Прикладная математика,
информатика,
информационные технологии

Э.И. Ватутин,
В.С. Титов,
С.Г. Емельянов

ОСНОВЫ ДИСКРЕТНОЙ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ

ПРЕДИСЛОВИЕ

Современный мир сложно представить себе без компьютеров и цифровой электроники, автоматизированных средств их проектирования (например, соответствующих САПРов и оптимизирующих компиляторов), средств планирования производства и т.п. Многие прикладные задачи и направления, возникающие при этом, требуют решения соответствующих фундаментальных задач, относящихся к областям теории графов, теории компиляции, теории расписаний, исследованию логических игр и пр. и оперирующих дискретными объектами. Многие из указанных задач являются экстремальными или оптимизационными, т.е. требуют выполнения оптимизации заданной математической модели с целью отыскания решения, характеризующегося минимумом или максимумом какой-либо функции или их совокупности (стоимости, аппаратной сложности, массогабаритных параметров и т.п.). Для их решения зачастую не применим обширный математический аппарат, используемый при оптимизации гладких дифференцируемых функций и базирующийся на понятии градиента, что выделяет указанные задачи в отдельный класс и вынуждает использовать для их решения специальные методы, от одних названий которых (например, «жадные методы» или «перебор грубой силой») ученым мужам, далеким от данной предметной области, временами становится плохо. А если при их решении используется колония муравьев (хорошо не тараканов, однако исследования в направлении биоинспирированных методов активно продолжаются) или пчел, то исследующему их ученому-математику пора на природу на отдых, а не в лабораторию за компьютер. Однако указанные задачи имеют важное практическое значение в различных областях науки и техники, следовательно их необходимо решать, а значит и с названиями методов придется мириться. Убедиться в том, хорош или плох выбранный метод, пусть он и имеет подозрительное название, достаточно просто: с его использованием необходимо найти решение известной задачи, качество которого будет превосходить известные решения.

Многие эвристические методы были разработаны и начали активно применятся относительно недавно (несколько десятков лет назад), соответственно их свойства и особенности применения на практике не до конца исследованы, поняты и приняты научным сообществом, что делает рассматриваемую область науки в достаточной степени новой и открывает простор для вычислительных экспериментов, направленных на изучение поведения методов в различных условиях.

Данное учебное пособие посвящено рассмотрению основных методов, подходов и стратегий, лежащих в основе алгоритмов и соответствующих им практических (в основном программных, хотя известны и

аппаратные) реализаций, направленных на решение задач дискретной комбинаторной оптимизации. При изложении материала рассмотрены известные классификации методов решения, приведены примеры практического использования методов, подробное описание соответствующих алгоритмов и программных реализаций. Последние являются в полной мере рабочими (в отличие, например, от псевдокода, на котором зачастую в литературе приводится краткое описание соответствующих методов) и могут быть взяты за основу как для снижения затрат вычислительного времени (например, в ходе оптимизации или распараллеливания), необходимых на отыскание решений, так и для повышения качества решений.

Задачи указанного «дискретного» направления можно рассматривать с позиции дискретной математики, приводя изложение теоретических положений, доказательства теорем и т.п. Указанные сущности безусловно являются важными, однако в данном учебном пособии использована иная стратегия изложения материала, в основе которой лежит практический «алгоритмический» взгляд на проблему. На наш взгляд ее использование позвляет беспрепятственное решение рассмотренных в книге задач и схожих с ними с использованием современных инстрементариев для разработки программных средств, не ограничивающихся Delphi, т.к. в современном мире указанные задачи без компьютера и соответствующего программного обеспечения, обычно разрабатываемого самостоятельно, не решаются. Приводимые примеры программ являются полными и работоспособными, они могут быть скопированы в соответствующую среду разработки и запущены на исполнение без каких-либо модификаций, что должно облегчить начальное освоение методов дискретной комбинаторной оптимизации.

В книге изложен материал, легший в основу курса «Основы комбинаторной оптимизации», читаемого на кафедре вычислительной техники Юго-Западного государственного университета. Учебное пособие соответствует Государственному образовательному стандарту высшего (профессионального) образования для направления подготовки 090301 и 090401 «Информатика и вычислительная техника» и может быть использовано студентами, магистрантами и аспирантами как для изучения основ указанной дисциплины, так и для углубления имеющихся знаний.

Авторы выражают благодарность и глубокую признательность рецензентам – Ларкину Е.В. и Леньшину А.В. и своим коллегам и товарищам – Григору В.В., Носову В.А. и Рожкову А.А. – за ряд ценных замечаний и предложений, поступивших в процессе подготовки материала книги к изданию, а также сотрудникам редакции за их нелегкий труд по подготовке пособия к печати. От лица авторского коллектива Ватутин Э.И. выражает глубокую благодарность своему школьному

учителю основ алгоритмизации, программирования и информатики – Киселеву Р.В., который на высоком профессиональном уровне, значительно превышавшем требования школьной программы того времени, не говоря про современное школьное образование, дал основы программирования, в том числе олимпиадного уровня, показал некоторые из задач и приемов их решения, которые нашли отражение в книге, а также обеспечил первую возможность для их практического программирования. Также Ватутин Э.И. выражает глубокую признательность своей жене – Ватутиной О.О. и ближайшим членам семьи за то терпение, понимание, надежную тыловую поддержку и посильную помощь, которых хватило вплоть до завершения работы над книгой выходными и бессонными ночами в свободное от бумажной кафедральной работы время.

Несмотря на тщательную кропотливую работу авторов, рецензентов и сочувствующих им лиц скорее всего книга не лишена мелких недостатков. Поэтому в случае их обнаружения просьба присылать конструктивные замечания и предложения на электронную почту по адресу evatutin@rambler.ru либо любым другим доступным образом.

Глава 1.
ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

1.1. Понятие задач оптимизации. Постановка задачи. Классы задач. Виды ограничений

Существует обширный класс задач, имеющих важное фундаментальное и/или прикладное значение, в рамках которых возникает необходимость в минимизации или максимизации функций различного вида (говоря другими словами, в отыскании их экстремумов). Данный раздел математики называется *оптимизацией* и включает в себя большое число задач различного вида и подходов к их решению. Математические методы оптимизации не следует путать с оптимизацией программных средств [1, 2] (как, например, математическое программирование не имеет ничего общего с программированием на языках высокого уровня), целью которой является снижение времени выполнения программы и/или объема памяти. Указанный набор приемов имеет важное значение в программировании, однако он выходит за пределы данной книги и далее под оптимизацией будем понимать именно ее математическую трактовку, если не оговорено иное.

Оптимизационные задачи могут быть разделены на два существенно различных класса: задачи *непрерывной* (гладкой) и задачи *дискретной* оптимизации. Их основным отличием являются типы функций, для которых производится отыскание экстремумов. В первом случае это, как правило, гладкие дифференцируемые функции (в некоторых случаях с рядом особенностей), определенные на множестве вещественных или комплексных значений аргументов (как правило, бесконечном или несчетном); во втором – как правило, функции от какого-либо дискретного аргумента или дискретной математической структуры (например, счетного множества, графа или матроида). Задачи непрерывной оптимизации известны с античных времен и возникают в ряде областей науки (физика, астрономия, теоретическая механика, экономика и пр.), и их решению посвящено достаточно большое количество литературы. Задачи дискретной оптимизации получили активное распространение с конца XIX – начала XX века в связи с развитием дискретной прикладной математики и ее составляющих, таких как теория графов, теория игр, исследование операций, математическое целочисленное программирование, теория расписаний.

В результате развития соответствующих направлений науки в XVII – XX вв. в работах Готфрида Лейбница (G. Leibniz), Леонарда Эйлера (L. Euler) и др. известных ученых зародилось новое направление, именуемое *комбинаторикой*, в рамках которого изучаются дискретные объ-

екты, их свойства и отношения между ними. Примерами комбинаторных объектов, изучаемых в рамках соответствующего направления, являются счетные множества и мультимножества, графы и гиперграфы, расписания различного вида (например, расписание занятий школы или университета, расписание движение поездов или самолетов, сетевой график работ и т.д.), логические игры (игральные карты, шахматы, пятнашки и т.д.), различные математические структуры (например, перестановки, группы, латинские квадраты и т.д.). Приведем несколько примеров задач, являющихся комбинаторными.

Задача 1.1. Какова вероятность того, что среди 4 карт колоды будут вытащены карты одинаковой масти?

Задача 1.2 (о ладьях). Сколькими способами можно расставить на шахматной доске размером $N \times N$ клеток N ладей так, чтобы они не атаковали друг друга?

Задача 1.3. Найти количество путей в графе между парой выбранных вершин.

Задача 1.4. Найти латинский квадрат порядка N с заданными свойствами.

Задача 1.5. Найти оптимальную линейку Голомба порядка N.

Как правило, комбинаторные задачи подразумевают организацию перебора того или иного множества решений с выполнением каких-либо дополнительных действий, например, подсчета числа решений или выбора одного из них. В некоторых частных случаях они допускают аналитическое решение, например, методами теории вероятностей, математической статистики, теории систем массового обслуживания и т.д. Примеры решения некоторых комбинаторных задач приведены ниже.

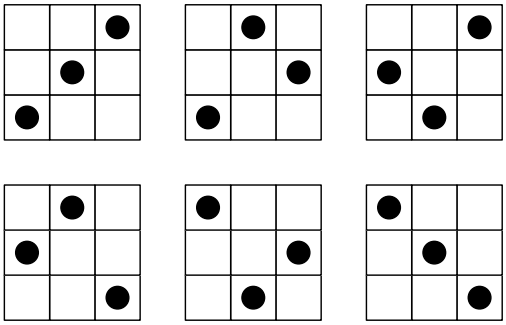


Рис. 1.1. Пример решения задачи о ладьях: для случая $N = 3$ существует $N! = 6$ решений

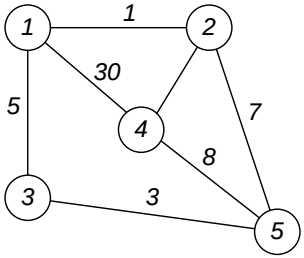


Рис. 1.2. Поиск пути в графе между вершинами 1 и 5: существуют 5 путей $P_1 = [1, 2, 4, 5]$, $P_2 = [1, 2, 5]$, $P_3 = [1, 3, 5]$, $P_4 = [1, 4, 2, 5]$ и $P_5 = [1, 4, 5]$ с длинами $L(P_1) = 14$, $L(P_2) = 8$, $L(P_3) = 8$, $L(P_4) = 42$, $L(P_5) = 38$, кратчайшими являются пути P_2 и P_3

а)

$$\begin{pmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
1 & 2 & 9 & 4 & 3 & 6 & 7 & 8 & 0 & 8 \\
2 & 9 & 3 & 1 & 7 & 0 & 5 & 8 & 4 & 6 \\
3 & 4 & 1 & 2 & 8 & 7 & 9 & 6 & 5 & 0 \\
4 & 3 & 5 & 9 & 2 & 1 & 8 & 0 & 6 & 7 \\
5 & 6 & 4 & 8 & 1 & 2 & 0 & 9 & 7 & 3 \\
6 & 5 & 8 & 7 & 0 & 3 & 2 & 1 & 9 & 4 \\
7 & 8 & 6 & 0 & 9 & 4 & 1 & 2 & 3 & 5 \\
8 & 7 & 0 & 5 & 6 & 9 & 3 & 4 & 1 & 2 \\
9 & 0 & 7 & 6 & 5 & 8 & 4 & 3 & 2 & 1
\end{pmatrix}$$

б)

$$\begin{pmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
7 & 2 & 4 & 9 & 0 & 6 & 5 & 1 & 3 & 8 \\
8 & 3 & 6 & 7 & 5 & 9 & 0 & 2 & 4 & 1 \\
2 & 6 & 8 & 5 & 1 & 7 & 4 & 0 & 9 & 3 \\
5 & 8 & 9 & 1 & 7 & 0 & 3 & 4 & 6 & 2 \\
9 & 4 & 1 & 2 & 8 & 3 & 7 & 6 & 0 & 5 \\
4 & 7 & 5 & 6 & 9 & 1 & 8 & 3 & 2 & 0 \\
3 & 0 & 7 & 8 & 2 & 4 & 1 & 9 & 5 & 6 \\
6 & 5 & 0 & 4 & 3 & 2 & 9 & 8 & 1 & 7 \\
1 & 9 & 3 & 0 & 6 & 8 & 2 & 5 & 7 & 4
\end{pmatrix}$$

