В. А. Балашевич

ОСНОВЫ МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Допущено Министерством высшего и среднего специального образования БССР в качестве учебного пособия для студентов инженерно-экономических и экономических специальностей

МИНСК, «ВЫШЭЙШАЯ ШКОЛА», 1985

ББК 22.18я73
Б20
УДК 519.85(075.8)

Рецензенты: кафедра математических методов и программирования Рижского института инженеров гражданской авиации; *А. М. Дубров*, д-р техн. наук, проф., зав. кафедрой математической статистики Московского экономико-статистического института

Б $\frac{1702000000-043}{М304(05)-85}$ 14—85

ПРЕДИСЛОВИЕ

Одним из основных вопросов экономической политики партии на современном этапе является ускорение экономического и социального развития общества. Решение этой задачи требует дальнейшего совершенствования системы управления экономикой, улучшения плановой работы на базе более широкого применения математических методов и ЭВМ. К настоящему времени наука достигла значительных успехов не только в разработке математического аппарата, соответствующего потребностям экономики, но и выработала определенную методологию его использования.

Наиболее разработанным и широко применяемым сейчас в экономике разделом прикладной математики является математическое программирование. Его рождение по праву связывают с опубликованием в 1939 г. работы советского ученого Л. В. Канторовича «Математические методы организации и планирования производства». Знаменательно следующее: академику Л. В. Канторовичу, математику по специальности, присвоены звания лауреата Ленинской (1965 г.) и Нобелевской (1975 г.) премий за вклад в развитие экономической науки.

В настоящее время курс математического программирования входит в учебные планы всех экономических и инженерно-экономических специальностей как одна из дисциплин, подготавливающих будущих экономистов к решению вопросов организации, планирования и управления на различных уровнях соответствующей отрасли с помощью математических методов и ЭВМ.

В пособии отражены основные разделы математического программирования, используемые при оптимальном планировании: линейное, нелинейное, дискретное программирование, задачи транспортного типа. Главное внимание уделяется методам решения рассматриваемых задач; их обоснования и доказательства даются с мень-

шей детализацией. Широко используются матрично-векторные обозначения, упрощающие изложение.

Приведено достаточное количество примеров, работа над которыми представляет важное условие овладения материалом пособия.

Автор признателен рецензентам профессору А. М. Андронову и коллективу возглавляемой им кафедры математических методов и программирования Рижского института инженеров гражданской авиации, а также профессору А. М. Дуброву за глубокий анализ рукописи и ценные рекомендации по ее улучшению.

Все замечания и пожелания по данному пособию просим присылать по адресу: 220048, Минск, проспект Машерова, 11, издательство «Вышэйшая школа».

Автор

1. ОБЩАЯ ЗАДАЧА ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

1.1. ПОСТАНОВКА ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Математические модели. Математическое программирование занимается исследованием свойств и разработкой методов решения задач следующего вида: найти значения переменных $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, принадлежащих некоторому заданному множеству X и доставляющих максимум (минимум) заданной функции $F(x_1, x_2, \ldots x_n)$ этих переменных. Если для определенности говорить о задаче максимизации, то ее краткая запись будет иметь вид $\max\{F(\mathbf{x})|\mathbf{x} \in X\}$. Источником такого рода задач является любая область практической деятельности человека и прежде всего производство, распределение, обмен и потребление материальных благ, т. е. сфера экономики.

Характерные особенности современной экономики, такие как усложнение связей и зависимостей между элементами экономических систем на всех уровнях, ускорение темпов внедрения научно-технических достижений в производство, необходимость учета ограниченности ресурсов при гигантских масштабах производства и капитальных вложений, приводят к одновременному существованию для каждой экономической задачи множества вариантов ее решения, существенно различающихся между собой по затратам, срокам исполнения, срокам окупаемости, экономическому, социальному эффекту. Если определены условия, при которых осуществляется выбор решения, и указан критерий, в соответствии с которым будет оцениваться его качество (другими словами, задано множество X и функция $F(\mathbf{x})$), то мы получаем сформулированную выше в общем виде задачу математического программирования.

В соответствии с принятой в математическом программировании терминологией элементы множества X называются *допустимыми планами (точками, векторами, ре-*

шениями). Задача математического программирования, для которой существуют допустимые планы, называется *допустимой.* Функция $F(\mathbf{x})$, подлежащая максимизации (или минимизации), называется *целевой.* Допустимый план, доставляющий наибольшее (наименьшее) значение целевой функции, называется *оптимальным планом (точкой, вектором, решением) задачи.*

Множество X допустимых планов задается обычно как множество решений некоторой системы уравнений и (или) неравенств. Система уравнений и неравенств, определяющих множество X, называется *системой ограничений (условий) задачи,* а каждое из уравнений или неравенств — *ограничением (условием) задачи.* В конкретных исследованиях ограничения отражают условия, в которых происходит выбор решения.

В зависимости от особенностей целевой функции и функций, задающих ограничения, задачи математического программирования разделяются на ряд классов.

Задачи линейного программирования возникают, если все соотношения в их формулировке линейны: целевая функция — линейная форма $z=\mathbf{cx}=c_1x_1+c_2x_2+...+c_nx_n$ переменных $x_1, x_2, ..., x_n$, ограничения — линейные равенства и неравенства, задающие выпуклое многогранное множество. Различия в свойствах этого множества, определяемые особенностями системы ограничений, приводят к тому, что среди задач линейного программирования выделяются частные постановки: транспортная, распределительная, блочная и другие задачи.

Задачи нелинейного программирования — такие задачи, в постановке которых нелинейна хотя бы одна из функций. Если о свойствах функций ничего более не сообщается, задача является общей задачей нелинейного программирования. Если же известно, что ограничения задают выпуклую область, на которой ищется максимум целевой функции, выпуклой вверх, или минимум функции, выпуклой вниз,— получаем *задачу выпуклого программирования.* Частным случаем задачи выпуклого программирования является *задача квадратичного программирования.* В ней множество допустимых решений задается линейными ограничениями и представляет, следовательно, выпуклое многогранное множество, а целевая функция — квадратичная форма, неположительно определенная для задач максимизации и неотрицательно определенная для задач минимизации.

Задачи целочисленного программирования — это те задачи математического программирования, в постановке которых присутствует требование целочисленности всех или части переменных (соответственно полностью или частично целочисленные задачи). Таким образом, можно говорить о целочисленных задачах нелинейного, выпуклого, линейного программирования. Они, в свою очередь, являются частным случаем более широкого класса *задач дискретного программирования*, которое охватывает задачи, определенные на конечных множествах допустимых решений.

При рассмотрении задач математического программирования будем различать два этапа: постановку задачи и ее решение. Математика, как известно, исследует не реальные объекты и отношения, а их абстракции. Одной из таких абстракций является *математическая модель* — формальное описание изучаемого явления, отражающее его наиболее существенные для данного конкретного случая черты. К модели предъявляются противоречивые требования: с одной стороны, она должна быть по возможности простой, а с другой,— достаточно точной. Вопрос о том, что значит «по возможности» и «достаточно» обычно решается вне рамок математического программирования. На некоторых примерах покажем, как могут возникать задачи математического программирования, в частности линейного.

Задача о раскрое. Листовой материал поступает на предприятия обычно в виде стандартных форм, из которых затем получают заготовки необходимых размеров. При разделении листов на заготовки все остатки идут в отходы. Количество отходов зависит от принятых вариантов раскроя. Каждый вариант характеризуется количеством заготовок различного вида, выкраиваемых из листа. Задача состоит в том, чтобы составить оптимальный план раскроя, т. е. указать, сколько листов кроить по каждому из вариантов, чтобы получить необходимое количество заготовок каждого типа при минимальных суммарных отходах.

Введем обозначения. Пусть имеется всего m различных видов заготовок, заготовки вида i требуются в количестве b_i, $i=1, ..., m$. Пусть с учетом возможностей расположения заготовок на листе, а также технологических, организационных и других факторов отобраны n рациональных вариантов раскроя листа. Пусть при этом коли-

чество заготовок i-го типа, получаемых из одного листа при использовании j-го варианта раскроя, равно a_{ij}, а отходы равны c_j, $j=1, \ldots, n$.

Если обозначить количество листов, раскраиваемых по варианту j, через $x_j \geqslant 0$, то набор чисел $\mathrm{x}=(x_1, x_2, \ldots, x_n)$ будет представлять некоторый план раскроя. Для того чтобы он был допустим, необходимо выпускать каждую заготовку i в количестве, не меньшем заданного, т. е. выполнить условие

$$\sum_{j=1}^{n} a_{ij}x_j \geqslant b_i, \quad i=1, \ldots, m. \tag{1.1}$$

Кроме того, по своему смыслу переменные x_j — компоненты плана — не могут быть отрицательными числами, что тоже должно быть отражено в условиях задачи:

$$x_j \geqslant 0, \quad j=1, \ldots, n. \tag{1.2}$$

Для того чтобы этот план был оптимален, необходимо подобрать значения x_j так, чтобы суммарные отходы z были минимальными. Это условие запишется в виде

$$\min z = \sum_{j=1}^{n} c_j x_j. \tag{1.3}$$

Таким образом, задача описывается условиями (1.1)—(1.3). Более сложные постановки могут включать требование целочисленности неизвестных x_j, существенное при малых значениях переменных, отражать наличие листов различного формата, учитывать другие критерии качества.

