphi с текстом программы вычисления двумерной экспоненциальной функции

Для использования данной функции в среде MS Excel необходимо скопировать файл MyDLLDelphi.dll в системную папку операционной системы MS Windows. Такой папкой является папка с именем system, которая имеется в папке с именем Windows на том диске, на который устанавливалась ОС. После выполнения этих действий функция `ExpFunc2` становится доступной для использования в качестве формулы в ячейках рабочего листа программы MS Excel. При этом никаких дополнительных действий по настройке программы MS Excel не требуется.

Функция `ExpFunc2` может быть непосредственно вызвана из ячеек любого рабочего листа данной книги. При этом следует использовать специальный формат вызова данной формулы из ячеек программы MS Excel. С этой целью, выделив нужную ячейку, необходимо вызвать мастер функций, в котором следует выбрать функцию: `ВЫЗВАТЬ`. Эта функция предназначена для вызова функций из внешних библиотек.

Примечание

Ситуация с функцией `ВЫЗВАТЬ` в различных версиях программы MS Excel может служить предметом специальных исследований, поскольку является при-

мером парадоксов в программной индустрии. Так, если в версии MS Excel 97 эта функция работает вполне корректно, то в версии MS Excel 2000 существует проблема с ее вызовом. Что касается версии MS Excel 2003, то там эта функция вообще отсутствует в перечне доступных функций. Видимо, с годами отношение к интеллекту пользователей у разработчиков Microsoft меняется не в лучшую сторону. Именно поэтому в дальнейшем все манипуляции с вызовом функций из библиотек DLL иллюстрируются применительно к программе MS Excel 97 без дополнительных комментариев.

Функция ВЫЗВАТЬ имеет следующий формат для своего использования:

```
ВЫЗВАТЬ (ИМЯ_МОДУЛЯ; ИМЯ_ФУНКЦИИ; ТИПЫ_ДАННЫХ; АРГУМЕНТ_1;
АРГУМЕНТ_2; ... АРГУМЕНТ_N).
```

Здесь ИМЯ_МОДУЛЯ — это текст в кавычках, задающий имя динамически связываемой библиотеки (DLL), которая содержит функцию, вызываемую в программе MS Excel. ИМЯ_ФУНКЦИИ — это текст в кавычках, задающий имя функции из DLL, вызываемой в программе MS Excel. ТИПЫ_ДАННЫХ — это текст в кавычках, задающий тип данных возвращаемого значения и типы данных всех аргументов для процедуры из DLL. При этом первая буква ТИПЫ_ДАННЫХ задает тип возвращаемого функцией значения. Коды, используемые для задания аргумента ТИПЫ_ДАННЫХ, подробно описаны в разделе справочной системы функции ВЫЗВАТЬ или Справочнике по функциям программы MS Excel.

АРГУМЕНТ_1; АРГУМЕНТ_2; ... АРГУМЕНТ_N — это аргументы, которые должны быть переданы функции для вычисления соответствующего значения.

Применительно к рассматриваемой функции соответствующая формула должна иметь следующий вид:

```
=ВЫЗВАТЬ("MyDLLDelphi"; "ExpFunc2"; "BBB"; ЯЧЕЙКА_1; ЯЧЕЙКА_2),
```

где в ячейках ЯЧЕЙКА_1 и ЯЧЕЙКА_2 заданы значения первого и второго аргумента двумерной экспоненциальной функции. Следует отметить, что в качестве типа данных возвращаемого значения и типа данных обоих аргументов используется тип Real, которому соответствует латинское B.

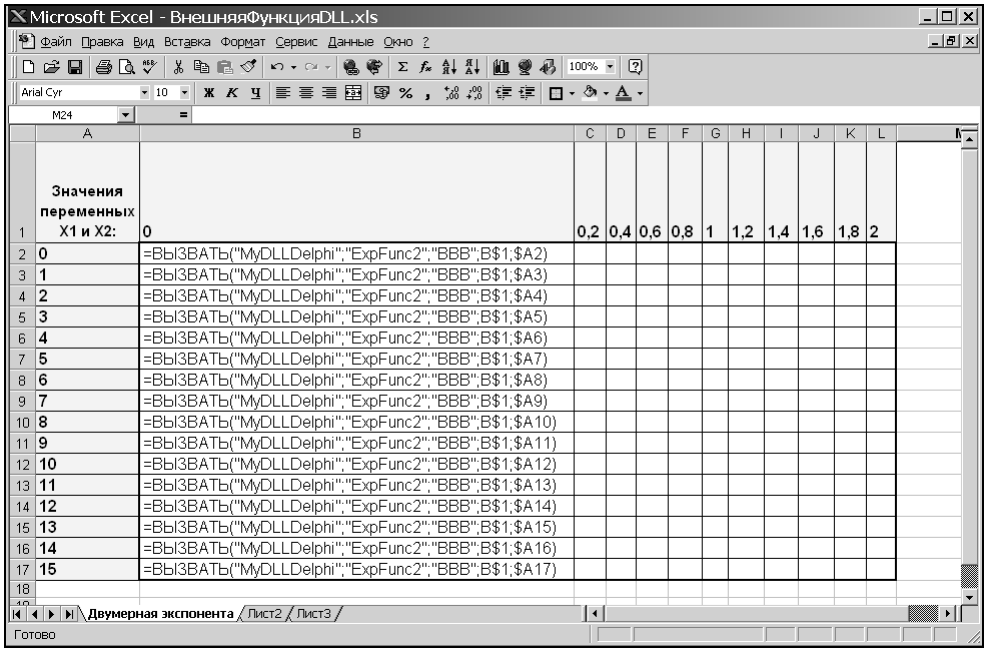
В качестве примера вызова данной функции рассмотрим подготовку исходных данных для построения графика двумерной экспоненциальной функции. С этой целью создадим новую книгу с именем Внешняя функция DLL и изменим имя ее первого листа на Двумерная экспонента. Далее следует выполнить следующие действия:

1. В ячейку **A1** рабочего листа с именем Двумерная экспонента книги Внешняя функция DLL введем текст "Значения переменных X1 и X2:".
2. С помощью операции автозаполнения зададим значения первой переменной x_1 в ячейках **B1:L1**, а значения второй переменной x_2 — в ячейках

A2:A17. Тем самым задаются последовательный диапазон значений первой переменной от 0 до 2 с интервалом их изменения, равным 0,2, и диапазон значений второй переменной от 0 до 15 с интервалом их изменения, равным 1.

3. Далее в ячейку **B2** введем формулу: `=ВЫЗВАТЬ("MyDLLDelphi"; "ExpFunc2"; "BBB"; ЯЧЕЙКА_1; ЯЧЕЙКА_2)`, которую с помощью автозаполнения скопируем в ячейки **B3:B17**.

Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 12.14).



	A	B	C	D	E	F	G	H	I	J	K	L
1	Значения переменных X1 и X2:	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2
2	0	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A2)										
3	1	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A3)										
4	2	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A4)										
5	3	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A5)										
6	4	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A6)										
7	5	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A7)										
8	6	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A8)										
9	7	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A9)										
10	8	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A10)										
11	9	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A11)										
12	10	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A12)										
13	11	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A13)										
14	12	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A14)										
15	13	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A15)										
16	14	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A16)										
17	15	=ВЫЗВАТЬ("MyDLLDelphi";"ExpFunc2";"BBB";B$1;$A17)										
18												

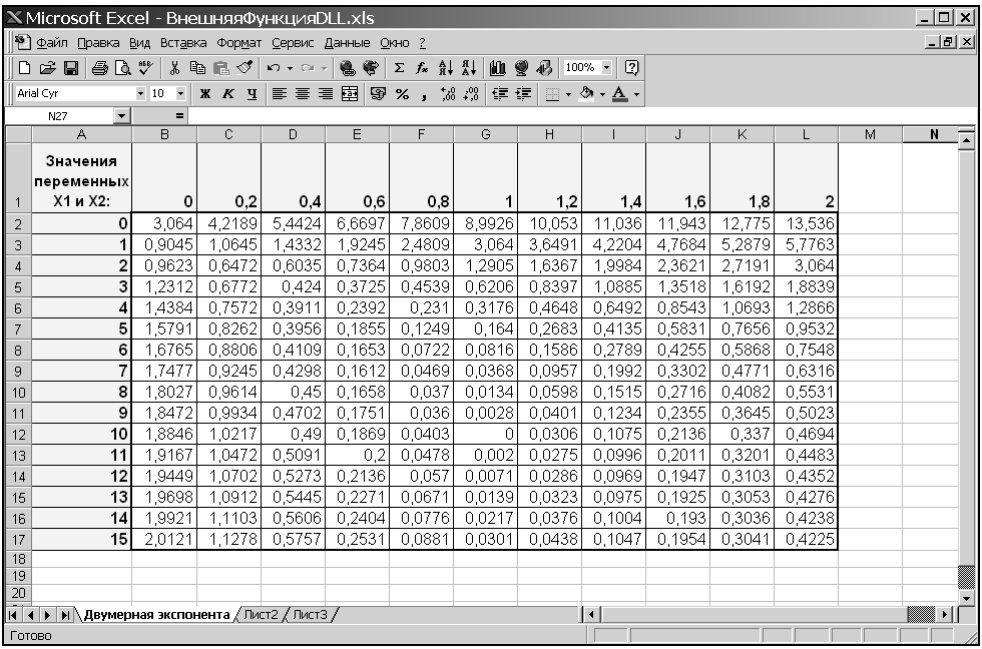
Рис. 12.14. Исходные данные для вычисления значений двумерной экспоненциальной функции

Выделив диапазон ячеек **B2:B17**, также с помощью автозаполнения скопируем соответствующие данные в ячейки **C2:L17**.

После ввода формул в ячейки данного рабочего листа будет выполнен расчет значений двумерной экспоненциальной функции для соответствующих значений аргументов. Результат вычисления значений двумерной экспоненциальной функции будет иметь следующий вид (рис. 12.15).

Как можно заметить, полученный результат вычисления двумерной экспоненциальной функции из библиотеки, созданной в среде Delphi, полностью

совпадает с исходными данными для построения графика двумерной экспоненциальной функции, полученными с помощью программы на VBA (см. рис. 11.4). Данный факт свидетельствует в пользу правильности вычислительных расчетов, выполненных программой MS Excel с помощью функции из библиотеки DLL, созданной в среде Delphi. Аналогичным образом могут быть представлены в форме внешних функций библиотек DLL и другие функции, предназначенные для решения отдельных типовых задач оптимизации.


Microsoft Excel - ВнешняяФункцияDLL.xls

Файл Правка Вид Вставка Формат Сервис Данные Окно ?

Arial Cyr 10

N27 =

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Значения переменных X1 и X2:	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2		
2	0	3,064	4,2189	5,4424	6,6697	7,8609	8,9926	10,053	11,036	11,943	12,775	13,536		
3	1	0,9045	1,0645	1,4332	1,9245	2,4809	3,064	3,6491	4,2204	4,7684	5,2879	5,7763		
4	2	0,9623	0,6472	0,6035	0,7364	0,9803	1,2905	1,6367	1,9984	2,3621	2,7191	3,064		
5	3	1,2312	0,6772	0,424	0,3725	0,4539	0,6206	0,8397	1,0885	1,3518	1,6192	1,8839		
6	4	1,4384	0,7572	0,3911	0,2392	0,231	0,3176	0,4648	0,6492	0,8543	1,0693	1,2866		
7	5	1,5791	0,8262	0,3956	0,1855	0,1249	0,164	0,2683	0,4135	0,5831	0,7656	0,9532		
8	6	1,6765	0,8806	0,4109	0,1653	0,0722	0,0816	0,1586	0,2789	0,4255	0,5868	0,7548		
9	7	1,7477	0,9245	0,4298	0,1612	0,0469	0,0368	0,0957	0,1992	0,3302	0,4771	0,6316		
10	8	1,8027	0,9614	0,45	0,1658	0,037	0,0134	0,0598	0,1515	0,2716	0,4082	0,5531		
11	9	1,8472	0,9934	0,4702	0,1751	0,036	0,0028	0,0401	0,1234	0,2355	0,3645	0,5023		
12	10	1,8846	1,0217	0,49	0,1869	0,0403	0	0,0306	0,1075	0,2136	0,337	0,4694		
13	11	1,9167	1,0472	0,5091	0,2	0,0478	0,002	0,0275	0,0996	0,2011	0,3201	0,4483		
14	12	1,9449	1,0702	0,5273	0,2136	0,057	0,0071	0,0286	0,0969	0,1947	0,3103	0,4352		
15	13	1,9698	1,0912	0,5445	0,2271	0,0671	0,0139	0,0323	0,0975	0,1925	0,3053	0,4276		
16	14	1,9921	1,1103	0,5606	0,2404	0,0776	0,0217	0,0376	0,1004	0,193	0,3036	0,4238		
17	15	2,0121	1,1278	0,5757	0,2531	0,0881	0,0301	0,0438	0,1047	0,1954	0,3041	0,4225		
18														
19														
20														

Двумерная экспонента / Лист2 / Лист3

Готово

Рис. 12.15. Результат вычисления значений двумерной экспоненциальной функции из библиотеки, созданной в среде Delphi

12.4.2. Разработка внешней функции в среде MS Visual Studio .NET и ее использование в среде MS Excel

В качестве среды для разработки внешних функций можно использовать среду MS Visual Studio .NET или среду MS Visual Studio .NET или предыдущую ее версию MS Visual Studio 6. В настоящем разделе рассматриваются особенности процесса разработки пользовательской функции на языке программирования Visual C++ в среде MS Visual Studio .NET и ее последующее ис-

пользование в среде MS Excel. При этом все описанные действия остаются справедливыми и могут быть аналогично реализованы в среде MS Visual Studio 6.

В качестве примера создания функции, которая может быть разработана с помощью среды MS Visual Studio .NET и в последующем использована в среде MS Excel, также рассматривается двумерная экспоненциальная функция. Напомним, что в общем случае двумерная экспоненциальная функция задается выражением (3.5.3). Соответствующая задача по вычислению значения этой функции для произвольной пары ее аргументов может быть решена с помощью специальной программы на языке Visual C++ в среде MS Visual Studio .NET.

Для разработки соответствующей программы в среде MS Visual Studio .NET необходимо прежде всего установить эту среду на компьютер пользователя. При загрузке этой среды появляется диалоговое окно с предложением выбрать тип нового проекта (рис. 12.16).

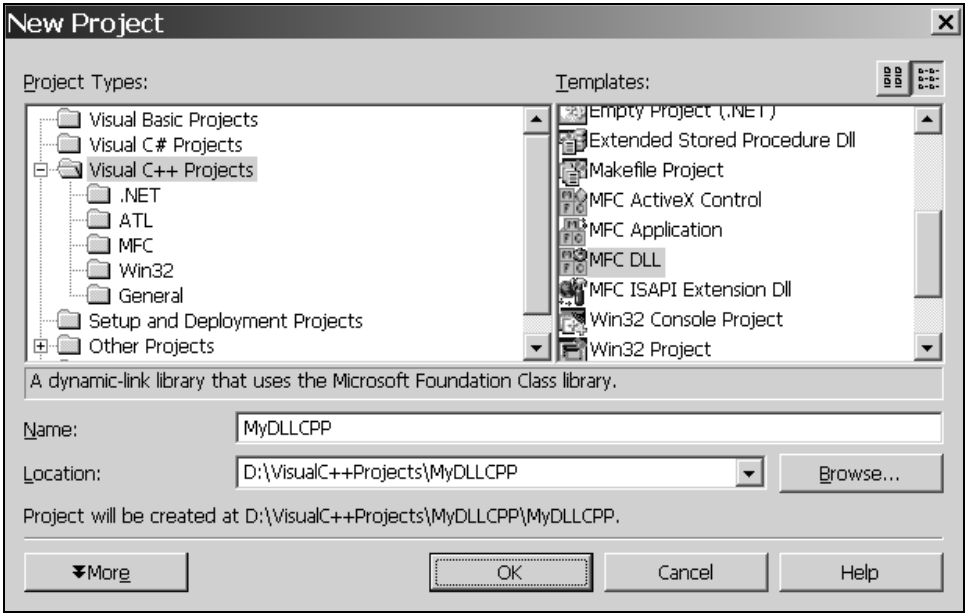


Рис. 12.16. Диалоговое окно мастера создания нового проекта в среде MS Visual Studio .NET

Если по какой-либо причине диалоговое окно мастера создания нового проекта не появилось, то для создания нового проекта следует выполнить операцию главного меню редактора MS Visual Studio .NET: **File** (Файл) | **New** (Новый) | **Project** (Проект). В результате этих действий будет открыто диалоговое окно мастера создания нового проекта в среде MS Visual Studio .NET (рис. 12.16).

В окне мастера создания нового проекта в среде MS Visual Studio .NET следует выделить в списке слева группу проектов: Visual C++ Projects, а в списке справа выделить пиктограмму MFC DLL (DLL на основе библиотеки классов MFC). Далее следует задать имя проекта MyDLLCPP, выбрать папку для размещения файлов проекта и нажать кнопку **OK**.

В результате выполнения этих действий будет создан новый проект с именем MyDLLCPP, доступный для редактирования в среде Visual Studio .NET. В редакторе MS Visual Studio .NET появится окно структуры проекта MyDLLCPP, который уже содержит необходимые файлы (рис. 12.17).

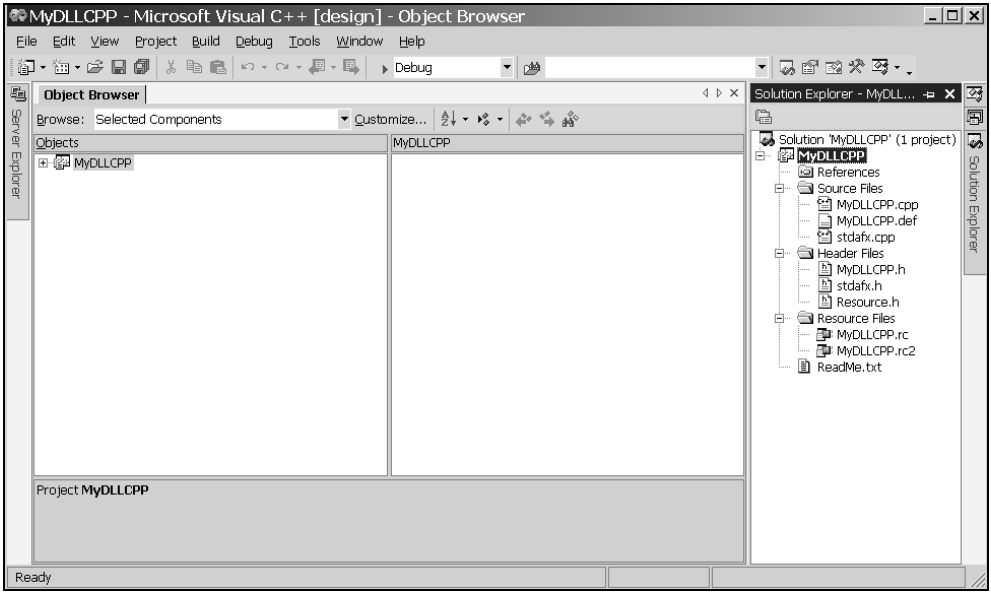


Рис. 12.17. Окно нового проекта по созданию DLL в среде Visual Studio .NET

В окне структуры проекта следует выполнить двойной щелчок на имени файла MyDLLCPP.CPP. В результате в редакторе кода MS Visual Studio .NET откроется этот файл, в конце которого следует ввести следующий текст программы на языке Visual C++ для вычисления двумерной экспоненциальной функции (листинг 12.8).

Листинг 12.8. Программа вычисления двумерной экспоненциальной функции на языке Visual C++

```
double __declspec(dllexport) __cdecl expFunc2 (double x1, double x2)
{
   register int i=1;
```

```
   double summa=0;
   double func=0;
   for (i = 1; i <= 10; i++)
     {
        func = exp(-0.1*i*x1)-exp(-0.1*i*x2)-exp(-0.1*i)+ exp(-1.0*i);
        summa=summa+func*func;
     }
    return summa;
}
```

Данная программа задает функцию `expFunc2`, которая имеет два формальных параметра: `x1` и `x2`. Текст программы этой функции начинается с помощью стандартного объявления подпрограммы — функции языка Visual C++. При этом имена формальных параметров функции `expFunc2` указываются в скобках. Перед именем функции следуют два ключевых слова: `__declspec(dllexport)` — для указания того факта, что рассматриваемая функция экспортируется из библиотеки DLL, и ключевое слово `__cdecl` — для указания компилятору совместимости синтаксиса переменных C и C++.

Текст функции начинается с явного объявления типов переменных, при этом функция возвращает действительное значение, что соответствует типу переменной `double`. Собственно вычисление значения двумерной экспоненциальной функции для двух значений независимых переменных `x1` и `x2` реализуется с помощью цикла `for`, который выполняется последовательно с помощью переменной `i`. Для вычисления обычной экспоненциальной функции применяется встроенная функция `exp` с единственным аргументом.

Наконец, последний оператор `return summa` задает возвращаемое данной функцией значение. Внешний вид редактора MS Visual Studio .NET с текстом программы, внесенной в файл с именем MyDLLCPP.cpp, изображен на рис. 12.18.

Следует заметить, что для использования встроенной экспоненциальной функции `exp` необходимо вставить в начало файла директиву: `#include "math.h"`, которая указывает ссылку на заголовочный файл с именами математических функций, доступных в языке программирования Visual C++.

Далее следует объявить данную функцию в заголовочном файле проекта — файле с именем MyDLLCPP.h. Для этого в файле окне структуры проекта следует выполнить двойной щелчок на имени файла MyDLLCPP.h. В результате в редакторе кода MS Visual Studio .NET откроется этот файл, в середину которого перед объявлением класса `CMyDLLCPPApp` следует ввести следующий текст объявления двумерной экспоненциальной функции (листинг 12.9).

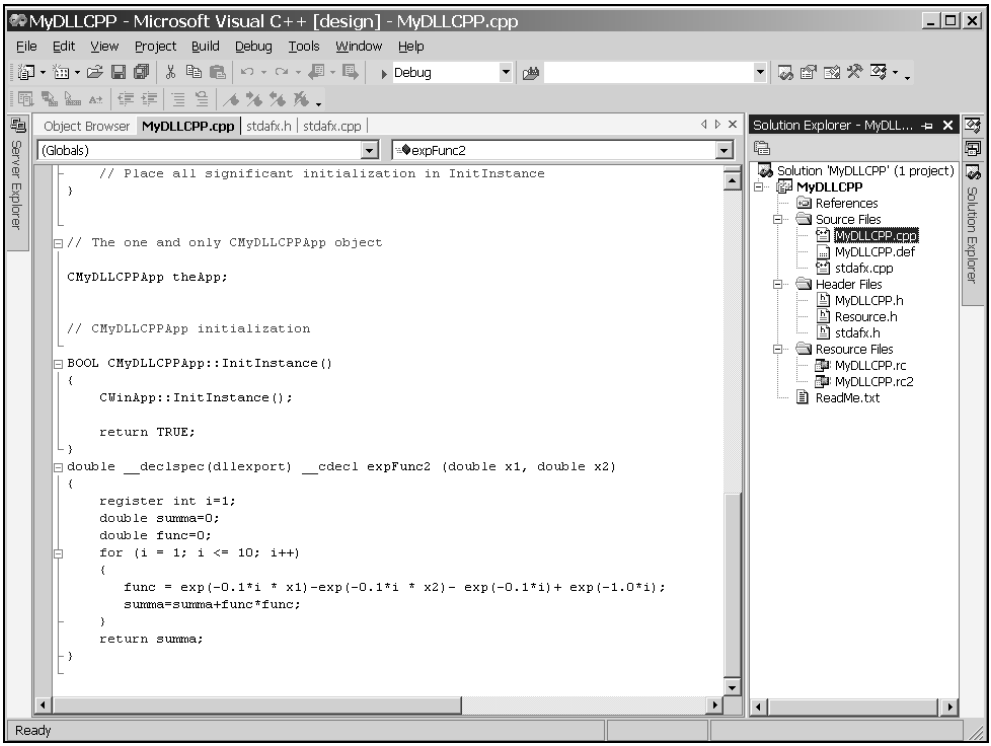


Рис. 12.18. Редактор MS Visual Studio .NET с текстом программы вычисления двумерной экспоненциальной функции

Листинг 12.9. Программа объявления двумерной экспоненциальной функции на языке Visual C++

```
extern "C"
{
       double __declspec(dllexport) __cdecl expFunc2 (double, double);
};
```

Внешний вид редактора MS Visual Studio .NET с текстом заголовочного файла с именем MyDLLCPP.h изображен на рис. 12.19.