в)

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
1	*2*	*0*	*4*	*3*	*7*	*9*	*8*	*5*	*6*
7	*3*	*5*	*9*	*0*	*4*	*8*	*6*	*2*	*1*
3	*5*	*6*	*8*	*9*	*0*	*4*	*1*	*7*	*2*
4	*9*	*7*	*2*	*6*	*8*	*1*	*5*	*0*	*3*
5	*8*	*4*	*6*	*7*	*1*	*3*	*2*	*9*	*0*
8	*4*	*9*	*1*	*2*	*3*	*7*	*0*	*6*	*5*
6	*7*	*3*	*0*	*1*	*2*	*5*	*9*	*4*	*8*
9	*0*	*1*	*5*	*8*	*6*	*2*	*4*	*3*	*7*
2	*6*	*8*	*7*	*5*	*9*	*0*	*3*	*1*	*4*

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
7	*5*	*1*	*9*	*2*	*8*	*0*	*4*	*6*	*3*
1	*0*	*3*	*4*	*6*	*7*	*5*	*2*	*9*	*8*
9	*8*	*4*	*7*	*5*	*2*	*1*	*0*	*3*	*6*
6	*7*	*9*	*0*	*8*	*3*	*2*	*1*	*5*	*4*
4	*6*	*5*	*1*	*0*	*9*	*8*	*3*	*2*	*7*
2	*3*	*8*	*5*	*1*	*6*	*4*	*9*	*7*	*0*
5	*2*	*7*	*8*	*3*	*4*	*9*	*6*	*0*	*1*
3	*4*	*6*	*2*	*9*	*0*	*7*	*8*	*1*	*5*
8	*9*	*0*	*6*	*7*	*1*	*3*	*5*	*4*	*2*

Рис. 1.3. Примеры латинского квадрата (а), диагонального латинского квадрата (б) и пары ортогональных диагональных латинских квадратов (в) порядка 10 [3–5]

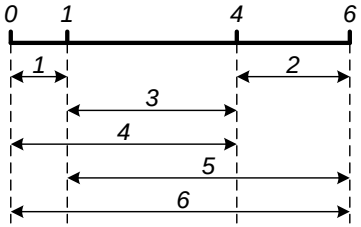


Рис. 1.4. Пример линейки Голомба 4-го порядка длины 6: расстояние между любой парой из 4 штрихов на числовой прямой уникально, расстояние между наиболее удаленными штрихами равно 6. Приведенная линейка оптимальна и совершенна [6, 7]

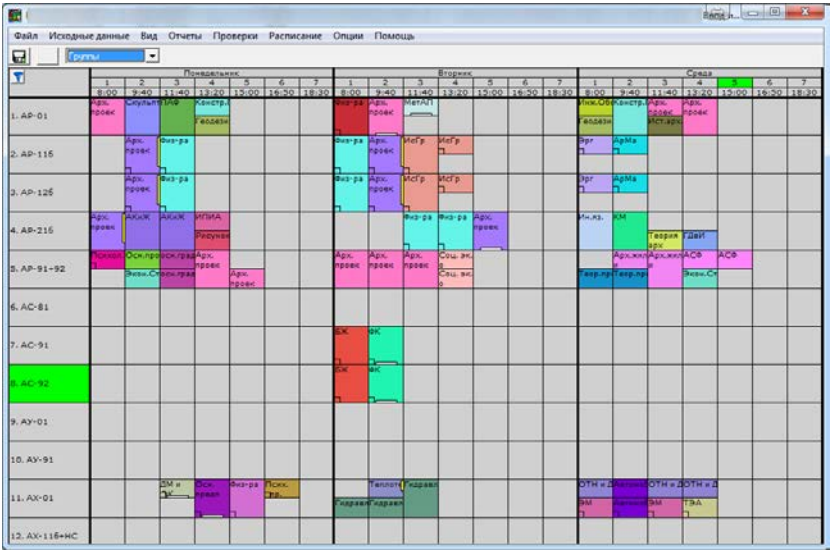


Рис. 1.5. Пример расписания занятий студенческих групп университета [8–10]

Подобные задачи теоретического плана находят применение при решении ряда практически важных задач: например, при прокладке маршрутов с использованием GPS-навигатора решается задача поиска кратчайшего пути в графе большой размерности; латинские квадраты находят применение в теории кодирования; линейки Голомба применяются при построении базовых станций CDMA (GSM), а также при проектировании фазированных антенных решеток и радиотелескопов.

При решении задач непрерывной оптимизации зачастую с успехом применяется математический аппарат, основанный на использовании производных и градиентов (более подробно данные задачи будет рассмотрены ниже). В задачах дискретной оптимизации он бесполезен ввиду того, что понятие производной для дискретных объектов не определено. Благодаря этому данные задачи выделяются в отдельный класс и решаются с использованием ряда специализированных подходов, подробно рассматриваемых ниже по мере изложения материала книги. Среди них можно выделить следующие основные подклассы задач:

- задачи на существование решения;
- задачи на поиск (суб)оптимального решения;
- задачи на пересчет объектов.

Для *задач на существование решения* сам факт наличия решения, удовлетворяющего всем поставленным условиям, неочевиден. В качестве примера подобной задачи можно рассмотреть задачу построения пар и троек попарно ортогональных диагональных латинских квадратов (ДЛК) порядка 10. Если просто ДЛК указанного порядка с успехом находятся с использованием ряда методов (например, полным перебором или с использованием взвешивающих эвристик [4]), то нахождение пары ортогональных ДЛК уже представляет собой нетривиальную задачу, решение которой может быть найдено с применением специализированных подходов [3, 5]. Существование тройки попарно ортогональных ДЛК на сегодняшний день не доказано [11], однако также не доказана невозможность ее построения. Задачи данного направления являются наиболее вычислительно сложными.

Задачи на поиск (суб)оптимального решения (задачи на оптимальность) как правило имеют множество корректных решений и некоторую целевую функцию, с использованием которой возможна оценка качества каждого из решений и выбор наилучших из них. Если найденное решение лучше всех остальных либо есть группа решений с совпадающим качеством и это доказано аналитически либо в ходе вычислительного эксперимента, оно называется *оптимальным*.

Получение оптимальных решений для задач некоторых типов не представляет труда, однако для большинства остальных из них поиск оптимальных решений затруднен ввиду необходимости комбинаторного перебора очень большого числа решений (например, пропорционального факториалу или экспоненте от размерности задачи), поэтому зачастую вместо оптимальных довольствуются решениями, которые являются неплохими по качеству, но уступающими оптимальному (либо их оптимальность не доказана). Т.е. теоретически возможно построение еще более хорошего решения по сравнению с найденным, однако на практике для этого могут потребоваться огромные затраты вычислительных ресурсов, что обычно нецелесообразно. Если для некоторого найденного решения известно, что оно хуже оптимального на некоторую известную величину (например, на 5%), данное решение называется *субоптимальным*. Если найденное решение лучше остальных, однако при этом неизвестно, насколько оно далеко от оптимального, оно называется *квазиптимальным*.

Существуют задачи, для которых отыскание оптимальных решений не представляет особого труда (например, кратчайший путь в графе находится с использованием алгоритма Дейкстры за квадратичное от числа вершин время [12]). Для некоторых задач решение в общем виде затруднительно, однако в некоторых частных случаях оно может быть найдено достаточно быстро (например, в задачах изоморфизма графов

особого вида [13] или оценки степени параллелизма граф-схемы алгоритма [14, 15], несмотря на то, что общие задачи изоморфизма и выделения максимальной по включению клики для графов общего вида на сегодняшний день не имеют быстрого алгоритма решения). Решения некоторых задач являются субоптимальными по одним показателям качества и квазиоптимальными по другим. Например, разбиения граф-схем параллельных алгоритмов субоптимальны по числу блоков и квазиоптимальны по остальным частным показателям качества [15, 16].

Методы, гарантирующие получение оптимальных решений, называются *точными* или *оптимальными*. Если получение оптимальных решений не гарантируется, соответствующий метод называется *приближенным* или *эвристическим* (основанным на применении тех или иных правил, именуемых *эвристиками*, с использованием которых достигается повышение качества решений или снижение затрат вычислительного времени).

В *задачах на пересчет* производится оценка:

- числа решений, например, в задаче о ладьях или ферзях;
- его асимптотического поведения, например, в задаче поиска латинских квадратов, где точное число решений заданного порядка неизвестно, однако известны его асимптотики;
- вероятностей наступления того или иного события как, например, в задаче 1.1.

Основу материала данной книги составляют задачи на оптимальность. Некоторые задачи на существование могут быть эффективно сведены к задачам на оптимальность [4, 17]. Задачи на пересчет, которые часто встречаются в теории вероятностей и математической статистике, в некоторых случаях также могут быть приближенно решены с использованием эвристических методов, основным назначением которых является решение задач на оптимальность.

Как уже было отмечено выше, для решения задач дискретной оптимизации существует множество подходов, которые можно разделить на два существенно отличающихся друг от друга класса [15, 16, 18].

К первому из них относятся *универсальные* подходы, которые могут применяться к решению различных задач (причем не обязательно дискретных) с незначительными изменениями. В качестве типичного примера можно рассмотреть группу стохастических методов, называемых также методами Монте-Карло. Они могут быть использованы для существенно различных целей начиная от грубой оценки числа π, например, с использованием игл Бюффона или бросанием песчинок в круг известного радиуса [19] или значений интегралов и заканчивая решением задач дискретной или непрерывной оптимизации [21, 22]. К ним также относятся методы полного перебора и жадные алгоритмы.

Ко второму классу относятся *специализированные* подходы, которые базируются на использовании специфики решаемой задачи или группы схожих задач, что позволяет существенно повысить качество решений, снизить затраты времени на их нахождение, либо добиться обеих целей одновременно. Так, например, для решения задачи построения минимального остовного дерева [23] может быть использован алгоритм Прима-Краскала [24, 25]; поиск кратчайших путей в графах без отрицательных весов эффективно реализуется с использованием уже упомянутого выше алгоритма Дейкстры; в задачах трассировки межсоединений при проектировании топологии печатных плат или полупроводниковых кристаллов применяется волновой алгоритм (поиск в ширину) [26, 27]; для поиска максимального паросочетания в двудольном графе – алгоритмы пополняющего пути или Хопкрофта-Карпа [28] (рис. 1.6).

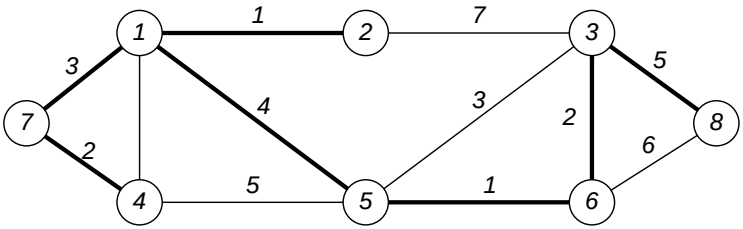


Рис. 1.6. Пример решения задачи построения минимального остовного дерева (выделено жирным)

Еще одним вариантом классификации методов решения задач дискретной комбинаторной оптимизации является их деление на *последовательные* и *итерационные*.

Последовательные методы обеспечивают получение единственного решения, которое может быть как оптимальным, так и суб- или квазиоптимальным. Данные методы, как правило, являются достаточно быстрыми, однако их недостатком является относительно низкое качество получаемых решений. В качестве примера методов данного направления можно привести жадные методы, более подробно рассмотренные в разд. 2.4.

Итерационные методы, в отличие от последовательных, обеспечивают получение нескольких решений с последующим выбором наилучшего из них. Качество решений, получаемых с их помощью, как правило, оказывается существенно выше, однако необходимые затраты времени также значительно выше, при этом обычно существует некоторая нелинейная зависимость между качеством найденного наилучшего решения и затратами времени (числом итераций).