Пример 1.1. Построить модель задачи раскроя при следующих исходных данных. Из листового проката необходимо вырезать заготовки двух типов (A и B) для производства 60 штук изделий. Для одного изделия требуются три заготовки типа A и восемь заготовок типа B. Размеры листа, а также размеры и конфигурация заготовок позволяют выбрать четыре рациональных варианта раскроя листа (табл. 1.1).

Так, при раскрое листа по первому варианту получаем четыре заготовки типа A, и отход составляет 12 условных единиц. При раскрое по второму варианту получаем три заготовки типа A, четыре заготовки типа B, и отход составляет 5 условных единиц и т. д. При этих исходных данных математическая модель задачи (1.1) — (1.3) примет вид

$$\min z = 12x_1+5x_2+3x_3;\ 4x_1+3x_2+2x_3+x_4 \geqslant 180;$$
$$4x_2+6x_3+10x_4 \geqslant 480;\ x_1 \geqslant 0,\ x_2 \geqslant 0,\ x_3 \geqslant 0,\ x_4 \geqslant 0.$$

Таблица 1.1

Заготовка i	Вариант раскроя j				Потребность b_i
	1	2	3	4	
A	4	3	2	1	180
B	0	4	6	10	480
Отходы c_j	12	5	3	0	

Производственная задача 1. Предприятие может использовать для производства однородного продукта n различных технологий и m видов ресурсов. Запас ресурса каждого вида i (сырье, полуфабрикаты, энергия, труд, оборудование и т. д.) ограничен величиной b_i, $i=1, ..., m$. Известно, что при использовании технологии j в единицу времени расходуется a_{ij} единиц i-го ресурса и производится c_j единиц продукта. Требуется определить интенсивность использования каждой технологии, чтобы при наличных ресурсах выпустить максимальное количество продукта.

Обозначим через $x_j \geqslant 0$ время, в течение которого применяется j-я технология, $j=1, ..., n$, а через $z(x_1, x_2, ..., x_n)$ — общий выпуск продукта. Тогда условия задачи можно формализовать в следующих соотношениях. Расход ресурса i за время x_j применения технологии j составит $a_{ij}x_j$, а его общее потребление всеми технологиями не должно превышать наличия ресурса, т. е. $a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \leqslant b_i$, $i=1, ..., m$. Выпуск продукта за время работы по технологии j составит величину c_jx_j, а по всем технологиям выразится функцией $z(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n$, значение которой должно быть максимизировано соответствующим выбором переменных x_j с учетом наложенных на них ограничений. В итоге получим задачу в виде

$$\max z = \sum_{j=1}^{n} c_j x_j; \tag{1.4}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i, \quad i = 1, ..., m; \tag{1.5}$$

$$x_j \geqslant 0, \quad j = 1, ..., n. \tag{1.6}$$

Производственная задача 2. Предприятие может выпускать n типов продукции, используя m видов ресурсов. Запас ресурса каждого вида i ограничен величиной b_i, $i=1, \ldots, m$. Известно, что на производство единицы продукта j расходуется a_{ij} единиц ресурса i, а доход от реализации единицы продукта j равен c_j, $j=1, \ldots, n$. Задача состоит в том, чтобы определить для каждого продукта объем производства, который позволит при наличных ресурсах получить максимальный общий доход. Сбыт всей выпущенной продукции обеспечен.

Обозначим через $x_j \geqslant 0$ планируемое к выпуску количество j-го продукта, $j = 1, \ldots, n$, а через $z(x_1, x_2, \ldots, x_n)$ — суммарный доход от реализации всей продукции. Расход ресурса i на производство x_j единиц продукта j составит $a_{ij}x_j$ единиц, а вся программа выпуска продукции $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ должна быть составлена так, чтобы расход данного ресурса на ее выполнение не превосходил его запаса: $\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i$, $i = 1, \ldots, m$.

Продукт j, произведенный в количестве x_j, обеспечит доход c_jx_j, а вся производственная программа $(x_1, x_2, \ldots, x_n)$ — доход $z(\mathbf{x}) = \sum_{j=1}^{n} c_jx_j$. Необходимо определить, при каких значениях неизвестных $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ функция $z(\mathbf{x}) = \sum_{j=1}^{n} c_jx_j$ принимает максимальное значение. Сводя вместе все соотношения, получаем ту же математическую модель (1.4)—(1.6), что и для производственной задачи 1.

Пример 1.2. Построить модель производственной задачи, использующей две технологии и три вида ресурсов при значениях величин a_{ij}, b_i и c_j, заданных в табл. 1.2.

Таблица 1.2

Вид ресурса i	Коэффициенты a_{ij}		Запас ресурса b_i
	a_{i1}	a_{i2}	
1	1	3	270
2	4	6	600
3	3	1	240
Доход c_j	3	2	

Подставляя коэффициенты из табл. 1.2 в модель (1.4)—(1.6), получаем задачу:

$$\max z = 3x_1 + 2x_2; \tag{1.7}$$

$$\left.\begin{aligned} x_1 + 3x_2 &\leqslant 270;\\ 4x_1 + 6x_2 &\leqslant 600;\\ 3x_1 + x_2 &\leqslant 240;\\ x_1 \geqslant 0,\ x_2 &\geqslant 0. \end{aligned}\right\} \tag{1.8}$$

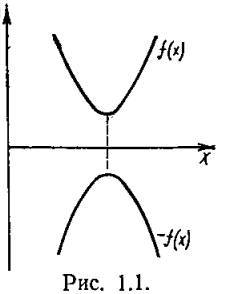

Рис. 1.1.

Эквивалентные формы задач линейного программирования. При постановке задач линейного программирования возможны различные случаи: целевая функция в одних задачах должна быть максимизирована, в других — минимизирована, ограничения на переменные могут быть заданы равенствами или неравенствами. Кроме того, в некоторых задачах требование неотрицательности распространяется не на все переменные. Важно однако, что во всех этих случаях различия носят формальный характер. В частности, с формальной точки зрения нет необходимости различать задачи максимизации и минимизации целевой функции z, так как одна задача сводится к другой изменением знака z. Точка оптимума и абсолютное значение целевой функции при этом не меняются, как показано на рис. 1.1 для произвольной функции одной переменной.

Что касается системы ограничений, то в зависимости от ее особенностей все постановки общей задачи линейного программирования укладываются в три основные формы: стандартную, каноническую и общую.

Стандартная (симметричная) задача имеет вид

$$\max z = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n;$$

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leqslant b_1;$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leqslant b_2;$$

$$\ldots\ldots\ldots\ldots\ldots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leqslant b_m;$$

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

Это задача с однотипными ограничениями-неравенствами и неотрицательными переменными. Очевидно, задача производственного планирования (1.4) — (1.6) является стандартной.

Введя обозначения $\mathbf{c}=(c_1, c_2, \ldots, c_n)$;

$$A=\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}; \quad \mathbf{x}=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}; \quad \mathbf{b}=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

можем записать стандартную задачу в *матричной форме*:

$$\max \mathbf{cx};\ A\mathbf{x}\leqslant\mathbf{b};\ \mathbf{x}\geqslant 0.$$

Здесь линейная форма $c_1x_1+c_2x_2+\ldots+c_nx_n$ записана в виде скалярного произведения $\mathbf{cx}$ векторов $\mathbf{c}$ и $\mathbf{x}$.

Введя для i-го столбца матрицы A обозначение $\mathbf{a}^i$, получим еще одну — *векторную форму* записи этой задачи:

$$\max z=c_1x_1+c_2x_2+\ldots+c_nx_n;$$

$$\mathbf{a}^1x_1+\mathbf{a}^2x_2+\ldots+\mathbf{a}^nx_n\leqslant\mathbf{b};$$

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

В стандартной задаче соотношение между числом неизвестных n и числом ограничений m может быть произвольным: задача имеет смысл при $n>m$, $n=m$, $n<m$.

Канонической задачей называется задача, где все переменные неотрицательны и ограничения имеют форму равенств:

$$\max z=\mathbf{cx};\ A\mathbf{x}=\mathbf{b};\ \mathbf{x}\geqslant 0.$$

В канонической задаче число неизвестных всегда больше числа уравнений ($n>m$). Действительно, если число неизвестных равно числу уравнений и уравнения линейно независимы, то система имеет единственное решение, и задача оптимизации не возникает. Если же уравнений больше числа неизвестных, то они либо линейно зависимы, и тогда часть из них можно отбросить, либо противоречивы, и тогда задача не имеет допустимых решений.

Замечание 1.1. Не умаляя общности, будем считать, что среди уравнений системы ограничений канонической задачи нет «лишних», т. е. что они образуют систему линейных независимых уравнений, и матрица условий задачи имеет ранг m.