После записи текста программы функции в редакторе MS Visual Studio .NET необходимо выполнить компиляцию и сборку проекта. С целью выявления ошибок в тексте программы целесообразно предварительно выполнить компиляцию проекта, для чего следует выполнить следующую операцию главного меню редактора MS Visual Studio .NET: **Build** (Выполнить сборку) |

Compile (Компилировать). При этом в случае наличия ошибок, они будут выявлены, а информация о них будет отображена в специальном окне редактора MS Visual Studio .NET.

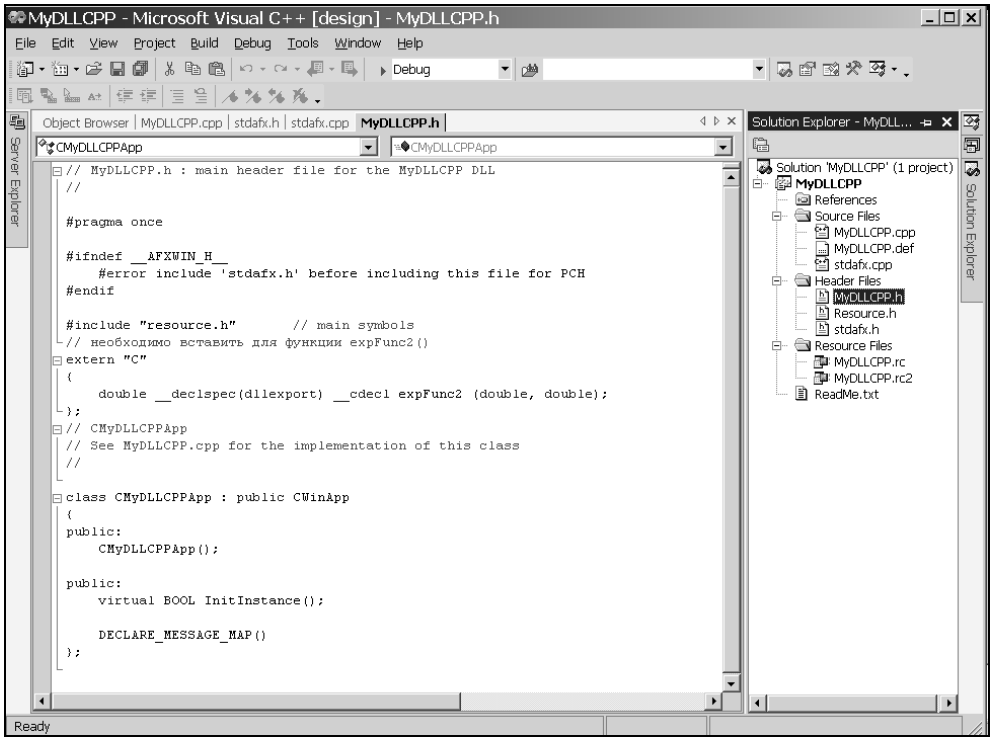


Рис. 12.19. Редактор MS Visual Studio .NET с текстом заголовочного файла проекта вычисления двумерной экспоненциальной функции

После устранения возможных ошибок следует выполнить сборку проекта, для чего нужно выполнить такую операцию главного меню редактора MS Visual Studio .NET: **Build** (Выполнить сборку) | **Build Solution** (Выполнить сборку решения). Окно сообщений при успешной сборке проекта MS Visual Studio .NET имеет следующий вид (рис. 12.20).

В результате этого в папке с именами файлов данного проекта будет создан файл с именем MyDLLCPP.dll, который представляет собой бинарный файл с исполнимым кодом вычисления значений двумерной экспоненциальной функции.

Для использования данной функции в среде MS Excel необходимо скопировать файл MyDLLCPP.dll в системную папку операционной системы MS Windows. Такой папкой является папка с именем system, которая имеется

в папке с именем Windows на том диске, на который устанавливалась ОС. После выполнения этих действий функция `expFunc2` становится доступной для использования в качестве формулы в ячейках рабочего листа программы MS Excel. Никаких дополнительных действий по настройке программы MS Excel не требуется.

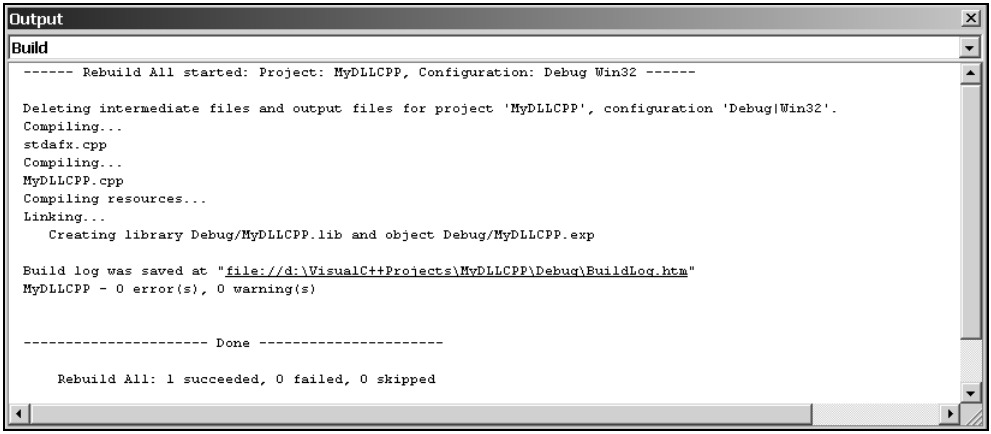


Рис. 12.20. Окно сообщений при успешной сборке проекта MS Visual Studio .NET

Функция `expFunc2` может быть вызвана из ячеек любого рабочего листа книги. При этом следует использовать специальный формат вызова формулы из ячеек программы MS Excel, который был рассмотрен ранее в *разд. 12.3.2*. С этой целью, выделив нужную ячейку, необходимо вызвать мастер функций, в котором следует выбрать функцию: `ВЫЗВАТЬ`. Эта функция предназначена для вызова функций из внешних библиотек.

Применительно к рассматриваемой функции соответствующая формула должна иметь следующий вид:

```
=ВЫЗВАТЬ("MyDLLCPP"; "expFunc2"; "BBB"; ЯЧЕЙКА_1; ЯЧЕЙКА_2),
```

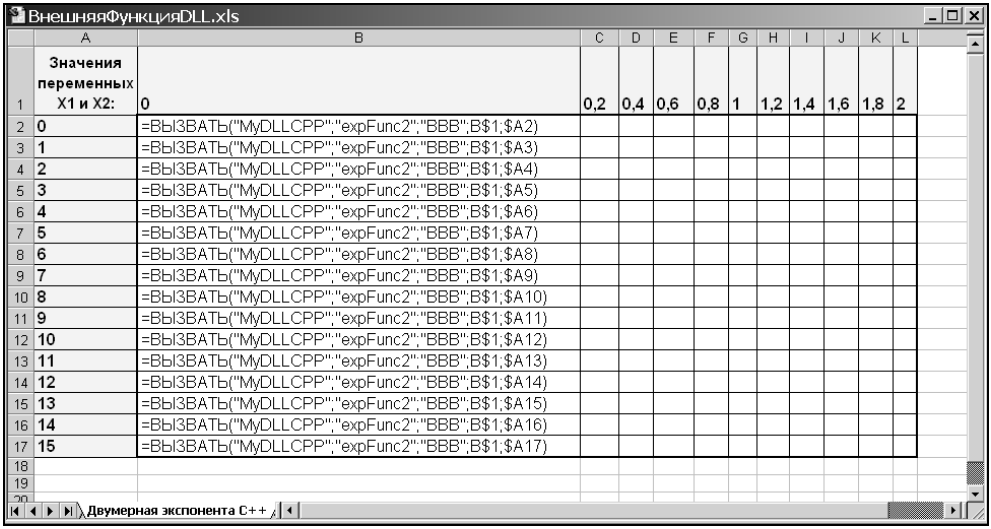
где в ячейках `ЯЧЕЙКА_1` и `ЯЧЕЙКА_2` заданы значения первого и второго аргумента двумерной экспоненциальной функции. Следует отметить, что в качестве типа данных возвращаемого значения и типа данных обоих аргументов используется тип `double`, которому соответствует латинское B.

В качестве примера вызова данной функции также рассмотрим подготовку исходных данных для построения графика двумерной экспоненциальной функции. С этой целью в книге с именем Внешняя функция DLL создадим новый лист с именем: Двумерная экспонента C++. Далее следует выполнить следующие действия:

1. В ячейку **A1** рабочего листа с именем Двумерная экспонента C++ книги Внешняя функция DLL введем текст "Значения переменных X1 и X2:".

2. С помощью операции автозаполнения зададим значения первой переменной x_1 в ячейках **B1:L1**, а значения второй переменной x_2 — в ячейках **A2:A17**. Тем самым задаются: последовательный диапазон значений первой переменной от 0 до 2 с интервалом их изменения, равным 0,2, и диапазон значений второй переменной от 0 до 15 с интервалом их изменения, равным 1.
3. Далее в ячейку **B2** введем формулу: `=ВЫЗВАТЬ("MyDLLCPP"; "expFunc2"; "BBB"; ЯЧЕЙКА_1; ЯЧЕЙКА_2)`, которую с помощью автозаполнения скопируем в ячейки **B3:B17**.

Результат выполнения данной последовательности операций по подготовке исходных данных будет иметь следующий вид (рис. 12.21).



ВнешняяФункцияDLL.xls

	A	B	C	D	E	F	G	H	I	J	K	L
1	Значения переменных X1 и X2:	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2
2	0	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A2)										
3	1	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A3)										
4	2	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A4)										
5	3	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A5)										
6	4	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A6)										
7	5	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A7)										
8	6	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A8)										
9	7	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A9)										
10	8	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A10)										
11	9	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A11)										
12	10	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A12)										
13	11	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A13)										
14	12	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A14)										
15	13	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A15)										
16	14	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A16)										
17	15	=ВЫЗВАТЬ("MyDLLCPP";"expFunc2";"BBB";B$1;$A17)										
18												
19												
20												

Двумерная экспонента C++

Рис. 12.21. Исходные данные для вычисления значений двумерной экспоненциальной функции

Выделив диапазон ячеек **B2:B17**, также с помощью автозаполнения скопируем соответствующие данные в ячейки **C2: L17**.

После ввода формул в ячейки данного рабочего листа будет выполнен расчет значений двумерной экспоненциальной функции для соответствующих значений аргументов. Результат вычисления значений двумерной экспоненциальной функции будет иметь следующий вид (рис. 12.22).

Как нетрудно заметить, полученный результат вычисления двумерной экспоненциальной функции из библиотеки, созданной на Visual C++ в среде MS Visual Studio .NET, полностью совпадают со значениями этой функции,

полученными с помощью программы на VBA (см. рис. 11.4), и значениями, полученными с помощью функции из библиотеки, созданной в среде Delphi (см. рис. 12.15). Данный факт свидетельствует в пользу правильности вычислительных расчетов, выполненных программой MS Excel с помощью функции из библиотеки DLL, созданной в среде MS Visual Studio .NET. Аналогичным образом могут быть созданы в среде MS Visual Studio .NET и представлены в форме внешних функций библиотек DLL другие функции, предназначенные для решения отдельных типовых задач оптимизации, что предлагается выполнить читателям самостоятельно в качестве упражнения.

ВнешняяФункцияDLL.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Значения переменных X1 и X2:	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2	
2	0	3,0640	4,2189	5,4424	6,6697	7,8609	8,9926	10,0528	11,0364	11,9429	12,7749	13,5362	
3	1	0,9045	1,0645	1,4332	1,9245	2,4809	3,0640	3,6491	4,2204	4,7684	5,2879	5,7763	
4	2	0,9623	0,6472	0,6035	0,7364	0,9803	1,2905	1,6367	1,9984	2,3621	2,7191	3,0640	
5	3	1,2312	0,6772	0,4240	0,3725	0,4539	0,6206	0,8397	1,0885	1,3518	1,6192	1,8839	
6	4	1,4384	0,7572	0,3911	0,2392	0,2310	0,3176	0,4648	0,6492	0,8543	1,0693	1,2866	
7	5	1,5791	0,8262	0,3956	0,1855	0,1249	0,1640	0,2683	0,4135	0,5831	0,7656	0,9532	
8	6	1,6765	0,8806	0,4109	0,1653	0,0722	0,0816	0,1586	0,2789	0,4255	0,5868	0,7548	
9	7	1,7477	0,9245	0,4298	0,1612	0,0469	0,0368	0,0957	0,1992	0,3302	0,4771	0,6316	
10	8	1,8027	0,9614	0,4500	0,1658	0,0370	0,0134	0,0598	0,1515	0,2716	0,4082	0,5531	
11	9	1,8472	0,9934	0,4702	0,1751	0,0360	0,0028	0,0401	0,1234	0,2355	0,3645	0,5023	
12	10	1,8846	1,0217	0,4900	0,1869	0,0403	0,0000	0,0306	0,1075	0,2136	0,3370	0,4694	
13	11	1,9167	1,0472	0,5091	0,2000	0,0478	0,0020	0,0275	0,0996	0,2011	0,3201	0,4483	
14	12	1,9449	1,0702	0,5273	0,2136	0,0570	0,0071	0,0286	0,0969	0,1947	0,3103	0,4352	
15	13	1,9698	1,0912	0,5445	0,2271	0,0671	0,0139	0,0323	0,0975	0,1925	0,3053	0,4276	
16	14	1,9921	1,1103	0,5606	0,2404	0,0776	0,0217	0,0376	0,1004	0,1930	0,3036	0,4238	
17	15	2,0121	1,1278	0,5757	0,2531	0,0881	0,0301	0,0438	0,1047	0,1954	0,3041	0,4225	
18													
19													
20													

Двумерная экспонента C++

Рис. 12.22. Результат вычисления значений двумерной экспоненциальной функции из библиотеки, созданной в среде MS Visual Studio .NET

12.4.3. Разработка функции нахождения минимального пути в среде Borland Delphi и ее использование в среде MS Excel

В качестве следующего примера создания функции, которая может быть разработана с помощью среды Borland Delphi 7 и в последующем использована в среде MS Excel, рассмотрим функцию, предназначенную для нахождения минимального пути в ориентированном графе. Напомним, что в *разд. 11.4.1* были рассмотрены особенности создания аналогичной функции на языке VBA, оформленной в виде отдельной функции с именем `MinPath`. Хотя задача поиска минимального пути, строго говоря, относится к задачам оптимизации на графах, приемы разработки соответствующей функции могут быть

использованы для создания аналогичных функций при решении задач комбинаторной оптимизации.

Для разработки программы, реализующей функцию нахождения минимального пути в графе на основе алгоритма пометок Дейкстры, в среде Borland Delphi 7 можно воспользоваться ранее созданным в *разд. 12.4.1* проектом с именем MyDLLDelphi.

Примечание

В случае создания для данной функции нового проекта следует выполнить операцию главного меню редактора Delphi: **File** (Файл) | **New** (Новый) | **Other** (Другой). В результате этого откроется диалоговое окно мастера создания нового проекта в среде Delphi, аналогичное изображенному на рис. 12.11, в котором следует выбрать пиктограмму DLL Wizard (Мастер DLL) и нажать кнопку **OK**. В результате выполнения этой операции будет создан новый проект, для которого следует задать имя: MyDLLDelphi либо любое другое.

В редакторе кода Delphi для проекта с именем MyDLLDelphi после текста ранее созданной двумерной экспоненциальной функции введем следующий текст программы на языке Pascal для функции нахождения минимального пути в графе (листинг 12.10).

Листинг 12.10. Программа вычисления функции нахождения минимального пути в графе на языке Pascal

```
Function MinPath(N:Integer; C: array of Real):PChar; export; stdcall;
 var
   dis : array of array of Real;     // dis- матрица расстояний графа
   Vmin : array of Real;
   Vpom,Vect : array of Integer;
   S1,S: String;
   i,j,k,im,jm : Integer;
   rec : Real;
 begin
    SetLength(Vmin, N);
    SetLength(Vpom, N);
    SetLength(Vect, N);
    SetLength(dis, N);
    for i:=0 to N-1 do
       SetLength(dis[i], N);
    for i:=0 to N-1 do
```

```
   for j:=0 to N-1 do
      dis[i,j]:=C[i*N+j+1];
for i:=0 to N-1 do
      begin
    Vmin[i]:=1000; // вектор минимальных расстояний
    Vpom[i]:=0; // 1 - постоянные пометки, 0 - временные пометки
    Vect[i]:=0  // минимальный путь
  end;
Vmin[0]:=0;
Vpom[0]:=1;
Vect[0]:=0;
k:=0;
im:=0;
S1:='';
repeat
  for i:=0 to N-2 do
    for j:=i+1 to N-1 do
      if(dis[i,j]<>0)and(Vpom[j]=0)and(Vmin[j]>dis[i,j]+Vmin[i])then
        begin
          Vmin[j]:=dis[i,j]+Vmin[i];
          Vect[j]:=i;
        end;
  rec:=1000;
  for i:=1 to N-1 do
    if(Vpom[i]=0)and(rec>Vmin[i]) then
      im:=i;
  Vpom[im]:=1;
  k:=k+1;
until(k=N-1);
S:='Минимальный путь из 1-й вершины в вершину '+IntToStr(N)+' :';
rec:=0;
for i:=0 to N-1 do
   Vpom[i]:=0;
jm:=N-1;
for j:=N-1 downto 1 do
   if(j=jm) then
       begin
          jm:=Vect[j];
          Vpom[jm]:=j;
```

```
            end;
      for i:=0 to N-2 do
         if(Vpom[i]<>0)then
            begin
               S1:=S1+'('+FloatToStr(i+1)+','+FloatToStr(Vpom[i]+1)+'),';
               rec:=rec+dis[i,Vpom[i]];
            end;
          Result:=PChar(S+S1+' Fmin ='+FloatToStr(rec));
    Vmin :=NIL;
    Vpom:=NIL;
    Vect:=NIL;
    dis:=NIL;
end;
exports MinPath;
```

Данная программа задает функцию `MinPath`, которая имеет два формальных параметра: `N` и `C`. Первый из них определяет значение общего количества вершин в исходном графе задачи, а второй — матрицу весов дуг, которая интерпретируется в виде расстояний между вершинами графа. При этом функция возвращает значение типа нуль-терминальной строки, который задается ключевым словом `PChar`. Этот тип позволяет оперировать строками текста при межпрограммном взаимодействии в среде MS Windows.

Текст программы этой функции начинается с помощью стандартного оператора языка Pascal — `Function`, предназначенного для объявления функции, после которого указывается имя функции `MinPath` и ее формальные параметры в скобках. После имени функции следуют два ключевых слова: `export` — для указания того факта, что данная функция может быть использована другими приложениями, и ключевое слово `stdcall` — для указания стандартного способа ее вызова.

Текст функции начинается с объявления типов переменных, при этом все массивы объявлены как динамические. Это позволяет добиться независимости программы от структуры и размерности исходного графа. При этом следует учесть некоторые дополнительные особенности передачи массивов в программу из рабочего листа MS Excel.

Во-первых, объявленные динамические массивы имеют нумерацию своих элементов в пределах от 0 до N – 1, где N — количество вершин исходного графа. Это требует особого внимания, поскольку отличается от традиционно используемой индексации массивов в языке Pascal.

Во-вторых, используемый в качестве параметра функции массив C представляет собой одномерный массив, размер которого равен N^2 и индексация элементов которого начинается с 1. Для преобразования его в двумерный массив расстояний `dis(N, N)` служит двойной цикл в программе после операторов переопределения размерности динамических массивов.

Собственно вычисление значения функции нахождения минимального пути в графе между вершиной с номером 1 и вершиной с номером N реализуется с помощью цикла `Repeat-Until` с постусловием. Данный блок функции реализует рассмотренный ранее алгоритм пометок Дейкстры, при этом назначение отдельных ее операторов прокомментировано в тексте самой программы.

Последний оператор `Export MinPath` служит для указания имени для обращения к данной функции из других приложений, включая программу MS Excel. Внешний вид редактора Delphi с текстом программы данной функции в файле с именем MyDLLDelphi.dpr изображен на рис. 12.23.

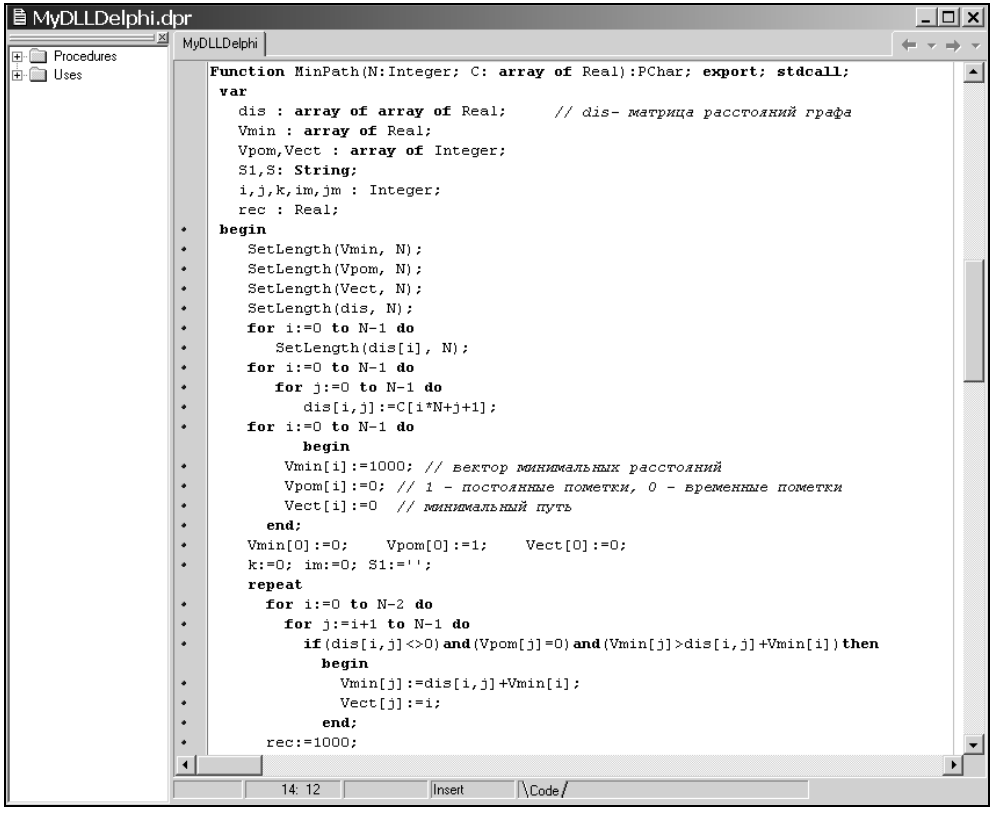


Рис. 12.23. Редактор Delphi с текстом программы функции нахождения минимального пути в графе

Примечание

Следует заметить, что в начало модуля с текстом программы после директивы `uses` следует вставить имя модуля `ShareMem`, как это настоятельно рекомендуется в тексте пространного комментария на английском (см. рис. 12.12). Этот комментарий помещается в текст автоматически после заголовка библиотеки при создании проекта мастером.

После записи текста программы функции в редакторе Delphi необходимо выполнить компиляцию и сборку проекта. С целью выявления ошибок в тексте программы целесообразно предварительно выполнить компиляцию проекта, для чего следует выполнить следующую операцию главного меню редактора Delphi: **Project** (Проект) | **Compile MyDLLDelphi** (Компилировать MyDLLDelphi). При этом в случае наличия ошибок, они будут выявлены, а информация о них будет отображена в специальном окне редактора Delphi.

После устранения возможных ошибок нужно выполнить сборку проекта, для чего следует выполнить следующую операцию главного меню редактора Delphi: **Project** (Проект) | **Build MyDLLDelphi** (Создать MyDLLDelphi). В результате этого в папке с именами файлов данного проекта будет создан файл с именем MyDLLDelphi.dll, который представляет собой бинарный файл с исполнимым кодом вычисления значений функции нахождения минимального пути в графе.

Для использования данной функции в среде MS Excel необходимо скопировать файл MyDLLDelphi.dll в системную папку операционной системы MS Windows. Такой папкой является папка с именем system, которая имеется в папке с именем Windows на том диске, на который устанавливалась ОС. После выполнения этих действий функция `MinPath` становится доступной для использования в качестве формулы в ячейках рабочего листа программы MS Excel. Никаких дополнительных действий по настройке программы MS Excel не требуется.