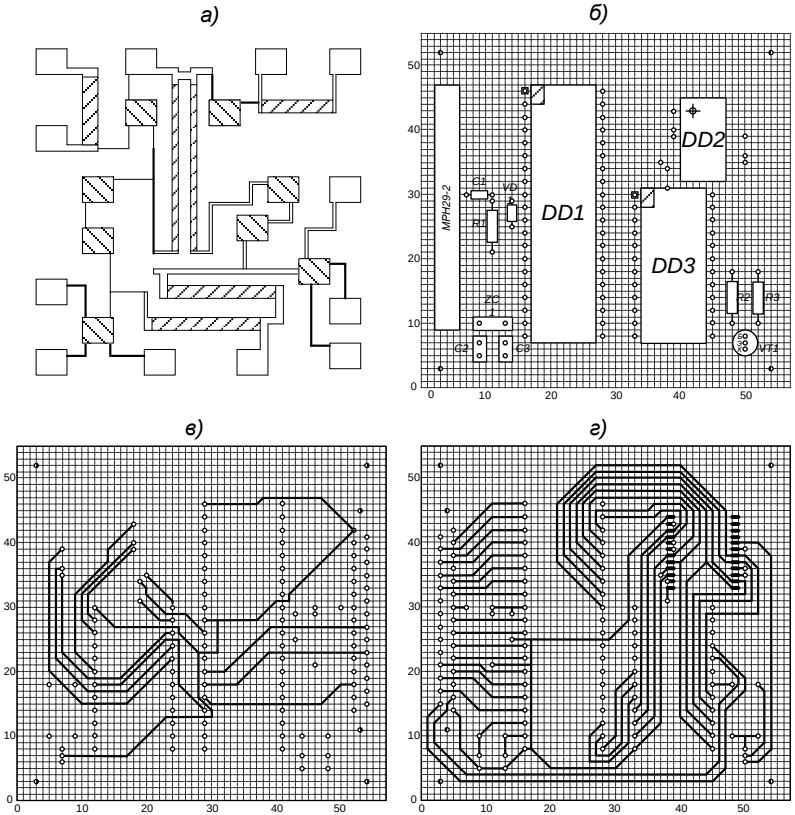


Рис. 1.7. Примеры решения задачи трассировки межсоединений: полупроводниковый кристалл (а), схема расположения элементов на печатной плате (б), два слоя межсоединений (в, г)

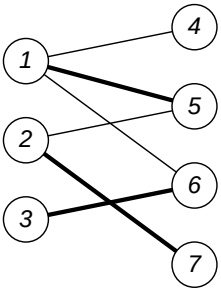


Рис. 1.8. Двудольный граф и пример решения задачи о максимальном паросочетании

Так, например, при синтезе разбиений жадным подходам [29, 30] необходимо порядка нескольких секунд времени процессора, в то время как метод случайного перебора [21, 22] требует до нескольких десятков часов, при этом в некоторых случаях качество решений последовательных методов оказывается выше [31–35].

Обобщенная постановка задачи оптимизации может быть представлена в следующем виде:

$$\begin{aligned}
&f(x_1, x_2, ..., x_n) \to \min\Big\} && \textit{целевая функция}\\
&\left.\begin{aligned}
&x_1^{(\min)} \le x_1 \le x_1^{(\max)}\\
&x_2^{(\min)} \le x_2 \le x_2^{(\max)}\\
&...\\
&x_n^{(\min)} \le x_n \le x_n^{(\max)}
\end{aligned}\right\} && \textit{ограничения векторного пространства}\\
&x_1 + x_2 + ... + x_n \le y\Big\} && \textit{дополнительные ограничения}\\
&x_1^*, x_2^*, ..., x_n^*, f^* - ?\Big\} && \textit{оптимальные значения}
\end{aligned} \qquad (1.1)$$

В рассмотрение вводится функция f, которая может не иметь аналитического выражения и именуется *целевой*, аргументом которой является *вектор параметров* $X = [x_1, x_2, ..., x_n]^T$, на значения которых накладываются определенные *ограничения*. Требуется отыскать такой вектор параметров $X^* = [x_1^*, x_2^*, ..., x_n^*]^T$, значения которых определяют минимальное значение целевой функции f^*.

Если по условию задачи требуется нахождение максимума, а не минимума целевой функции, постановка задачи (1.1) изменяется незначительно, т.к. несложно показать, что

$$f(x_1, x_2, ..., x_n) \to \max \Leftrightarrow -f(x_1, x_2, ..., x_n) \to \min, \qquad (1.2)$$

ввиду чего задачи поиска минимума и максимума обычно не различают, говоря о минимизации либо просто об оптимизации, понимая под этим поиск минимума.

Ограничения на значения параметров целевой функции в простейшем случае могут быть записаны в интервальном виде

$$x_i^{(\min)} \le x_i \le x_i^{(\max)}, \; i = \overline{1, n}, \qquad (1.3)$$

что является следствием некоторых физических ограничений математической модели. Например, при нормальных условиях вода в жидком состоянии находится только при температуре $0 < t < 100$ градусов Цельсия; ферромагнетики обладают магнитными свойствами только

при температуре ниже температуры Кюри; пары в расписании не могут образовывать «накладок», т.е. невозможно присутствие учебной группы или преподавателя в двух различных местах одновременно (по крайней мере, за пределами Хогвартса); работа *A* (возведение стен) не может выполняться раньше работы *B* (заливка фундамента) и т.п. Во многих задачах по соображениям здравого смысла значения параметров должны быть неотрицательными (например, в рассматриваемой ниже задаче о аренде верблюдов число верблюдов в караване не может быть меньше нуля).

Важным типом ограничений являются линейные ограничения вида

$$k_1x_1 + k_2x_2 + ... + k_nx_n < y, \tag{1.4}$$

где $k_i, i = \overline{1, n}$ – некоторые коэффициенты, y – некоторое пороговое значение. Сравнение на меньше «<» представляет собой лишь один из видов и при необходимости может быть изменено на больше «>» или на нестрогие операции сравнения «$\leq$» и «$\geq$». При этом, по аналогии с (1.2), следует помнить, что

$$k_1x_1 + k_2x_2 + ... + k_nx_n < y \Leftrightarrow -k_1x_1 - k_2x_2 - ... - k_nx_n > y. \tag{1.5}$$

Ограничения подобного вида в совокупности с линейной целевой функцией образуют важный класс задач, решаемых в рамках *линейного программирования* [36]. Например, многие стоимостные ограничения могут быть представлены в виде (1.4), где x_i – количество некоторого компонента, k_i – его стоимость, а y – значение максимальной разрешенной стоимости. С геометрической точки зрения ограничение данного вида представляют собой гиперплоскость в *n*-мерном пространстве с уравнением

$$k_1x_1 + k_2x_2 + ... + k_nx_n = y, \tag{1.6}$$

рассекающую *n*-мерное пространство на два подпространства. В двумерном случае уравнение (1.6) представляет собой уравнение прямой, рассекающей двумерную плоскость на две части (рис. 1.9).

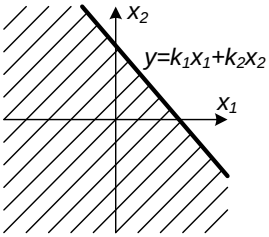


Рис. 1.9. Геометрический смысл линейных ограничений вида (1.4) на плоскости. Область решений, удовлетворяющих ограничению, обозначена штриховкой

Линейных ограничений вида (1.4) может быть несколько (например, m):

$$\begin{aligned} &k_{11}x_1 + k_{12}x_2 + ... + k_{1n}x_n < y_1, \\ &k_{21}x_1 + k_{22}x_2 + ... + k_{2n}x_n < y_2, \\ &... \\ &k_{m1}x_1 + k_{m2}x_2 + ... + k_{mn}x_n < y_m. \end{aligned} \qquad (1.7)$$

В таком случае говорят о поиске решения в некоторой области, ограниченной m гиперплоскостями и называемой n-мерным многогранником (возможно незамкнутым) или *симплексом* (рис. 1.10), откуда происходит название класса специализированных методов, именуемых *симплекс-методами*.

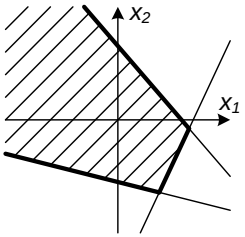


Рис. 1.10. Пример многогранника (симплекса), образованного тремя линейными ограничениями, на плоскости

В общем случае ограничения могут быть нелинейными (рис. 1.11):

$$g_i(x_1, x_2, ..., x_n) < y_i, \ i = \overline{1, m}. \qquad (1.8)$$

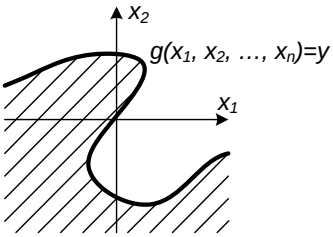


Рис. 1.11. Пример нелинейного ограничения на плоскости $(n=2)$

Простейший пример подобных ограничений – квадратичные ограничения вида

$$\sum_{i=1}^{n}\alpha_i x_i^2+\sum_{i=1}^{n}\sum_{j=i+1}^{n}\beta_{ij}x_i x_j+\sum_{i=1}^{n}\gamma_i x_i<y, \qquad (1.9)$$

где $\alpha_i, \beta_{ij}, \gamma_i$ – некоторые коэффициенты. В невырожденном $(\exists\alpha_i\neq 0)\vee(\exists\beta_{ij}\neq 0)$ двумерном случае $(n=2)$ им соответствуют кривые второго порядка (параболы, гиперболы, окружности, эллипсы или их вырожденные случаи), ограничивающие область корректных решений. Простейшие и наиболее важные частные случаи ограничений приведены ниже:

- $x_1^2+x_2^2\le y$ – круг с центром в начале координат и радиусом $\sqrt{y}$;
- $\alpha_1 x_1^2+\alpha_2 x_2^2\le y$ – эллипс с центром в начале координат и полуосями α_1 и α_2;
- $x_1^2+x_2^2+x_3^2\le y$ – шар с центром в начале координат и радиусом $\sqrt{y}$;
- $\sum_{i=1}^{n}x_i^2\le y,\ n>3$ – n-мерный шар с центром в начале координат и радиусом $\sqrt{y}$.

Для решения задач с нелинейными целевыми функциями и/или ограничениями разработан специальный математический аппарат, называемый *нелинейным математическим программированием* и его частные разделы, например, *квадратичное математическое программирование*. Как правило, подобные задачи намного сложнее задач линейного программирования и могут быть решены аналитически только в некоторых частных случаях.

В ряде оптимизационных задач могут присутствовать различные дополнительные ограничения и особенности. Например, целевых функций может быть несколько:

$$\begin{gathered} f_1(x_1, x_2, ..., x_n) \to \min, \\ f_2(x_1, x_2, ..., x_n) \to \min, \\ ... \\ f_l(x_1, x_2, ..., x_n) \to \min. \end{gathered} \tag{1.10}$$

Подобные задачи называются *задачами многокритериальной оптимизации*. У них может не быть однозначного решения, т.к. зачастую более сильная минимизация одной из целевых функций (частного показателя качества) приводит к ухудшению (увеличению) значений остальных и необходимо соблюдение некого баланса между ними (в соответствии с принципами Парето-оптимальности [37] или отталкиваясь от некой экспертной оценки). Иногда в подобных задачах вводится значение интегрального показателя качества F [38, 39], объединяющего в своем составе частные показатели (например, в виде взвешенной суммы):

$$F\left(f_1(X),\ f_2(X), ...,\ f_l(X)\right) \to \min. \tag{1.11}$$

В некоторых задачах на значения параметров накладываются ограничения целочисленности или бинарности, некоторые примеры подобных задач будут рассмотрены ниже. Данные задачи образуют отдельные важные классы и решаются с использованием специализированных разделов математики (например, методами *целочисленного математического программирования*). Иногда данные задачи можно решать, оперируя вещественными значениями параметров с их последующим «округлением» до ближайших целых значений, однако это удается не всегда, что схематично показано на рис. 1.12.

Важным частным случаем целочисленных задач являются задачи, в которых допустимые значения параметров x_i ограничены множествами $\{0, 1\}$. Подобные задачи называются бинарными или булевыми, а соответствующий раздел математики – *бинарное* или *булево математическое программирование*. Примером подобной задачи является задача о том, можно ли составить заданную сумму S из имеющегося множества монет различного достоинства $\{c_1, c_2, ..., c_n\}$. Ее можно свести к решению диофантова уравнения $c_1x_1 + c_2x_2 + ... + c_nx_n = S$ с дополнительным ограничением $\forall x_i \in \{0, 1\}$. Данная задача является примером задачи на существование и ее решение в общем случае нетривиально [40], несмотря на кажущуюся простоту.