Частным случаем канонической задачи является *задача в базисной форме*, отличающаяся тем, что все коэффициенты вектора ограничений $\mathbf{b}$ неотрицательны, $b_i\geqslant$

$\geqslant 0$, $i=1, \ldots, m$, и в каждом уравнении имеется переменная с коэффициентом 1, которая не входит ни в одно из остальных уравнений. Переменная с таким свойством называется *базисной*.

Общей задачей линейного программирования называется задача, в которой имеются ограничения в виде как равенств, так и неравенств. Кроме того, требование неотрицательности может быть наложено не на все переменные.

Эквивалентные преобразования. Нетрудно видеть, что стандартная и каноническая задачи являются частными случаями общей. Каждая из них используется в своей определенной области. Так, формализация производственных и планово-экономических задач приводит чаще всего к построению стандартной или общей задачи. Основные же методы решения разработаны для канонической задачи. При этом все три формулировки эквивалентны между собой: общая задача простыми приемами приводится к стандартной или канонической, а последние могут быть преобразованы друг в друга так, что решение одной задачи однозначно определяет решение другой.

Опишем способы этих преобразований. Для перехода от канонической к стандартной задаче надо каждое уравнение $\sum_{j=1}^{n} a_{ij}x_j = b_i$ заменить двумя неравенствами $\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i$; $\sum_{j=1}^{n} a_{ij}x_j \geqslant b_i$.

Для перехода от стандартной задачи с числом неравенств m и неизвестных n к канонической вводятся дополнительные неотрицательные переменные x_{n+i}, $i = 1, \ldots, m$, и каждое ограничение-неравенство вида $\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i$ заменяется ограничением-равенством $\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i$

Нетрудно показать, что если решение сформулированной при этом канонической задачи имеет вид $(x_1^*, x_2^*, \ldots, x_n^*, x_{n+1}^*, \ldots, x_{n+m}^*)$, то решением стандартной задачи будет вектор $(x_1^*, x_2^*, \ldots, x_n^*)$. Наконец, если некоторая переменная x_j не имеет ограничения на знак, ее следует заменить разностью двух переменных, например $x_j = u_j - v_j$, наложив условия $u_j \geqslant 0$, $v_j \geqslant 0$.

В результате этих преобразований любая задача может быть приведена к нужному виду. Таким образом, чтобы уметь решать все задачи линейного программирования, достаточно владеть способом решения одной из них — канонической.

Пример 1.3. Привести к каноническому виду задачу

$$\max z = 8x_1 + 2x_2;\ x_1 + x_2 \leqslant 16;$$
$$4x_1 - 2x_2 \leqslant 20;\ x_1 \geqslant 0.$$

В данной задаче оба ограничения имеют вид неравенств, на переменную x_2 не наложено требование неотрицательности. Для приведения задачи к каноническому виду введем в ограничения-неравенства дополнительные неотрицательные переменные x_3, x_4, которые войдут в целевую функцию с коэффициентами 0, а неограниченную по знаку переменную x_2 заменим разностью двух неотрицательных переменных u_2, v_2. Получим

$$\max z = 8x_1 + 2u_2 - 2v_2;\ x_1 + u_2 - v_2 + x_3 = 16;$$
$$4x_1 - 2u_2 + 2v_2 + x_4 = 20;\ x_1 \geqslant 0,\ u_2 \geqslant 0,\ v_2 \geqslant 0,\ x_3 \geqslant 0,\ x_4 \geqslant 0.$$

1.2. СВОЙСТВА ЗАДАЧИ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Геометрическое представление задачи. В задаче линейного программирования всегда имеются ограничения, задающие область допустимых решений, из которой допускается выбирать значения переменных. Без указания такой области задача отыскания экстремума линейной функции не имеет смысла, поскольку частные производные такой функции — коэффициенты при переменных — нигде не обращаются в нуль, что необходимо для существования экстремума во внутренней точке области. Отсюда видно, что точка максимума линейной формы задачи должна лежать на границе допустимой области. Это свойство задачи линейного программирования в случае двух переменных получает простое геометрическое обоснование. Рассмотрим для определенности условия задачи (1.7) — (1.8). Построим для нее область допустимых решений и исследуем поведение целевой функции на ней.

Как известно, каждая прямая $a_1x_1 + a_2x_2 = b$ делит плоскость на две полуплоскости. Точки (x_1, x_2) плоскости, удовлетворяющие неравенству $a_1x_1 + a_2x_2 \leqslant b$, находятся по одну сторону от граничной прямой и на ней, удовлетворяющие неравенству $a_1x_1 + a_2x_2 \geqslant b$ по другую сторону и на самой прямой. Следовательно, для построения допустимой области, заданной неравенствами (1.8),

необходимо прежде всего построить на плоскости многоугольник, ограниченный координатными осями $x_1=0$; $x_2=0$ и прямыми $x_1+3x_2=270$; $4x_1+6x_2=600$; $3x_1+x_2=240$ (рис. 1.2, линии *1*, *2*, *3* соответственно). Учитывая требования $x_1\geqslant 0$, $x_2\geqslant 0$, будем рассматривать только первый октант координатной плоскости. Любая точка, лежащая внутри или на границе заштрихованной области, удовлетворяет всем неравенствам системы (1.8). Для того чтобы убедиться в этом, достаточно подставить любую из точек заштрихованной области, например начало координат (0, 0), в каждое из неравенств системы. Аналогично проверяется, что любая точка, лежащая вне заштрихованной области, не удовлетворяет хотя бы одному из неравенств системы 1.8.

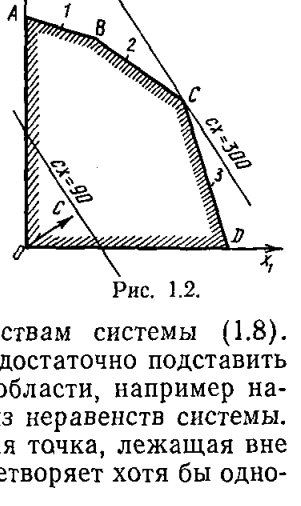

Рис. 1.2.

Итак, допустимыми решениями являются только точки заштрихованной области. Оптимальным решением будет та из точек, в которой целевая функция (1.7) максимальна. Уравнение целевой функции $c_1x_1+c_2x_2=z$ для любого фиксированного z представляет собой прямую. При различных значениях z будем иметь параллельные между собой прямые. Действительно, если представить их уравнения в виде $x_2=z/c_2-(c_1/c_2)x_1$, то видно, что коэффициент наклона c_1/c_2 не зависит от z. Приравняем правую часть равенства (1.7) к некоторой постоянной z и проведем прямую $3x_1+2x_2=z$ так, чтобы она пересекала заштрихованную область. В нашем примере можно положить $z=90$. Увеличивая z, будем передвигать прямую $3x_1+2x_2=z$ параллельно самой себе до тех пор, пока она и заштрихованная область будут иметь хотя бы одну общую точку. Направление перемещения задается направляющим вектором $\mathbf{c}=(c_1, c_2)=(3, 2)$, координатами которого являются коэффициенты при неизвестных линейной формы. Он перпендикулярен к прямой $c_1x_1+c_2x_2=z$ и указывает направление возрастания z.

В нашем примере оказывается, что максимальное значение $z=300$ достигается в точке C с координатами (60, 60), которая является точкой пересечения прямых $4x_1+$

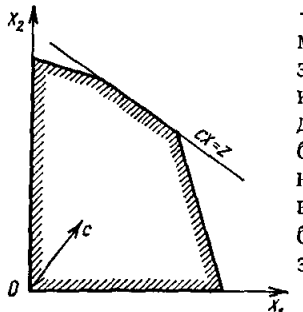

Рис. 1.3.

$+6x_2=600$, $3x_1+x_2=240$. Если бы мы начали процесс с большего значения z, допустим $z=450$, при котором прямая не пересекает допустимую область, то ее надо было бы двигать, уменьшая z, в направлении, противоположном вектору $\mathbf{c}$, параллельно самой себе до встречи с первой точкой этой области.

Очевидно, при других соотношениях между коэффициентами c_1, c_2 угол наклона целевой функции был бы другим и оптимальной могла бы стать одна из точек A, B или D. В частном случае, когда коэффициенты целевой функции пропорциональны коэффициентам какого-либо ограничения, получаем прямую, проходящую через соответствующую сторону многоугольника (рис. 1.3). Оптимум достигается сразу в двух крайних точках, а вместе с этим — и в любой точке соединяющего их отрезка.

Если допустимый многоугольник неограничен, возможны следующие случаи: когда существует единственное решение задачи (рис. 1.4, *а*); или существует бесчисленное множество решений, одно из которых совпадает с крайней точкой многоугольника (рис. 1.4, *б*); или оптимального решения не существует, поскольку на точках допустимого множества можно получить как угодно большие значения целевой функции (рис. 1.4, *в*). Наконец, на рис. 1.4, *г* показан случай, когда ограничения противоречивы и множество допустимых решений пусто.

Таким образом, в задачах линейного программирования вида (1.4) — (1.6) имеется одна из четырех возможностей: 1) задача имеет единственное решение, лежащее

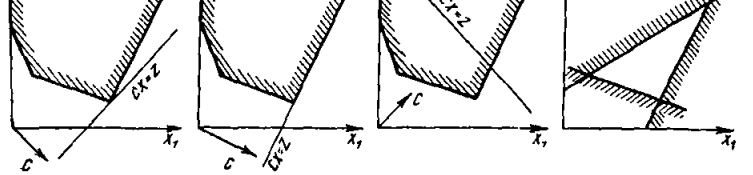

Рис. 1.4.