Функция `MinPath` может быть непосредственно вызвана из ячеек любого рабочего листа данной книги. При этом следует использовать специальный формат вызова данной формулы из ячеек программы MS Excel. С этой целью, выделив нужную ячейку, необходимо вызвать мастер функций, в котором следует выбрать функцию: `ВЫЗВАТЬ`. Напомним, что эта функция предназначена для вызова функций из внешних библиотек.

Применительно к рассматриваемой функции нахождения минимального пути в графе соответствующая формула должна иметь следующий вид:

```
=ВЫЗВАТЬ("MyDLLDelphi"; "MinPath"; "СIK"; ЯЧЕЙКА_1; ДИАПАЗОН_ЯЧЕЕК_2),
```

где в ячейке `ЯЧЕЙКА_1` должно быть задано значение количества вершин исходного графа, а `ДИАПАЗОН_ЯЧЕЕК_2` определяет адреса ячеек со значениями

матрицы весов дуг исходного графа. Следует отметить, что в качестве типа данных возвращаемого значения используется строковый тип, которому соответствует символ `C`. В качестве типа первого параметра используется целочисленный тип данных, которому соответствует символ `I`. В качестве типа второго параметра используется массив данных, которому соответствует символ `K`.

Для вызова разработанной функции нахождения минимального пути в графе воспользуемся книгой с именем Внешняя функция DLL, в которой имеется рабочий лист с именем Минимальный путь. В данном листе содержатся все необходимые исходные данные для выполнения соответствующих расчетов (см. рис. 11.15). Дополнительно к этому следует выполнить следующие действия:

1. В ячейку **I11** введем число 8, которое определяет общее количество вершин исходного графа.
2. В некоторую ячейку, например, в **A12**, введем формулу: `=ВЫЗВАТЬ("MyDLLDelphi"; "MinPath"; "CIK";"I11"; "B3:I10")`.

После ввода формулы в ячейку **A12** рабочего листа будет выполнен расчет значения функции нахождения минимального пути в графе для соответствующих значений аргументов. Результат использования функции нахождения минимального пути в графе будет иметь следующий вид (рис. 12.24).

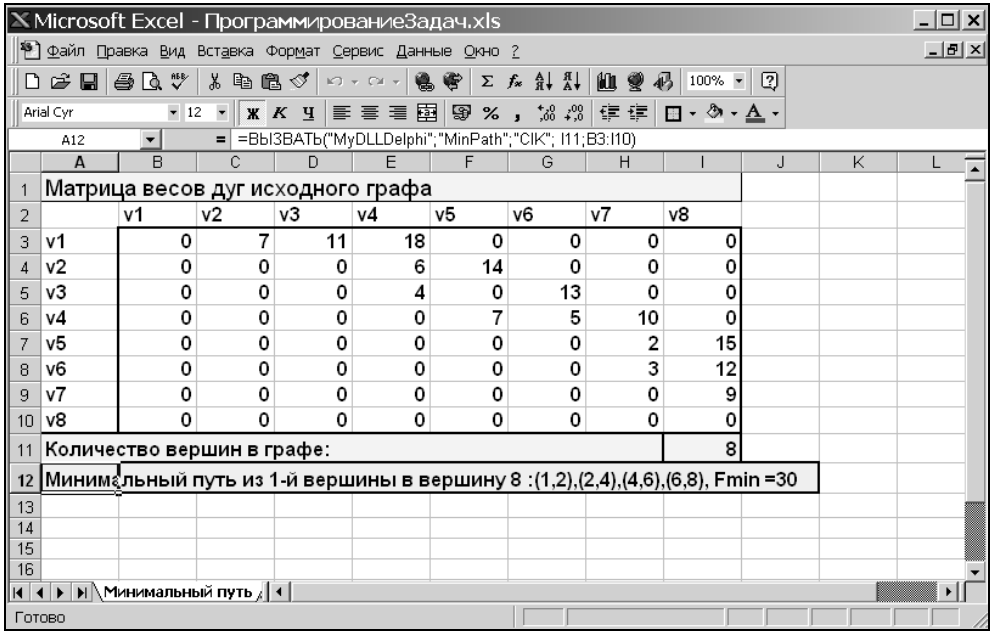


Рис. 12.24. Результат использования функции нахождения минимального пути в графе из библиотеки, созданной в среде Delphi

Как можно заметить, полученный результат вычисления функции нахождения минимального пути в графе из библиотеки, созданной в среде Delphi, полностью совпадают с результатами решения соответствующей задачи оптимизации, полученными с помощью программы на VBA (см. рис. 11.4). Данный факт свидетельствует в пользу правильности вычислительных расчетов, выполненных программой MS Excel с помощью функции из библиотеки DLL, созданной в среде Delphi.

12.5. Упражнения

В качестве упражнений для самостоятельного решения с помощью программирования на VBA, Pascal Delphi и MS Visual C++ предлагаются несколько задач, расширяющих функциональность программы MS Excel.

12.5.1. Модификация программы приближенного решения задачи коммивояжера

Для задачи коммивояжера модифицировать программу на VBA, которая реализует алгоритм локальной оптимизации и позволяет задавать исходное решение случайным образом. Решить несколько индивидуальных задач коммивояжера с помощью двух программ нахождения приближенного решения и сравнить полученные результаты решения. Сделать выводы об эффективности этих программ. Представить данную программу в качестве пользовательской функции и сохранить в качестве надстройки MS Excel.

12.5.2. Модификация программы приближенного решения задачи о разбиении

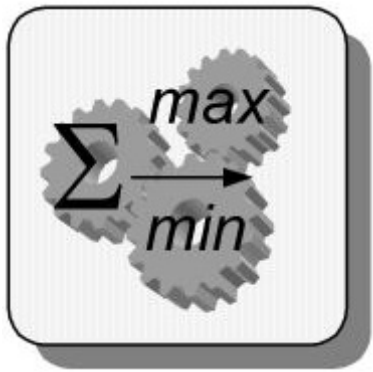
Для задачи о разбиении модифицировать программу на VBA, которая реализует алгоритм локальной оптимизации и позволяет задавать исходное решение случайным образом. Решить несколько индивидуальных задач о разбиении с помощью двух программ нахождения приближенного решения и сравнить полученные результаты решения. Сделать выводы об эффективности этих программ. Представить данную программу в качестве пользовательской функции и сохранить в качестве надстройки MS Excel.

12.5.3. Разработка программы нахождения максимального покрывающего дерева

Для задачи о максимальном покрывающем дереве в графе написать программу на Pascal Delphi или MS Visual C++, которая реализует жадный алгоритм и позволяет найти покрывающее дерево максимальной длины для произвольного неориентированного графа. Представить данную программу в качестве пользовательской функции, сохранить в виде библиотеки динамической компоновки (DLL) и протестировать ее использование в программе MS Excel посредством решения нескольких индивидуальных задач нахождения максимального покрывающего дерева.

12.5.4. Разработка программы нахождения критического пути

Для задачи о критическом пути в сетевом графе написать программу на Pascal Delphi или MS Visual C++, которая реализует алгоритм постоянных пометок и позволяет найти критический путь для произвольного сетевого графа. Представить данную программу в качестве пользовательской функции, сохранить в виде библиотеки динамической компоновки (DLL) и протестировать ее использование в программе MS Excel посредством решения нескольких индивидуальных задач нахождения критического пути для различных сетевых графов.

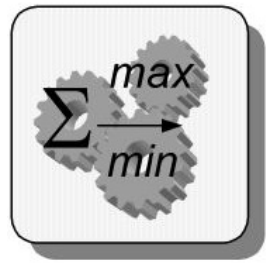


Приложения

Приложение 1. Основные понятия теории множеств, теории графов и комбинаторного анализа

Приложение 2. Назначение операций главного меню программы электронных таблиц MS Office Excel 2003

Приложение 3. Назначение операций главного меню редактора Visual Basic пакета MS Office System 2003



Приложение 1

Основные понятия теории множеств, теории графов и комбинаторного анализа

Основы классической теории множеств как специального раздела математики были разработаны немецким математиком Георгом Кантором (1845—1918) в серии его работ в период 1871—1883 гг. Поскольку теория графов использует понятия обычных множеств и отношений, в настоящем приложении рассматриваются основные теоретико-множественные понятия и обозначения, необходимые для понимания соответствующих конструкций теории графов. В завершение приводятся основные понятия комбинаторного анализа, которые используются при анализе и решении задач комбинаторной оптимизации. При этом следует отметить, что сама теоретико-множественная терминология имеет непосредственное отношение не только к современным концепциям естествознания и методологии математических исследований, но и к адекватному описанию базовых концепций системного моделирования.

Множество и способы его задания

Исходным понятием теории множеств является само понятие *множество*, под которым принято понимать некоторую совокупность объектов, хорошо различимых нашей мыслью или интуицией. При этом не делается никаких априорных предположений ни о природе этих объектов, ни о способе их включения в данную совокупность. Отдельные объекты, составляющие то или иное множество, называют *элементами* данного множества. В классической теории множеств природа элементов, из которых состоят множества, не имеет принципиального значения. Предметом данной теории является изучение таких свойств множеств, которые не зависят от природы составляющих их элементов.

Для записи множеств, их свойств и операций используются специальные обозначения. В общем случае сами множества принято обозначать прописными буквами латинского алфавита. При этом отдельные элементы множества, которые могут иметь различную природу, обозначаются строчными буквами, иногда с индексами, которые хотя и вносят некоторую упорядоченность в последовательность рассмотрения этих элементов, но не являются необходимым атрибутом их задания. Важно понимать, что какой бы то ни было порядок, вообще говоря, не входит в исходное определение множества, а может быть установлен на основе аксиоматизации свойств отношений.

Принято называть элементы отдельного множества *принадлежащими* данному множеству. Данный факт записывается с использованием специального символа "$\in$", который так и называется — *символом принадлежности*. Например, запись $a_i \in A$ означает тот факт, что отдельный элемент с индексом или номером i принадлежит множеству A. Если через a_{110} обозначить квартиру с номером 110, то применительно к данному примеру она не входит во множество A, или, говоря более строго, *не* принадлежит этому множеству. Формально это записывается как $a_{110} \notin A$, где символ "$\notin$" означает отрицание принадлежности элемента множеству. По определению кроме указанных квартир в множество A не входят никакие другие объекты. Например, такие элементы, как *дача* и *автомобиль*, по определению не принадлежат множеству A.

Как следствие сказанного, с любым множеством можно связать или ассоциировать некоторую функцию, которая принимает значение 1 для каждого из элементов данного множества, и значение 0 — для всех остальных элементов, не входящих в рассматриваемое множество. Такая функция получила название *характеристической* функции множества. С использованием специальных обозначений характеристическая функция множества A, обозначаемая через χ_A и формально заданная на некотором универсуме $\boldsymbol{X}$, определяется следующим образом:

$$\chi_A(x) = \begin{cases} 1, & \text{если } x \in A \\ 0, & \text{если } x \notin A \end{cases} \quad (\text{для любого } x \in \boldsymbol{X}) \qquad (\text{П1.1})$$

Можно сказать, что задание любого конкретного множества *неявно* определяет соответствующую характеристическую функцию и наоборот. Обычно в классической теории множеств характеристическая функция χ редко используется и играет второстепенную роль. Однако в теории нечетких множеств обобщение этой функции имеет фундаментальное значение, поскольку именно с этой конструкцией связано базовое определение нечеткого множества.

Между множествами могут иметь место различные отношения, простейшим из которых является *равенство* двух множеств. Два множества считаются

равными, если они состоят из одних и тех же элементов. При этом порядок следования элементов в этих множествах не имеет значения. Формально равенство двух множеств можно записать с помощью характеристических функций этих множеств. А именно множество A равно множеству B (записывается как $A = B$) тогда и только тогда, когда их характеристические функции равны, т. е. выполняется следующее условие:

$$\chi_A(x) = \chi_B(x) \qquad \text{(для любого } x \in \boldsymbol{X}\text{)}, \qquad \text{(П1.2)}$$

где $\chi_A(x)$, $\chi_B(x)$ — характеристические функции множеств A и B соответственно.

Для иллюстрации различных теоретико-множественных операций традиционно используются не характеристические функции, а специальные графические конструкции — *диаграммы Венна*. Последние получили свое название в честь английского логика Джона Венна (1834—1923), который предложил эти обозначения для наглядной интерпретации множеств. Тот факт, что некоторая совокупность элементов образует множество, на диаграмме Венна обозначается графически в форме круга. В этом случае окружность приобретает содержательный смысл или, выражаясь более точным языком, семантику *границы* данного множества. В этом случае равенство двух множеств можно изобразить графически следующим образом (рис. П1.1).

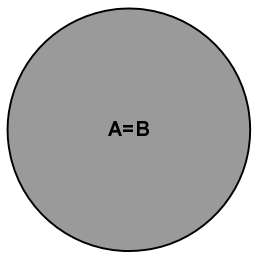


Рис. П1.1. Диаграмма Венна для равенства двух множеств A и B

Следующим важным понятием, которое служит прототипом для многих других терминов системного и нечеткого моделирования, является понятие *подмножества*. Интуитивно ситуация достаточно прозрачна. Если есть некоторая совокупность, рассматриваемая как множество, то любая ее часть и будет являться подмножеством этого множества. Так, например, совокупность квартир на первом этаже жилого дома есть не что иное, как подмножество множества квартир рассматриваемого нами примера. Для обозначения подмножества используется специальный символ "$\subset$". Если утверждается, что множество A является подмножеством множества B, то это записывается как: $A \subset B$.

Примечание

Строго говоря, следует различать два различных варианта подмножества. Рассмотренное ранее определение характерно для так называемого *собственного* подмножества, когда исключается случай возможного равенства двух множеств $A = B$. Если же при определении подмножества допускается равенство двух множеств, то оно называется *несобственным* подмножеством и имеет обозначение $A \subseteq B$.

Формально это отношение можно записать и с помощью характеристических функций этих множеств. А именно множество A является подмножеством (собственным подмножеством) множества B тогда и только тогда, когда справедливо следующее неравенство:

$$\chi_A(x) \le \chi_B(x), \text{ соответственно, } \chi_A(x)<\chi_B(x), \text{ для любого } x \in \boldsymbol{X} \qquad (П1.3)$$

Здесь $\chi_A(x)$, $\chi_B(x)$ — характеристические функции множеств A и B соответственно.

Подмножество или факт включения элементов одного множества в другое множество можно изобразить графически следующим образом (рис. П1.2). На этом рисунке бóльшему множеству B соответствует внешний круг, а меньшему множеству (подмножеству) A — внутренний круг.

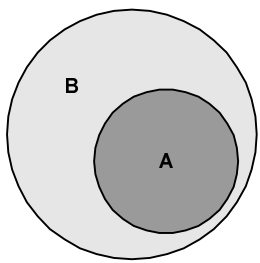


Рис. П1.2. Диаграмма Венна для строгого включения двух множеств $A \subset B$.

В теории множеств некоторые специальные множества играют особую роль. Одним из таких множеств является так называемое *пустое* множество или множество, которое не содержит ни одного элемента. Это множество имеет специальное обозначение: $\varnothing$. Из этого определения следует, что пустое множество является собственным подмножеством любого множества, не являющегося в свою очередь пустым. То есть для любого множества A всегда справедливо утверждение: $\varnothing \subset A$. Очевидно, что характеристическая функция пустого множества тождественно равна нулю для каких бы то ни было элементов: $\chi_\varnothing = 0$.

Другим специальным множеством является так называемый *универсум* или множество, содержащее все возможные элементы. Это множество также имеет специальное обозначение: $\boldsymbol{X}$. Из определения универсума следует, что любое множество, не являющееся в свою очередь универсумом, является собственным подмножеством универсума. То есть для любого множества A всегда справедливо утверждение: $A \subset \boldsymbol{X}$. Очевидно, что характеристическая функция универсума тождественно равна 1 для каких бы то ни было элементов: $\chi_X = 1$.

В зависимости от количества элементов множества бывают конечные и бесконечные. Множество называется *конечным*, если оно содержит конечное число элементов. Про такое множество еще говорят, что оно имеет конечную *мощность*, которая численно равна количеству элементов этого множества. Для обозначения мощности произвольного множества A используется специальный символ $\boldsymbol{card}(A)$.

Примечание

В литературе для обозначения мощности множества A также используются и другие символы, как, например: $|A|$ или $\overline{\overline{A}}$. Последний из символов может ввести в заблуждение, поскольку, как будет видно из последующего изложения, совпадает по внешнему виду с символом двойного дополнения. По этой причине в дальнейшем мы не будем его использовать.

Возвращаясь к примеру с множеством квартир жилого дома, можно сказать, что его мощность равна 100. Множество всех возможных комбинаций из 8 символов, которые могут служить для ввода некоторого пароля с клавиатуры компьютера, конечное, хотя и достаточно большое. Или, говоря строгим языком, это множество имеет конечную мощность.

Ситуация усложняется, когда рассматриваются *бесконечные* множества, т. е. множества, не являющиеся конечными. Эта сложность связана с тем, что бесконечные множества могут быть счетными и несчетными. *Счетным* множеством принято называть множество, содержащее бесконечное число элементов, которые, однако, можно перенумеровать натуральными числами 1, 2, 3 и т. д. При этом важно иметь в виду, что достичь последнего элемента при такой нумерации принципиально невозможно, иначе множество окажется конечным. Про такие множества говорят, что они имеют *счетную мощность*, которая обозначается $\aleph_0$ (читается — *алеф нуль*). Есть все основания считать множество всех звезд бесконечным, хотя многие из звезд имеют свое уникальное название. Именно по этой причине это множество может быть отнесено к категории счетных.

Несчетным множеством принято называть множество, содержащее бесконечное число элементов, которые принципиально нельзя перенумеровать натуральными числами. Г. Кантор доказал, что множество всех действительных чисел несчетно. В этой связи иногда говорят, что данное множество имеет несчетную мощность или мощность *континуума* (обозначается символами $\boldsymbol{c}$ или $\aleph$).

Множества могут быть заданы двумя основными способами:

1. Явным *перечислением* или указанием всех элементов, входящих в рассматриваемое множество. Очевидно, что этот способ подходит только для задания конечных множеств с небольшим числом элементов. Например, множество из пяти первых натуральных чисел или множество планет Солнечной системы.
2. Указанием некоторого *свойства*, которым обладает каждый из элементов рассматриваемого множества и не обладает всякий другой элемент, не входящий в рассматриваемое множество. Этот способ может быть использован для задания как конечных, так, что наиболее важно, и бесконечных множеств. Например, только таким способом можно определить множество всех четных натуральных чисел или множество действительных чисел, строго больших 10.

В первом случае множество символически записывается в виде: $A = \{a_1, a_2, ..., a_n\}$, где n — общее число элементов множества A. Другими словами, множество A имеет конечную мощность $n = \boldsymbol{card}(A)$.

Во втором случае вводят в рассмотрение некоторое характеристическое свойство, которое может быть записано в виде одноместного *предиката* $P(x)$. Для формальной строгости предикат $P(x)$ определяется на универсуме $\boldsymbol{X}$ элементов, из которых формируется множество A. Предполагается, что в качестве универсума $\boldsymbol{X}$ в каждом конкретном случае используется такая совокупность объектов, для которых рассматриваемое свойство имеет содержательный смысл или, другими словами, означивание соответствующего предиката семантически корректно. При этом предикат может принимать одно из двух значений истинности: *"истина"* или *"ложь"*. Если элемент $x \in \boldsymbol{X}$ обладает рассматриваемым свойством, то соответствующий предикат $P(x)$ принимает значение *"истина"*. Если же элемент $x \in \boldsymbol{X}$ не обладает рассматриваемым свойством, то соответствующий предикат $P(x)$ принимает значение *"ложь"*. Тогда в общем случае задание множества A с использованием специального свойства, выраженного в форме одноместного предиката $P(x)$, может быть записано в виде: $A = \{a \mid P(a),\ a \in \boldsymbol{X}\}$.

Например, множество неотрицательных действительных чисел, которое имеет специальное обозначение $\boldsymbol{R}_+$ или $[0, +\infty)$, может быть задано в виде: $\boldsymbol{R}_+ = \{x \mid x \in \boldsymbol{R},\ x \geq 0\}$. В этом случае предикат $P(x)$ определяется в форме неко-

торого составного высказывания *"x является действительным числом и, одновременно, x больше или равно* 0", которое допускает подстановку вместо x любого числа (т. е. $\boldsymbol{X} = \boldsymbol{R}$, где $\boldsymbol{R}$ — множество всех действительных чисел). Другим интересным множеством, которое также может быть задано вторым способом, является *замкнутый интервал* действительных чисел, заключенных между 0 и 1, включая сами эти числа. Этот интервал, который традиционно обозначается как I_R или [0, 1], определяется следующим образом: $I_R = \{x | \ x \in \boldsymbol{R}, \ 0 \le x \le 1\}$.

Примечание

В связи с рассмотрением последнего примера следует обратить внимание на различие множеств {0, 1} и [0, 1]. Первое из них является конечным множеством, которое состоит только из двух элементов: чисел 0 и 1. Второе является бесконечным множеством и, более того, имеет мощность континуума. Тем не менее, справедливо включение: $\{0, 1\} \subset [0, 1]$.

Как нетрудно заметить, в общем случае для произвольного множества A существует формальная взаимосвязь между характеристической функцией множества $\chi_A(x)$ и некоторым предикатом $P_A(x)$, который, возможно неявно, выражает некоторое характеристическое свойство этого множества. А именно если элемент $a \in \boldsymbol{X}$ обладает свойством $P_A(x)$, т. е. $P_A(a)$ — *"истинно"*, то $\chi_A(a) = 1$. Если же элемент $a \in \boldsymbol{X}$ не обладает свойством $P_A(x)$, т. е. $P_A(a)$ — *"ложно"*, то $\chi_A(a) = 0$. Верно и обратное заключение. Эта связь характеристической функции множества и соответствующего ей предиката играет важную роль при установлении взаимосвязей между теорией нечетких множеств и нечеткой логикой.

Основные теоретико-множественные операции

Пусть A и B — произвольные (конечные или бесконечные) множества. *Пересечением* двух множеств A и B называется некоторое третье множество C, которое состоит из тех и только тех элементов двух исходных множеств, которые *одновременно* принадлежат и множеству A, и множеству B. Для этой операции имеется специальное обозначение: "$\cap$". Тогда результат операции пересечения двух множеств можно записать в виде: $C = A \cap B$, где $C = \{x | x \in A$ *и одновременно* $x \in B\}$.

Примечание

При этом важно обратить внимание на тот факт, что характеристическая функция χ_C множества $C = A \cap B$ зависит некоторым образом от характеристических функций исходных множеств χ_A и χ_B. А именно значение функции $\chi_C(x)$ для любого $x \in \boldsymbol{X}$ может быть получено как минимальное из двух значений: $\chi_A(x)$ и $\chi_B(x)$. Это свойство может быть положено в основу определения операции пересечения множеств. Формально же оно записывается в следующем виде:

$$\chi_C(x) = min\{\chi_A(x), \chi_B(x)\} \qquad \text{(для любого } x \in \boldsymbol{X}) \qquad \text{(П1.4)}$$

Например, если в качестве множества A рассмотреть множество сотрудников некоторой фирмы, а в качестве множества B — множество всех мужчин, то нетрудно догадаться, что множество C как результат операции пересечения $A \cap B$ будет состоять из элементов — всех сотрудников мужского пола данной фирмы. Операция пересечения множеств может быть проиллюстрирована с использованием диаграмм Венна (рис. П1.3). На этом рисунке условно изображены два множества A и B, затемненной области как раз и соответствует множество C, являющееся пересечением множеств A и B.

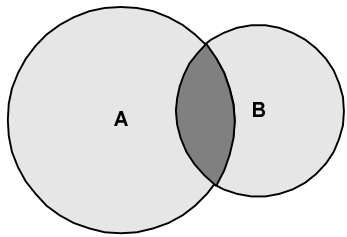


Рис. П1.3. Диаграмма Венна для пересечения двух множеств A и B

Под *объединением* двух множеств A и B понимается некоторое третье множество, которое обозначим как D, состоящее из тех и только тех элементов, которые принадлежат или A, или B, или им обоим одновременно. Для этой операции также существует специальное обозначение: $D = A \cup B$, где $D = \{x | x \in A \text{ или } x \in B\}$.