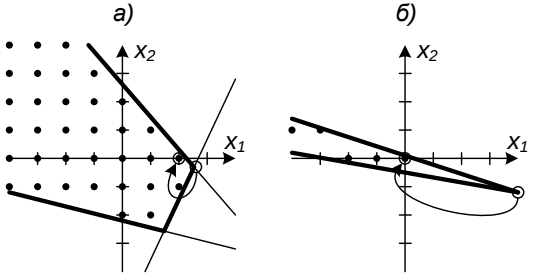


Рис. 1.12. Примеры вариантов решений. Корректные целочисленные решения отмечены точками, оптимальные решения без ограничения на целочисленность и попытки их «округления» до ближайшего целого решения отмечены кругами (а – тривиальный случай, б – нетривиальный случай)

В некоторых случаях ограничения могут быть противоречивы, например, $(x_1 + x_2 < 1) \wedge (2x_1 + 2x_2 > 3)$, в таком случае множество решений задачи пусто и говорить об оптимальном решении нет смысла. Далеко не всегда можно заранее сказать о том, есть ли вообще решения у поставленной задачи (например, для пар ортогональных диагональных латинских квадратов (ОДЛК) порядка 10 решения известны (см. рис. 1.3), хотя факт их существования не очевиден, для троек ОДЛК порядка 10 существование решений не доказано, однако и не опровергнуто).

Рассмотрим несколько примеров оптимизационных задач различного типа.

Задача 1.6. Найти минимум функции $f(x) = x^2 + 2x + 1{,}5$ на отрезке $-2 \le x \le 1$.

Данная задача в полной мере соответствует постановке (1.1) и является задачей минимизации гладкой непрерывной дифференцируемой аналитически заданной функции одного переменного с простейшим интервальным ограничением. Как известно из школьного курса математики, она решается путем нахождения производной целевой функции, затем путем решения соответствующего уравнения производится нахождение экстремумов, проверка на нахождение их в пределах заданного отрезка и затем – выбор интересующего значения (в данном случае – минимума) путем сравнения значений функции на границах отрезка и в найденных экстремумах:

$$\frac{df}{dx} = 2x + 2,\ \ 2x + 2 = 0,\ \ x = -1,\ \ x \in [-2; 1],$$

$$f(-2) = 1{,}5,\ \ f(-1) = 0{,}5,\ \ f(1) = 4{,}5,$$

$$x^* = -1,\ \ f^* = 0{,}5.$$

В правильности решения несложно убедиться графически (рис. 1.13).

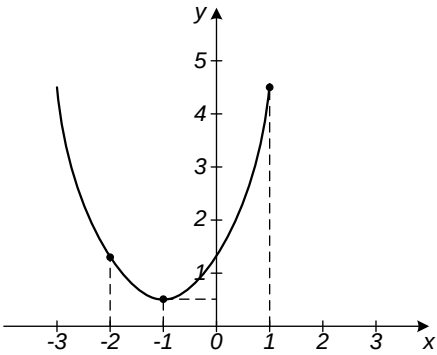


Рис. 1.13. Графическая интерпретация решения задачи

Задача 1.7. Найти минимум функции $f(x, y) = x^2 + y^2 - 2x + 4y + 5$ при ограничениях значений аргументов $-3 \le x \le 3, -3 \le y \le 3$.

Данная задача также в полной мере соответствует постановке (1.1) и отличается от предыдущей наличием нескольких (в данном случае двух) аргументов. Как известно, стратегия ее решения незначительно отличается от предыдущей и заключается в нахождении частых производных, составлении и решении соответствующей системы уравнений и проверки соответствующих граничных условий:

$$\frac{\partial f}{\partial x} = 2x - 2,\ \ 2x - 2 = 0,\ \ x = 1,\ \ x \in [-3; 3],$$

$$\frac{\partial f}{\partial y} = 2y + 4,\ \ 2y + 4 = 0,\ \ y = -2,\ \ y \in [-3; 3],$$

$$f(1, -2) = 0,$$

$$x^* = 1,\ y^* = -2,\ f^* = 0.$$

Замечание 1.1. В данном упрощенном решении нам повезло и экстремум (минимум) «попал» в прямоугольник, образованный ограничениями (рис. 1.14), и оказался решением. Если, например, нам по условию задачи потребовалось бы найти максимум, а не минимум, той же целевой функции, то необходимо было бы сперва найти уравнения границ области допустимых решений (в данном случае их четыре: $x=-3$, $x=3$, $y=-3$, $y=3$) и решить 4 независимых друг от друга задачи оптимизации (в данном примере однопараметрической) на них с последующим выбором решения с максимальным значением целевой функции *f*. Существуют и другие приемы решения подобных задач, например, с использованием *штрафных функций* [41, 42].

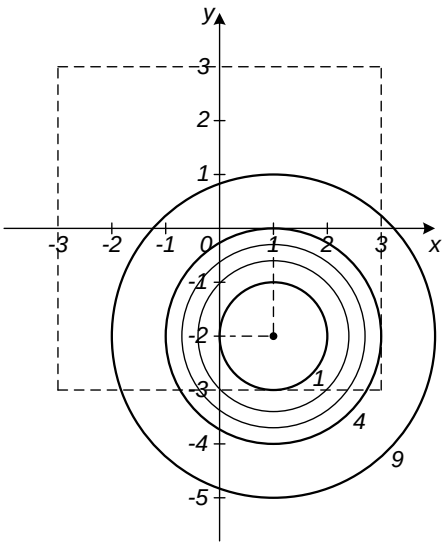


Рис. 1.14. Графическая интерпретация решения задачи (окружностями показана часть линий уровня, соответствующих двумерной кривой – параболоиду вращения)

Задача 1.8. Найти минимум функции $f(x)=\dfrac{\sin 5x}{\sqrt{x}}+\cos 3x$ на отрезке $0\le x\le 10$.

На первый взгляд данная задача не отличается от рассмотренной выше задачи 1.6 и может быть решена тем же способом через нахожде-

ние производной с последующим решением соответствующего уравнения. Однако после нахождения производной целевой функции

$$\frac{df}{dt}=\left(\frac{\sin 5x}{\sqrt{x}}+\cos 3x\right)'=\left(\frac{\sin 5x}{\sqrt{x}}\right)'+(\cos 3x)'=$$

$$=\frac{5\cos 5x\cdot\sqrt{x}-\sin 5x\cdot\dfrac{1}{2\sqrt{x}}}{x}-3\sin 3x=\frac{5\cos 5x}{\sqrt{x}}-\frac{\sin 5x}{2x\sqrt{x}}-3\sin 3x$$

и получения нелинейного уравнения

$$\frac{5\cos 5x}{\sqrt{x}}-\frac{\sin 5x}{2x\sqrt{x}}-3\sin 3x=0$$

несложно убедиться в том, что его аналитическое решение затруднительно.

График целевой функции приведен на рис. 1.15.

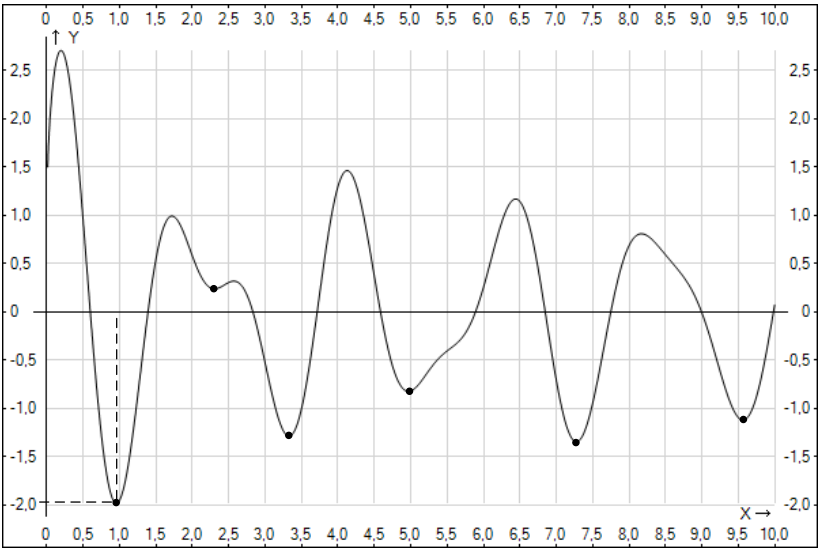


Рис. 1.15. График целевой функции, точками отмечено множество минимумов

Из графика видно, что функция имеет на заданном отрезке несколько минимумов, причем значения функции в них отличаются:

$$x_1 \approx 1{,}0 \quad f_1 \approx -2{,}0,$$
$$x_2 \approx 2{,}3 \quad f_2 \approx 0{,}2,$$
$$x_3 \approx 3{,}3 \quad f_3 \approx -1{,}3,$$
$$x_4 \approx 5{,}0 \quad f_4 \approx -0{,}8,$$
$$x_5 \approx 7{,}3 \quad f_5 \approx -1{,}4,$$
$$x_6 \approx 9{,}6 \quad f_6 \approx -1{,}1.$$

По условию задачи нас интересует «самый маленький» минимум, в данном случае $x^* \approx 1{,}0 \quad f^* \approx -2{,}0$, называемый *глобальным*. Остальные минимумы называются *локальными*, т.к. по определению экстремума функции в каждом случае можно подобрать некоторую ε-окрестность $\left(x_i - \varepsilon \le x \le x_i + \varepsilon\right)$, в пределах которой найденный минимум будет глобальным, однако на заданном по условию отрезке локализации данное свойство в общем случае не выполняется.

Для нахождения решения поставленной задачи численно существует ряд методов, одним из которых является широко известный метод Ньютона (метод касательных). Он заключается в выборе некоторого начального значения x_0 с его последующим итеративным уточнением с использованием формулы

$$x_n = x_{n-1} - \frac{f\left(x_{n-1}\right)}{f'\left(x_{n-1}\right)}, \tag{1.12}$$

что схематично показано на рис. 1.16.

При нахождении минимума функции многих аргументов применяются схожие методы, например, метод наискорейшего спуска, в котором из начальной точки $X_0 = \left[x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}\right]^T$ происходит движение в сторону антиградиента с использованием формулы

$$X_i = X_{i-1} - \mu \nabla f\left(X_{i-1}\right), \tag{1.13}$$

где

$$\nabla f\left(X_i\right) = \left\{\frac{\partial f\left(X_i\right)}{\partial x_1}, \frac{\partial f\left(X_i\right)}{\partial x_2}, ..., \frac{\partial f\left(X_i\right)}{\partial x_n}\right\} \tag{1.14}$$

– градиент (вектор наискорейшего возрастания функции), μ – настроечный параметр, определяющий величину шага при движении в векторном пространстве. Известны также и другие методы, такие как метод покоординатного спуска, метод сопряженных градиентов, метод деформированного многогранника (метод Нелдера-Мида [43]), метод Хука-Дживса [44], метод Ньютона-Рафсона и др.

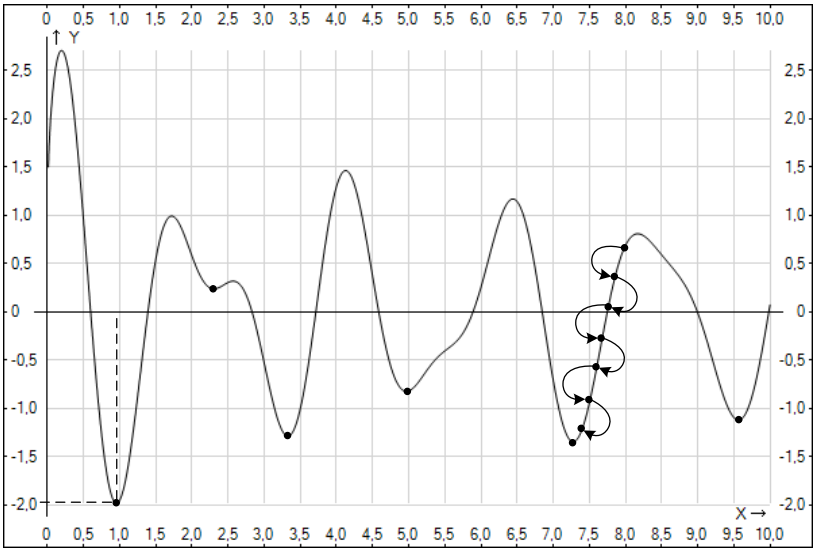


Рис. 1.16. Иллюстрация, поясняющая работу метода Ньютона

Большинству из них свойственен ряд недостатков. Основным из них является возможность захода в ближайший локальный экстремум, что схематично показано на рис. 1.16. Для борьбы с этим негативным явлением применяются различные приемы: попытка совершения шага в случайном направлении с целью выхода за пределы бассейна притяжения текущего локального экстремума, выбор случайного расположения начальной точки и пр. Они используются, как правило, при решении непрерывных задач, но имеющие в достаточной степени схожие дискретные аналоги, например, метод имитации отжига (разд. 2.9) или алгоритм пчелиной колонии (разд. 2.11), которые применимы для решения как непрерывных, так и дискретных комбинаторных задач.