в крайней точке — вершине выпуклого многоугольника (рис. 1.2, рис. 1.4, *а*); 2) задача имеет бесконечное число решений, среди которых имеется хотя бы одно решение, лежащее в крайней точке допустимого многоугольника (рис. 1.3, рис. 1.4, *б*); 3) задача имеет допустимые решения, но не имеет оптимального решения (рис. 1.4, *в*); 4) задача не имеет допустимых решений, а следовательно, не имеет и оптимального решения (рис. 1.4, *г*).

Эти выводы сохраняются и при переходе к задачам с большим числом переменных, где вместо многоугольного рассматривается многогранное множество допустимых решений. Отсюда вытекает и общая схема решения, которая должна предусматривать проверку совместности условий и ограниченности допустимой области, а также выбор среди всех вершин многогранного выпуклого множества вершины, на которой достигается максимальное значение целевой функции.

Опорный план и его базис. Дальнейшее изучение свойств задачи линейного программирования потребует использования ряда понятий и определений линейной алгебры.

Определение 1.1. Множество X называется *выпуклым*, если оно вместе с двумя своими произвольными точками $\mathbf{x}^1 \in X$, $\mathbf{x}^2 \in X$ содержит и *все точки вида*

$$\mathbf{x} = \alpha\mathbf{x}^1 + (1 - \alpha)\mathbf{x}^2, \qquad (1.9)$$

где $0 \leqslant \alpha \leqslant 1$.

Принимая во внимание, что (1.9) есть параметрическое уравнение прямой, проходящей через точки $\mathbf{x}^1$, $\mathbf{x}^2$, можем дать геометрическое определение выпуклого множества.

Определение 1.2. Множество X называется *выпуклым*, если оно вместе с двумя своими произвольными точками $\mathbf{x}^1 \in X$, $\mathbf{x}^2 \in X$ содержит и *соединяющий их отрезок*.

На рис. 1.5, *а, б* представлены выпуклые множества, на рис. 1.5, *в, г* — невыпуклые множества.

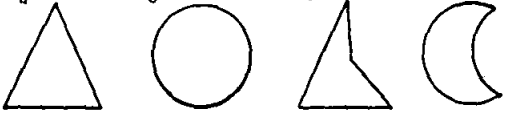

Рис. 1.5.

Определение 1.3. *Выпуклой линейной комбинацией* точек $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ называется сумма вида

$$\alpha_1\mathbf{x}^1+\alpha_2\mathbf{x}^2+\ldots+\alpha_k\mathbf{x}^k, \tag{1.10}$$

где $\alpha_i \geqslant 0$, $i=1, \ldots, k$, $\sum\limits_{i=1}^{k} \alpha_i = 1$.

В соответствии с определением 1.1 множество X называется выпуклым, если оно содержит выпуклую линейную комбинацию любых двух своих точек.

Определение 1.4. Точка выпуклого множества называется *крайней*, если она не может быть представлена в виде выпуклой комбинации каких-либо двух различных точек этого множества.

Например, крайними точками многоугольника являются его вершины.

Определение 1.5. Система векторов $\mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^n$ называется *линейно независимой*, если из равенства

$$\alpha_1\mathbf{a}^1+\alpha_2\mathbf{a}^2+\ldots+\alpha_n\mathbf{a}^n=0 \tag{1.11}$$

следует равенство нулю всех коэффициентов α_i линейной комбинации.

Если же в равенстве (1.11) хотя бы один из коэффициентов α_i отличен от нуля, то система *линейно зависима*. В этом случае вектор $\mathbf{a}^k$, соответствующий коэффициенту $\alpha_k \neq 0$, может быть выражен в виде линейной комбинации остальных векторов:

$$\mathbf{a}^k = -\frac{\alpha_1}{\alpha_k}\mathbf{a}^1 - \ldots - \frac{\alpha_{k-1}}{\alpha_k}\mathbf{a}^{k-1} - \frac{\alpha_{k+1}}{\alpha_k}\mathbf{a}^{k+1} - \ldots - \frac{\alpha_n}{\alpha_k}\mathbf{a}^n.$$

И наоборот, если один из векторов системы линейно выражается через остальные, то система линейно зависима.

Рассмотрим каноническую задачу

$$\max z = c_1x_1+c_2x_2+\ldots+c_nx_n; \tag{1.12}$$

$$\mathbf{a}^1x_1+\mathbf{a}^2x_2+\ldots+\mathbf{a}^nx_n=\mathbf{b}; \tag{1.13}$$

$$x_1 \geqslant 0, \quad x_2 \geqslant 0, \ldots, x_n \geqslant 0, \tag{1.14}$$

где система ограничений записана в виде векторного уравнения (1.13). Из этой записи видно, что вектор $\mathbf{x}=(x_1, x_2, \ldots, x_n)$, $x_j \geqslant 0$, $j=1, \ldots, n$, является планом задачи (1.12) — (1.14) в том и только том случае, когда век-

тор $\mathbf{b}$ выражается как неотрицательная линейная комбинация столбцов матрицы A, где в качестве коэффициентов берутся координаты вектора $\mathbf{x}$.

Определение 1.6. План $\mathbf{x}=(x_1, x_2, ..., x_n)$ задачи (1.12) — (1.14) называется *опорным*, если система векторов-столбцов $\mathbf{a}^j$ матрицы A, соответствующих положительным координатам x_j, линейно независима.

Пример 1.4. Пусть система ограничений некоторой задачи линейного программирования имеет вид:

$$3x_1+x_2+2x_3=1;\ x_1-2x_3=0,\ x_1\geqslant 0,\ x_2\geqslant 0,\ x_3\geqslant 0.$$

Определить, является ли план $\mathbf{x}=(1/2, 0, 1/4)$ опорным.

Положительным координатам плана $x_1=1/2$, $x_3=1/4$ соответствуют векторы-столбцы $\mathbf{a}^1$ и $\mathbf{a}^3$ матрицы A. Если эти векторы линейно независимы, то план $\mathbf{x}=(1/2, 0, 1/4)$ будет опорным.

Для суждения о линейной независимости векторов $\mathbf{a}^1$ и $\mathbf{a}^3$ нужно вычислить определитель, составленный из координат этих векторов:

$$\Delta=\begin{vmatrix}3 & 2\\ 1 & -2\end{vmatrix}=-6-2=-8\neq 0.$$

Определитель Δ отличен от нуля, следовательно векторы $\mathbf{a}^1$ и $\mathbf{a}^3$ линейно независимы и план $\mathbf{x}=(1/2, 0, 1/4)$ является опорным.

Из определения опорного плана следует, что число его положительных координат не может превышать m, где m — число ограничений задачи. Действительно, векторы $\mathbf{a}^j$ являются m-мерными, а в m-мерном пространстве максимальное число линейно независимых векторов равно m.

Определение 1.7. Пусть ранг матрицы A равен m. *Базисом* опорного плана называется система из m линейно независимых векторов-столбцов матрицы A, которая включает все векторы, отвечающие положительным координатам опорного плана.

В примере 1.4 векторы $\mathbf{a}^1$, $\mathbf{a}^3$ составляют базис, соответствующий плану $\mathbf{x}=(1/2, 0, 1/4)$.

Определение 1.8. Опорный план называется *невырожденным*, если число его положительных координат равно m, и *вырожденным* в противном случае.

Из определений 1.7 и 1.8 следует, что базис невырожденного опорного плана определяется единственным образом, а у вырожденного их может быть несколько.

Свойства допустимых планов. Рассмотрим задачу линейного программирования

$$\max\sum_{j=1}^{n}c_jx_j; \qquad (1.15)$$

$$\sum_{j=1}^{n} a_{ij}x_j = b_i,\ i = 1,\ \ldots,\ m; \tag{1.16}$$

$$x_j \geqslant 0,\ j = 1,\ \ldots,\ n, \tag{1.17}$$

которую при необходимости будем записывать также в матричной форме

$$\max \mathbf{cx}; \tag{1.18}$$

$$A\mathbf{x} = \mathbf{b}; \tag{1.19}$$

$$\mathbf{x} \geqslant 0. \tag{1.20}$$

Основные свойства этой задачи формулируются в следующих теоремах.