Примечание

При этом важно заметить, что характеристическая функция χ_D множества $D = A \cup B$ также зависит некоторым образом от характеристических функций исходных множеств χ_A и χ_B. А именно значение функции $\chi_D(x)$ для любого $x \in \boldsymbol{X}$ может быть получено как максимальное из двух значений: $\chi_A(x)$ и $\chi_B(x)$. Это свойство может быть положено в основу определения операции объединения множеств. Формально же оно записывается в следующем виде:

$$\chi_D(x) = max\{\chi_A(x), \chi_B(x)\} \text{ для любого } x \in \boldsymbol{X}. \qquad \text{(П1.5)}$$

Так, например, если в качестве множества A рассмотреть множество, состоящее из клавиатуры и мыши, а в качестве множества B — множество, состоящее из системного блока и монитора, то нетрудно догадаться, что их объединение, т. е. множество $D = A \cup B$ будет сдержать все основные компоненты персонального компьютера. Операция объединения множеств также может быть проиллюстрирована с использованием диаграмм Венна (рис. П1.4). На этом рисунке объединению двух исходных множеств также соответствует затемненная область, только размеры и форма ее отличаются от случая пересечения двух множеств.

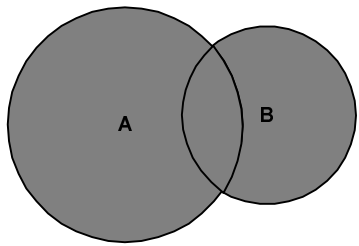


Рис. П1.4. Диаграмма Венна для объединения двух множеств A и B

Разностью двух множеств A и B называется некоторое третье множество E (обозначается $E = A \backslash B$, которое состоит из тех и только тех элементов, которые принадлежат множеству A и не принадлежат множеству B, т. е. $E = \{x | x \in A$ *и одновременно* $x \notin B\}$. Если множества A и B не пересекаются (т. е. $A \cap B = \varnothing$), то $A \backslash B = A$. Из определения этой операции также следует, что: $A \backslash A = \varnothing$ и $A \backslash \varnothing = A$. Формально справедливо следующее утверждение:

$$\chi_E(x) = max\{\chi_A(x) - \chi_B(x), 0\} \text{ для всех } x \in \boldsymbol{X}, \qquad (П1.6)$$

где под знаком максимума применяется обычная операция арифметической разности двух чисел. Это свойство, в свою очередь, может быть использовано для исходного определения операции разности множеств.

Так, например, если в качестве множества A рассмотреть множество, состоящее из студентов некоторой группы, а в качестве множества B — множество студентов, получивших неудовлетворительные оценки на сессии или не аттестованных вовсе, то их теоретико-множественная разность $E = A \backslash B$ — множество студентов группы, успешно сдавших сессию. Операция разности множеств также может быть проиллюстрирована с использованием диаграмм Венна (рис. П1.5). На этом рисунке разности двух исходных множеств также соответствует затемненная область.

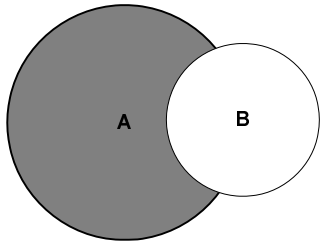


Рис. П1.5. Диаграмма Венна для разности двух множеств A и B

Следует заметить, что операция разности множеств в отличие от операций объединения и пересечения не является симметричной, т. е. в общем случае $A\backslash B \neq B\backslash A$. Поэтому иногда удобно рассматривать так называемую *симметрическую разность* двух множеств A и B (обозначается через $A\Delta B$). По определению $A\Delta B = (A\backslash B)\cup(B\backslash A)$, т. е. симметрическая разность двух множеств представляет собой объединение двух разностей множеств A и B. При этом оказывается справедливым следующее утверждение:

$$\chi_{A\Delta B}(x) = max\{|\chi_A(x) - \chi_B(x)|, 0\} \text{ для всех } x \in \boldsymbol{X}, \qquad (П1.7)$$

где под знаком максимума применяется операция модуля (или вычисления абсолютного значения) числа. Это свойство также может быть использовано для исходного определения операции симметрической разности множеств.

Операция симметрической разности множеств иллюстрируется с помощью диаграмм Венна следующим образом (рис. П1.6). На этом рисунке симметрической разности двух исходных множеств соответствует затемненная область. Можно показать, что для операции симметрической разности выполняется следующее тождество: $A\Delta B = (A\cup B)\backslash(A\cap B)$.

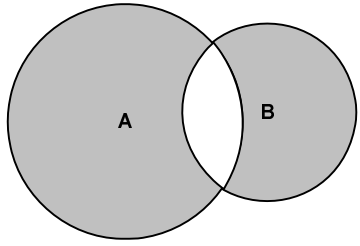


Рис. П1.6. Диаграмма Венна для симметрической разности двух множеств A и B

В целом ряде случаев оказывается полезной унарная операция *дополнения* множества. Дополнение множества A обозначается через $\overline{A}$ и определяется следующим образом: $\overline{A} = \{x | \, x \in \boldsymbol{X}$ *и одновременно* $x \notin A\}$ или $\overline{A} = \boldsymbol{X} \setminus A$,

где множество $\boldsymbol{X}$ — универсум. Легко проверяется следующее свойство характеристической функции для дополнения множества:

$$\chi_{\bar{A}} = 1 - \chi_{\bar{A}} \text{ для всех } x \in \boldsymbol{X}. \qquad (П1.8)$$

Операция дополнения множества иллюстрируется с помощью диаграмм Венна следующим образом (рис. П1.7). На этом рисунке дополнению множества A соответствует затемненная область.

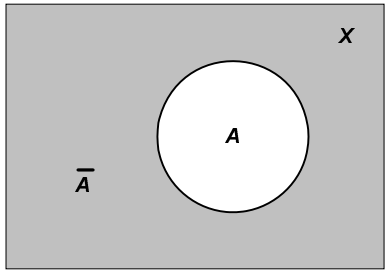


Рис. П1.7. Диаграмма Венна для дополнения множества A

Пример П1.1. Рассмотрим два числовых множества: $A = \{1, 3, 5, 7, 8, 9\}$ и $B = \{2, 4, 6, 7, 8, 10\}$. Для этих множеств результаты выполнения рассмотренных выше теоретико-множественных операций будут следующими: $A \cap B = \{7, 8\}$, $A \cup B = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, $A \setminus B = \{1, 3, 5, 9\}$, $A \Delta B = \{1, 2, 3, 4, 5, 6, 9, 10\}$. Если в качестве универсума $\boldsymbol{X}$ взять множество натуральных чисел $\boldsymbol{N}$, то $\bar{A} = \{2, 4, 6, 10, 11, 12, 13, ...\}$ и $\bar{B} = \{1, 3, 5, 9, 11, 12, 13, \ldots\}$.

Для теоретико-множественных операций имеют место следующие свойства, часть из которых непосредственно следует из их определения, а доказательство других может служить в качестве упражнения:

- *Коммутативность* операций объединения и пересечения:

$$A \cup B = B \cup A; \qquad A \cap B = B \cap A \qquad (П1.9)$$

- *Ассоциативность* операций объединения и пересечения:

$$A \cup (B \cup C) = (A \cup B) \cup C; \qquad A \cap (B \cap C) = (A \cap B) \cap C \qquad (П1.10)$$

- *Дистрибутивность* операций объединения и пересечения относительно друг друга:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C); \quad A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \qquad (П1.11)$$

- *Идемпотентность* операций объединения и пересечения:

$$A \cup A = A; \qquad A \cap A = A \qquad (П1.12)$$

- *Поглощение* пустого множества и универсума при операциях объединения и пересечения соответственно:

$$A \cup \varnothing = A; \qquad A \cap \boldsymbol{X} = A \tag{П1.13}$$

- Универсальные границы (верхняя и нижняя):

$$A \cup \boldsymbol{X} = \boldsymbol{X}; \qquad A \cap \varnothing = \varnothing \tag{П1.14}$$

- *Дополняемость* (или дополнительность):

$$A \cap \overline{A} = \varnothing; \qquad A \cup \overline{A} = \boldsymbol{X} \tag{П1.15}$$

- *Инволюция* или двойное дополнение:

$$\overline{\overline{A}} = A \tag{П1.16}$$

- Законы де Моргана:

$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}; \qquad \overline{(A \cap B)} = \overline{A} \cup \overline{B} \tag{П1.17}$$

Перечисленные свойства получили название аксиом *булевой алгебры* в честь английского логика Дж. Буля (1815—1864), отмечая тем самым его вклад в разработку основ математической логики.

Формула (П1.9) означает, что теоретико-множественные операции объединения и пересечения конечного числа множеств не зависят от порядка следования отдельных множеств при выполнении этих операций. Формула (П1.10) означает, что эти операции не зависят от того, выполняем ли мы их в некоторой очередности или разбиваем на отдельные группы с последующим выполнением соответствующих операций над этими группами. Первый из законов дистрибутивности (П1.11) полностью аналогичен закону дистрибутивности операции умножения относительно сложения в арифметике. Как и в арифметике, из этого закона следует, что для того чтобы найти пересечение двух групп множеств, каждая из которых представляет собой объединение конечного числа множеств, следует найти пересечение каждого из множеств первой группы с каждым из множеств второй группы, а к полученным результатам применить операцию объединения. Например, $(A \cup B \cup C \cup D) \cap (E \cup F \cup G) = (A \cap E) \cup (A \cap F) \cup (A \cap G) \cup (B \cap E) \cup \ldots \cup (D \cap E) \cup (D \cap F) \cup (D \cap G)$. Идемпотентность операций объединения и пересечения не имеет аналогии в арифметике, поскольку в результате их применения к одному и тому же множеству всегда получается исходное множество. Свойства поглощения и универсальных границ аналогичны наличию единицы для операции умножения и нуля для операции сложения в арифметике (кроме первой формулы (П1.14), которая не имеет аналогии).

Первая формула дополняемости (П1.15) имеет аналогию в формальной логике в виде так называемого закона *противоречия*. Этот закон утверждает, что

в процессе рассуждения о каком-либо определенном предмете или явлении нельзя одновременно утверждать и отрицать что-либо в одном и том же отношении, в противном случае оба суждения не могут быть одновременно истинными. Действительно, если мы утверждаем, что некоторый элемент a_i принадлежит множеству A, то тем самым мы исключаем противоположное утверждение, что этот же элемент a_i *не* принадлежит множеству A. И наоборот, истинность утверждения "элемент a_j не принадлежит множеству A" влечет ложность противоположного утверждения "элемент a_j принадлежит множеству A". Например, утверждая, что квартира с номером 100 одновременно принадлежит и не принадлежит множеству A рассмотренного ранее примера П1.1, мы неизменно приходим к противоречию, поскольку в данном примере по определению возможен только один случай, а именно: $a_{100} \in A$.

Вторая формула дополняемости (П1.15) также имеет аналогию в формальной логике в виде так называемого закона *исключенного третьего*. Этот закон утверждает, что в процессе рассуждения о каком-либо определенном предмете или явлении необходимо доводить дело до некоторого законченного утверждения или его отрицания. При этом если одно из таких утверждений истинно, то другое должно быть обязательно ложно, и наоборот. Третьего в этом отношении быть не может: либо истина, либо ложь. Действительно, можно сформулировать только *два* утверждения относительно принадлежности того или иного элемента множеству: *"элемент a_i принадлежит множеству A"* и *"элемент a_i не принадлежит множеству A"*. Третьего утверждения в классической теории множеств быть не может. Например, относительно квартиры с номером 100 можно утверждать только следующее: *"квартира a_{100} принадлежит множеству A"* или *"квартира a_{100} не принадлежит множеству A"*. В этом контексте важно понимать, что нельзя сформулировать какое бы то ни было третье утверждение относительно принадлежности данной квартиры рассматриваемому множеству A.

Что касается инволюции или двойного дополнения, то это свойство легко доказывается, если обозначить $\overline{A} = B$. Тогда, если некоторый элемент $a_i \in A$, то тем самым $a_i \notin \overline{A}$ и, следовательно, $a_i \notin B$ или $a_i \in \overline{B}$. Таким образом, $A = \overline{B}$ или, вводя операцию двойного дополнения, $\overline{A} = \overline{\overline{A}}$. В обратную сторону доказательство формулы (П1.16) аналогично. Предположим, что $a_i \in \overline{B}$. Это означает, что $a_i \notin B$ или, с учетом введенных обозначений, $a_i \notin \overline{A}$. Последнее утверждение в свою очередь влечет $a_i \in A$. Поскольку $\overline{B}$ по определению равно $\overline{\overline{A}}$, то тем самым $\overline{\overline{A}} = A$, что и завершает доказательство данного свойства.

В заключение докажем справедливость первого из законов, получивших свое название в честь шотландского математика и логика Огастеса де Моргана

(De Morgan Augustus, 1806—1871): $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$. Для этого обозначим $A \cup B$ через C, т. е. $C = A \cup B$. Предположим, что некоторый элемент $a_i \in \overline{C}$ или $a_i \notin C$. Тем самым справедливо $a_i \notin A$, поскольку $A \subset C$, и одновременно $a_i \notin B$, поскольку также и $B \subset C$. С другой стороны, это означает, что $a_i \in \overline{A}$ и одновременно $a_i \in \overline{B}$. Последнее означает не что иное, как: $a_i \in \overline{A} \cap \overline{B}$. Это доказывает первую часть закона. Теперь докажем рассматриваемое равенство в обратную сторону. Если предположить, что некоторый элемент $a_i \notin A$ и одновременно $a_i \notin B$, то это будет означать, что $a_i \notin A \cup B$. Действительно, если существует такой элемент a_j, для которого справедливо условие: $a_j \in A \cup B$, то тем самым исключается одновременное выполнение условий: $a_j \notin A$ и в то же время $a_j \notin B$. Помня, что $C = A \cup B$, условие $a_i \notin A \cup B$ будет означать $a_i \notin C$, а значит $a_i \in C$. Поскольку $a_i \notin A$ означает $a_i \in \overline{A}$, а $a_i \notin B$ означает $a_i \in \overline{B}$, одновременное выполнение этих условий означает не что иное, как: $a_i \in \overline{A} \cap \overline{B}$. Поскольку мы начали рассуждения с последнего утверждения, а пришли к утверждению $a_i \in \overline{C} = \overline{A \cup B}$, то тем самым получили доказательство первого закона де Моргана в обратную сторону. Второй закон доказывается аналогично и предоставляется выполнить читателю в качестве самостоятельного упражнения.

Булеан или множество всех подмножеств

Как можно заметить, в определении множества присутствует некоторая двойственность, которая касается входящих в множество элементов. С одной стороны, между элементом и множеством имеет место отношение принадлежности $a_i \in A$, которое устанавливает исходный состав множества $A = \{a_1, a_2, \ldots, a_i, \ldots\}$. С другой стороны, каждый из элементов множества A можно рассматривать как одноэлементное подмножество $\{a_i\}$, и тогда имеет место отношение включения между каждым из этих одноэлементных подмножеств и исходным множеством: $\{a_i\} \subset A$. При этом оказывается справедливым следующее утверждение: $\bigcup_i \{a_i\} = A$, где операция объединения выполняется по всем одноэлементным подмножествам, соответствующим элементам множества A.

Именно последний аспект рассмотрения множеств послужил исходной идеей для введения специальной конструкции, которая играет особую роль при определении целого ряда других понятий. Речь идет о множестве всех подмножеств (булеане или степенном множестве), элементами которого по определению являются все без исключения подмножества некоторого фиксированного

множества A. А именно формально *булеан* определяется как $\boldsymbol{B}(A) = \{A_i \mid A_i \subseteq A\}$. По этому определению пустое множество $\varnothing$ и само множество A входят в булеан или являются его элементами. Если исходное множество A конечное, т. е. имеет конечную мощность $n = \boldsymbol{card}(A)$, то булеан также будет являться конечным множеством, а его мощность будет равна $\boldsymbol{card}(\boldsymbol{B}(A)) = 2^n$ (доказывается методом индукции по количеству элементов множества A).

Таким образом, имеет место следующее утверждение: $\boldsymbol{card}(\boldsymbol{B}(A)) = 2^{\boldsymbol{card}(A)}$. По аналогии с конечными множествами это свойство распространяют и на бесконечные множества, допуская при этом некоторую вольность, поскольку показатель степени в этом случае не является числом.

Отношения и способы их задания

Понятие отношения наряду с понятием самого множества является фундаментальным не только в теории множеств, но и в других теоретических и прикладных дисциплинах. Следует помнить, что это понятие часто заменяется терминами *связь*, *ассоциация*, *взаимосвязь* или *соотношение*. В общем случае отношение определяется как любое подмножество упорядоченных кортежей, построенных из элементов некоторых исходных множеств. При этом под *кортежем* понимается набор или список упорядоченных элементов.

Хотя и существует некоторая неоднозначность в принятых обозначениях, кортеж из двух элементов удобно обозначать как $<a_1, a_2>$, из трех элементов — $<a_1, a_2, a_3>$ и т. д. Кортеж из k элементов будем обозначать через $<a_1, a_2, \ldots, a_k>$ и говорить, что он имеет длину k, где k — некоторое натуральное число. При этом отдельные элементы кортежа могут принадлежать как одному множеству, так и различным множествам. Важно иметь в виду, что порядок выбора элементов для построения кортежей строго *фиксирован* для каждого конкретного случая. Речь идет о том, что первый элемент всегда выбирается из первого множества, второй — из второго и т. д.

Совокупность всех кортежей длины 2, образованных из элементов двух произвольных (конечных или бесконечных) множеств A и B, образует множество, которое получило специальное название — *декартово произведение* двух множеств. Данное произведение названо в честь известного французского философа и математика Рене Декарта (1596—1650). Формально декартово произведение определяется как: $A \times B = \{<a_i, b_j> \mid a_i \in A, b_j \in B\}$. Поскольку декартово произведение является множеством в обычном смысле, то для него справедливы все теоретико-множественные свойства и операции. В частности, мощность декартова произведения двух конечных множеств

$A = \{a_1, a_2, \ldots, a_n\}$ и $B = \{b_1, b_2, \ldots, b_m\}$ равна произведению (арифметическому) мощностей этих множеств: $\boldsymbol{card}(A \times B) = n \cdot m$. Этот факт непосредственно следует из простого подсчета всех кортежей, составленных из элементов этих множеств.

Декартово произведение может быть расширено на случай произвольного конечного числа множеств: $A_1 \times A_2 \times \ldots \times A_k$ на основе рассмотрения всех кортежей длины k, составленных из элементов этих множеств. С другой стороны, в качестве множеств декартова произведения может выступать и одно множество A. Соответствующее декартово произведение примет вид: $A \times A$ или $A \times A \times \ldots \times A$ (k *раз*). В последнем случае иногда используется запись: $\prod_{i=1}^{k} A$.

В общем случае *отношением,* заданным на множествах $A_1, A_2, \ldots, A_k$ (иногда говорят — над множествами), называется некоторое подмножество декартова произведения этих множеств. Другими словами, если обозначить произвольное отношение через Q, то по определению $Q \subseteq A_1 \times A_2 \times \ldots \times A_k$. При этом не исключается случай равенства отношения самому декартову произведению множеств.

Чтобы характеризовать количество множеств, на основе которых строится то или иное отношение, принято называть отношение между двумя множествами — *бинарным*, между тремя множествами — *тернарным*, а в общем случае — *k-арным* отношением. В каждом конкретном случае рассматриваемое отношение неявно предопределяет некоторый способ или семантику выбора отдельных элементов из одного или нескольких множеств для образования соответствующего множества кортежей.

Примечание

Поскольку отношение по своей сути является множеством, то каждому отношению будет соответствовать некоторая характеристическая функция этого отношения χ_Q, заданная на декартовом произведении исходных множеств данного отношения. При этом если кортеж $<a_1, a_2, \ldots, a_k>$ входит или принадлежит отношению Q, то значение характеристической функции этого отношения χ_Q для данного кортежа равно 1, и 0 — в противном случае.

Существуют различные способы, которыми могут быть заданы те или иные конкретные отношения. Наибольшее распространение из них получили следующие способы:

1. Непосредственное *перечисление кортежей* отношения. Как и для случая одного множества, этот способ может быть использован только для зада-

ния конечных отношений произвольной арности с небольшим количеством кортежей.

2. *Матричный* способ. Основан на представлении отношения в форме матрицы, строки которой соответствуют одному из множеств отношения, а столбцы — другому множеству. При этом элементы матрицы принимают одно из двух значений: 1 или 0, в зависимости от того, принадлежит или не принадлежит соответствующий кортеж данному отношению. Как нетрудно представить, данный способ может быть использован только для задания конечных бинарных отношений. Иногда определенную таким образом матрицу называют *матрицей бинарного отношения*, которую будем обозначать через $\mathbf{M}_Q$.

3. *Табличный* способ, с одной стороны, может рассматриваться как разновидность матричного, поскольку конечная матрица всегда может быть представлена в форме таблицы. С другой стороны, при представлении отношений (не обязательно бинарных) в форме таблиц удобно указывать только те кортежи, которые принадлежат рассматриваемому отношению. Вообще говоря, таблицы отношений являются центральным объектом теории и практики *реляционных баз данных*. Данная тематика, в рамках которой разработаны специальные средства манипулирования данными (реляционная алгебра, правила нормализации, язык SQL, коммерческие базы данных и пр.) выходит за рамки настоящей книги.

4. *Графический* способ. В общем случае основан на представлении отношения с использованием некоторой графической нотации или системы специальных обозначений. Наиболее распространены две разновидности этого способа, которые подходят для задания бинарных отношений:

   - Представление отношения в виде некоторой геометрической фигуры (кривой или совокупности отдельных точек) на плоскости или в пространстве. При этом могут использоваться различные системы координат. График математической функции такой как, например, парабола $y = x^2$ является примером этого способа задания отношения. Здесь функция рассматривается как частный случай отношения, о чем будет сказано далее. Этот способ зачастую является единственно возможным для визуализации бесконечных отношений.

   - Представление отношения в виде ориентированного (направленного) графа. Хотя основные понятия теории графов получили свое развитие задолго до появления теории множеств как самостоятельной научной дисциплины, формальное определение графа удобно представить в теоретико-множественных терминах. Рассмотрим этот способ более подробно.

Ориентированным графом называется совокупность двух множеств: множества точек или *вершин* и множества соединяющих их линий или дуг. Формально граф задается в виде двух конечных множеств: $G = (V, E)$, где $V = \{v_1, v_2, \ldots, v_n\}$ — множество вершин графа, $E = \{e_1, e_2, \ldots, e_m\}$ — множество дуг графа. Вершины графа изображаются точками, а дуги — отрезками прямых линий со стрелкой на одном из концов. Рядом с вершинами и дугами записываются условные обозначения соответствующих вершин и дуг, что позволяет идентифицировать их однозначным образом. Натуральное число n определяет общее количество вершин конкретного графа, а натуральное число m — общее количество дуг графа (рис. П1.8, *а*).

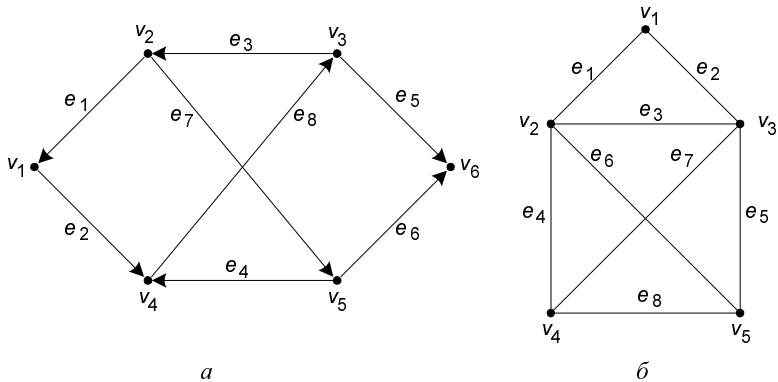


Рис. П1.8. Примеры ориентированного (*а*) и неориентированного (*б*) графов

Следует заметить, что в общем случае не все вершины в графе могут соединяться между собой. Именно этот факт ставит в соответствие каждому ориентированному графу некоторое бинарное отношение Q_G, состоящее из всех пар вида $<v_i, v_j>$, где $v_i, v_j \in V$. При этом пара $<v_i, v_j>$ принадлежит соответствующему отношению Q_G в том и только в том случае, если вершины v_i и v_j соединяются в графе G некоторой дугой $e_k \in E$, направленной из вершины v_i в вершину v_j. При графическом задании отношения $Q \subseteq A \times A$ каждому элементу множества $a_i \in A$ будет соответствовать отдельная вершина $v_i \in G$ графа этого отношения, а каждому кортежу отношения $<a_i, a_j> \in Q$ будет соответствовать дуга графа $<v_i, v_j> \in E$ с началом в вершине v_i и концом в вершине v_j.

Вообще говоря, графы бывают различных типов. Кроме рассмотренных выше ориентированных графов, в теории графов рассматриваются *неориентированные* графы, т. е. такие графы, у которых соединяющие вершины ребра не имеют направления или ориентации. Неориентированные графы удобно считать частным случаем ориентированных, у которых каждая дуга имеет дугу противоположной ориентации (рис. П1.8, *б*).