Отталкиваясь от того, какой минимум находит тот или иной метод, различают методы *локальной* и *глобальной оптимизации*. Методам первого направления свойственна возможность попадания в локальные экстремумы, при этом нахождение глобального экстремума возможно, но не гарантировано; методы второго направления гарантируют нахождение глобального экстремума. В общем случае нахождение глобального экстремума произвольной функции является достаточно сложной задачей, и на практике зачастую используются дополнительные приемы, основанные на знании некоторых особенностей поведения целевой функции.

Выбор шага μ при движении в векторном пространстве параметров целевой функции также является достаточно непростой задачей. Так выбор слишком большого значения μ способен привести к «прыжку» за пределы локального минимума, в сторону которого указывает антиградиент, а следствием выбора слишком маленького μ является большое число шагов (итераций) на пути к локальному минимуму и, соответственно, большие затраты вычислительного времени. Для различных задач значения оптимального шага могут существенно различаться. В некоторых из них различные параметры характеризуются различными диапазонами значений и скоростью убывания/возрастания функции (значения соответствующих компонент в векторе градиента (1.14) могут отличаться на несколько порядков). В некоторых случаях применяются адаптивные схемы, при использовании которых шаг изменяется по мере работы соответствующего алгоритма. Например, при обучении нейросетей типа Персептрон в алгоритме обратного распространения ошибки, основанном на вычислении градиентов функций активации нейронов, имеется параметр скорости обучения, значение которого постепенно уменьшается [45, 46].

Еще одним существенным недостатком данной группы методов является необходимость вычисления значения целевой функции на каждом шаге. Для рассматриваемой в качестве примера задачи 1.8 это не существенно, однако существуют задачи, в которых оценка качества решения является трудоемкой операцией, определяющей общее время работы, ввиду чего число подобных вычислений необходимо по возможности минимизировать. Например, в задаче составления расписаний [9, 10, 39] оценка качества решения может составлять величину до нескольких минут на одно решение, а число пробных решений может исчисляться тысячами и более. В задаче построения модели галактики Млечный путь, вращающихся вокруг нее звездных потоков и плотности темной материи [47, 48] оценка ошибки (невязки) для заданного набора параметров выполняется в ходе численного моделирования и занимает до нескольких часов на CPU и до нескольких минут на GPU. С целью получения решения за разумное время в обозримом будущем (в пределах нескольких лет) в данной задаче применяется распределенная организация моделирования с использованием грид-системы на базе платформы BOINC [49], включающей в своем составе десятки тысяч компьютеров.

Однако наиболее существенным недостатком методов данной группы с позиции дискретной комбинаторной оптимизации является необходимость вычисления значения производной целевой функции в указанной точке. В простейших случаях (как в рассматриваемом примере

1.8) эта задача решается аналитически и не представляется трудоемкой. В более сложных случаях, когда получение аналитических выражений для градиента (1.14) затруднительно или невозможно, допускается его приближенная численная оценка, например, в виде

$$\frac{\partial f}{\partial x_i} \approx \frac{f\left(x_1, x_2, ..., x_i + h, ..., x_n\right) - f\left(x_1, x_2, ..., x_i - h, ..., x_n\right)}{h}, \quad (1.15)$$

где h – некоторый настроечный параметр, определяющий размер окрестности, либо с использованием более сложных формул, имеющих больший порядок точности. Однако даже с использованием формулы (1.15) при наличии функции с n аргументами необходимо вычисление ее значений в $2n$ точках, что, как уже было рассмотрено выше, не всегда удобно и возможно. При решении задач дискретной комбинаторной оптимизации, представляющих собой предмет данной книги, в абсолютном большинстве случаев понятие производной не определено для дискретных объектов (графов, перестановок, расписаний и пр.) и не допускает какой-либо разумной аппроксимации, что автоматически делает невозможным применение данной группы методов для решения указанных задач.

Еще одним классом задач, в которых применение градиентных методов существенно ограничено, являются задачи стохастической оптимизации, в которых целевая функция имеет некоторую стохастическую компоненту (рис. 1.17).

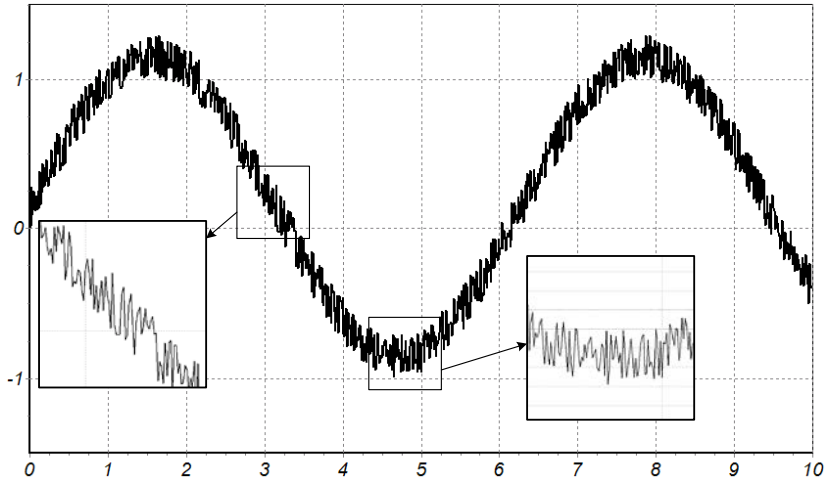


Рис. 1.17. Пример графика стохастической функции $y = \sin x + r$, где r – стохастическая составляющая

Источники наличия данной составляющей могут быть различными, например, физические особенности датчика радиотелескопа, гравитационного интерферометра или КМОП-сенсора [50], невозможность полного описания модели поведения системы частиц – броуновского движения, плазмы, системы электронов в атоме и пр., однако принципиальным свойством данных функций является невозможность полного устранения стохастической (шумовой) составляющей. Непосредственное использование градиентных методов в указанных задачах невозможно по ряду причин: нестационарной природы шума во времени, наличия очень большого числа локальных экстремумов и пр.

В некоторых случаях допускается снижение влияния шумовой составляющей с использованием приемов фильтрации (например, медианной, усреднения скользящим окном, согласованной, с использованием вейвлетов и пр. [51–53]), однако в общем случае проблема имеет место. Данный класс задач имеет важное практическое значение в теории автоматического управления: стохастические задачи идентификации, упреждения, оптимального управления [54–56], такие как стыковка космических аппаратов, системы мягкой посадки, системы управления расходом топлива и пр.; при решении стохастических дифференциальных уравнений в области моделирования броуновского движения, физики плазмы, теплопереноса; при построении моделей биржевых индексов и т.п. К подобным задачам также сводятся задачи метаоптимизации эвристических методов (см. разд. 2.7).

Задача 1.9 (о сбалансированном питании). Для организации экспедиции на Марс требуется разработать меню для питания космонавтов. Меню может включать n продуктов питания $x_1, x_2, ..., x_n$, каждый из которых обладает определенной калорийностью (англ. energy) e_i, а также содержанием белков (англ. proteins) p_i, жиров (англ. fats) f_i и углеводов (англ. carbohydrates) h_i. Масса одного килограмма продукта x_i составляет m_i кг, цена – c_i руб. Суммарная масса продуктов не должна превышать m. Необходимо подобрать оптимальное меню минимальной стоимости при условии, что космонавт должен получать не менее: e калорий, p белков, f жиров и h углеводов в день.

Приведенная постановка задачи упрощена по сравнению с реальным практическим случаем, в котором потребуется учесть ряд дополнительных требований: предпочтения космонавтов, необходимость наличия разнообразия продуктов, их срок годности и др. Целевая функция в данном случае представляется в виде

$$f(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n \to \min$$

и направлена на минимизации стоимости меню (однокритериальная оптимизация). При этом требуется учесть ограничения на необходимую минимальную калорийность и количество белков, жиров и углеводов:

$$e_1x_1 + e_2x_2 + ... + e_nx_n \geq e,$$
$$p_1x_1 + p_2x_2 + ... + p_nx_n \geq p,$$
$$f_1x_1 + f_2x_2 + ... + f_nx_n \geq f,$$
$$h_1x_1 + h_2x_2 + ... + h_nx_n \geq h.$$

Также необходим учет ограничения на максимальную разрешенную массу продуктов:

$$m_1x_1 + m_2x_2 + ... + m_nx_n \leq m,$$

и ограничения физического смысла: количество каждого из продуктов должно быть неотрицательной величиной:

$$x_1 \geq 0,\ x_2 \geq 0,\ \ldots,\ x_n \geq 0.$$

В указанной математической постановке поставленная задача в полной мере соответствует (1.1) с линейной целевой функцией и линейными ограничениями (1.7).

Задача 1.10 (об аренде верблюдов). Для перевозки инжира из Багдада в Мекку Мухаммед использует одно- и двугорбых верблюдов. Одногорбый верблюд может перевезти за один переход f_1 фунтов инжира (англ. figs), а двугорбый – f_2. За один переход одногорбый потребляет h_1 кип сена (англ. hay) и w_1 ведер воды (англ. water), двугорбый – h_2 кип сена и w_2 ведер воды. При этом запас сена не может превышать h кип, а запас питья – w ведер воды. Все верблюды арендуются в Багдаде по цене c_1 динаров за одногорбого и по цене c_2 за двугорбого верблюда. Определить, сколько одногорбых и двугорбых верблюдов следует арендовать, если Мухаммед должен перевезти более f фунтов инжира, при условии, что арендная плата за верблюдов должна быть минимальной.

Как и в предыдущем примере, целевая функция ориентирована на минимизацию стоимости аренды и может быть представлена как

$$g(x_1, x_2) = c_1x_1 + c_2x_2 \to \min.$$

По условию задачи имеют место ограничение на минимальное количество перевозимого инжира

$$f_1x_1 + f_2x_2 \geq f$$

и ограничения сверху на количество сена и воды:

$$h_1 x_1 + h_2 x_2 \le h,$$
$$w_1 x_1 + w_2 x_2 \le w.$$

Кроме того, число верблюдов в караване не может быть отрицательным:

$$x_1 \ge 0\,,\ x_2 \ge 0.$$

Несмотря на отличающуюся предметную область, данная задача в математической постановке, также как и предыдущая, имеет линейную целевую функцию и линейные ограничения.

Задача 1.11 (о рюкзаке). Имеется набор предметов $x_1, x_2, ..., x_n$, каждый из которых характеризуется весом $w_1, w_2, ..., w_n$ и стоимостью $c_1, c_2, ..., c_n$. Необходимо укомплектовать рюкзак ограниченной емкости w так, чтобы стоимость вошедших в него предметов была максимальной.

Целевая функция в данном случае отражает стремление к максимизации стоимости предметов

$$f(x_1, x_2, ..., x_n) = c_1 x_1 + c_2 x_2 + ... + c_n x_n \to \max,$$

в отличие от минимизации стоимости издержек в предыдущих двух задачах (что, как уже было отмечено выше, не является сколь-нибудь значимым отличием ввиду (1.2)). Единственным линейным ограничением данной задачи является ограничение на суммарный вес предметов:

$$w_1 x_1 + w_2 x_2 + ... + w_n x_n \le w.$$

Таким образом, во всех трех задачах имеет место линейная целевая функция и одно или несколько линейных ограничений, а следовательно они относятся к одному классу и могут решаться с использованием одного и того же инструментария линейного программирования несмотря на различие в предметных областях? Ответ на поставленный вопрос является отрицательным, а рассмотренные задачи имеют существенное отличие.