Теорема 1.1. *Множество планов задачи линейного программирования выпукло (если оно не пусто).*

◆ Необходимо показать, что если $\mathbf{x}^1$ и $\mathbf{x}^2$ — планы задачи линейного программирования (1.18) — (1.20), то их выпуклая линейная комбинация $\mathbf{x} = \alpha_1\mathbf{x}^1 + \alpha_2\mathbf{x}^2$, $\alpha_1 \geqslant 0$, $\alpha_1 + \alpha_2 = 1$ также является ее планом. Итак, $\mathbf{x}^1$ и $\mathbf{x}^2$ — планы задачи. Тогда справедливы равенства $A\mathbf{x}^1 = \mathbf{b}$; $A\mathbf{x}^2 = \mathbf{b}$. Возьмем два любых числа α_1 и α_2, такие, что $\alpha_1 \geqslant 0$, $\alpha_2 \geqslant 0$, $\alpha_1 + \alpha_2 = 1$, и умножим первое равенство на α_1, второе — на α_2. Складывая результаты, получаем: $A(\alpha_1\mathbf{x}^1 + \alpha_2\mathbf{x}^2) = \mathbf{b}$, т. е. линейная комбинация $\mathbf{x} = \alpha_1\mathbf{x}^1 + \alpha_2\mathbf{x}^2$ удовлетворяет системе (1.19). Но так как $\mathbf{x}^1 \geqslant 0$, $\mathbf{x}^2 \geqslant 0$, $\alpha_1 \geqslant 0$, $\alpha_2 \geqslant 0$, то и $\mathbf{x} \geqslant 0$, т. е. удовлетворяет условию (1.20). Таким образом, $\mathbf{x}$ — план задачи линейного программирования. ◆

Так как каждое из m уравнений системы (1.16) с геометрической точки зрения представляет уравнение гиперплоскости в n-мерном пространстве, то выпуклое множество из теоремы 1.1 есть множество всех точек с неотрицательными координатами, принадлежащих всем этим m гиперплоскостям. Это множество может быть ограниченным, и тогда оно называется *выпуклым многогранником*, а его крайние точки — вершинами. Если множество неотрицательных решений системы (1.16) неограниченно, то его называют *выпуклым многогранным множеством*. Будем обозначать выпуклое множество планов задачи линейного программирования через X.

Теорема 1.2. *Если целевая функция $z = \mathbf{cx}$ имеет максимум на выпуклом многограннике допустимых решений X, то он достигается в вершине этого многогранника.*

◆ Обозначим вершины многогранника решений через $\mathbf{x}^1$, $\mathbf{x}^2$, ..., $\mathbf{x}^n$, а точку, где функция z имеет максимум, через $\mathbf{x}^0$. Если среди вершин $\mathbf{x}^i$, $i = 1, ..., n$, найдется хотя бы одна, допустим $\mathbf{x}^k$, для которой $\max z = \mathbf{c}\mathbf{x}^0 = \mathbf{c}\mathbf{x}^k$, то теорема доказана. Предположим противное, т. е. что $\mathbf{x}^0$ не является крайней точкой. Отсюда для каждой из вершин $\mathbf{x}^i$ выполняется соотношение $\mathbf{c}\mathbf{x}^0 > \mathbf{c}\mathbf{x}^i$, $i = 1, ..., n$. Тогда, представляя точку $\mathbf{x}^0$ в виде выпуклой линейной комбинации крайних точек $\mathbf{x}^0 = \sum_{i=1}^{n} \alpha_i \mathbf{x}^i$, $\alpha_i \geqslant 0$, $\sum_{i=1}^{n} \alpha_i = 1$, получаем $\mathbf{c}\mathbf{x}^0 = \mathbf{c}\sum_i \alpha_i \mathbf{x}^i = \sum_i \alpha_i \mathbf{c}\mathbf{x}^i < \sum_i \alpha_i \mathbf{c}\mathbf{x}^0 = \mathbf{c}\mathbf{x}^0 \sum_i \alpha_i = \mathbf{c}\mathbf{x}^0$, что является противоречием. ◆

Итак, в ограниченной выпуклой многогранной области всегда существует вершина, в которой целевая функция принимает максимальное значение. Однако это не означает, что оптимум всегда достигается в одной точке.

Теорема 1.3. *Если целевая функция имеет максимум более чем в одной вершине многогранника X, то этот максимум достигается в любой точке, являющейся выпуклой линейной комбинацией этих вершин.*

◆ Пусть линейная форма $z = \mathbf{c}\mathbf{x}$ достигает максимума в нескольких вершинах $\mathbf{x}^1$, $\mathbf{x}^2$, ..., $\mathbf{x}^n$ многогранника, т. е. $\max z = \mathbf{c}\mathbf{x}^i = d$, $i = 1, ..., n$. Тогда для любой выпуклой линейной комбинации $\mathbf{x} = \sum_{i=1}^{n} \alpha_i x^i$, $\alpha_i \geqslant 0$, $\sum_{i=1}^{n} \alpha_i = 1$, этих вершин имеем $z = \mathbf{c}\mathbf{x} = \mathbf{c} \sum_{i=1}^{n} \alpha_i \mathbf{x}^i = \sum_{i=1}^{n} \alpha_i \mathbf{c}\mathbf{x}^i = d \sum_{i=1}^{n} \alpha_i = d$. ◆

Итак, для отыскания оптимального решения канонической задачи достаточно исследовать вершины выпуклого многогранника X, или, что то же, опорные планы, как это будет показано в следующей теореме.

Теорема 1.4. *Вектор* $\mathbf{x}$ *является опорным планом задачи* (1.12) — (1.14) *тогда и только тогда, когда* $\mathbf{x}$ *— вершина многогранника X допустимых планов.*

◆ Необходимость. Пусть опорным является план $\mathbf{x} = (x_1, x_2, ... x_k, 0, ..., 0)$, где равны нулю последние $n-k$ координат. Тогда в соответствии с (1.13) можно записать $\mathbf{a}^1 x_1 + \mathbf{a}^2 x_2 + ... + \mathbf{a}^k x_k = \mathbf{b}$, где по определению векторы $\mathbf{a}^i$, $i = 1, ..., k$, образующие базис опорного пла-

на, линейно независимы. Предположим, что точка $\mathbf{x}$ не является вершиной многогранника X. Тогда она может быть представлена в виде выпуклой комбинации любых двух других точек $\mathbf{x}^1, \mathbf{x}^2 \in X$: $\mathbf{x} = \alpha_1 \mathbf{x}^1 + \alpha_2 \mathbf{x}^2$, $\alpha_1 \geqslant 0$, $\alpha_2 \geqslant 0$, $\alpha_1 + \alpha_2 = 1$. Поскольку $\mathbf{x}^1 \geqslant \mathbf{0}$, $\mathbf{x}^2 \geqslant 0$, $\alpha_1 \geqslant 0$, $\alpha_2 \geqslant 0$, то нулевым координатам вектора $\mathbf{x}$ могут отвечать только такие же нулевые координаты векторов $\mathbf{x}^1$, $\mathbf{x}^2$, т. е. $\mathbf{x}^1 = (x_1^1, x_2^1, \ldots, x_k^1, 0, \ldots, 0)$; $\mathbf{x}^2 = (x_1^2, x_2^2, \ldots, x_k^2, 0, \ldots 0)$. Планы $\mathbf{x}^1$ и $\mathbf{x}^2$ удовлетворяют (1.13):

$$\mathbf{a}^1 x_1^1 + \mathbf{a}^2 x_2^1 + \ldots + \mathbf{a}^k x_k^1 = \mathbf{b}; \quad \mathbf{a}^1 x_1^2 + \mathbf{a}^2 x_2^2 + \ldots + \mathbf{a}^k x_k^2 = \mathbf{b}.$$

Вычитая второе равенство из первого, получаем

$$(x_1^1 - x_1^2)\,\mathbf{a}^1 + (x_2^1 - x_2^2)\,\mathbf{a}^2 + \ldots + (x_k^1 - x_k^2)\,\mathbf{a}^k = \mathbf{0}.$$

Отсюда следует, что если не выполняется хотя бы одно из равенств $x_i^1 = x_i^2$, то векторы $\mathbf{a}^1$, $\mathbf{a}^2$, ...$\mathbf{a}^k$ линейно зависимы, что противоречит предположению о том, что они образуют базис опорного плана. Следовательно, $x_i^1 = x_i^2$, $i = 1, \ldots, k$, и точку x невозможно представить как выпуклую линейную комбинацию других двух точек многогранника X, т. е. $\mathbf{x}$ — крайняя точка.

Достаточность. Пусть вектор $\mathbf{x} = (x_1, x_2, \ldots, x_k, 0, \ldots, 0)$ определяет некоторую вершину многогранника X. Для доказательства того, что $\mathbf{x}$ — опорный план, достаточно показать, что векторы-столбцы $\mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^k$ матрицы A линейно независимы. Проведем доказательство от противного. Пусть векторы $\mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^k$ линейно зависимы. Тогда существует такая их линейная комбинация, что

$$\sum_{i=1}^{k} \alpha_i \mathbf{a}^i = \mathbf{0}, \tag{1.21}$$

где не все α_i равны нулю. Как точка допустимого множества вектор $\mathbf{x}$ удовлетворяет условию

$$\sum_{i=1}^{k} x_i \mathbf{a}^i = \mathbf{b}. \tag{1.22}$$

Умножим равенство (1.21) на некоторое число $d > 0$, результат сначала прибавим к равенству (1.22), а затем вычтем из него:

$$\sum_{i=1}^{k} (x_i + d\alpha_i)\,\mathbf{a}^i = \mathbf{b}; \qquad \sum_{i=1}^{k} (x_i - d\alpha_i)\,\mathbf{a}^i = \mathbf{b}.$$

При достаточно малом d в силу положительности x_i все величины $x_i \pm da_i$ положительны. Поэтому векторы

$$\mathbf{x}^1 = (x_1 + da_1,\ x_2 + da_2,\ \ldots,\ x_k + da_k,\ 0,\ \ldots,\ 0);$$

$$\mathbf{x}^2 = (x_1 - da_1,\ x_2 - da_2,\ \ldots,\ x_k - da_k,\ 0,\ \ldots,\ 0)$$

являются планами задачи, при этом $\mathbf{x} = \frac{1}{2}\mathbf{x}^1 + \frac{1}{2}\mathbf{x}^2$ — их выпуклая комбинация. Но это противоречит предположению, что точка $\mathbf{x}$ есть вершина, т. е. крайняя точка многогранника X. Поэтому векторы $\mathbf{a}^1$, $\mathbf{a}^2$, ... $\mathbf{a}^k$ линейно независимы. ◆

Из теоремы 1.4 вытекает, что множество опорных планов задачи линейного программирования совпадает с множеством крайних точек или вершин многогранника X, определяемого условиями задачи. Из эквивалентности алгебраического понятия опорного плана геометрическому понятию вершины многогранника вытекают очевидные следствия.