Примечание

В современной теории графов рассматриваются также и *бесконечные* графы, т. е. графы с бесконечным числом вершин. Однако в задачах оптимизации на графах бесконечные графы не рассматриваются. Именно поэтому графический способ задания отношений используется только для конечных отношений.

В общем случае графы широко используются для представления информации о структуре систем и процессов. Примерами подобных графических моделей могут служить: схемы автомобильных дорог региона, которые соединяют отдельные населенные пункты; схемы телекоммуникаций, используемых для передачи информации между отдельными узлами системы; схемы программ, на которых указываются варианты ветвления вычислительного процесса. Общим для всех подобных моделей является возможность представления информации в графическом виде. Достоинством данного способа является наглядность представления отношений и возможность их визуализации. При этом отдельные модели могут обладать дополнительной семантикой и специальными обозначениями, характерными для той или иной предметной области.

Важными понятиями теории графов являются понятия маршрута и пути, которые ассоциируются с последовательным перемещением от вершины к вершине по соединяющим их ребрам или дугам. Для неориентированного графа *маршрут* определяется как конечная или бесконечная упорядоченная последовательность ребер $S = <..., e_{s1}, e_{s2}, ..., e_{sk}, ...>$, таких, что каждые два соседних ребра имеют общую вершину. Нас будут интересовать только конечные маршруты $S = <e_{s1}, e_{s2}, ..., e_{sk}>$, т. е. такие маршруты, которые состоят из конечного числа ребер. При этом ребро e_{s1} принято считать началом маршрута S, а ребро e_{sk} — концом маршрута S. Для ориентированного графа соответствующая последовательность дуг $S = <e_{s1}, e_{s2}, ..., e_{sk}>$ называется *ориентированным маршрутом*, если две соседние дуги имеют общую вершину, которая является концом предыдущей и началом последующей дуги.

Примерами ориентированных маршрутов для графа (рис. П1.8, *а*) являются такие последовательности дуг: $S_1 = <e_2, e_8, e_5>$, $S_2 = <e_3, e_7, e_6>$, $S_3 = <e_8, e_3, e_7, e_4, e_8>$. Если в ориентированном маршруте не повторяются ни ребра, ни вершины, как в случае S_1 и S_2, то такой ориентированный маршрут называется *путем.* Последнее понятие также иногда применяется для обозначения простой цепи в неориентированных графах и для определения специального класса графов, так называемых деревьев. В общем случае деревья служат для графического представления иерархических структур или иерархий, а задачи нахождения деревьев с некоторыми дополнительными свойствами являются типовыми задачами оптимизации на графах.

Примерами маршрутов для неориентированного графа (рис. П1.8, *б*) являются последовательности ребер: $S_1 = <e_1, e_2, e_5, e_8>$, $S_2 = <e_1, e_2, e_3, e_1>$, $S_3 = <e_3, e_5, e_8>$. Если в маршруте не повторяются ни ребра, ни вершины, как в случае S_1 и S_3, то такой неориентированный маршрут называется *простой цепью.*

Деревом в теории графов называется такой граф $D = <V, E>$, между любыми двумя вершинами которого существует единственная простая цепь, т. е. неориентированный маршрут, у которого вершины и ребра не повторяются. Применительно к ориентированным графам соответствующее определение является более сложным, поскольку основывается на выделении некоторой специальной вершины v_0, которая получила специальное название *корневой* вершины или просто — *корня.* В этом случае ориентированный граф $D = <V, E>$ называется ориентированным деревом или сокращенно — деревом, если между корнем дерева v_0 и любой другой вершиной существует единственный путь, берущий начало в v_0. Ниже представлены два примера деревьев: ориентированного дерева (рис. П1.9, *а*) и неориентированного дерева (рис. П1.9, *б*).

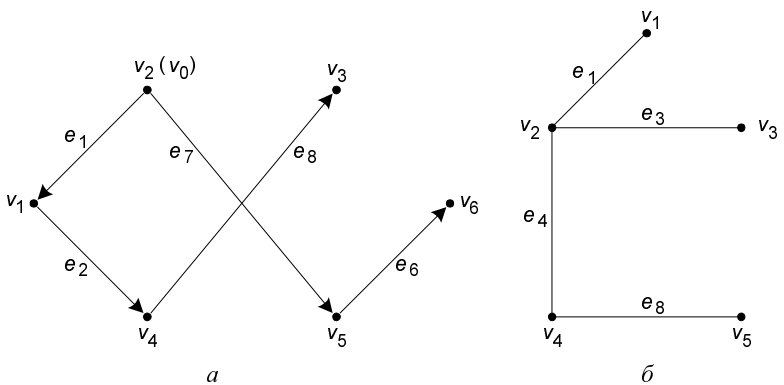


Рис. П1.9. Примеры ориентированного (*а*) и неориентированного (*б*) деревьев

Для случая ориентированного дерева (рис. П1.9, *а*) вершина v_2 является единственным его корнем и имеет специальное обозначение v_0. Единственность корня в ориентированном дереве следует из того факта, что ориентированный путь всегда имеет единственную вершину, которая является его началом. Поскольку в теории графов имеет значение только наличие или отсутствие связей между отдельными вершинами, деревья, как правило, изображаются специальным образом в виде иерархической структуры. При этом корень дерева изображается самой верхней вершиной в данной иерархии. Далее следуют вершины уровня 1, которые связаны с корнем одним ребром или одной

дугой. Следующий уровень будет иметь номер 2, поскольку соответствующие вершины должны быть связаны с корнем двумя последовательными ребрами или дугами. Процесс построения иерархического дерева продолжается до тех пор, пока не будут рассмотрены вершины, которые не связаны с другими вершинами, кроме рассмотренных, или из которых не выходит ни одна дуга. В этом случае самые нижние вершины иногда называют *листьями* дерева. Важно иметь в виду, что в теории графов дерево "растет" вниз, а не вверх, как в реальной жизни.

В случае неориентированного дерева (рис. П1.9, *б*) любая из вершин графа может быть выбрана в качестве корня. Подобный выбор определяется специфическими особенностями решаемой задачи. Так, вершина v_1 может рассматриваться в качестве корня неориентированного дерева, поскольку между v_1 и любой другой вершиной дерева всегда существует единственная простая цепь по определению (или, что менее строго, единственный неориентированный путь).

Изображенные ранее деревья (рис. П1.9) можно преобразовать к виду иерархий. Например, ориентированное дерево (рис. П1.9, *а*) может быть изображено в форме иерархического дерева (рис. П1.10, *а*), и такое представление является единственным. Неориентированное дерево (рис. П1.9, *б*) также может быть представлено в виде иерархического дерева (рис. П1.10, *б*), однако такое представление не является единственным. В данном случае корнем иерархии является вершина v_1.

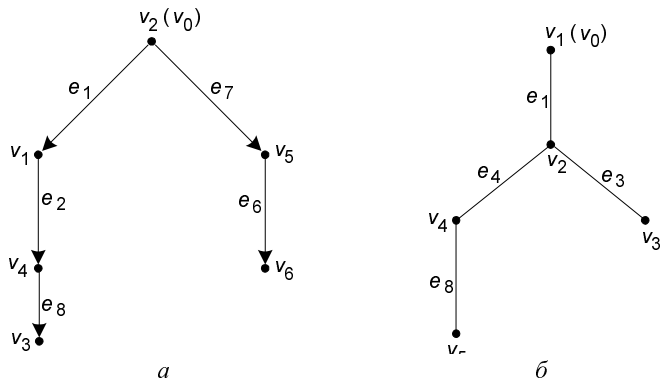


Рис. П1.10. Иерархические схемы ориентированного дерева (*а*) и неориентированного дерева (*б*)

В первом случае (рис. П1.10, *а*) вершины v_1 и v_5 образуют первый уровень иерархии, вершины v_4 и v_6 — второй уровень иерархии, вершина v_3 — третий

и последний уровень иерархии. Листьями данного ориентированного дерева являются вершины v_3 и v_6. При этом листьями данного неориентированного дерева являются вершины v_3 и v_5. Во втором случае (рис. П1.10, *б*) вершина v_2 образует первый уровень иерархии, вершины v_4 и v_3 — второй уровень иерархии, вершина v_5 — третий и последний уровень иерархии.

Наконец, последний способ задания отношений основан на явном указании некоторого *свойства*, которым должны обладать все элементы рассматриваемого отношения. Этот способ аналогичен описанному ранее способу для множеств и подходит для задания как конечных, так бесконечных отношений. В этом случае вводится в рассмотрение некоторое характеристическое свойство, которое может быть записано в виде *многоместного предиката* $P(<x_1, x_2, \ldots, x_k>)$. Данный предикат $P(<x_1, x_2, ..., x_k>)$ определяется на декартовом произведении $A_1 \times A_2 \times \ldots \times A_k$. При этом предикат может принимать одно из двух значений истинности: "*истина*" или "*ложь*". Если кортеж $<a_1, a_2, \ldots, a_k>$ принадлежит рассматриваемому отношению Q, то соответствующий предикат $P_Q(<a_1, a_2, \ldots, a_k>)$ принимает значение "*истина*". Если же кортеж $<a_1, a_2, ..., a_k>$ не принадлежит отношению Q, то соответствующий предикат $P_Q(<a_1, a_2,..., a_k>)$ принимает значение "*ложь*". Тогда в общем случае отношение Q может быть записано в виде: $Q=\{<a_1, a_2, \ldots, a_k> \mid P_Q(<a_1, a_2, ..., a_k>), <a_1, a_2, ..., a_k> \in A_1 \times A_2 \times \ldots \times A_k\}$.

Пример П1.2. Для иллюстрации различных способов задания отношений рассмотрим в качестве примера фрагмент расписания международных авиарейсов аэропорта Пулково-2 (Санкт-Петербург). Для простоты ограничим наше внимание только номерами авиарейсов, аэропортами назначения и типами самолетов. С этой целью рассмотрим следующие базисные множества: множество номеров авиарейсов A = {LH6370, MA232, Z8221, Z8229, Z8257, Z8277, Z8285}, множество аэропортов назначения B = {*Амстердам*, *Афины*, *Барселона*, *Каир*, *Нью-Йорк*, *Париж*, *Хельсинки*}, множество типов самолетов C = {Боинг-737, Боинг-747, Ил-86, Ту-134, Ту-154}. Одним из бинарных отношений $Q_1 \subseteq A \times B$ является отношение, устанавливающее соответствие между каждым номером авиарейса и аэропортом назначения. Другим отношением $Q_2 \subseteq A \times C$ является отношение между номерами авиарейсов и типом самолета, осуществляющим перелет по соответствующему маршруту.

Традиционно подобная информация записывается в форме таблиц, которые являются исходными объектами представления информации в реляционных базах данных. В нашем простом случае эти таблицы имеют следующий вид (табл. П1.1 и П1.2), что соответствует табличному способу задания этих отношений.

Таблица П1.1. *Авиарейсы и аэропорты назначения*

Номер авиарейса	**Аэропорт назначения**
Z8277	Амстердам
MA232	Афины
Z8221	Барселона
Z8285	Каир
LH6370	Нью-Йорк
Z8257	Париж
Z8229	Хельсинки

Таблица П1.2. *Авиарейсы и типы самолетов*

Номер авиарейса	**Тип самолета**
Z8277	Ту-154
MA232	Боинг-737
Z8221	Ту-134
Z8285	Ту-134
LH6370	Боинг-747
Z8257	Ил-86
Z8229	Ту-154

Первый из указанных выше способов задания отношений — непосредственное перечисление кортежей. В этом случае рассматриваемые отношения могут быть записаны в следующем виде: Q_1 = {<Z8277, *Амстердам*>, <MA232, *Афины*>, <Z8221, *Барселона*>, <Z8285, *Каир*>, <LH6370, *Нью-Йорк*>, <Z8257, *Париж*>, <Z8229, *Хельсинки*>}; Q_2 = {<Z8277, Ту-154>, <MA232, Боинг-737>, <Z8221, Ту-134>, <Z8285, Ту-134>, <LH6370, Боинг-747>, <Z8257, Ил-86>, <Z8229, Ту-154>}.

Второй из способов — матричный, с использованием которого эти отношения могут быть заданы в виде двух матриц $\mathbf{M}_1$ и $\mathbf{M}_2$, первая из которых соответствует отношению Q_1, а вторая — отношению Q_2. При этом неявно предполагается, что на исходных базисных множествах введено естественное отношение порядка, которое соответствует перечислению элементов в множествах A, B, C.

$$\mathbf{M}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{M}_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Примечание

Напомним, что при задании отношений матричным способом строки матрицы соответствуют элементам первого множества, а столбцы матрицы — элементам второго множества. В этом случае каждый из элементов матрицы может принимать одно из двух возможных значений: 0 или 1 в зависимости от того, принадлежит или нет соответствующий кортеж исходному отношению. Как нетрудно заметить, количество единиц в матрице отношения в точности равно количеству различных кортежей, из которых состоит то или иное отношение.

Третий из способов задания отношений (табличный) был использован для исходной записи рассматриваемых отношений.

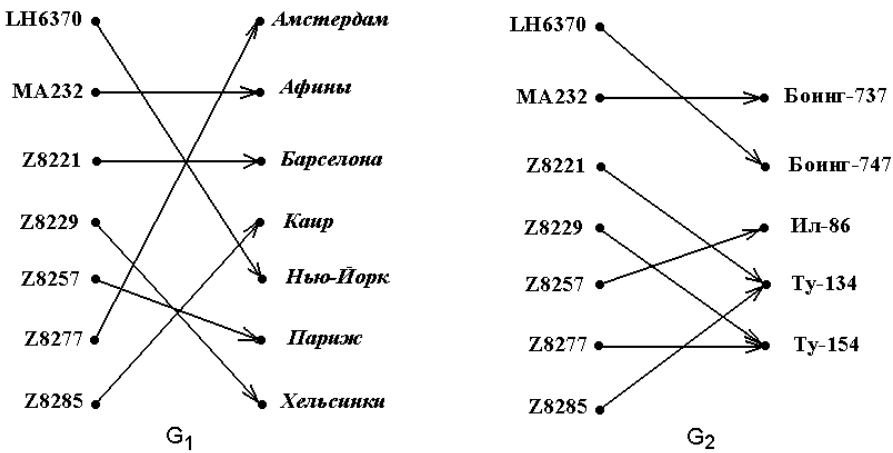


Рис. П1.11. Графы отношений Q_1 и Q_2

Рассмотрим четвертый способ задания отношений — графический. В этом случае отношения Q_1 и Q_2 могут быть заданы в виде двух ориентированных графов: $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$. При этом множество вершин V_1 графа G_1 должно соответствовать объединению множеств $A \cup B$, а множество вершин V_2 графа G_2 — объединению множеств $A \cup C$. Соответствующие графы отношений представлены на рис. П1.11.

Что касается пятого способа задания отношений, то применительно к рассматриваемому примеру для кортежей отношений Q_1 и Q_2 нельзя выявить общие свойства или характеристические признаки. Поскольку в этом случае для задания самих предикатов $P_1(<x_1, x_2>)$ и $P_2(<x_1, x_2>)$ отношений Q_1 и Q_2 необходимо использовать один из способов задания соответствующих отношений, этот способ из соображений конструктивности здесь не приводится.

Операции над бинарными отношениями

Поскольку отношения являются некоторыми специальным образом определенными множествами, то для них в полной мере применимы все теоретико-множественные операции: пересечение, объединение, разность, симметрическая разность и дополнение. В то же время для отношений могут быть определены дополнительные операции, не имеющие аналогов среди множеств в общем случае. Ограничив наше внимание бинарными отношениями, рассмотрим унарную операцию *"обратное отношение"* и бинарную операцию *композиции отношений*.

Обратное отношение. Пусть задано некоторое бинарное отношение $Q \subseteq A \times B$. *Обратным отношением* Q^{-1} к отношению Q называется такое бинарное отношение, заданное на декартовом произведении $B \times A$, которое содержит кортеж элементов $<b_j, a_i>$ $(b_j \in B,\ a_i \in A)$ тогда и только тогда, когда $<a_i, b_j> \in Q$. Другими словами: $Q^{-1} = \{<b_j, a_i> | <a_i, b_j> \in Q,\ a_i \in A,\ b_j \in B\}$.

В качестве примера рассмотрим обратное отношение Q_2^{-1} к отношению $Q_2 \subseteq A \times C$ из примера П1.4. В этом случае $Q_2^{-1} \subseteq C \times A$ и может быть задано матричным способом в виде:

$$\mathrm{M}_{Q_2^{-1}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Примечание

Как нетрудно заметить, матрица обратного отношения может быть получена транспонированием матрицы исходного бинарного отношения, а граф обратного отношения может быть получен обращением стрелок (изменением их ориентации на противоположную) графа исходного отношения и возможно перемещением некоторых его вершин для большей наглядности.

Композиция отношений. Пусть задано два бинарных отношения $Q_1 \subseteq A \times B$ и $Q_2 \subseteq B \times C$ (правое базисное множество отношения Q_1 должно быть обязательно равно левому базисному множеству отношения Q_2). *Композицией бинарных отношений* Q_1 и Q_2 (обозначается $Q_1 \bullet Q_2$) называется такое бинарное отношение Q, которое заданно на декартовом произведении $A \times C$ и содержит кортеж элементов $<a_i, c_k>$ ($a_i \in A$, $c_k \in C$) тогда и только тогда, когда существует элемент $b_j \in B$, такой что одновременно $<a_i, b_j> \in Q_1$ и $<b_j, c_k> \in Q_2$. Операцию композиции бинарных отношений иногда называют *произведением* отношений.

В качестве примера рассмотрим композицию конечных отношений Q_2^{-1} и Q_1, где Q_2^{-1} — обратное к отношению $Q_2 \subseteq A \times C$ из примера П1.3. В этом случае $Q_2^{-1} \subseteq C \times A$, $Q_1 \subseteq A \times B$, что обеспечивает выполнение формального условия, необходимого для этой операции. Результат композиции этих отношений, который содержательно устанавливает соответствие между типами самолетов и аэропортами назначения, может быть представлен в матричном виде следующим образом:

$$\mathrm{M}_{Q_2^{-1} \bullet Q_1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Для наглядности представим этот же результат в форме таблицы (табл. П1.3).

Таблица П1.3. *Типы самолетов и аэропорты назначения*

Тип самолета	Аэропорт назначения
Боинг-737	Афины
Боинг-747	Нью-Йорк
Ил-86	Париж
Ту-134	Барселона, Каир
Ту-154	Амстердам, Хельсинки

Отображение

Бинарное отношение F, заданное на декартовом произведении $A \times B$, называется *отображением*, если из одновременного выполнения условий $<a_i, a_j> \in F$ и $<a_i, a_s> \in F$ всегда следует равенство вторых элементов этих кортежей, а именно $a_j = a_s$. Другими словами, каждому из элементов множества A отображение F ставит в соответствие не более одного элемента множества B. В этом случае говорят, что отображение F действует *из* множества A *в* множество B. Для формальной записи отображения используется следующее обозначение $F: A \to B$, при этом не исключается случай, когда $A = B$.

С точки зрения теории множеств известное в математике понятие *функции* является частным случаем отображения, когда множества A и B являются числовыми. Общепринятый способ обозначения функциональной зависимости малыми латинскими буквами часто используется и в теории множеств. В этом случае отображение может быть записано в форме $f: A \to B$ или более привычной $y = f(x)$, что, если не указано дополнительно, не имеет принципиального значения. Обе эти формы записи отображений (функций) будут использоваться далее как эквивалентные.

Понятие отображения допускает обобщение на декартово произведение произвольного конечного числа множеств слева от стрелки. Поэтому в общем случае отображение записывается в виде $f: A_1 \times A_2 \times \ldots \times A_k \to B$ и ставит в соответствие каждому кортежу $<a_1, a_2, \ldots, a_k> \in A_1 \times A_2 \times \ldots \times A_k$ некоторый единственный элемент b из множества B. Известное в алгебре понятие *операция* является частным случаем такого отображения, когда все множества $A_1, A_2, \ldots, A_k$ и B равны. В этом случае операция, точнее, *k-местная операция*, может быть записана в форме $f: A \times A \times \ldots \times A \to A$ или в более привычной форме: $y = x_1 \bullet x_2 \bullet \ldots \bullet x_k$, где "$\bullet$" — символ данной операции. Так, например, арифметическая операция сложения есть не что иное, как бинарная операция $+: \boldsymbol{R} \times \boldsymbol{R} \to \boldsymbol{R}$, где через $\boldsymbol{R}$ обозначено множество всех действительных чисел.

При анализе отображений часто вводят в рассмотрение два отдельных множества, которые характеризуют особенности структурного строения отображений. Во-первых, отображение f может быть задано не на всем множестве A, а только на некотором его подмножестве $D_f \subseteq A$. В этом случае множество D_f называется *областью* (*множеством*) *определения* отображения f. Если же $D_f = A$, то отображение f (соответственно, функцию) называют *всюду определенным*. Во-вторых, подмножество $Im_f \subseteq B$, определяемое как $Im_f = \{f(x) \mid x \in D_f\}$, получило название *области* (*множества*) *значений* отображения f. Здесь можно отметить тот случай, когда множество обозначено двумя буквами (сокращение от *англ.* image).

Имеются некоторые дополнительные понятия, связанные с произвольным отображением $f: A \to B$. Зафиксируем произвольный элемент $a \in D_f$. По определению этому элементу a отображение f ставит в соответствие некоторый единственный элемент $b \in Im_f$. При этом элемент b называется *образом* элемента a при отображении f, а сам элемент a — *прообразом* элемента b при отображении f.

Примечание

Как нетрудно заметить, характеристическая функция множества по определению является отображением $\chi_A : X \to \{0, 1\}$, которое задается выражением (П1.1). Здесь в качестве области определения отображения используется некоторый универсум X, а в качестве области значений — двухэлементное множество $\{0, 1\}$. Важно понимать, что задание в качестве области значений множества $\{0, 1\}$ гарантирует, что никаких других значений, кроме 0 и 1, данная функция принимать не может.

Свойства бинарных отношений, заданных на одном базисном множестве

Как уже было упомянуто выше, в теории множеств физическая природа элементов, из которых могут быть образованы множества и отношения, не имеет принципиального значения. Предметом исследования являются такие особенности множеств и отношений, которые характеризуют отдельные абстрактные свойства их структуры. В контексте нечеткого моделирования наибольший интерес представляют такие общие свойства бинарных отношений, заданных на одном множестве, как рефлексивность, симметричность и транзитивность. Эти свойства играют важную роль при определении некоторых специальных отношений. Наш интерес к этим отношениям обусловлен тем обстоятельством, что эти свойства допускают естественное нечеткое обобщение, что широко используется в нечетком моделировании.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *рефлексивным*, если каждый из кортежей $<a_i, a_i>$ принадлежит отношению Q. Формально это можно записать в следующем виде: Q — рефлексивно тогда и только тогда, когда $<a_i, a_i> \in Q$, $\forall a_i \in A$. Следует заметить, что главная диагональ матрицы рефлексивного отношения состоит из одних 1.

Примечание

В определении рефлексивного отношения использован специальный символ "$\forall$", который называется *квантором общности* и читается *"для любого, для*

каждого, для всех" (этот символ ассоциируется с перевернутой первой буквой английского *All*). Квантор общности широко применяется в различных логических исчислениях, включая нечеткую логику. Здесь его употребление обусловливается соображениями удобства и сокращением формальных записей.

Кроме квантора общности имеется двойственный ему *квантор существования*, который обозначается символом "∃" (этот символ ассоциирован с первой буквой англ. *Exist*). В математических и логических выражениях он читается как *"существует, найдется хотя бы один"*. При этом не исключается случай, когда искомых элементов может оказаться несколько, а в предельном случае — даже все. Важно понимать, что применение квантора существования гарантирует наличие *хотя бы одного* элемента, удовлетворяющего заданному логическому условию. Более подробно семантика использования этих кванторов рассмотрена в *приложении 2*.