В задаче о сбалансированном питании решения могут быть дробными (например, можно взять 1,5 кг моркови и 0,7 кг картофеля), т.е. $x_1, x_2, ..., x_n \in \mathbb{R}$, сама задача относится к линейному программированию и решается с использованием симплекс-метода. В задаче о верблюдах решение может быть только целочисленным, т.е. $x_1, x_2, ..., x_n \in \mathbb{N}$, задача относится к целочисленному линейному программированию и решается с использованием более сложных метода Гомори или М-метода [36]. А задача о рюкзаке допускает лишь наличие или отсутствие

предмета в наборе, т.е. $x_1, x_2, ..., x_n \in \{0,1\}$, что относит ее к существенно более сложным задачам булева программирования. Она относится к классу *NP*-полных (см. разд. 1.3), для отыскания ее оптимального решения не существует быстрого алгоритма, поэтому при ее решении на практике организуют перебор всех вариантов с выбором наилучшего (их число составляет 2^n), либо применяют эвристические подходы, дающие суб- или квазиоптимальные решения в ходе ограниченного перебора [57–59]. Подобные задачи и способы их решения составляют предмет изучения данной книги.

1.2. Основные понятия и определения теории графов

Многие из задач дискретной комбинаторной оптимизации имеют непосредственное отношение к теории графов либо допускают полиномиальное сведение к задачам данного направления. Поэтому перед их подробным рассмотрением во избежание терминологической путаницы необходимо ввести основные понятия, определения и обозначения. Более подробно с теорией графов можно ознакомиться путем изучения специализированной литературы (например, [60]).

Графом $G = \langle A, V \rangle$ в дискретной математике и теории графов называется совокупность двух множеств – множества вершин $A = \{a_1, a_2, ..., a_N\}$, $|A| = N$ и множества соединяющих их *дуг* или *ребер* $V = \{v_1, v_2, ..., v_M\} \subseteq A \times A$, $|V| = M$, $v_i = \left(a_{нач}^{(i)}, a_{кон}^{(i)}\right)$, $i = \overline{1, M}$, $a_{нач}^{(i)} \in A$, $a_{кон}^{(i)} \in A$ (операция $|X|$ обозначает мощность множества – число элементов множества *X*; $A \times B$ – декартово произведение множеств – множество, образованное всеми возможными парами вида (a_i, b_j), $a_i \in A$, $b_j \in B$). Обычно рассматриваются графы без петель:

$$\forall v_i = \left(a_{нач}^{(i)}, a_{кон}^{(i)}\right), i = \overline{1, M}: \; a_{нач}^{(i)} \neq a_{кон}^{(i)}$$

и парных дуг или ребер:

$$\overline{\exists} i, \; j = \overline{1, M}, i \neq j: \left(a_{нач}^{(i)} = a_{нач}^{(j)}\right) \wedge \left(a_{кон}^{(i)} = a_{кон}^{(j)}\right).$$

Пары $v_i = \left(a_{нач}^{(i)}, a_{кон}^{(i)}\right)$ могут рассматриваться как упорядоченные, тогда v_i называется дугой, или как неупорядоченные $v_i = \left(a_{нач}^{(i)}, a_{кон}^{(i)}\right) = \left(a_{кон}^{(i)}, a_{нач}^{(i)}\right) = \left(a_1^{(i)}, a_2^{(i)}\right)$, тогда v_i называется ребром.

```
        Result := True;
        BestLen := CurrLen;
        BestPath := CurrPath;
      end;
end;


begin
  if not GetWeightedRandomPath(AdjacencyMatrix, 1, 10, 1.0, 1)
  then
    Writeln('Path not found')
  else
    PrintPath(AdjacencyMatrix, BestPath);

  Readln;
end.
```

Приведенная программная реализация для примера графа на рис. 2.21 обеспечивает нахождение оптимального пути $P^* = [a_1, a_2, a_3, a_5, a_{10}]$, длина которого $L(P^*) = 18$, начиная с первой итерации. В общем случае для графов с б**о**льшим числом вершин нахождение оптимальных решений обеспечивается за меньшее число итераций по сравнению с методом случайного перебора, что подтверждает эффективность использоваться эвристики (2.7) на практике.

При построении очередного решения опыт использования различных переходов по ветвям дерева комбинаторного перебора, произведенных на прошлых итерациях, не учитывается. Его можно учесть при использовании метода муравьиной колонии, подробно рассмотренного в следующем разделе, что позволяет дополнительно снизить число итераций для получения решений сопоставимого качества.

2.8. Метод муравьиной колонии

Как уже было отмечено в предыдущих разделах, использование комбинации случайного перебора и взвешивающих эвристик обеспечивает на практике ощутимую выгоду, заключающуюся в возможности быстрого получения решений неплохого качества, что не обеспечивается методами на базе случайной или жадной стратегий по отдельности. Однако при итеративном повторении поиска решений опыт прошлых итераций не учитывается, плохие или хорошие решения фактически отбрасываются, запоминается лишь лучшее из них (рекорд). Исправить данную ситуацию можно с использованием *метода муравьиной колонии* (англ. Ant Colony optimization, сокр. AC), в котором старые решения запоминаются путем специальной пометки ветвей дерева комбинатор-

ного перебора и соответствующих им элементов решения. Данная пометка выражается в использовании специальных дополнительных слагаемых или множителей в составе эвристики, на основе которой принимается решение о выборе направления движения.

Метод муравьиной колонии [109] был предложен итальянским ученым Марко Дориго (M. Dorigo) в 1992 г. как результат наблюдения за поведением муравьев в природе, где их целью является нахождение кратчайшего пути между муравейником и источником пищи. При выборе направления движения в пространстве муравей выбирает конкретное решение, руководствуясь двумя факторами: длиной пути l_i в выбранном *i*-м направлении и количеством *феромона* τ_i, оставленном на *i*-м пути другими муравьями, прошедшими по нему ранее.

Замечание 2.10. Феромоны – это химические вещества, выделяемые некоторыми видами животных и растений, и обеспечивающие химическую коммуникацию между особями одного вида. Муравьи используют их для пометки путей, которые оказались перспективными. Другие виды животных могут использовать феромоны для привлечения брачных партнеров или в качестве сигналов об опасности. Их активное изучение началось во второй половине XX века.

В природе муравей имеет ограниченный запас феромона и при достижении цели (обычно источника пищи) помечает понравившийся ему путь дополнительной порцией феромона, причем на более коротком пути остается большее количество феромона, что при итеративном повторении (движении группы муравьев по пути в различное время) обеспечивает отметку феромоном более перспективных путей, оставляя при этом небольшую возможность для отклонения от них в отличие от жадного метода. С течением времени по наиболее перспективным путям происходит перемещение большего числа муравьев, что дополнительно увеличивает общее количество феромона на них. С течением времени феромон испаряется, что приводит к уменьшению его количества на путях, оказавшихся неперспективными.

В результате колония из достаточно примитивных по отдельности агентов (муравьев) в совокупности получает способность решать непростые практические задачи, связанные с ориентированием на местности. Существование муравьиных колоний в природе на протяжении миллионов лет, достигающих с использованием группового интеллекта явных успехов в адаптации к условиям обитания, является неоспоримым эмпирическим доказательством правильности используемого муравьями подхода, что стимулирует попытки адаптации рассмотренного подхода

к решению различных практически важных задач. При решении задач дискретной комбинаторной оптимизации данный принцип может быть использован путем отметки феромоном перспективных решений (путей в дереве комбинаторного перебора) и испарения феромона с неперспективных. Данная идея и легла в основу диссертации М. Дориго, в которой он проанализировал возможность применения электронных аналогов биологических муравьев в задаче поиска путей в графах.

Направление движения электронного муравья в каждом конкретном случае выбирается пропорционально вероятности

$$p_i = \frac{\eta_i^{\alpha}\tau_i^{\beta}}{\sum_{j=1}^{K}\eta_j^{\alpha}\tau_j^{\beta}},\ i=\overline{1,K}, \qquad (2.8)$$

где $\eta_i = \frac{1}{l_i}$ в рассматриваемой задаче поиска путей; α, β – настроечные параметры, определяющие реакцию муравья на длину участка пути (жадность) и количество феромона на нем. В предельных случаях при $\alpha = 1$ и $\beta = 0$ метод превращается в жадный, т.к. электронный муравей руководствуется только длиной пути, а при $\alpha = 0$ и $\beta = 1$ – только количеством феромона. На практике обычно используется компромиссный вариант, когда $(\alpha > 0) \wedge (\beta > 0)$.

Замечание 2.11. В некоторых источниках [123] критерии вроде (2.2), (2.7) и (2.8) называют *метаэвристиками*, а методы на их базе – *метаэвристическими*.

При достижении цели, критерием чего в природе является нахождение маршрута к источнику пищи, а в рассматриваемой цифровой модели муравьиной колонии – нахождение решения, электронный муравей помечает пройденный путь, добавляя к имеющемуся количеству феромона на каждом из его участков приращение

$$\Delta\tau = \frac{Q}{L}, \qquad (2.9)$$

где Q – количество феромона в распоряжении муравья (настроечный параметр),

$$L = \sum_i l_i$$

– длина найденного пути (оценка качества решения). При этом учитывается испарение феромона

$$\tau_i^{(t)} = \gamma\tau_i^{(t-1)} + \Delta\tau, \qquad (2.10)$$

где t – дискретное время (номер итерации); $\gamma \in [0;1]$ – настроечный параметр, влияющий на скорость испарения: при $\gamma = 0$ происходит полное испарение феромона прошлых итераций, при $\gamma = 1$ – сохранение значения феромона с «момента сотворения мира» (иногда вместо γ в литературе используется параметр $p = 1 - \gamma$, обладающий обратным смыслом и не меняющий сути подхода).

При реализации алгоритма на практике обычно рассматривается колония из Z муравьев, поведение каждого из которых в простейшем случае определяется рассмотренными выше правилами. В некоторых случаях при построении решений некоторые муравьи в составе колонии могут действовать по отличающимся правилам. Так, например, иногда вводится понятие элитарных муравьев, обладающих повышенной жадностью.

Для построения решений достаточно колонии из $Z = 1$ муравья. Если же $Z > 1$, то каждый из муравьев колонии производит нахождение своего решения, некоторые из которых могут совпадать, и его пометку феромоном, а затем выполняется испарение феромона со всех путей (2.10). По сравнению с единственным муравьем колонией производится локальная разведка признакового пространства в окрестности текущего неплохого решения, отмеченного максимальным количеством феромона.

Метод муравьиной колонии относится к эвристическим, т.к. не гарантирует получение оптимальных решений, стохастическим, т.к. базируется на вероятностных принципах выбора направления движения, и итерационным, т.к. подразумевает перебор некоторого множества из $C_{\max}$ решений.

Рассмотрим применение указанного алгоритма для решения задачи 1.12 поиска кратчайшего пути в графе. При использовании алгоритма муравьиной колонии выбор направления движения из текущей вершины определяется пропорционально вероятности (2.8). С учетом того, что нормирующий знаменатель $\sum_{j=1}^{K} \eta_j^{\alpha} \tau_j^{\beta}$ в формуле (2.8) является константой [110], его можно исключить из рассмотрения, осуществляя выбор направления движения с использованием эвристики

$$F_i = \eta_i^{\alpha} \tau_i^{\beta} r_k \to \max \qquad (2.11)$$

либо по принципу рулетки (см. разд. 2.5). Максимальное значение эвристики (2.11) будет соответствовать предпочтительному направлению движения

$$j = \arg\max_{i=\overline{1,K}} \eta_i^{\alpha} \tau_i^{\beta} r_k .$$

В ходе работы алгоритма каждый муравей проделывает путь по одной из ветвей дерева комбинаторного перебора от корня к одному из листьев, причем общее количество проходов совпадает с числом итераций $C_{\max}$ для рассмотренных выше методов, помечая его феромоном в соответствии с (2.10) случае попадания в конечную вершину. В случае попадания муравья в тупик в простейшем случае пометка пути феромоном не производится. При реализации более сложных вариантах поведения муравьев [110], применяемых при решении задач на существование, возможно использование стратегии комбинаторных возвратов с ограничением на их число [103] (англ. Ant Colony with Returns, сокр. ACR) или разрешение движения муравьев по отсутствующим дугам графа, которым приписывается заведомо большое значение длины V. При этом путь, включающий в своем составе запрещенные переходы, будет обладать гораздо большей длиной L по сравнению с допустимыми путями, что обеспечит для него малую прибавку феромона $\Delta\tau$ и, как следствие, малую вероятность использования в дальнейшем.