Следствие 1. Каждая вершина многогранника X имеет не более m положительных координат.

Это очевидно, поскольку число положительных координат опорного плана по его определению не может превышать m.

Следствие 2. Каждой вершине многогранника X соответствует $k \leqslant m$ линейно независимых векторов-столбцов матрицы A.

Это по существу определение 1.6 с заменой понятия опорного плана понятием вершины многогранника.

Итак, если целевая функция задачи линейного программирования ограничена на выпуклом многогранном множестве, заданном условиями задачи, то существует крайняя точка (опорный план) этого множества, в которой целевая функция достигает максимума. Поэтому для решения задачи линейного программирования необходимо исследовать только вершины многогранного множества — опорные планы задачи линейного программирования. В этом и состоит основная идея симплекс-метода решения задач линейного программирования.

1.3. СИМПЛЕКС-МЕТОД

Системы линейных алгебраических уравнений. Основным методом решения задач линейного программирования до настоящего времени остается симплекс-метод,

разработанный американским математиком Д. Данцигом в 1949 г. для задач в канонической форме записи:

$$\max z = \mathbf{cx}; \quad A\mathbf{x} = \mathbf{b}; \quad \mathbf{x} \geqslant \mathbf{0}, \tag{1.23}$$

где A — матрица условий задачи размеров $m \times n$, ранг которой будем всегда считать равным m; $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ — вектор-строка коэффициентов целевой функции; $\mathbf{b}^T = (b_1, b_2, \ldots, b_n)$ — вектор-столбец коэффициентов ограничений. Напомним, что вектор-столбец $\mathbf{x}^T = (x_1, x_2, \ldots, x_n)$, удовлетворяющий условиям $A\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geqslant \mathbf{0}$, называется *допустимым решением* или *планом*, а допустимое решение, доставляющее максимум целевой функции, называется *оптимальным решением* или *оптимальным планом*. Множество векторов $\{\mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant 0\}$ называется *областью допустимых решений* задачи линейного программирования и является выпуклым.

Поскольку в канонических задачах ограничения задаются системой линейных алгебраических уравнений, преобразования которых являются существенным элементом симплекс-метода, приведем сначала необходимые сведения о решении систем линейных уравнений.

Рассмотрим две системы:

$$\left.\begin{aligned} 2x_1 - x_2 + 2x_3 + x_4 &= 1; \\ x_1 + x_2 - x_3 + 3x_4 &= 5; \\ 3x_1 - x_2 + 3x_3 - x_4 &= 5; \end{aligned}\right\} \tag{1.24}$$

$$\left.\begin{aligned} x_1 + 3x_4 &= 1; \\ x_2 - 5x_4 &= 7; \\ x_3 - 5x_4 &= 3. \end{aligned}\right\} \tag{1.25}$$

В системе (1.25) каждое из уравнений содержит переменную, которая входит в него с коэффициентом 1 и исключена из остальных уравнений (входит в них с коэффициентом 0). Такие переменные называются *базисными*, а система вида (1.25) — *системой в базисной форме*. Так, базисными переменными системы (1.25) являются переменные x_1, x_2, x_3. Векторы-столбцы

$$\mathbf{a}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; \qquad \mathbf{a}^2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}; \qquad \mathbf{a}^3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

соответствующие этим переменным, составляют единичный базис системы (1.25). Базис системы (1.24) также может быть образован из векторов-столбцов при пере-

менных x_1, x_2, x_3. Для проверки достаточно убедиться, что определитель третьего порядка, составленный из компонент векторов $\mathbf{a}^1$, $\mathbf{a}^2$, $\mathbf{a}^3$ системы (1.24), отличен от нуля:

$$\Delta = \begin{vmatrix} 2 & -1 & 2 \\ 1 & 1 & -1 \\ 3 & -1 & 3 \end{vmatrix} = 7 - 5 = 2.$$

Преимущество системы (1.25) в том, что ее *общее решение* находится непосредственно: $x_1 = 1 - 3x_4$; $x_2 = 7 + 5x_4$; $x_3 = 3 + 5x_4$. Здесь *базисные (зависимые) переменные* x_1, x_2, x_3 выражены через *свободную (независимую) переменную* x_4. Положив ее равной нулю, получим *частное* решение $x_1 = 1$, $x_2 = 7$, $x_3 = 3$. Частное решение, в котором свободные неизвестные равны нулю, называется *базисным*. Подстановкой проверяем, что вектор $\mathbf{x} = (1, 7, 3, 0)$ является также решением системы (1.24). При этом матрица коэффициентов системы (1.25) в отличие от системы (1.24) содержит единичную подматрицу, благодаря чему и становится возможным непосредственное решение системы (1.25).

Из теории матриц известно, что единичная матрица есть результат умножения квадратной неособенной матрицы на обратную ей. Отсюда вытекает следующая схема решения системы линейных уравнений $A\mathbf{x} = \mathbf{b}$. Разобьем матрицу A на две подматрицы B и R так, что B — квадратная неособенная матрица размеров $m \times m$, R — матрица размеров $m \times (n - m)$. Поскольку матрица B неособенная, ее векторы-столбцы образуют базис системы векторов-столбцов матрицы A. Назовем поэтому матрицу B базисной. Не умаляя общности, будем считать, что матрица B составлена из первых m столбцов матрицы A, т. е. $A = (B, R)$, так как этого всегда можно достичь перестановкой столбцов. Аналогичным образом разобьем компоненты вектора $\mathbf{x}$, так что $\mathbf{x} = (\mathbf{x}_B, \mathbf{x}_R)$, где $\mathbf{x}_B$ — вектор базисных, $\mathbf{x}_R$ — вектор небазисных переменных. В результате система $A\mathbf{x} = \mathbf{b}$ может быть записана в виде $B\mathbf{x}_B + R\mathbf{x}_R = \mathbf{b}$, а ее общее решение — в виде $(\mathbf{x}_B, \mathbf{x}_R)$, где $\mathbf{x}_R$ — произвольный $(n - m)$-мерный вектор, а $\mathbf{x}_B$ выражается как

$$\mathbf{x}_B = B^{-1}\mathbf{b} - B^{-1}R\mathbf{x}_R. \qquad (1.26)$$

Базисное решение $(\mathbf{x}_B, \mathbf{x}_R) = (B^{-1}\mathbf{b}, 0)$, полученное при $\mathbf{x}_R = 0$, единственно, так как B — неособенная матрица.

Если рассматривается система $A\mathbf{x} = \mathbf{b}$ задачи (1.23), то базисное решение $(B^{-1}\mathbf{b}, 0)$ и соответствующий ему базис B называются *допустимыми* при условии $B^{-1}\mathbf{b} \geqslant 0$. Базисное допустимое решение называется *невырожденным*, если все его компоненты положительны, и *вырожденным* в противном случае. Таким образом, понятие базисного допустимого решения совпадает с введенным ранее понятием опорного плана.

Метод исключений Жордана — Гаусса. С вычислительной точки зрения схема решения, основанная на обращении базисной матрицы, может быть реализована различными способами. Рассмотрим один из наиболее удобных — метод исключений Жордана — Гаусса, преобразующий исходную систему

$$\sum_{j=1}^{n} a_{ij}x_j = b_i,\ i = 1,\ \ldots,\ m, \tag{1.27}$$

в базисную

$$x_i + \sum_{j=m+1}^{n} a'_{ij}x_j = b'_i,\ i = 1,\ \ldots,\ m, \tag{1.28}$$

где коэффициенты a'_{ij} являются элементами матрицы $B^{-1}R$ размеров $m \times (n - m)$; коэффициенты b'_i — элементами m-мерного вектора $B^{-1}\mathbf{b}$.

Метод исключений Жордана — Гаусса позволяет решить систему преобразованием ее к эквивалентной системе с единичным базисом или установить ее несовместность. В процессе преобразований при необходимости может быть построена также обратная матрица. На каждом шаге исключений одна из переменных становится базисной: исключается из всех уравнений, кроме одного, где остается с коэффициентом 1.