Примечание

При использовании квантора существования ничего не говорится о способе определения указанного элемента (или элементов, которых, вообще говоря, может быть несколько). Данное обстоятельство послужило источником критики классической теории множеств и целых разделов математики, особенно теорем существования, со стороны так называемого *конструктивного* направления. Однако эти аспекты теории множеств лежат за пределами тематики настоящей книги.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *антирефлексивным*, если ни один из кортежей $<a_i, a_i>$ не принадлежит отношению Q. Формально это можно записать в следующем виде: Q — антирефлексивно тогда и только тогда, когда $<a_i, a_i> \notin Q$, $\forall a_i \in A$. Следует заметить, что главная диагональ матрицы антирефлексивного отношения состоит из одних 0.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *симметричным*, если наряду с каждым кортежем $<a_i, a_j>$, принадлежащим отношению Q, этому отношению принадлежит и кортеж $<a_j, a_i>$, в котором элементы $a_i, a_j \in A$ взяты в обратном порядке. Формально это можно записать в следующем виде: Q — симметрично тогда и только тогда, когда $<a_i, a_j> \in Q$ и $<a_j, a_i> \in Q$ одновременно. Следует заметить, что матрица симметричного отношения симметрична относительно главной диагонали.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *асимметричным*, если из двух кортежей $<a_i, a_j>$ и $<a_j, a_i>$ по меньшей мере один не принадлежит отношению Q (а может быть и оба). Формально это можно записать в следующем виде: Q — асимметрично тогда и только

тогда, когда условия $<a_i, a_j> \in Q$ и $<a_j, a_i> \in Q$ не могут выполняться одновременно.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *антисимметричным*, если оба кортежа $<a_i, a_j>$ и $<a_j, a_i>$ могут принадлежать отношению Q только в одном случае, а именно когда $a_i = a_j$. Формально это можно записать в следующем виде: Q — антисимметрично тогда и только тогда, когда условия $<a_i, a_j> \in Q$ и $<a_j, a_i> \in Q$ могут выполняться одновременно только при $a_i = a_j$.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *транзитивным*, если из условия принадлежности двух кортежей $<a_i, a_j>$ и $<a_j, a_s>$ всегда следует, что данному отношению принадлежит и кортеж $<a_i, a_s>$. Формально это можно записать в следующем виде: Q — транзитивно тогда и только тогда, когда из условий $<a_i, a_j> \in Q$ и $<a_j, a_s> \in Q$ всегда следует выполнение условия: $<a_i, a_s> \in Q$.

Проиллюстрируем свойства отношений на следующем примере.

Пример П1.3. Каждая бизнес-система имеет некоторую структуру подчинения своих сотрудников согласно занимаемым ими должностям. Рассмотрим подобное отношение *"субординации"* на множестве сотрудников в общем случае. Данное отношение антирефлексивно (сотрудник не может быть подчинен самому себе); асимметрично (если сотрудник a_i подчинен сотруднику a_j, то сотрудник a_j не может быть подчинен сотруднику a_i) и транзитивным (если сотрудник a_i подчинен сотруднику a_j, а сотрудник a_j подчинен сотруднику a_s, то сотрудник a_i подчинен сотруднику a_s).

Рассмотренный пример иллюстрирует тот важный факт, что те или иные отношения, рассматриваемые в самом общем контексте, могут одновременно обладать несколькими из введенных в рассмотрение свойств.

Некоторые специальные виды бинарных отношений, заданных на одном базисном множестве

Совместное наличие нескольких свойств у тех или иных бинарных отношений позволяет выполнить их дальнейшую специализацию с целью получения более конструктивных результатов относительно наличия целого ряда дополнительных свойств и уточнения характера взаимосвязей элементов этих бинарных отношений.

Отношение строгого частичного порядка

Бинарное отношение Q, заданное на одном базисном множестве $Q \subset A \times A$, называется отношением *строгого частичного порядка*, если оно одновременно является антирефлексивным, асимметричным и транзитивным. Если дополнительно отношение удовлетворяет условию: для любой пары элементов $<a_i, a_j>$, где $a_i, a_j \in A$, справедливо либо $<a_i, a_j> \in Q$, либо $<a_j, a_i> \in Q$, то такое отношение называется отношением *строгого линейного порядка.*

Примерами отношений строгого частичного порядка могут служить отношение субординации на множестве сотрудников некоторой бизнес-системы (пример П1.3), а также отношение включения на множестве всех подмножеств (булеане) некоторого базисного множества. Примерами отношений строгого линейного порядка могут служить отношение *"строго больше"* на множестве действительных чисел, а также отношения типа *"быть выше ростом"*, *"иметь более высокую скорость движения"* на соответствующих базисных множествах.

Отношение толерантности

Бинарное отношение Q, заданное на одном базисном множестве $Q \subset A \times A$, называется отношением *толерантности*, если оно является рефлексивным и симметричным.

Отношение толерантности иногда называют отношением покрытия или сходства, поскольку оно связано с определением соответствующей системы множеств. А именно система подмножеств $\Im(A) = \{A_k \mid A_k \subseteq A\}$ множества A называется *покрытием*, если выполняется следующее условие:

$$\bigcup_k A_k = A, \qquad (A_k \in \Im), \qquad \text{(П1.18)}$$

т. е. объединение всех или части подмножеств из $\Im(A)$ совпадает с исходным множеством A (или *"покрывает"* исходное множество A).

Пример П1.4. Примером отношения толерантности может служить факт знакомства среди некоторого множества субъектов. Рассмотрим в качестве исходного множества $A = \{a_1, a_2, a_3, \ldots, a_n\}$ совокупность сотрудников некоторой достаточно крупной бизнес-системы. Сформируем бинарное отношение Q на этом множестве, включив в него такие пары элементов $<a_i, a_j>$, где $a_i, a_j \in A$, для которых выполняется условие: *"сотрудник a_i знаком с сотрудником a_j"*. Легко проверить, что это отношение является симметричным, поскольку если *"сотрудник a_i знаком с сотрудником a_j"*, то определенно и *"сотрудник a_j знаком с сотрудником a_i"*. Свойство рефлексивности также

выполняется, поскольку по определению будем считать, что каждый из сотрудников знаком с самим собой.

С другой стороны, это же отношение порождает некоторое покрытие $\Im(A) = \{A_k \mid A_k \subseteq A\}$ множества A, если в качестве подмножеств $A_k \subseteq A$ взять указанные ранее пары элементов $A_k = \{a_i, a_j\}$ — знакомых между собой сотрудников.

Отношение эквивалентности

Бинарное отношение Q, заданное на одном базисном множестве $Q \subset A \times A$, называется отношением *эквивалентности*, если оно одновременно является рефлексивным, симметричным и транзитивным.

Отношение эквивалентности также называют отношением разбиения, поскольку оно связано с определением соответствующей системы множеств. А именно система подмножеств $\Re(A) = \{A_k \mid A_k \subseteq A\}$ множества A называется *разбиением*, если выполняются следующие условия:

$$\bigcup_k A_k = A, \qquad (A_k \in \Re), \tag{П1.19}$$

$$A_l \bigcap A_m = \varnothing, \qquad (\forall A_l, A_m \in \Re), \tag{П1.20}$$

т. е. объединение всех или части подмножеств из $\Re(A)$ совпадает с исходным множеством A (или *"покрывает"* исходное множество A), при этом подмножества разбиения попарно не пересекаются между собой. В этом случае отдельные множества $A_k \in \Re$ разбиения получили специальное название — *классы* разбиения.

Нетрудно заметить, что каждое разбиение является в то же время и покрытием, которое дополнительно удовлетворяет свойству (П1.20). Вообще говоря, это свойство существенно обогащает структуру соответствующих отношений эквивалентности, позволяя утверждать, что каждый из элементов исходного множества A может принадлежать не более чем одному из классов разбиения. Свойство (П1.19) гарантирует наличие такого класса A_k для каждого элемента $a_l \in A$.

Примерами отношений эквивалентности могут служить отношение *"быть родственником"*, *"иметь одинаковый рост"*, *"иметь одинаковую цену"*, *"обладать равным доходом"* и многие другие. В общем случае с отношением эквивалентности связаны логические правила "хорошей" классификации, которые используются в системном моделировании для концептуальной структуризации предметной области при построении моделей сложных систем.

Основы комбинаторного анализа

Комбинаторный анализ, называемый также комбинаторной математикой или просто *комбинаторикой*, представляет собой раздел математики, посвященный изучению специальных свойств конечных или счетных множеств, а также разработке методов решения задач выбора и перечисления элементов этих множеств в соответствии с заданными правилами. Эти правила определяют способы задания или построения из элементов исходного множества той или иной комбинаторной конструкции, которая в общем случае называется *комбинаторным объектом*. При этом природа исходного множества не оказывает влияния на особенности определения и анализа свойств комбинаторных объектов. Примерами комбинаторных объектов являются перестановки, размещения и сочетания.

Развитие методов комбинаторного анализа связывают с именами Г. Лейбница, Я. Бернулли, Л. Эйлера. В настоящее время интерес к комбинаторному анализу возрождается в связи с развитием информатики, теории передачи информации, средств шифрования и защиты информации. Целью комбинаторного анализа является выяснение условий существования различных комбинаторных объектов и разработка алгоритмов их построения.

При разработке методов и алгоритмов решения различных задач оптимизации находят применение такие комбинаторные объекты, как перестановки, сочетания и размещения. Для их определения рассмотрим некоторое конечное множество $A = \{a_1, a_2, \ldots, a_n\}$.

Перестановка

Перестановкой элементов множества A называется их произвольная упорядоченная совокупность $(a_i, a_j, \ldots, a_k)$, в которой каждый из элементов множества A встречается ровно один раз.

Например, если в качестве исходного множества рассмотреть множество, содержащее первые 5 натуральных чисел: $A = \{1, 2, 3, 4, 5\}$, то следующие кортежи: (1, 2, 3, 5, 4), (2, 1, 3, 5, 4) и (3, 2, 1, 4, 5) являются примерами перестановок элементов множества A.

Один из основных вопросов комбинаторного анализа связан с нахождением общего количества различных комбинаторных объектов у рассматриваемого множества элементов.

Как несложно доказать методом индукции, общее количество перестановок элементов конечного множества $A = \{a_1, a_2, \ldots, a_n\}$ определяется выражением: $P_n = n!$, где P_n обозначает количество перестановок из n элементов, а операция "!" — операцию вычисления факториала натурального числа.

Для нахождения числа перестановок в программе MS Excel можно использовать, например, специальную функцию: ФАКТР. В качестве единственного аргумента этой функции используется число n — количество элементов рассматриваемого множества. Так, например, чтобы вычислить количество перестановок множества: $A = \{1, 2, 3, 4, 5\}$, следует в любую ячейку рабочего листа ввести формулу: =ФАКТР(5). В результате будет получено значение: $P_5 = 120$.

Сочетание

Сочетанием из n элементов множества A по m элементов называется их произвольная неупорядоченная совокупность $\{a_i, a_j, \ldots, a_k\}$, содержащая ровно m элементов ($m < n$), при этом каждый из элементов множества A встречается ровно один раз.

Например, если в качестве исходного множества рассмотреть множество: $A = \{1, 2, 3, 4, 5\}$, то подмножества: {1, 2, 3}, {1, 2, 4} и {3, 4, 5} являются примерами сочетаний из 5 элементов множества A по 3. Заметим, что сочетания {1, 2, 3}, {2, 1, 3} и {3, 2, 1} равны по определению.

Можно доказать, что общее количество сочетаний из n элементов конечного множества $A = \{a_1, a_2, \ldots, a_n\}$ по m элементов определяется выражением: $C_n^m = n!/(m! \cdot (n - m)!)$, где C_n^m обозначает количество сочетаний из n элементов по m элементов, а операция "!" — операцию вычисления факториала натурального числа.

Для нахождения числа перестановок в программе MS Excel можно использовать специальную функцию: ЧИСЛКОМБ. Эта функция имеет два аргумента, первый из которых равен n — количеству элементов рассматриваемого множества, а второй m — количеству элементов в сочетаниях. Так, например, чтобы вычислить количество сочетаний из 5 элементов множества: $A = \{1, 2, 3, 4, 5\}$ по 3 элемента, следует в любую ячейку рабочего листа ввести формулу: =ЧИСЛКОМБ(5; 3). В результате будет получено значение: $C_5^3 = 10$.

Размещение

Размещением из n элементов множества A по m элементов называется их произвольная упорядоченная совокупность $(a_i, a_j, \ldots, a_k)$, содержащая ровно m элементов ($m < n$), при этом каждый из элементов множества A встречается ровно один раз.

Например, если в качестве исходного множества рассмотреть множество: $A = \{1, 2, 3, 4, 5\}$, то следующие кортежи: (1, 2, 3, 5), (2, 3, 4, 5) и (3, 1, 2, 4)

являются примерами размещений элементов множества из 5 элементов множества A по 4. Заметим, что размещения (1, 2, 3, 5), (2, 3, 1, 5) и (3, 1, 5, 2) различны по определению.

Можно доказать, общее количество размещений из n элементов конечного множества $A = \{a_1, a_2, \ldots, a_n\}$ по m элементов определяется выражением: $A_n^m = n!/(n-m)!)$, где A_n^m — общепринятое обозначение количества размещений из n элементов по m элементов, а операция "!" обозначает операцию вычисления факториала натурального числа.

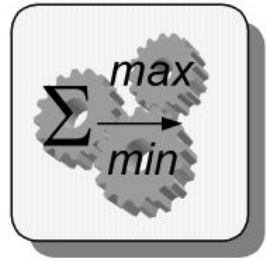
Для нахождения числа размещений в программе MS Excel можно использовать специальную функцию: `ПЕРЕСТ`. Эта функция имеет два аргумента, первый из которых равен n количеству элементов рассматриваемого множества, а второй m — количеству элементов в размещениях. Так, например, чтобы вычислить количество размещений из 5 элементов множества: $A = \{1, 2, 3, 4, 5\}$ по 3 элемента, следует в любую ячейку рабочего листа ввести формулу: `=ПЕРЕСТ(5; 3)`. В результате будет получено значение: $A_n^m = 60$.

Примечание

Эта же функция, как следует из ее названия, может быть использована и для нахождения количества перестановок заданного множества элементов. В этом случае второй аргумент функции `ПЕРЕСТ` должен быть равен первому аргументу, который задает количество элементов исходного множества. Так, например, чтобы вычислить количество перестановок множества: $A = \{1, 2, 3, 4, 5\}$, следует в любую ячейку рабочего листа ввести формулу: `=ПЕРЕСТ(5; 5)`. В результате будет получено значение: $P_5 = 120$.

При решении задач комбинаторного анализа необходимо правильно определить искомые комбинаторные объекты. Для этой цели можно воспользоваться следующей рекомендацией. Первоначально следует установить, все ли элементы исходного множества используются в определении комбинаторного объекта или нет? Если при определении комбинаторного объекта используются все элементы исходного множества, то соответствующим комбинаторным объектом может быть только перестановка. Если же при определении комбинаторного объекта используются не все элементы, то соответствующим комбинаторным объектом может быть либо сочетание, либо размещение.

В последнем случае следует дополнительно установить, имеет ли значение порядок расположения элементов или нет? Если в определении комбинаторного объекта важен порядок, то соответствующим комбинаторным объектом является размещение. Если же порядок не имеет значения, то соответствующим комбинаторным объектом является сочетание.



Приложение 2

Назначение операций главного меню программы электронных таблиц MS Office Excel 2003

Окно рабочего интерфейса программы электронных таблиц MS Office Excel 2003 имеет главное меню, операции которого обеспечивают реализацию широкого диапазона функциональных возможностей программы. В данном приложении рассматривается краткое назначение отдельных пунктов главного меню программы MS Office Excel 2003, что позволяет пользователю более уверенно подойти к выполнению операций, необходимых при решении задач оптимизации всех рассмотренных классов.

❒ Пункт меню **Файл** главного меню содержит следующие операции (рис. П2.1).

- **Создать** — позволяет создать новую книгу программы электронных таблиц MS Office Excel 2003.
- **Открыть** — вызывает стандартное диалоговое окно открытия внешнего файла с диска. По умолчанию в этом окне отображаются файлы книг электронных таблиц (файлы с расширением xls).
- **Закрыть** — закрывает окно редактирования открытой книги, при этом сама программа электронных таблиц MS Office Excel 2003 не закрывается. В случае внесения изменений в редактируемую книгу открывается дополнительное окно с предложением сохранить внесенные в книгу изменения либо отказаться от них.
- **Сохранить** — сохраняет внесенные в редактируемую книгу изменения в файле с текущим именем.
- **Сохранить как** — позволяет сохранить редактируемую книгу в файле с новым именем. В этом случае открывается дополнительное окно с предложением задать имя нового файла и его местоположение на диске.

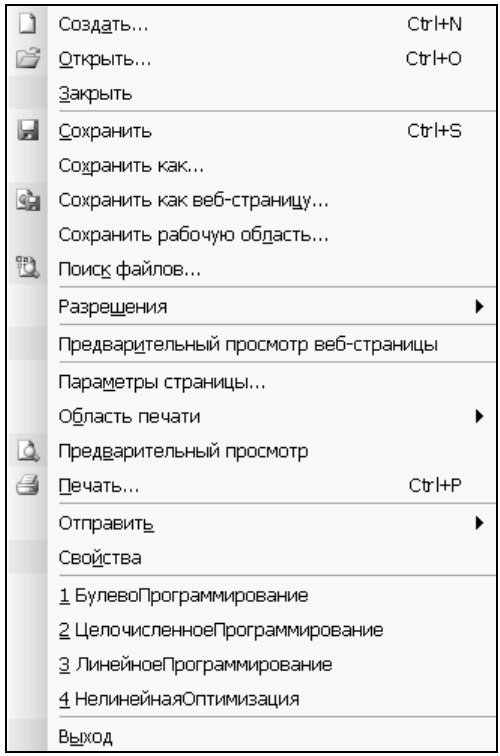


Рис. П2.1. Операции пункта меню **Файл** главного меню

- **Сохранить как веб-страницу** — позволяет сохранить редактируемую книгу в форме веб-страницы, которая может быть открыта для просмотра в любом интернет-браузере. В этом случае открывается дополнительное окно с предложением задать имя нового файла с расширением htm или html и его местоположение на диске.
- **Сохранить рабочую область** — позволяет сохранить рабочую область редактируемой книги во внешнем файле на диске. При этом вызывается стандартное диалоговое окно сохранения файла на диске с предложением задать имя соответствующего файла с расширением xwl.
- **Поиск файлов** — открывает диалоговое окно задания строки текста для поиска файлов на диске, в которых она содержится.
- **Разрешения** — позволяет установить различные уровни доступа к редактируемой рабочей книге.
- **Предварительный просмотр веб-страницы** — позволяет выполнить предварительный просмотр редактируемой рабочей книги в форме веб-страницы на установленном по умолчанию интернет-браузере.

- **Параметры страницы** — открывает диалоговое окно для редактирования параметров страницы с целью последующей печати рабочей книги на подключенном к данному компьютеру принтере.
- **Область печати** — позволяет отобразить или убрать разделитель в форме пунктирной линии, указывающий область, которая при печати разместится на одной странице.
- **Предварительный просмотр** — позволяет выполнить предварительный просмотр рабочих листов редактируемой книги на странице с целью визуального контроля их содержимого при подготовке к печати на подключенном к данному компьютеру принтере.
- **Печать** — позволяет распечатать на принтере информацию рабочего листа или выделенной его части. При этом вызывается стандартное диалоговое окно настройки свойств печати на подключенном к данному компьютеру принтере.
- **Отправить** — содержит дополнительные операции, позволяющие отправить редактируемую рабочую книгу по факсу или по электронной почте.
- **Свойства** — открывает диалоговое окно для редактирования свойств редактируемой рабочей книги.
- Секция с именами последних файлов, с которыми осуществлялась работа в программе MS Office Excel 2003.
- **Выход** — закрывает программу MS Office Excel 2003, при этом пользователю предлагается сохранить внесенные в рабочую книгу изменения в файле с текущим именем.

❒ Пункт меню **Правка** содержит следующие операции (рис. П2.2).

- **Отменить** — отменяет выполнение последнего действия.
- **Повторить** — отменяет выполнение последней операции **Отменить**.
- **Вырезать** — вырезает выделенный фрагмент из редактируемого рабочего листа и помещает его в буфер обмена.
- **Копировать** — копирует выделенный фрагмент из редактируемого рабочего листа и помещает его в буфер обмена.
- **Буфер обмена Office** — позволяет просмотреть информацию, хранящуюся в буфере обмена пакета Office, и при необходимости вставить нужный фрагмент в выделенную ячейку.
- **Вставить** — вставляет хранящийся в буфере обмена пакета Office фрагмент в выделенную ячейку редактируемого рабочего листа.

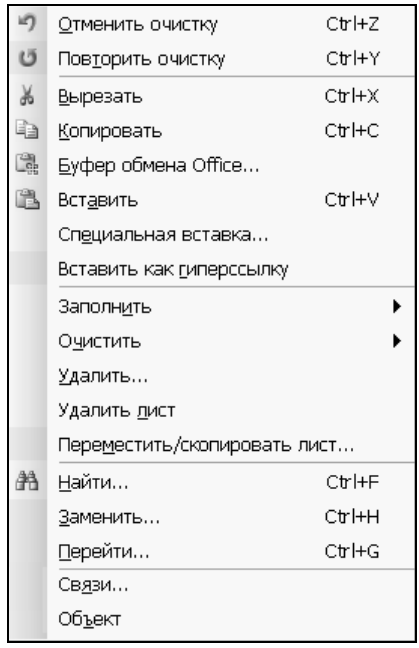


Рис. П2.2. Операции пункта меню **Правка** главного меню

- **Специальная вставка** — вызывает диалоговое окно мастера выбора объекта для вставки из числа зарегистрированных приложений.
- **Вставить как гиперссылку** — позволяет вставить хранящийся в буфере обмена пакета Office фрагмент в выделенную ячейку редактируемого рабочего листа в форме гиперссылки.
- **Заполнить** — содержит дополнительные операции, позволяющие заполнять информацией выделенные ячейки редактируемого рабочего листа.
- **Очистить** — содержит дополнительные операции, позволяющие удалить информацию из выделенных ячеек редактируемого рабочего листа.
- **Удалить** — удаляет выделенные ячейки или строки с выполнением различных сдвигов оставшихся ячеек.
- **Удалить лист** — удаляет рабочий лист из редактируемой книги.
- **Переместить/скопировать лист** — открывает дополнительное окно, которое позволяет выбрать лист для перемещения или копирования в одну из открытых рабочих книг.
- **Найти** — открывает дополнительное окно, которое позволяет задать строку текста для поиска в редактируемой рабочей книге.

- **Заменить** — открывает дополнительное окно, которое позволяет задать 2 строки текста для поиска и замены в редактируемой рабочей книге.
- **Перейти** — открывает дополнительное окно, которое позволяет выбрать элемент для быстрого перехода в редактируемой рабочей книге.
- **Связи** — открывает дополнительное окно, которое отображает информацию о связях ячеек в редактируемой рабочей книге. Эта операция активна только в том случае, если ранее были установлены связи ячеек рабочего листа с внешними объектами.
- **Объект** — содержит дополнительные операции, позволяющие изменить связанный объект или открыть соответствующее приложение для редактирования внешнего объекта. Эта операция активна только в том случае, если ранее были установлены связи ячеек рабочего листа с внешними объектами.

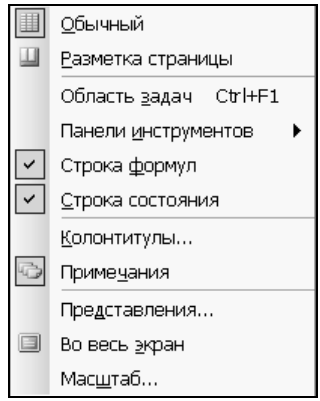


Рис. П2.3. Операции пункта меню **Вид** главного меню

❒ Пункт меню **Вид** позволяет отображать на экране различные компоненты программы и содержит следующие операции (рис. П2.3).

- **Обычный** — отображает рабочий лист программы MS Office Excel 2003 в обычном представлении.
- **Разметка страницы** — отображает рабочий лист программы MS Office Excel 2003 в виде страницы установленного формата.
- **Область задач** — делает видимой или невидимой область задач программы MS Office Excel 2003.
- **Панели инструментов** — содержит дополнительные операции, позволяющие сделать видимой или невидимой ту или иную панель инструментов.

- **Строка формул** — делает видимой или невидимой строку формул программы MS Office Excel 2003.
- **Строка состояния** — делает видимой или невидимой строку состояния программы MS Office Excel 2003.
- **Колонтитулы** — позволяет создавать и редактировать верхний и нижний колонтитулы рабочего листа.
- **Примечания** — позволяет создавать и редактировать примечания для различных ячеек рабочего листа.
- **Представления** — отображает дополнительное окно для задания и редактирования параметров представлений.
- **Во весь экран** — отображает рабочий лист программы MS Office Excel 2003 во весь экран без панелей инструментов и строки состояния.
- **Масштаб** — позволяет изменить масштаб отображения рабочего листа программы MS Office Excel 2003.