Перед практической реализацией метода муравьиной колонии необходимо определиться со значениями настроечных параметров $\alpha, \beta, \gamma, Q, \tau^{(0)}, V, Z$, где $\tau^{(0)}$ – начальное значение феромона на дугах графа, т.е. выполнить метаоптимизацию. В отличие от рассмотренного в предыдущем разделе метода взвешенного случайного перебора в данном случае число настроечных параметров 6, что требует б**о**льших вычислительных затрат при оптимизации стохастической функции, соответствующей усредненному качеству решений.

Рассмотрим применение метода муравьиной колонии на примере поиска пути между вершинами a_1 и a_{10} в графе, изображенном на рис. 2.45.

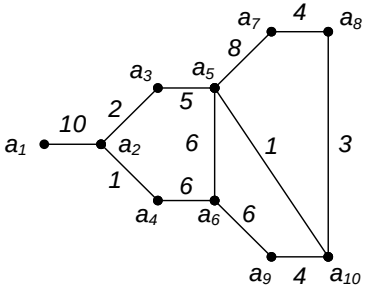


Рис. 2.45. Пример графа

Для данного примера настроечные параметры выбраны равными $\alpha = 0{,}1$, $\beta = 0{,}9$, $\gamma = 0{,}9$, $Q = 1{,}0$, $\tau^{(0)} = 10{,}0$, $Z = 2$. Движение по отсутствующим дугам запрещено, поэтому значение параметра V для данного примера не важно. Муравей начинает движение из вершины a_1 и на первом участке пути имеет возможность попасть лишь в вершину a_2 независимо от длины ребра и количества феромона на нем. Далее, находясь в вершине a_2, возможны два направления движения: в вершину a_3 или a_4. Для выбора одного из них производится расчет значения эвристики (2.11):

$$a_2 \to a_3:\ l_{23} = 2,\ \eta_{23} = \frac{1}{l_{23}} = 0{,}5,\ \tau_{23}^{(0)} = 10{,}0,\ r_k = 0{,}030 \ \Rightarrow$$

$$\Rightarrow\ f_{23} = 0{,}5^{0{,}1} \cdot 10{,}0^{0{,}9} \cdot 0{,}03 = 0{,}233,$$

$$a_2 \to a_4:\ l_{24} = 1,\ \eta_{24} = \frac{1}{l_{24}} = 1,\ \tau_{24}^{(0)} = 10{,}0,\ r_k = 0{,}861 \ \Rightarrow$$

$$\Rightarrow\ f_{24} = 1^{0{,}1} \cdot 10{,}0^{0{,}9} \cdot 0{,}86 = 6{,}840,$$

и в результате выбирается направление движения $a_2 \to a_4$, т.к. $f_{24} > f_{23}$. Далее выбор вершин пути итеративно повторяется до тех пор, пока не будет достигнута конечная вершина a_{10} (табл. 2.2).

Таблица 2.2

Формирование пути *P* первого муравья

P	Возможные направления движения	η	τ	r_k	F	Выбранное направление
$[a_1, a_2, a_4]$	a_6	0,167	10,0	0,201	1,345	+
$[a_1, a_2, a_4, a_6]$	a_5	0,167	10,0	0,273	1,812	
	a_9	0,167	10,0	0,672	4,460	+
$[a_1, a_2, a_4, a_6, a_9]$	a_{10}	0,250	10,0	0,319	2,204	+

Таким образом, первый муравей колонии проходит путь $P = [a_1, a_2, a_4, a_6, a_9, a_{10}]$, причем его длина $L(P) = 27$. Далее муравей производит пометку ребер найденного пути порцией феромона $\Delta\tau = \frac{Q}{L(P)} = \frac{1}{27} = 0{,}04$:

$$\tau_{12}^{(1)} = \tau_{12}^{(0)} + \Delta\tau = 10{,}0 + 0{,}04 = 10{,}04,$$
$$\tau_{24}^{(1)} = \tau_{24}^{(0)} + \Delta\tau = 10{,}0 + 0{,}04 = 10{,}04,$$
$$\tau_{46}^{(1)} = \tau_{46}^{(0)} + \Delta\tau = 10{,}0 + 0{,}04 = 10{,}04,$$
$$\tau_{69}^{(1)} = \tau_{69}^{(0)} + \Delta\tau = 10{,}0 + 0{,}04 = 10{,}04,$$
$$\tau_{9,10}^{(1)} = \tau_{9,10}^{(0)} + \Delta\tau = 10{,}0 + 0{,}04 = 10{,}04.$$

После этого производится построение пути вторым муравьем (табл. 2.3).

Таблица 2.3

Формирование пути *P* второго муравья

P	Возможные направления движения	η	τ	r_k	F	Выбранное направление
$[a_1]$	a_2	0,1	10,04	0,162	1,024	+
$[a_1, a_2]$	a_3	0,5	10	0,372	2,759	
	a_4	1	10,04	0,426	3,393	+
$[a_1, a_2, a_4]$	a_6	0,167	10,04	0,082	0,546	+
$[a_1, a_2, a_4, a_6]$	a_5	0,167	10	0,475	3,153	+
	a_9	0,167	10,04	0,071	0,470	
$[a_1, a_2, a_4, a_6, a_5]$	a_3	0,2	10	0,841	5,686	+
	a_7	0,125	10	0,060	0,385	
	a_{10}	1	10	0,293	2,330	

Таким образом, второй муравей попал в тупик, пойдя по пути $P = [a_1, a_2, a_4, a_6, a_5, a_3]$. Отметка феромоном данного пути не производится.

После выполнения движения каждого из муравьев колонии производится испарение феромона со всех ребер графа:

$$\tau_{12}^{(2)} = \tau_{12}^{(1)} \cdot \gamma = 10{,}04 \cdot 0{,}9 = 9{,}03,$$
$$\tau_{13}^{(2)} = \tau_{13}^{(1)} \cdot \gamma = 10 \cdot 0{,}9 = 9,$$
$$\ldots$$

Далее движение муравьев повторяется аналогично рассмотренному выше. При этом первый муравей находит тот же путь $P=[a_1, a_2, a_4, a_6, a_9, a_{10}]$, $L(P)=27$ и помечает его феромоном:

$$\tau_{12}^{(3)}=\tau_{12}^{(2)}+\Delta\tau=9{,}03+0{,}04=9{,}07,$$
$$\tau_{24}^{(3)}=\tau_{24}^{(2)}+\Delta\tau=9{,}03+0{,}04=9{,}07,$$
$$\tau_{46}^{(3)}=\tau_{46}^{(2)}+\Delta\tau=9{,}03+0{,}04=9{,}07,$$
$$\tau_{69}^{(3)}=\tau_{69}^{(2)}+\Delta\tau=9{,}03+0{,}04=9{,}07,$$
$$\tau_{9,10}^{(3)}=\tau_{9,10}^{(2)}+\Delta\tau=9{,}03+0{,}04=9{,}07.$$

Второй муравей повторяет движение первого: $P=[a_1, a_2, a_4, a_6, a_9, a_{10}]$, $L(P)=27$, и помечает феромоном тот же путь:

$$\tau_{12}^{(4)}=\tau_{12}^{(3)}+\Delta\tau=9{,}07+0{,}04=9{,}11,$$
$$\tau_{24}^{(4)}=\tau_{24}^{(3)}+\Delta\tau=9{,}07+0{,}04=9{,}11,$$
$$\tau_{46}^{(4)}=\tau_{46}^{(3)}+\Delta\tau=9{,}07+0{,}04=9{,}11,$$
$$\tau_{69}^{(4)}=\tau_{69}^{(3)}+\Delta\tau=9{,}07+0{,}04=9{,}11,$$
$$\tau_{9,10}^{(4)}=\tau_{9,10}^{(3)}+\Delta\tau=9{,}07+0{,}04=9{,}11.$$

После этого производится испарение феромона:

$$\tau_{12}^{(5)}=\tau_{12}^{(4)}\cdot\gamma=9{,}11\cdot 0{,}9=8{,}20,$$
$$\tau_{13}^{(2)}=\tau_{13}^{(1)}\cdot\gamma=9\cdot 0{,}9=8{,}10,$$
$$\ldots$$

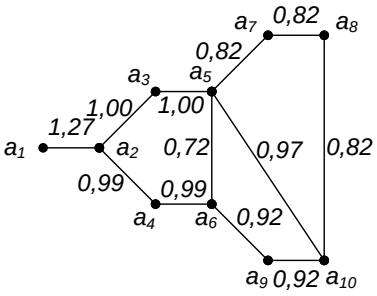
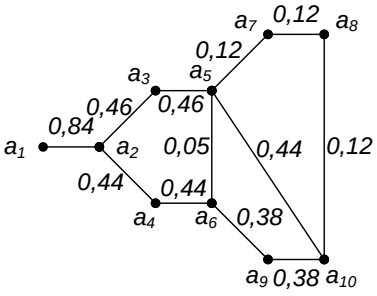
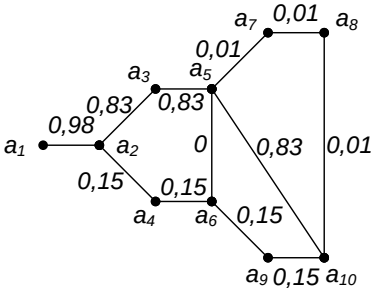
Уже на данном этапе видно, что на пути $P=[a_1, a_2, a_4, a_6, a_9, a_{10}]$, по которому «ходят» муравьи и который на данном этапе считается неплохим, начинает скапливаться большее число феромона по сравнению с другими путями.

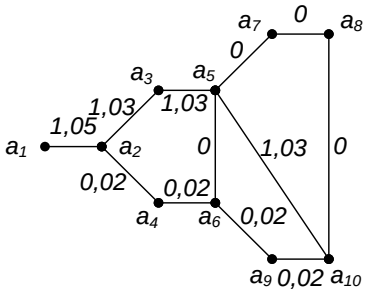
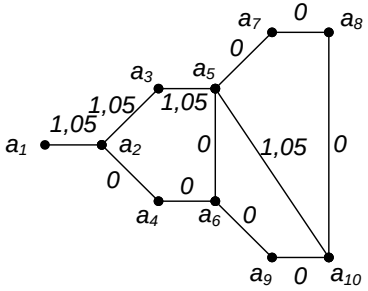
На следующей итерации первый муравей находит тот же самый путь, а второй выбирает путь $P=[a_1, a_2, a_3, a_5, a_7, a_8, a_{10}]$, длина которого $L(P)=32$, соответственно прибавка феромона $\Delta\tau=0{,}03$ на его ребрах меньше, чем в предыдущем случае.

Повторяя процесс движения муравьев, значение феромона на ребрах графа изменяется, как показано в табл. 2.4.

Таблица 2.4

Изменение количества феромона на ребрах графа в зависимости от номера итерации

Номер итерации t	Количество феромона $\tau_{ij}^{(t)}$ на ребрах графа
(1)	(2)
50	
100	
150	

(1)	(2)
200	
250	

Из приведенных в таблице результатов видно, что на ребрах графа, посещаемых муравьями чаще, собирается большее количество феромона. Так к итерации № 100 большинство муравьев помечают феромоном пути $P_1 = [a_1, a_2, a_3, a_5, a_{10}]$, $L(P_1) = 18$ и $P_2 = [a_1, a_2, a_4, a_6, a_9, a_{10}]$, $L(P_2) = 27$. К итерации № 200 большинство муравьев выбирает путь P_1, а феромон с пути P_2 постепенно испаряется, достигая почти нулевого значения к итерации № 250. Начиная с этого момента абсолютное большинство муравьев колонии производят движение по пути P_1, который является оптимальным.

Замечание 2.12. В некоторых задачах колония может прийти подобным образом в локальный, но не глобальный экстремум, и чтобы выбраться из него потребуются дополнительные действия, подобные рассмотренным в разд. 1.1 для задач непрерывной оптимизации.

В природе целью муравьиной колонии является нахождение оптимального или близкого к нему пути, который помечается феромоном сильнее остальных, что в будущем обеспечивает движение по нему абсолютного большинства муравьев. При решении задач дискретной оптимизации важна не сама пометка пути, а факт нахождения решения, лучшего, чем предыдущий рекорд, на что может потребоваться существенно меньшее время (число итераций). Динамика обновления рекорда в рассматриваемом примере приведена в табл. 2.5.