Рассмотрим этот процесс подробнее. Пусть переменная x_s входит в r-е уравнение с коэффициентом $a_{rs} \neq 0$. Для того чтобы она стала базисной, разделим r-е уравнение на a_{rs} (получим x_s с коэффициентом 1) и результат вычтем из каждого оставшегося уравнения $i = 1,\ \ldots,\ m$, $i \neq r$, умножая его каждый раз на соответствующий коэффициент a_{is} (получим x_s с коэффициентами 0). Совокупность операций, составляющих шаг жордановых исключений, называется *жордановым преобразованием*, коэффициент a_{rs} — *ведущим (разрешающим) элементом*, строка r

и столбец s — соответственно *ведущими строкой* и *столбцом*. Формулы для расчета коэффициентов a'_{ij}, b'_i новой системы, получаемой в результате одного шага жордановых исключений с ведущим элементом a_{rs}, имеют вид:

$$\left.\begin{aligned} a'_{rj} &= \frac{1}{a_{rs}}\, a_{rj},\ j = 1,\ \ldots,\ n; \\ b'_r &= \frac{1}{a_{rs}}\, b_r; \end{aligned}\right\} \qquad (1.29)$$

$$\left.\begin{aligned} a'_{ij} &= a_{ij} - \frac{a_{is}}{a_{rs}}\, a_{rj},\ j = 1,\ \ldots,\ n; \\ b'_i &= b_i - \frac{a_{is}}{a_{rs}}\, b_r; \\ i &= 1,\ \ldots,\ m,\ i \neq r. \end{aligned}\right\} \qquad (1.30)$$

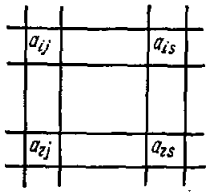

Рис. 1.6.

Вычисления по формулам (1.30) можно описать с помощью так называемого *правила прямоугольника* (рис. 1.6): чтобы найти преобразованный элемент a'_{ij}, надо из элемента a_{ij} вычесть произведение коэффициентов, стоящих напротив его в ведущих столбце и строке, деленное на ведущий элемент, расположенный по диагонали от элемента a_{ij}. При этом элементы ведущего столбца принимают значения $a'_{is} = 0$ для всех $i = 1,\ \ldots,\ m,\ i \neq r$, $a'_{rs} = 1$.

Итак, последовательность действий, выполняемых на одном шаге жордановых исключений в соответствии с формулами (1.29), (1.30), такова: ведущий элемент заменяется единицей; все остальные элементы ведущей строки делятся на ведущий элемент; все остальные элементы ведущего столбца заменяются нулями; элементы, не принадлежащие ведущей строке или столбцу, вычисляются по правилу прямоугольника.

Для преобразования системы (1.27) в базисную систему (1.28) требуется не более m шагов жордановых исключений. На первом шаге в качестве ведущего элемента выбирается любой элемент $a_{rs} \neq 0$. На втором шаге ведущий элемент выбирается в любом уравнении (кроме r-го) среди ненулевых элементов системы, полученной в результате первого шага, и т. д. Если в процессе исключений появляется уравнение, в котором левая часть равна нулю, а свободный член отличен от нуля, — это признак несовместности системы. Если левая и правая части какого-либо уравнения обращаются в нуль, оно является линейной комбинацией остальных уравнений, и его следует исключить

из рассмотрения. Таким образом, в процессе жордановых исключений либо устанавливается несовместность системы уравнений, либо система приводится к эквивалентной базисной форме (1.28), откуда решение получается непосредственно. Формулы жордановых преобразований (1.29), (1.30) применяются одинаково как в случае $m = n$, так и в случае $m < n$.

Пример 1.5. Решить методом жордановых исключений систему (1.24) и получить для нее обратную базисную матрицу.

Запишем систему в виде таблицы, дополненной единичной матрицей (табл. 1.3). Здесь через $\mathbf{e}^i$ обозначен вектор, i-й элемент которого равен 1, а остальные — 0. После трех жордановых преобразований с ведущими элементами, взятыми в рамку, в последних строках табл. 1.3 получаем эквивалентную базисную систему в виде

Таблица 1.3

$\mathbf{a}^1$	$\mathbf{a}^2$	$\mathbf{a}^3$	$\mathbf{a}^4$	$\mathbf{e}^1$	$\mathbf{e}^2$	$\mathbf{e}^3$	$\mathbf{b}$
$\boxed{2}$	—1	2	1	1	0	0	1
1	1	—1	3	0	1	0	5
3	—1	3	—1	0	0	1	5
1	—1/2	1	1/2	1/2	0	0	1/2
0	$\boxed{3/2}$	—2	5/2	—1/2	1	0	9/2
0	1/2	0	—5/2	—3/2	0	1	7/2
1	0	1/3	4/3	1/3	1/3	0	2
0	1	—4/3	5/3	—1/3	2/3	0	3
0	0	$\boxed{2/3}$	—10/3	—4/3	—1/3	1	2
1	0	0	3	1	1/2	—1/2	1
0	1	0	—5	—3	0	2	7
0	0	1	—5	—2	—1/2	3/2	3

$$\begin{aligned} x_1 + \quad & \quad + 3x_4 = 1; \\ x_2 - \quad & \quad - 5x_4 = 7; \\ x_3 - 5x_4 &= 3, \end{aligned}$$

откуда запишем общее решение $x_1 = 1 - 3x_4$; $x_2 = 7 + 5x_4$; $x_3 = 3 + 5x_4$. При этом на месте базисной матрицы исходной системы

$$B = \begin{bmatrix} 2 & -1 & 2 \\ 1 & 1 & -1 \\ 3 & -1 & 3 \end{bmatrix}$$

получаем единичную матрицу, а на месте присоединенной единичной матрицы — обратную базисную матрицу

$$B^{-1} = \begin{bmatrix} 1 & 1/2 & -1/2 \\ -3 & 0 & 2 \\ -2 & -1/2 & 3/2 \end{bmatrix}.$$

Действительно,

$$BB^{-1} = \begin{bmatrix} 2 & -1 & 2 \\ 1 & 1 & -1 \\ 3 & -1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1/2 & -1/2 \\ -3 & 0 & 2 \\ -2 & -1/2 & 3/2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Имея обратную матрицу, можно записать также общее решение в соответствии с выражением (1.26):

$$\mathbf{x}_B = B^{-1}\mathbf{b} - B^{-1}R\mathbf{x}_R,$$

где в нашем случае $R = \mathbf{a}^4 = \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix}$; $\mathbf{x}_R = x_4$. Находим

$$\mathbf{x}_B = \begin{bmatrix} 1 & 1/2 & -1/2 \\ -3 & 0 & 2 \\ -2 & -1/2 & 3/2 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \\ 5 \end{bmatrix} - \begin{bmatrix} 1 & 1/2 & -1/2 \\ -3 & 0 & 2 \\ -2 & -1/2 & 3/2 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix} x_4 =$$

$$= \begin{bmatrix} 1 \\ 7 \\ 3 \end{bmatrix} - \begin{bmatrix} 3 \\ -5 \\ -5 \end{bmatrix} x_4$$

или $x_1 = 1 - 3x_4$; $x_2 = 7 + 5x_4$; $x_3 = 3 + 5x_4$.

Отметим, что во многом выбор преобразований был произвольным: мы могли бы получить то же решение через другую последовательность ведущих элементов, кроме того, в качестве базисной могла быть выбрана переменная x_4, а в качестве свободной — одна из переменных x_1, x_2, x_3.

Пример 1.6. Решить систему

$$x_1 - 2x_2 + 2x_3 = 3;\ 2x_1 + x_2 - x_3 = 1;$$

$$3x_1 - 4x_2 + 2x_3 = 1;\ x_1 + 2x_2 - x_3 = 2.$$

Таблица 1.4

$\mathbf{a}^1$	$\mathbf{a}^2$	$\mathbf{a}^3$	$\mathbf{b}$
$\boxed{1}$	−2	2	3
2	1	−1	1
3	−4	2	1
1	2	−1	2
1	−2	2	3
0	$\boxed{5}$	−5	−5
0	2	−4	−8
0	4	−3	−1

Продолжение табл. 1.4

a^1	a^2	a^3	b
1	0	0	1
0	1	−1	−1
0	0	**−2**	−6
0	0	1	3
1	0	0	1
0	1	0	2
0	0	1	3
0	0	0	0

После трех жордановых преобразований с ведущими элементами 1, 5, −2 (табл. 1.4) устанавливаем, что решением системы являются значения переменных $x_1=1$; $x_2=2$; $x_3=3$. В четвертом уравнении левая и правая части обратились в нуль, следовательно, оно является линейной комбинацией остальных уравнений. Действительно, нетрудно заметить, что это уравнение получается суммированием первых трех уравнений, взятых с коэффициентами соответственно 1/2, 1, −1/2.