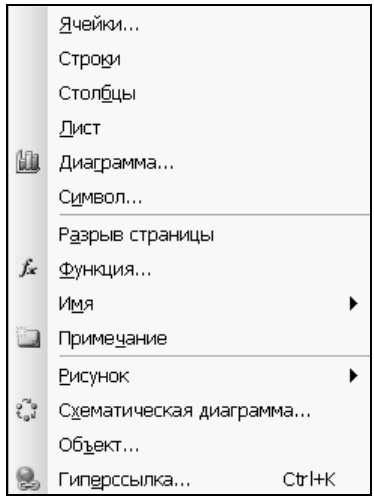


Рис. П2.4. Операции пункта меню **Вставка** главного меню

❒ Пункт меню **Вставка** содержит следующие операции (рис. П2.4).

- **Ячейки** — позволяет вставить в рабочий лист дополнительные ячейки, строку или столбец.
- **Строки** — позволяет вставить в рабочий лист дополнительную строку.
- **Столбцы** — позволяет вставить в рабочий лист дополнительный столбец.
- **Лист** — позволяет вставить в рабочую книгу дополнительный лист.

- **Диаграмма** — открывает дополнительное окно мастера диаграмм для задания типа диаграммы, исходных данных, подписей рядов данных и осей.
- **Символ** — открывает дополнительное окно со списком специальных символов для добавления их в редактируемую ячейку рабочего листа.
- **Разрыв страницы** — позволяет вставить в рабочий лист разрыв страницы.
- **Функция** — открывает дополнительное окно мастера функций для задания имени функции и ее аргументов.
- **Имя** — позволяет задать дополнительные имена для последующего применения к отдельным ячейкам рабочего листа.
- **Примечание** — позволяет вставить в рабочий лист дополнительное примечание.
- **Рисунок** — содержит дополнительные операции, позволяющие вставить в рабочий лист рисунок различных графических форматов.
- **Схематическая диаграмма** — открывает дополнительное окно со списком специальных видов диаграмм для добавления их в рабочий лист и последующего редактирования.
- **Объект** — открывает дополнительное окно со списком специальных объектов из числа зарегистрированных приложений для добавления их в рабочий лист.
- **Гиперссылка** — открывает дополнительное окно для выбора папки или файла, ссылка на которые может быть добавлена в редактируемую ячейку рабочего листа в форме гиперссылки.

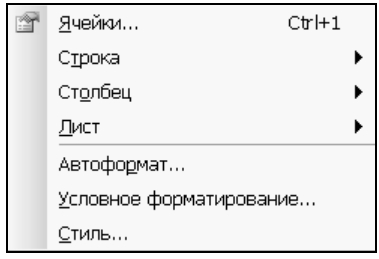


Рис. П2.5. Операции пункта меню **Формат** главного меню

❐ Пункт меню **Формат** содержит следующие операции (рис. П2.5).

- **Ячейки** — открывает дополнительное окно редактирования параметров и формата данных для выделенной ячейки или диапазона ячеек рабочего листа.

- **Строка** — содержит дополнительные операции, позволяющие изменить высоту выбранной строки, сделать ее видимой или невидимой.
- **Столбец** — содержит дополнительные операции, позволяющие изменить ширину выбранного столбца, сделать его видимым или невидимым.
- **Лист** — содержит дополнительные операции, позволяющие изменить имя редактируемого рабочего листа, сделать его видимым или невидимым.
- **Автоформат** — открывает дополнительное окно со списком специальных форматов для автоматического форматирования выделенного диапазона ячеек рабочего листа.
- **Условное форматирование** — открывает дополнительное окно задания условий и значений для выполнения условного форматирования.
- **Стиль** — открывает дополнительное окно для редактирования параметров стиля представления данных редактируемого рабочего листа.

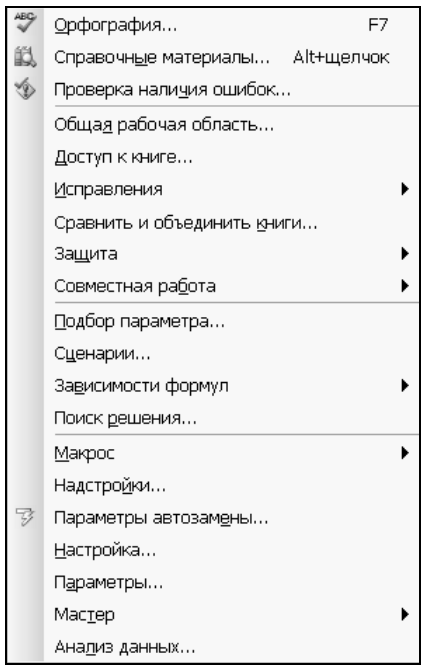


Рис. П2.6. Операции пункта меню **Сервис** главного меню

❒ Пункт меню **Сервис** содержит следующие операции (рис. П2.6).

- **Орфография** — открывает дополнительное окно для спецификации параметров проверки орфографии и выполнения этой проверки для всех ячеек редактируемого рабочего листа.

- **Справочные материалы** — открывает дополнительное окно Справки для задания параметров поиска справочной информации по ключевому слову или тексту.
- **Проверка наличия ошибок** — запускает встроенную программу поиска ошибок в редактируемом рабочем листе.
- **Общая рабочая область** — открывает дополнительное окно Общей рабочей области для получения доступа к документам и задачам коллективной разработки.
- **Доступ к книге** — открывает дополнительное окно для задания параметров доступа к редактируемой рабочей книге.
- **Исправления** — содержит дополнительные операции, позволяющие начать запись исправлений, а также принять или отклонить внесенные в рабочий лист исправления.
- **Сравнить и объединить книги** — позволяет сравнить и объединить книги, к которым установлен коллективный доступ.
- **Защита** — содержит дополнительные операции, позволяющие установить защиту от изменений для отдельных листов и рабочей книги в целом.
- **Совместная работа** — открывает дополнительное окно программы NetMeeting для задания параметров совместного обсуждения редактируемой рабочей книги.
- **Подбор параметра** — открывает дополнительное окно мастера подбора параметра, который используется для решения уравнений.
- **Сценарии** — открывает дополнительное окно Диспетчера сценариев для добавления, объединения и спецификации параметров сценариев.
- **Зависимости формул** — позволяет отобразить имеющиеся зависимости между ячейками рабочего листа.
- **Поиск решения** — открывает дополнительное окно мастера поиска решения, который используется для спецификации параметров решения задач оптимизации.
- **Макрос** — содержит дополнительные операции, позволяющие начать запись макроса и открыть редактор Visual Basic для редактирования текста программного кода макросов и модулей.
- **Надстройки** — открывает дополнительное окно менеджера добавления и удаления надстроек для программы электронных таблиц MS Office Excel 2003.
- **Параметры автозамены** — открывает дополнительное окно спецификации параметров автозамены текста при вводе и параметров смарт-тегов.

- **Настройка** — открывает дополнительное окно для настройки панелей инструментов программы электронных таблиц MS Office Excel 2003.
- **Параметры** — открывает дополнительное окно для спецификации параметров программы электронных таблиц MS Office Excel 2003.
- **Мастер** — содержит дополнительные операции, позволяющие автоматизировать задание отдельных функций дополнительных компонентов программы электронных таблиц MS Office Excel 2003.
- **Анализ данных** — открывает дополнительное окно со списком доступных инструментов анализа данных для выбора из них необходимого и задания соответствующих исходных данных.

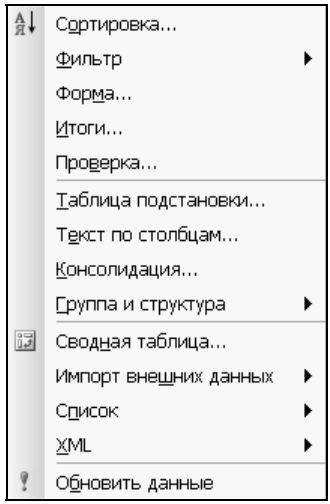


Рис. П2.7. Операции пункта меню **Данные** главного меню

❒ Пункт меню **Данные** содержит следующие операции (рис. П2.7).

- **Сортировка** — открывает дополнительное окно спецификации параметров сортировки и ее выполнения для выделенного диапазона ячеек.
- **Фильтр** — открывает дополнительное окно спецификации параметров фильтра для отображения необходимой информации в выделенном диапазоне ячеек.
- **Форма** — открывает дополнительное окно специальной формы для просмотра, ввода и редактирования данных листа, а также поиска данных по заданному критерию.
- **Итоги** — открывает дополнительное окно **Промежуточные итоги** для спецификации параметров и вычисления итогов по заданному столбцу.

- **Проверка** — открывает дополнительное окно для спецификации параметров автоматической проверки вводимых в ячейки данных.
- **Таблица подстановки** — открывает дополнительное окно спецификации параметров для создания таблицы подстановки с одной или несколькими переменными.
- **Текст по столбцам** — открывает дополнительное окно мастера для спецификации параметров форматирования и отображения текста в выделенном столбце.
- **Консолидация** — открывает дополнительное окно для спецификации параметров консолидации данных из различных листов и рабочих книг.
- **Группа и структура** — содержит дополнительные операции, позволяющие группировать и разгруппировать выделенный диапазон ячеек, а также создавать, удалять и редактировать структуры данных.
- **Сводная таблица** — открывает дополнительное окно мастера для спецификации параметров создания и редактирования сводных таблиц данных из различных листов и рабочих книг.
- **Импорт внешних данных** — содержит дополнительные операции, позволяющие выбрать формат данных и выполнить импорт данных из внешних источников, а также создать и выполнить запрос данных из различных источников.
- **Список** — содержит дополнительные операции, позволяющие создавать список, а также изменять его размер, выполнять синхронизацию данных списка и преобразовывать его в диапазон ячеек.
- **XML** — содержит дополнительные операции, позволяющие выбрать источник XML-данных и выполнить импорт этих данных из внешних источников, а также настроить пакеты расширения XML.
- **Обновить данные** — позволяет выполнить обновление данных из внешних источников, которые были помещены ранее в сводную таблицу с использованием исходного списка.

❐ Пункт меню **Окно** содержит следующие операции (рис. П2.8).

- **Новое** — позволяет создать новое окно для отображения редактируемого рабочего листа.
- **Расположить** — содержит дополнительные операции, позволяющие расположить открытые окна рабочих листов вертикально, горизонтально или каскадно.
- **Сравнить рядом с** — позволяет отобразить рядом открытые окна рабочих листов.

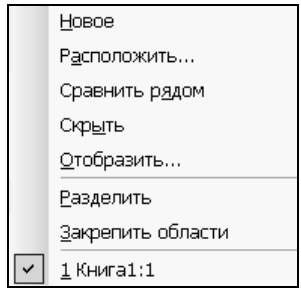


Рис. П2.8. Операции пункта меню **Окно** главного меню

- **Скрыть** — позволяет скрыть открытое окно редактируемого рабочего листа без его закрытия.
- **Отобразить** — позволяет отобразить скрытое окно редактируемого рабочего листа.
- **Разделить** — позволяет разделить или отменить разделение редактируемого рабочего листа на отдельные области.
- **Закрепить области** — позволяет закрепить или отменить закрепление отдельных областей в редактируемом рабочем листе.
- Секция содержит имена открытых окон редактируемых рабочих книг и листов для переключения между ними.

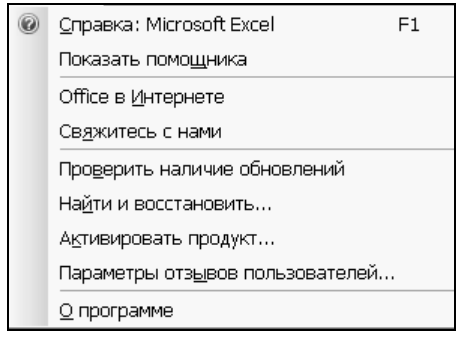
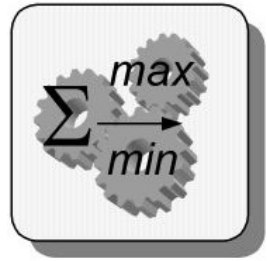


Рис. П2.9. Операции пункта меню **Справка** главного меню

❐ Пункт меню **Справка** содержит следующие операции (рис. П2.9).

- **Справка: Microsoft Excel** — вызывает программу просмотра справочной системы с началом знакомства с MS Office Excel 2003.
- **Показать помощника** — отображает помощника программы MS Office Excel 2003, если он установлен при инсталляции программы.

- **Office в Интернете** — вызывает интернет-браузер, установленный на компьютере пользователя по умолчанию, и реализует попытку установить связь с домашней страницей разработчиков пакета MS Office System.
- **Свяжитесь с нами** — вызывает программу просмотра справочной системы с информацией о Службе поддержки продуктов корпорации Майкрософт через Интернет, находящейся по адресу: **http://support.microsoft.com**.
- **Проверить наличие обновлений** — вызывает интернет-браузер, установленный на компьютере пользователя по умолчанию, и реализует попытку установить связь с веб-узлом разработчиков пакета MS Office System с целью проверки наличия последних обновлений.
- **Найти и восстановить** — открывает дополнительное окно спецификации параметров для нахождения ошибок и восстановления программы MS Office Excel 2003.
- **Активировать продукт** — позволяет активировать программу MS Office Excel 2003 после ее инсталляции.
- **Параметры отзывов пользователей** — открывает дополнительное окно для спецификации параметров пользователя с целью его участия в обсуждении вопросов по улучшению качества программы электронных таблиц MS Office Excel 2003.
- **О программе** — отображает информацию о текущей рабочей версии программы электронных таблиц MS Office Excel 2003.



Приложение 3

Назначение операций главного меню редактора Visual Basic пакета MS Office System 2003

Окно рабочего интерфейса редактора Visual Basic пакета MS Office System 2003 имеет главное меню, которое позволяет пользователю выполнять весь набор операций, обеспечивающих реализацию широкого диапазона функциональных возможностей программы. В данном приложении рассматривается назначение отдельных пунктов главного меню редактора Visual Basic, входящего в состав офисного пакета MS Office System 2003. Поскольку редактор Visual Basic является нелокализованным даже для локализованной версии пакета MS Office System 2003, названия пунктов главного меню приводятся на 2-х языках: на английском и русском, пункты русского меню указаны после английского в скобках.

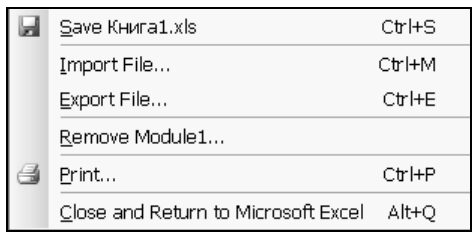


Рис. П3.1. Операции пункта меню **File** (Файл) главного меню

- Пункт меню **File** (Файл) главного меню содержит следующие операции (рис. П3.1).
  - **Save** (Сохранить) — сохраняет редактируемый проект в соответствующем документе MS Office System 2003; в случае вызова редактора из программы MS Office Excel 2003 — в рабочей книге.

- **Import File** (Импорт файла) — позволяет добавлять в проект файлы одного из следующих форматов: файлы форм Visual Basic с расширением FRM, файлы программ Visual Basic с расширением BAS, файлы форм VBA с расширением FRM, файлы с описанием классов Visual Basic с расширением CLS. При выполнении этой операции меню открывается дополнительное окно с предложением выбрать соответствующий файл на диске.
- **Export File** (Экспорт файла) — позволяет экспортировать редактируемый модуль или форму в файл программы Visual Basic с расширением BAS. При выполнении этой операции меню открывается дополнительное окно с предложением задать имя нового файла и его местоположение на диске.
- **Remove** (Удалить) — удаляет выделенный модуль или форму из редактируемого проекта. При выполнении этой операции пользователю предлагается экспортировать модуль или форму в файл программы Visual Basic, при этом вызывается стандартное диалоговое окно сохранения файла на диске с предложением задать имя соответствующего файла и его местоположение.
- **Print** (Печать) — позволяет распечатать на принтере информацию о содержании редактируемого модуля или формы. При этом вызывается стандартное диалоговое окно настройки свойств печати на подключенном к данному компьютеру принтеру.
- **Close and Return to Microsoft Excel** (Закрыть и вернуться в MS Excel) — закрывает редактор Visual Basic и возвращает пользователя к программе MS Office Excel 2003.

❒ Пункт меню **Edit** (Правка) содержит следующие операции по редактированию программного кода или формы пользователя (рис. П3.2).

- **Undo** (Отменить) — отменяет выполнение одного или нескольких последних действий.
- **Redo** (Повторить) — отменяет выполнение последней операции **Undo** (Отменить).
- **Cut** (Вырезать) — вырезает выделенный фрагмент текста из окна редактирования программного кода или элемент управления из формы пользователя и помещает его в буфер обмена.
- **Copy** (Копировать) — копирует выделенный фрагмент из окна редактирования кода или выделенный элемент управления из формы пользователя и помещает его в буфер обмена.

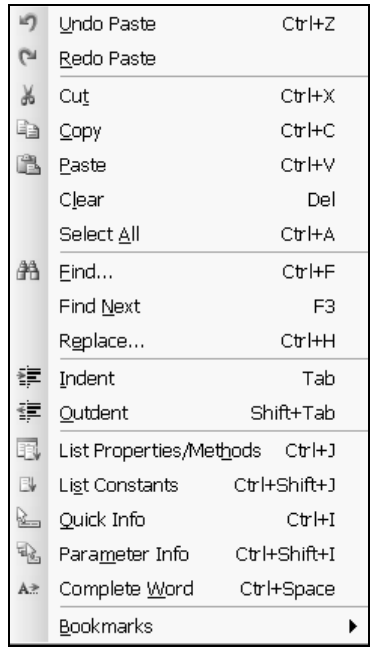


Рис. П3.2. Операции пункта меню **Edit** (Правка) главного меню

- **Paste** (Вставить) — вставляет хранящийся в буфере обмена фрагмент текста в программный код или элемент управления на форму пользователя.
- **Clear** (Очистить) — удаляет выделенный текст из программного кода редактируемого модуля. При редактировании формы пользователя вместо данной операции отображается операция **Delete** (Удалить), которая удаляет выделенный элемент управления с формы пользователя.
- **Select All** (Выделить все) — позволяет выделить всю информацию окна редактирования или все элементы управления формы пользователя.
- **Find** (Найти) — вызывает диалоговое окно задания параметров поиска необходимого элемента проекта или текста.
- **Find Next** (Найти далее) — вызывает диалоговое окно с параметрами последнего процесса поиска необходимого элемента проекта или текста для нахождения следующего объекта заданного типа.
- **Replace** (Заменить) — вызывает диалоговое окно задания параметров поиска и замены необходимого элемента проекта или текста.
- **Indent** (Установить отступ) — смещает текст справа от курсора на заданный интервал табуляции вправо.

- **Outdent** (Отменить отступ) — смещает текст справа от курсора на заданный интервал табуляции влево.
- **List Properties/Methods** (Список свойств/методов) — отображает во всплывающем окне список доступных свойств и методов для выбранного элемента программного кода. После двойного щелчка на выбранном свойстве или методе соответствующее слово добавляется в код.
- **List Constants** (Список констант) — отображает во всплывающем окне список доступных констант для выбранного элемента программного кода. После двойного щелчка на выбранном значении константы оно добавляется в код.
- **Quick Info** (Быстрая информация) — вызывает всплывающую подсказку о синтаксисе текущего выражения программного кода.
- **Parameter Info** (Информация о параметрах) — вызывает всплывающую подсказку о синтаксисе текущего параметра программного кода.
- **Complete Word** (Завершить слово) — обеспечивает автоматическое завершение набора выражения с клавиатуры, если данное выражение является стандартной синтаксической конструкцией языка VBA.
- **Bookmarks** (Закладки) — содержит вложенное подменю, операции которых позволяют задавать в тексте программного кода закладки и быстро перемещаться по тексту от одной закладки к другой в прямом и обратном направлениях.

❒ Пункт меню **View** (Вид) позволяет отображать на экране различные компоненты программы и содержит следующие операции (рис. П3.3).

- **Code** (Код) — открывает окно редактора программного кода для выбранного элемента проекта.
- **Object** (Объект) — открывает окно редактора формы пользователя для выбранного элемента проекта.
- **Definition** (Определение) — указывает для выбранной переменной или выражения место программного кода, в котором выполняется их определение.
- **Last Position** (Последняя позиция) — перемещает курсор к последней редактируемой строке в тексте программного кода.
- **Object Browser** (Браузер объектов) — вызывает окно браузера объектов с именами всех доступных для использования классов и модулей с соответствующими процедурами и функциями.

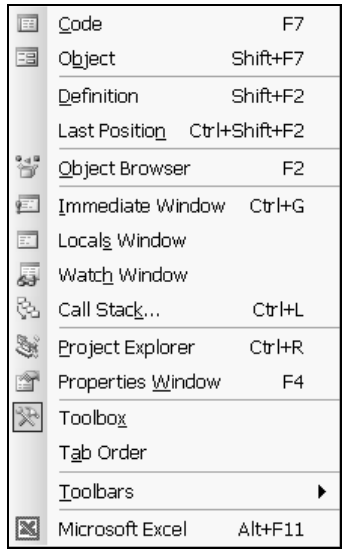


Рис. П3.3. Операции пункта меню **View** (Вид) главного меню

- **Immediate Window** (Окно непосредственных вычислений) — открывает дополнительное окно, в котором можно выполнить простейшие действия, записанные в одной строке, и непосредственно получить соответствующий результат после нажатия клавиши **Enter (**Ввод**)**.
- **Locals Window** (Окно локальных значений) — открывает дополнительное окно, в котором содержится информация обо всех переменных и объектах, содержащихся в редактируемой процедуре или функции.
- **Watch Window** (Окно контрольных значений) — открывает дополнительное окно, в котором содержится информация о текущих значениях выбранных переменных или группы переменных.
- **Call Stack** (Стек вызова) — открывает дополнительное окно, в котором отображается информация обо всех вызванных и незавершенных процедурах.
- **Project Explorer** (Проводник проекта) — открывает и активизирует окно **Проводника проекта**, в котором содержится информация обо всех элементах, содержащихся в редактируемом проекте.
- **Properties Window** (Окно свойств) — открывает и активизирует окно свойств выделенного на форме пользователя объекта или элемента редактируемого проекта.
- **Toolbox** (Панель инструментов) — открывает и активизирует специальное окно панели инструментов, содержащей кнопки с элементами

управления, которые могут быть добавлены на редактируемую форму пользователя.

- **Tab Order** (Порядок следования) — открывает дополнительное окно, в котором отображается список имен всех размещенных на форме пользователя элементов управления с возможностью изменить порядок их следования, реализуемый при нажатии клавиши **Tab** (Табуляция).
- **Toolbars** (Панели инструментов) — содержит вложенные операции, позволяющие открыть окна дополнительных панелей инструментов для выполнения специальных операций по редактированию элементов управления формы пользователя или отладке текста программного кода.
- **Microsoft Excel** — возвращает пользователя к программе MS Office Excel 2003, при этом редактор Visual Basic не закрывается.

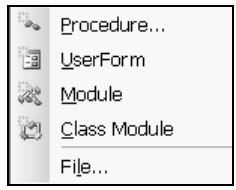


Рис. П3.4. Операции пункта меню **Insert** (Вставка) главного меню

❐ Пункт меню **Insert** (Вставка) содержит следующие операции (рис. П3.4).

- **Procedure** (Процедура) — открывает дополнительное окно, в котором предлагается специфицировать параметры новой процедуры, после чего в программный код будет добавлено объявление этой процедуры.
- **UserForm** (Форма пользователя) — добавляет в проект новую форму пользователя, на которой могут быть размещены различные элементы управления.
- **Module** (Модуль) — добавляет в проект новый программный модуль.
- **Class Module** (Модуль класса) — добавляет в проект программный модуль для создания файла заголовка или файла исходного текста для нового класса.
- **File** (Файл) — вызывает стандартное окно открытия файла с предложением выбрать файл, содержимое которого будет добавлено в выбранный рабочий лист проекта.

❐ Пункт меню **Format** (Формат) содержит следующие операции (рис. П3.5).

- **Align** (Выравнивание) — содержит список дополнительных операций, позволяющих выравнивать выделенные элементы управления друг относительно друга.