Таблица 2.5

Изменение рекорда в зависимости от числа итераций

Номер итерации	Текущий кратчайший путь (рекорд)	Длина кратчайшего пути
1	$P_1^+ = [a_1, a_2, a_4, a_6, a_9, a_{10}]$	27
7	$P_2^+ = [a_1, a_2, a_4, a_6, a_5, a_{10}]$	24
8	$P_3^+ = [a_1, a_2, a_3, a_5, a_{10}]$	18

Анализ данных таблицы показывает, что первый муравей посетил кратчайший путь уже на 8-й итерации алгоритма, а его более сильная отметка феромоном, обеспечивающая движение по нему всех муравьев колонии, – только к 200-й итерации, т.е. существенно позже.

Алгоритм муравьиной колонии, соответствующий решению задачи о поиске кратчайшего пути, приведен ниже.

1. (инициализация) Положить $L(P^+) := \infty$, $P^+ := []$, $C := 1$, $\tau_{a_i, a_j} := \tau^{(0)}$, $i, j = \overline{1, N}$.
2. (построение пути) Положить $P := [a_{нач}]$, $a_{тек} := a_{нач}$.
3. Сформировать множество S еще не посещенных вершин, достижимых из вершины $a_{тек}$: $S := \{a_{j_1}, a_{j_2}, ..., a_{j_Q}\}$, $(a_{j_k} \notin P) \wedge (l(a_{тек}, a_{j_k}) \neq \infty)$, $k = \overline{1, Q}$.
4. Если $S = \varnothing$, то путь не найден, перейти к п. 11.
5. Для каждой вершины $a_{j_i} \in S$ рассчитать эвристику $F(a_{j_i}) = \left[\frac{1}{l(a_{тек}, a_{j_i})}\right]^{\alpha} \left[\tau_{a_{тек}, a_{j_i}}\right]^{\beta} r_k$.

6. Выбрать вершину $a_{j_i} \in S$, такую что $j_i = \arg\max_{i=\overline{1,|S|}} F\left(a_{j_i}\right)$. Положить $P \coloneqq P \odot a_{j_i}$, $a_{тек} \coloneqq a_{j_i}$.
7. Если $a_{j_i} \neq a_{кон}$, перейти к п. 3.
8. (обновление рекорда) Если $L(P) < L\left(P^+\right)$, положить $L\left(P^+\right) \coloneqq L(P)$, $P^+ \coloneqq P$.
9. (пометка пути феромоном) Положить $\Delta\tau = \frac{1}{L(P)}$. Для всех ребер $\left(a_{j_i}, a_{j_{i+1}}\right) \in P$ произвести их отметку феромоном: $\tau_{a_{i_j}, a_{i_{j+1}}} \coloneqq \tau_{a_{i_j}, a_{i_{j+1}}} + \Delta\tau$.
10. (испарение феромона) Если $C \bmod Z = 0$, положить $\tau_{a_i, a_j} \coloneqq \tau_{a_i, a_j} \cdot \gamma$, $i, j = \overline{1, N}$.
11. Положить $C \coloneqq C + 1$.
12. Если $C \leq C_{\max}$, перейти к п. 2.
13. Конец алгоритма.

Лучшее решение выбирается из $C_{\max}$ кандидатов, полученных на различных итерациях работы алгоритма. На каждой итерации производится построение пути не более чем из N вершин, для каждой из них производится анализ направлений возможного продолжения пути не более чем из N вершин, соответственно одна итерация построения пути в ходе движения муравья характеризуется временной сложностью $O\left(N^2\right)$. Добавление феромона после движения муравья производится для не более чем N ребер за время $O(N)$. Испарение феромона, требующее обновления феромона на $O(M) \simeq O\left(N^2\right)$ ребрах, производится один раз в Z итераций, соответственно его временная сложность $O\left(\frac{N^2}{Z}\right) \simeq O\left(N^2\right)$, т.к. размер колонии Z – константа. Соответственно, асимптотическая временная сложность алгоритма составляет $t \simeq O\left(C_{\max} N^2\right)$. При работе алгоритм хранит текущий путь, на что необходимо $O(N)$ ячеек памяти, и матрицу со значениями феромона на ребрах, на что необходимо $O(M) \simeq O\left(N^2\right)$ ячеек памяти. Следователь-

но, асимптотическая емкостная сложность алгоритма составляет $m \simeq O\left(N + N^2\right) \simeq O\left(N^2\right)$.

Замечание 2.13. Если рассматриваемый граф является неплотным $\left(M \ll N^2\right)$, приведенные выше асимптотические оценки сложности алгоритма могут быть снижены.

Программная реализация, соответствующая приведенному выше алгоритму, представлена ниже.

```
uses
  SysUtils, Math;

const
  N = 10;   { Число вершин в графе }

type
  TMatrix = array [1..N, 1..N] of Double; { Матрица смежности }

  TPath = record
    Path: array [1..N] of Integer;       { Путь в графе }
    Len: Integer;                        { Число вершин в пути }
  end;

var
  { Матрица смежности (отсутствующим ребрам соответствуют
    нулевые элементы) }
  AdjacencyMatrix: TMatrix = (
      ( 0, 10,  0,  0,  0,  0,  0,  0,  0,  0),
      (10,  0,  2,  1,  0,  0,  0,  0,  0,  0),
      ( 0,  2,  0,  0,  5,  0,  0,  0,  0,  0),
      ( 0,  1,  0,  0,  0,  6,  0,  0,  0,  0),
      ( 0,  0,  5,  0,  0,  6,  8,  0,  0,  1),
      ( 0,  0,  0,  6,  6,  0,  0,  0,  6,  0),
      ( 0,  0,  0,  0,  8,  0,  0,  4,  0,  0),
      ( 0,  0,  0,  0,  0,  0,  4,  0,  0,  3),
      ( 0,  0,  0,  0,  0,  6,  0,  0,  0,  4),
      ( 0,  0,  0,  0,  1,  0,  0,  3,  4,  0)
  );

  { Кратчайший путь }
  BestPath: TPath;


{ Определение длины пути }
function GetPathLen(
    const AM: TMatrix; const Path: TPath): Double;
var
  I: Integer;
```

```
begin
  Result := 0;

  for I := 1 to Path.Len-1 do
    Result := Result + AM[Path.Path[I]][Path.Path[I+1]];
end;


{ Вывод найденного пути }
procedure PrintPath(const AM: TMatrix; const Path: TPath);
var
  I: Integer;
begin
  { Путь }
  Write('Path = [');

  for I := 1 to Path.Len do
    Write('a', Path.Path[I], ', ');

  Writeln(#8#8'] ');

  { Длина }
  Writeln('Len = ', GetPathLen(AM, Path):5:2);
end;


function GetAntColonyPath(
    const AM: TMatrix; BegVertex, EndVertex: Integer;
    TriesCnt: Integer; var Path: TPath): Boolean;
var
  I, J, K, CurrVertex, BestVertex: Integer;
  CurrPath: TPath;
  CurrPathLen, BestPathLen, DT, P, BestP: Double;
  Pheromone: TMatrix;           { Количество феромона на дугах }
  Used: array [1..N] of Boolean;
  BestVertexFound, CurrPathFound: Boolean;
const
  { Настроечные параметры }
  PHEROMONE_START_VALUE = 10.0;
  PHEROMONE_PORTION = 1;        { Количество феромона, которое
                                  несет один муравей }
  ALPHA = 0.1;
  BETA = 0.9;
  GAMMA = 0.9; { Коэффициент испарения феромона за итерацию:
                 1.0 - ничего не испаряется (помним все),
                 0.0 - испаряется весь (забываем все) }
  Z = 100;     { Число муравьев в колонии }
begin
  { Начальный путь не найден }
  BestPathLen := 1e100;
  Result := False;

  { Инициализация феромона }
  for I := 0 to High(Pheromone) do
```

```
  for J := 0 to High(Pheromone) do
    Pheromone[I][J] := PHEROMONE_START_VALUE;

{ Перемещение муравьев }
for I := 1 to TriesCnt do begin
  { Муравей находится в начальной вершине }
  CurrVertex := BegVertex;

  for J := 1 to N do
    Used[J] := False;

  CurrPath.Path[1] := BegVertex;
  CurrPath.Len := 1;
  Used[BegVertex] := True;

  CurrPathFound := False;

  repeat
    { Расчет вероятностей переходов }
    BestVertexFound := False;

    for J := 1 to N do
      if (not Used[J]) and          { Вершина еще не посещена }
         (Abs(AM[CurrVertex][J]) > 1e-50) { Есть путь из
                                            текущей вершины }
      then begin
        try
          P := Power(1/AM[CurrVertex][J], ALPHA) *
              Power(Pheromone[CurrVertex][J], BETA) * Random;
        except
          on Exception do     { Защита от переполнения }
            P := 1e10;
        end;

        if not BestVertexFound then begin
          BestVertexFound := True;
          BestVertex := J;
          BestP := P;
        end else if P > BestP then begin
          BestVertex := J;
          BestP := P;
        end;
      end;

    { Из текущей вершины идти некуда, тупик, переходим к
      следующей итерации }
    if not BestVertexFound then
      break;

    { Добавление вершины в путь }
    Inc(CurrPath.Len);
    CurrPath.Path[CurrPath.Len] := BestVertex;

    Used[BestVertex] := True;
```

```
      CurrVertex := BestVertex;

      if CurrVertex = EndVertex then begin
        CurrPathFound := True;
        break;
      end;
    until False;

    CurrPathLen := GetPathLen(AM, CurrPath);

    { Корректировка количества феромона в случае
      нахождения пути }
    if CurrPathFound then
      DT := PHEROMONE_PORTION / CurrPathLen
    else
      DT := 0.0;

    { Испарение предыдущего феромона }
    if I mod Z = 0 then   { Данное условие соответствует
                            пробеганию колонии из M муравьев
                            перед испарением феромона }
      for J := 1 to N do
        for K := 1 to N do
          Pheromone[J][K] := Pheromone[J][K] * GAMMA;

    { Добавление нового феромона }
    for J := 1 to CurrPath.Len-1 do
      Pheromone[CurrPath.Path[J]][CurrPath.Path[J+1]] :=
          Pheromone[CurrPath.Path[J]][CurrPath.Path[J+1]] + DT;

    { Обновление рекорда }
    if CurrPathFound and (CurrPathLen < BestPathLen) then begin
      Path := CurrPath;
      BestPathLen := CurrPathLen;
      Result := True;
    end;
  end;
end;


begin
  if not GetAntColonyPath(AdjacencyMatrix, 1, 10, 8, BestPath)
  then
    Writeln('Path not found')
  else
    PrintPath(AdjacencyMatrix, BestPath);

  Readln;
end.
```

Замечание 2.14. В рассмотренном выше примере и соответствующих ему алгоритму и программной реализации движение муравьев колонии

происходит последовательно, каждый *i*-й муравей имеет возможность использовать пометку феромоном, произведенную $(i-1)$-м муравьем, а испарение феромона производится через каждые *Z* проходов муравьев. Иногда на практике этот процесс организуется в параллельной форме, когда движение *Z* муравьев колонии происходит независимо друг от друга.

Замечание 2.15. В приведенном выше алгоритме и соответствующей ему программной реализации не использовано раннее отсечение неперспективных решений с использованием стратегии ветвей и границ. Это сделано умышленно, т.к. в данном случае нахождение нового пути, пусть и более длинного, чем текущий рекорд, способно «выбросить» колонию из локального экстремума. В некоторых задачах подобное отсечение может быть оправданным, однако в общем случае этого утверждать нельзя.

Замечание 2.16. Использование константных начальных значений феромона на ребрах $\tau^{(0)}$ в некоторых случаях способно снижать разнообразие получаемых решений. Пример подобной ситуации рассмотрен в работе [4]. Выходом из ситуации является использование случайных начальных значений феромона, например, выбираемых в диапазоне $\left[\tau_{\min}^{(0)}; \tau_{\max}^{(0)}\right]$.

Замечание 2.17. В задачах, не связанных непосредственно с обходом некоторого пути в графе (например, при поиске минимальной раскраски графа), рассмотренный выше способ пометки феромоном не подходит, поэтому вместо него может быть использована отметка ребер двудольного графа, отмечающего соответствия между элементами решения [112].

Рассмотренные выше жадный, случайный и взвешенный случайный методы в совокупности с методом муравьиной колонии образуют категорию методов, в основе которых лежит построение решения элемент за элементом. При этом на каждом шаге метода производится выяснение значения очередного элемента решения без изменения предыдущих элементов.

В отличие от этой категории, рассматриваемые в разд. 2.9 и 2.10 методы, напротив, оперируют подмножеством готовых решений и пытаются произвести их модификацию с целью повышения качества.

Содержание