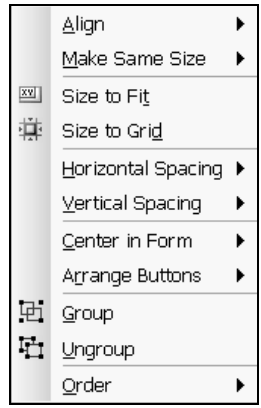


Рис. П3.5. Операции пункта меню **Format** (Формат) главного меню

- **Make Same Size** (Сделать одинакового размера) — содержит список дополнительных операций, позволяющих сделать выделенные элементы управления одинакового размера по ширине, по высоте или по обоим этим измерениям.
- **Size to Fit** (Установить оптимальный размер) — позволяет установить некоторый оптимальный размер всех выделенных элементов управления на форме пользователя.
- **Size to Grid** (Установить размер по сетке) — позволяет привязать к сетке размер всех выделенных элементов управления на форме пользователя.
- **Horizontal Spacing** (Горизонтальный интервал) — содержит список дополнительных операций, позволяющих изменять горизонтальный интервал между всеми выделенными элементами управления на форме пользователя.
- **Vertical Spacing** (Вертикальный интервал) — содержит список дополнительных операций, позволяющих изменять вертикальный интервал между всеми выделенными элементами управления на форме пользователя.
- **Center in Form** (Расположить по центру формы) — содержит список дополнительных операций, позволяющих расположить все выделенные элементы управления по центру формы пользователя относительно горизонтальной или вертикальной оси.
- **Arrange Buttons** (Упорядочить кнопки) — содержит список дополнительных операций, позволяющих расположить все выделенные кнопки управления внизу или справа на форме пользователя.

- **Group** (Группировка) — выполняет группировку выделенных элементов управления, после чего все действия с этими элементами можно выполнять как с одним объектом.
- **Ungroup** (Разгруппировка) — позволяет разгруппировать выделенный элемент управления.
- **Order** (Порядок) — содержит список дополнительных операций, позволяющих изменять порядок отображения выделенных элементов управления на форме пользователя.

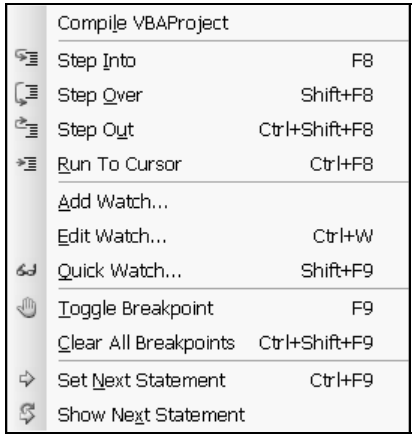


Рис. П3.6. Операции пункта меню **Debug** (Отладка) главного меню

❒ Пункт меню **Debug** (Отладка) содержит следующие операции (рис. П3.6).

- **Compile VBAProject** (Компилировать проект VBA) — выполняет компиляцию редактируемого проекта VBA.
- **Step Into** (Шаг с заходом) — реализует однократное выполнение процедуры вместе с вызываемыми из нее другими процедурами.
- **Step Over** (Шаг с обходом) — реализует однократное выполнение процедуры без выполнения вызываемых из нее других процедур.
- **Step Out** (Шаг с выходом) — реализует прекращение выполнения вызываемой процедуры и возврат к выполнению основной процедуры.
- **Step To Cursor** (Пошаговое выполнение до курсора) — реализует пошаговое выполнение редактируемой процедуры до позиции в тексте, в которой расположен курсор.
- **Add Watch** (Добавить контрольное значение) — открывает дополнительное окно для добавления выражений с целью последующего по-

шагового контроля их значений на этапе отладки редактируемой процедуры.

- **Edit Watch** (Редактировать контрольное значение) — открывает дополнительное окно для редактирования выражений с целью последующего пошагового контроля их значений на этапе отладки процедуры.
- **Quick Watch** (Контрольное значение) — открывает дополнительное окно, в котором содержатся значения заданных выражений при пошаговой отладке редактируемой процедуры.
- **Toggle Breakpoint** (Установить точку останова) — позволяет установить точку останова в тексте редактируемой процедуры.
- **Clear All Breakpoints** (Снять все точки останова) — удаляет все точки останова в тексте редактируемой процедуры.
- **Set Next Statement** (Установить следующий оператор) — позволяет установить точку выполнения на выбранную строку текста программного кода.
- **Show Next Statement** (Показать следующий оператор) — позволяет выделить цветом следующий выполняемый оператор.

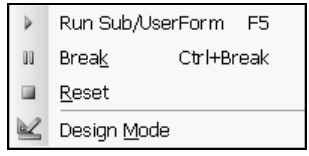


Рис. П3.7. Операции пункта меню **Run** (Выполнить) главного меню

❒ Пункт меню **Run** (Выполнить) содержит следующие операции (рис. П3.7).

- **Run Sub/UserForm** (Выполнить процедуру/ Форму пользователя) — позволяет запустить редактируемую процедуру или форму пользователя на выполнение.
- **Break** (Прервать) — временно останавливает выполнение ранее запущенной процедуры или формы пользователя.
- **Reset** (Сброс) — позволяет очистить стек вызова и обновить значения переменных уровня модуля.
- **Design Mode** (Режим проектирования) — позволяет перейти в режим проектирования формы пользователя.

❒ Пункт меню **Tools** (Сервис) содержит следующие операции (рис. П3.8).

- **References** (Ссылки) — открывает дополнительное окно редактирования ссылок на внешние библиотеки ресурсов и объектов проекта.

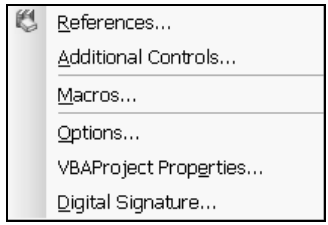


Рис. П3.8. Операции пункта меню **Tools** (Сервис) главного меню

- **Additional Controls** (Дополнительные элементы управления) — открывает дополнительное окно, позволяющее добавить дополнительные элементы управления на специальную панель инструментов, которые впоследствии могут быть размещены на форме пользователя.
- **Macros** (Макросы) — открывает дополнительное окно для редактирования и запуска макросов проекта.
- **Options** (Параметры) — открывает дополнительное окно для редактирования параметров настройки редактора Visual Basic.
- **VBAProject Properties** (Свойства VBA-проекта) — открывает дополнительное окно для редактирования свойств VBA проекта.
- **Digital Signature** (Цифровая подпись) — открывает дополнительное окно для задания цифровой подписи проекта.

❒ Пункт меню **Add-Ins** (Расширения) — содержит единственную операцию **Add-In Manager** (Менеджер расширений), которая вызывает диалоговое окно с информацией об установленных расширениях для их регистрации, загрузки и контроля поведения.

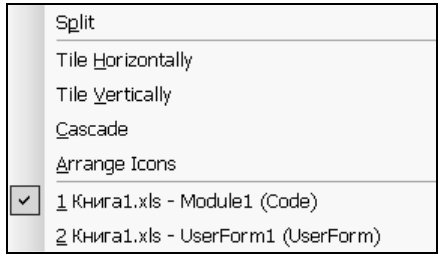


Рис. П3.9. Операции пункта меню **Window** (Окно) главного меню

❒ Пункт меню **Window** (Окно) содержит следующие операции (рис. П3.9).

- **Split** (Разделить) — разделяет окно редактора с текстом программного кода на 2 подокна горизонтальной разделительной линией.

- **Tile Horizontally** (Разместить горизонтально) — позволяет разместить все открытые окна с элементами проекта таким образом, что каждое из них занимает всю ширину основной рабочей области.
- **Tile Vertically** (Разместить вертикально) — позволяет разместить все открытые окна с элементами проекта таким образом, что каждое из них занимает всю высоту основной рабочей области.
- **Cascade** (Разместить каскадом) — позволяет разместить все открытые окна с элементами проекта каскадно в основной рабочей области.
- **Arrange Icons** (Упорядочить пиктограммы) — позволяет упорядочить все открытые окна с элементами проекта в основной рабочей области.
- Секция отображает информацию об открытых окнах редактора с различными элементами проекта для переключения между ними.

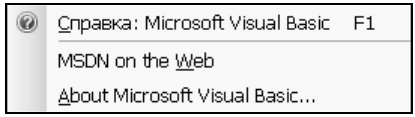


Рис. П3.10. Операции пункта меню **Help** (Справка) главного меню

❐ Пункт меню **Help** (Справка) содержит следующие операции (рис. П3.10).

- **Справка: Microsoft Visual Basic** — открывает окно справочной системы о языке и редакторе MS Visual Basic.
- **MSDN on the Web** (Справка в Интернет от MS Developer Network) — вызывает браузер Интернета, установленный по умолчанию на компьютере пользователя, для отображения справки сети разработчиков MS по адресу: **http://msdn.microsoft.com/developer/default.htm**.
- **About Microsoft Visual Basic** (О редакторе MS Visual Basic) — отображает информацию о текущей рабочей версии редактора MS Visual Basic.

В соответствующих разделах встроенной справки можно получить и дополнительную информацию об операциях главного меню редактора Visual Basic пакета MS Office System 2003.

Список литературы

1. Абрамов Л. М., Капустин В. Ф. Математическое программирование. — Л.: ЛГУ, 1981. — 328 с.
2. Адельсон-Вельский Г. М., Диниц Е. А., Карзанов А. В. Потоковые алгоритмы. — М.: Наука, 1975. — 120 с.
3. Айгнер М. Комбинаторная теория. — М.: Мир, 1982. — 560 с.
4. Айзерман М. А., Алескеров Ф. Т. Выбор вариантов: основы теории. — М.: Наука, 1990. — 240 с.
5. Акулич И. Л. Математическое программирование в примерах и задачах. — М.: Высшая школа, 1986. — 320 с.
6. Алексеев О. Г. Комплексное применение методов дискретной оптимизации. — М.: Наука, 1987. — 248 с.
7. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979. — 536 с.
8. Балашевич В. А. Математические методы в управлении производством. — Минск: Высшая школа, 1976. — 250 с.
9. Банди Б. Методы оптимизации. Вводный курс. — М.: Радио и связь, 1988. — 128 с.
10. Баранов В. И., Стечкин Б. С. Экстремальные комбинаторные задачи и их приложения. — М.: Наука, 1989. — 160 с.
11. Беклемешев Д. В. Дополнительные главы линейной алгебры. — М.: Наука, 1983. — 336 с.
12. Белов В. В., Воробьев Е. М., Шаталов В. Е. Теория графов. — М.: Высшая школа, 1976. — 392 с.
13. Блюмберг В. А., Глущенко В. Ф. Какое решение лучше? Метод постановки приоритетов. — Л.: Лениздат, 1982. — 160 с.
14. Ван Гиг Дж. Прикладная общая теория систем. Том 1, 2. — М.: Мир, 1981. — 732 с.
15. Васильев Ф. П. Лекции по методам решения экстремальных задач. — М.: МГУ, 1974. — 375 с.

16. Васильев Ф. П. Методы решения экстремальных задач. — М.: Наука, 1981. — 400 с.

17. Васильев Ф. П. Численные методы решения экстремальных задач. — М.: Наука, 1980. — 520 с.

18. Виленкин Н. Я. Комбинаторика. — М.: Наука, 1969. — 328 с.

19. Волгин Л. Н. Принцип согласованного оптимума. — М.: Сов. радио, 1977. — 144 с.

20. Вязгин В. А., Федоров В. В. Математические методы автоматизированного проектирования. — М.: Высшая школа, 1989. — 184 с.

21. Гаас С. Линейное программирование. — М.: ГИФМЛ, 1961. — 304 с.

22. Гаас С. Путешествие в страну линейного программирования. — М.: Мир, 1973. — 176 с.

23. Гавурин М. К., Малоземов В. Н. Экстремальные задачи с линейными ограничениями. — Л.: ЛГУ, 1984. — 176 с.

24. Галеев Э. М., Тихомиров В. М. Краткий курс теории экстремальных задач. — М.: МГУ, 1989. — 204 с.

25. Гермейер Ю. Б. Введение в теорию исследования операций. — М.: Наука, 1971. — 384 с.

26. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. — М.: Мир, 1985. — 512 с.

27. Гирсанов И. В. Лекции по математической теории экстремальных задач. — М.: МГУ, 1970. — 120 с.

28. Гольштейн Е. Г. Теория двойственности в математическом программировании и ее приложения. — М.: Наука, 1971. — 352 с.

29. Горбатов В. А. Основы дискретной математики. — М.: Высшая школа, 1986. — 312 с.

30. Губанов В. А., Захаров В. В., Коваленко А. Н. Введение в системный анализ. — Л.: ЛГУ, 1988. — 232 с.

31. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982. — 416 с.

32. Даффин Р. и др. Геометрическое программирование. — М.: Мир, 1972. — 312 с.

33. Дегтярев Ю. И. Методы оптимизации. — М.: Сов. радио, 1980. — 272 с.

34. Дегтярев Ю. Н. Исследование операций. — М.: Высшая школа, 1986. — 320 с.

35. Дубов Ю. А., Травкин С. И., Якимец В. Н. Многокритериальные модели формирования и выбора вариантов систем. — М.: Наука, 1986. — 296 с.

36. Евстигнеев В. А. Применение теории графов в программировании. — М.: Наука, 1985. — 352 с.

37. Евтушенко Ю. Г. Методы решения экстремальных задач и их применение в системах оптимизации. — М.: Наука, 1982. — 432 с.

38. Емеличев В. А., Ковалев М. М., Кравцов М. К. Многогранники, графы, оптимизация (комбинаторная теория многогранников). — М.: Наука, 1981. — 208 с.

39. Емельянов С. В., Ларичев О. И. Многокритериальные методы принятия решений. — М.: Знание, 1985. — 32 с.

40. Ермольев Ю. М., Ляшко И. И., Михалевич В. С., Тюптя В. И. Математические методы исследования операций. — Киев.: Вища школа, 1979. — 312 с.

41. Зайченко Ю. П. Исследование операций. — Киев.: Высшая школа, 1979.— 392 с.

42. Зангвилл У. И. Нелинейное программирование. — М.: Сов. радио, 1973. — 312 с.

43. Заславский Ю. Л. Сборник задач по линейному программированию. — М.: Наука, 1969. — 256 с.

44. Зыков А. А. Основы теории графов. — М.: Наука, 1987. — 384 с.

45. Исследование операций/Под ред. Дж. Моудера, С. Элмаграби. Том 1, 2. — М.: Мир, 1981. — 712 с., — 677 с.

46. Исследования по дискретной оптимизации/Под ред. А. А. Фридмана. — М.: Наука, 1976. — 448 с.

47. Исследования по теории структур./Под ред. М. А. Айзермана, Э. Р. Каянелло. — М.: Наука, 1988. — 208 с.

48. Калинин В. Н., Резников Б. А. Теория систем и управления (структурно-математический подход). — Л.: ВИКИ им. А. Ф. Можайского, 1987. — 417 с.

49. Калинин В. Н., Резников Б. А., Варакин Е. И. Теория систем и оптимального управления. Часть 1 — МО СССР, 1979. — 319 с.

50. Калинин В. Н., Резников Б. А., Варакин Е. И. Теория систем и оптимального управления. Часть 2. — МО СССР, 1987. — 589 с.

51. Калихман И. Л. Сборник задач по линейной алгебре и программированию. — М.: Высшая школа, 1969. — 160 с.

52. Карлин С. Математические методы в теории игр, программировании и экономике. — М.: Мир, 1964. — 838 с.

53. Карманов В. Г. Математическое программирование. — М.: Наука, 1980. — 256 с.

54. Кини Р. Л., Райфа Х. Принятие решений при многих критериях. — М.: Радио и связь, 1981. — 560 с.

55. Клир Дж. Системология. Автоматизация решения системных задач. — М.: Радио и связь, 1990. — 544 с.

56. Конвей Р. В., Максвелл В. Л., Миллер Л. В. Теория расписаний. — М.: Наука, 1975. — 360 с.

57. Корбут А. А., Финкельштейн Ю. Ю. Дискретное программирование. — М.: Наука, 1969. — 368 с.

58. Кофман А., Анри-Лабордер А. Методы и модели исследования операций. — М.: Мир, 1977. — 432 с.

59. Кузнецов О. П., Адельсон-Вельский Г. М. Дискретная математика для инженера. — М.: Энергия, 1980. — 344 с.

60. Куратовский К., Мостовский А. Теория множеств. — М.: Мир, 1970. — 416 с.

61. Ларичев О. И. Объективные модели и субъективные решения. — М.: Наука, 1987. — 144 с.

62. Лекции по теории графов/В. А. Емеличев, О. И. Мельников, В. И. Сарванов и др. — М.: Наука, 1990. — 384 с.

63. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. — СПб.: БХВ-Петербург, 2003. — 736 с.

64. Леоненков А. В. Самоучитель UML. 2-е изд. — СПб.: БХВ-Петербург, 2004. — 432 с.

65. Линейное и нелинейное программирование/Ляшенко И. Н. и др. — Киев: Выща школа, 1975. — 372 с.

66. Лэсдон Л. С. Оптимизация больших систем. — М.: Наука, 1975. — 432 с.

67. Майника Э. Алгоритмы оптимизации на сетях и графах. — М.: Мир, 1981. — 328 с.

68. Месарович М., Мако Д., Такахара И. Теория иерархических многоуровневых систем. — М.: Мир, 1973. — 344 с.

69. Михалевич В. С., Волкович В. Л. Вычислительные методы исследования и проектирования сложных систем. — М.: Наука, 1982. — 288 с.

70. Михалевич В. С., Кукса А. И. Методы последовательной оптимизации. — М.: Наука, 1983. — 208 с.

71. Многокритериальные задачи принятия решений/Под ред. Д. Н. Гвишкана и С. В. Емельянова. — М.: Машиностроение, 1978. — 192 с.

72. Моисеев Н. Н. Математические задачи системного анализа. — М.: Наука, 1981. — 488 с.

73. Моисеев Н. Н., Иванилов Ю. П., Столярова Е. М. Методы оптимизации. — М.: Наука, 1978. — 352 с.

74. Морозов В. В. и др. Исследование операций в задачах и упражнениях. — М.: Высшая школа, 1986. — 287 с.

75. Муртаф Б. Современное линейное программирование. — М.: Мир, 1984. — 224 с.

76. Нейман Дж., Моргенштерн О. Теория игр и экономическое поведение. — М.: Наука, 1970. — 708 с.

77. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. — М.: Мир, 1985. — 512 с.

78. Перегудов Ф. И., Тараненко Ф. П. Введение в системный анализ: Учебное пособие для вузов. — М.: Высшая школа, 1989.— 368 с.

79. Подиновский В. В. Математическая теория выработки решений в сложных ситуациях. — МО СССР, 1981. — 256 с.

80. Подиновский В. В., Гаврилов В. М. Оптимизация по последовательно применяемым критериям. — М.: Сов. радио, 1975. — 192 с.

81. Подиновский В. В., Ногин В. Д. Парето-оптимальные решения многокритериальных задач. — М.: Наука, 1982.— 254 с.

82. Полак Б. Т. Введение в оптимизацию. — М.: Наука, 1983. — 384 с.

83. Полак Э. Численные методы оптимизации. — М.: Мир, 1974. — 376 с.

84. Резников Б. А. Методы и алгоритмы оптимизации на дискретных моделях сложных систем. — Л.: ВИКИ, 1983. — 250 с.

85. Резников Б. А. Системный анализ и методы системотехники. Часть 1. Методология системных исследований. Моделирование сложных систем. — М.: МО СССР, 1990. — 522 с.

86. Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. — М.: Мир, 1980. — 480 с.

87. Рихтер К. Динамические задачи дискретной оптимизации. — М.: Радио и связь, 1985. — 136 с.

88. Рыбников К. А. Введение в комбинаторный анализ. — М.: МГУ, 1972. — 256 с.

89. Саати Т. Целочисленные методы оптимизации и связанные с ними экстремальные проблемы. — М.: Мир, 1973. — 304 с.

90. Саати Т., Кернс К. Аналитическое планирование. Организация систем. — М.: Радио и связь, 1991. — 224 с.

91. Сачков В. Н. Введение в комбинаторные методы дискретной оптимизации. — М.: Наука, 1982. — 384 с.

92. Свами М., Тхуласираман К. Графы, сети и алгоритмы. — М.: Мир, 1984. — 456 с.

93. Сергиенко Н. В. Математические модели и методы решения задач дискретной оптимизации. — Киев: Наукова Думка, 1985. — 384 с.

94. Системный анализ в экономике и организации производства/Под ред. С. А. Валуева, В. Н. Волковой. — Л.: Политехника, 1991. — 398 с.

95. Скорняков Л. А. Элементы теории структур. — М.: Наука, 1982. — 160 с.

96. Современное состояние теории исследования операций/Под ред. Н. Н. Моисеева. — М.: Наука, 1979. — 464 с.

97. Сухарев А. Г., Тимохов А. В., Федоров В. В. Курс методов оптимизации. — М.: Наука, 1986.— 328 с.

98. Танаев В. С. Теория расписаний. — М.: Знание, 1988. — 32 с.

99. Танаев В. С., Шкурба В. В. Введение в теорию расписаний. — М.: Наука, 1975.— 256 с.

100. Тараканов В. Е. Комбинаторные задачи и (0, 1) — матрицы. — М.: Наука, 1985. — 192 с.

101. Таха Х. Введение в исследование операций. Том 1, 2. — М.: Мир, 1985. — 480 с., — 495 с.

102. Теория расписаний и вычислительные машины/Под ред. Э. Г. Кофмана. — М.: Наука, 1984. — 336 с.

103. Технология системного моделирования/Под общ. ред. акад. С. В. Емельянова. — М.: Машиностроение, — Берлин: Техник, 1988. — 520 с.

104. Уилсон Р. Введение в теорию графов. — М.: Мир, 1977. — 246 с.

105. Филлипс Д., Гарсиа-Диас А. Методы анализа сетей. — М.: Мир, 1984. — 496 с.

106. Форд Л., Фалкерсон Д. Потоки в сетях. — М.: Мир, 1966. — 276 с.

107. Харари Ф. Теория графов. — М.: Мир, 1973. — 304 с.

108. Хачиян Л. Г. Сложность задач линейного программирования.— М.: Знание, 1987. — 32 с.

109. Ху Т. Целочисленное программирование и потоки в сетях. — М.: Мир, 1974. — 520 с.

110. Цурков В. И. Декомпозиция в задачах большой размерности. — М.: Наука, 1981. — 352 с.

111. Юдин Д. Б. Вычислительные методы теории принятия решений. — М.: Наука, 1989. — 320 с.

112. Юдин Д. Б., Гольштейн Е. Г. Линейное программирование. Теория, методы и приложения. — М.: Наука, 1969. — 424 с.

113. Яблонский С. В. Введение в дискретную математику. — М.: Наука, 1979. — 272 с.

Предметный указатель

А

Автозаполнение ячеек 73
Аддитивность 155
Активная ячейка 71
Алгоритм:
  метода минимального отклонения 410
  метода уступок 408
  пометок Дейкстры 321
  пометок Форда — Фалкерсона 350
  расстановки постоянных пометок 337
Альтернатива 42

Б

Библиотека динамической компоновки 596
Булеан 636

В, Г

Венгерский метод 283
Графическое решение 106

Д

Двойственная задача ЛП 175
Диаграммы Венна 625
Диапазон ячеек 72
Дополнение множества 632

Ж

Жадный алгоритм 308
  модифицированный 312

З

Задача:
  водопроводчика 26
  многокритериальной оптимизации 399
  коммивояжера 20
Запуск подпрограмм VBA 528

И

Идеальная точка, 410

К

Комбинаторная оптимизация 359
Комбинаторный анализ, 655
Композиция бинарных отношений 648
Критериальная функция 44
Критический путь 330

Л

Локальная окрестность решения 567

М

Мастер:
  диаграмм 88
  функций 78
  подбора параметра 162
  поиска решения 110
Математическая модель 41

Метод:
локальной оптимизации 568, 582
минимального элемента 193
потенциалов 188
Модуль 508
Мощность множества 627

Н

Надстройка пользователя 594
Нелинейное
программирование 103
Неотрицательность 155

О

Обратное отношение 647
Объединение множеств 630
Ограничение 42
Оператор:
безусловного перехода 515
условного перехода 515
Оптимальное решение 48
Отношение 638
доминирования по Парето, 401
толерантности, 653
эквивалентности 654
Отображение 649

П

Параметры поиска решения 113
Пересечение множеств 629
Перестановка 655
Покрывающее или остовное
дерево 298
Приближенное решение 107
Программирование:
булево 253
геометрическое 232
динамическое 262
линейное 141
целочисленное линейное 205
Процедура VBA 510

Р

Размещение, 656
Разность множеств, 631
Решение:
аналитическое 105
алгоритмическое 106

С

Симметрическая разность
множеств 632
Симплекс-метод, 165
Системное моделирование 27
Сочетание 656

Ф

Форматирование данных 71
Формула 77
Функция:
пользователя 509
целевая 44

Ц

Цикл:
с параметрами 515
с предусловием 516

Э

Экспорт модуля VBA 591
Эмерджентность 27