Пример 1.7. Решить систему

$$2x_1-x_2+3x_3=3;\ -x_1+3x_2-x_3=1;\ x_1+2x_2+2x_3=2.$$

Таблица 1.5

a^1	a^2	a^3	b
2	−1	3	3
−1	3	−1	1
1	2	2	2
0	**5**	1	5
1	−3	1	−1
0	5	1	3
0	1	1/5	1
1	0	8/5	2
0	0	0	−2

После двух жордановых преобразований, выполненных в табл. 1.5, третье уравнение приняло вид $0=-2$, что свидетельствует о несовместности системы. Действительно, левая часть этого уравнения является линейной комбинацией первых двух уравнений, взятых с коэффициентами 1, а правая часть — этих же уравнений, взятых с коэффициентами 1/2.

Алгебра симплекс-метода. Основу вычислительной схемы симплекс-метода составляют преобразования базисных систем линейных уравнений. Каждое преобразование связано с переводом одной базисной переменной в список небазисных, а одной небазисной — в список базисных. Замена переменных организуется так, что целевая функция с каждым преобразованием возрастает, если только получаемые допустимые базисные решения (опорные планы) невырождены.

Рассмотрим задачу (1.23) в предположении, что система $A\mathbf{x}=\mathbf{b}$ приведена к эквивалентной базисной форме (1.28). Пусть для простоты записи базисными являются первые m переменных. Тогда общее и базисное решения системы соответственно имеют вид:

$$x_i = b_i - \sum_{j=m+1}^{n} a_{ij}x_j, \quad i = 1, \ldots, m; \tag{1.31}$$

$$x_i = \begin{cases} b_i, & \text{если } i = 1, \ldots, m; \\ 0, & \text{если } i = m+1, \ldots, n. \end{cases} \tag{1.32}$$

Целевую функцию $z=\mathbf{cx}$ преобразуем так, чтобы она зависела только от свободных переменных. Подставляя в нее выражения базисных переменных x_i из равенства (1.31), получаем

$$z = \sum_{i=1}^{m} c_i \left(b_i - \sum_{j=m+1}^{n} a_{ij}x_j \right) + \sum_{j=m+1}^{n} c_j x_j.$$

Введя обозначения

$$z_0 = \sum_{i=1}^{m} c_i b_i; \tag{1.33}$$

$$z_j = \sum_{i=1}^{m} c_i a_{ij}, \quad j = 1, \ldots, n, \tag{1.34}$$

приведем целевую функцию к виду

$$z = z_0 - \sum_{j=m+1}^{n} (z_j - c_j)\, x_j$$

или

$$z = z_0 - \sum_{j=m+1}^{n} \Delta_j x_j, \tag{1.35}$$

где

$$\Delta_j = z_j - c_j, \quad j = 1, \ldots, n. \tag{1.36}$$

Равенство (1.35) называется *приведенным выражением целевой функции*, а коэффициенты Δ_j — *приведенными коэффициентами целевой функции* или *оценками* соответствующих переменных x_j.

Замечание 1.2. Поскольку каждая базисная переменная входит в одну строку системы с коэффициентом 1, а в остальные — с коэффициентами 0, то из (1.34) следует, что для базисных переменных имеет место равенство $z_j=c_j$, откуда получаем, что оценки базисных переменных равны нулю: $\Delta_j=z_j-c_j=0$, $j=1, \ldots, m$.

Проанализируем связь между величиной целевой функции и свободными переменными. Возможны три случая.

Случай 1. Все коэффициенты Δ_j при свободных переменных x_j функции (1.35) неотрицательны, $\Delta_j \geqslant 0$, $j=m+1, \ldots, n$. Тогда любое увеличение свободных переменных $x_{m+1}, x_{m+2}, \ldots, x_n$, которые в базисном решении равны нулю, привело бы к уменьшению функции (1.35). Следовательно, допустимое базисное решение (1.32) дает максимум целевой функции $\max z=z_0$ и является оптимальным. Таким образом, условие

$$\Delta_j=z_j-c_j \geqslant 0, \quad j=1, \ldots, n,$$

является *условием оптимальности плана* задачи.

Случай 2. Среди оценок Δ_j свободных переменных имеется хотя бы одна отрицательная, и при этом среди коэффициентов соответствующего ей столбца базисной системы нет положительных. В таком случае функция z не ограничена в области допустимых решений.

Действительно, пусть $\Delta_s<0$ и все $a_{is} \leqslant 0$, $i=1, \ldots, m$. Тогда, полагая в выражениях (1.31) и (1.35) все свободные переменные, кроме x_s, равными нулю, получаем

$$x_i=b_i-a_{is}x_s, \quad i=1, \ldots, m; \tag{1.37}$$

$$z=z_0-\Delta_s x_s. \tag{1.38}$$

Так как $\Delta_s<0$, то при увеличении x_s функция (1.38) монотонно возрастает. Вместе с тем из выражения (1.37) следует, что при $a_{is} \leqslant 0$ все переменные x_i, зависящие от x_s, остаются неотрицательными при любых сколь угодно больших значениях x_s. Следовательно, функция z максимума не имеет, оптимального плана у задачи не существует, хотя допустимых планов бесконечно много.

Случай 3. Имеются свободная переменная x_s с оценкой $\Delta_s<0$ и хотя бы один коэффициент $a_{is}>0$. Тогда

можно найти новый опорный план, для которого значения целевой функции больше, если рассматриваемый опорный план невырожден.

Действительно, как видно из (1.38), при $\Delta_s<0$ увеличение x_s приводит к возрастанию целевой функции. Увеличение x_s допустимо до тех пор, пока правые части выражения (1.37) остаются неотрицательными при всех i. Максимально возможное значение $x_s=\theta_s$, при котором обеспечивается неотрицательность переменных в (1.37), определяется из условия

$$\theta_s = \min_{a_{is}>0} \frac{b_i}{a_{is}} = \frac{b_r}{a_{rs}}. \tag{1.39}$$

Для невырожденного опорного плана все $b_i>0$, следовательно, $\theta_s>0$. Присваивая в соответствии с (1.39) переменной x_s значение

$$x_s=\theta_s=b_r/a_{rs}, \tag{1.40}$$

мы переводим свободную переменную x_s в число базисных вместо базисной переменной x_r. Для определения значений остальных базисных переменных x_i, $i=1, \ldots, m$, $i\neq s$, выполним по формулам (1.30) шаг жорданова исключения с ведущим элементом a_{rs}. В итоге базисное решение преобразованной системы примет вид

$$\left.\begin{aligned} &x_i = b_i - a_{is}b_r/a_{rs},\ i=1, \ldots, m,\ i\neq r;\\ &x_r = 0;\\ &x_s = b_r/a_{rs};\\ &x_j = 0,\ j=m+1, \ldots, n,\ j\neq s. \end{aligned}\right\} \tag{1.41}$$

Замечание 1.3. Если минимум отношений b_i/a_{is} в (1.39) достигается в нескольких строках $i=r, t, \ldots$, то нулевые значения примут несколько соответствующих переменных $x_r, x_t, \ldots$ Новая базисная переменная x_s вводится вместо любой из них, а остальные остаются в базисе. Следовательно, получаемый в этом случае план является вырожденным.

Рассмотрим, как изменяется целевая функция при переходе к новому опорному плану. В соответствии с равенствами (1.38), (1.40) ее новое значение определяется как

$$z'=z_0-(\Delta_s/a_{rs})b_r. \tag{1.42}$$

Поскольку $\Delta_s<0$, $b_r/a_{rs}>0$, то целевая функция возрастает на величину $\Delta_s\theta_s=\Delta_s b_r/a_{rs}$. Нетрудно заметить, что выражение (1.42) повторяет формулу (1.30) жордановых исключений для расчета правых частей системы (1.28)

с заменой коэффициента a_{is} строки i матрицы A на коэффициент Δ_s приведенной целевой функции. Действительно, выражение (1.35), записанное в виде

$$z + \sum_{j=m+1}^{n} \Delta_j x_j = z_0, \tag{1.43}$$

формально не отличается от уравнений системы (1.28). Поэтому его можно рассматривать как $(m+1)$-е уравнение, разрешенное относительно базисной переменной z, и выполнять над элементами Δ_j, z_0 те же преобразования, что и над элементами a_{ij}, b_i. Формула для расчета преобразованных элементов Δ'_j примет вид

$$\Delta'_j = \Delta_j - (\Delta_s/a_{rs})a_{rj}, \; j = 1, \ldots, n. \tag{1.44}$$

Итак, выполнив жорданово преобразование системы (1.28), расширенной добавлением строки приведенной целевой функции (1.43), получим новое базисное допустимое решение системы (1.28), связанное с большим значением целевой функции.

Чтобы решить, является ли новое базисное решение оптимальным, надо провести анализ оценок Δ_j. При этом возникает опять один из трех взаимно исключающих случаев. Если имеет место случай 3, процесс жордановых преобразований продолжается. После конечного числа преобразований при невырожденном начальном опорном плане и невырожденных опорных планах на каждой итерации возможен случай 1 или 2: либо будет найдено оптимальное решение, либо будет установлена неограниченность целевой функции. Конечность симплекс-метода является следствием конечности числа различных опорных планов: их верхняя граница равна C_n^m.

Если в процессе работы алгоритма появляется вырожденный опорный план, то на этой итерации переход к следующему опорному плану происходит без увеличения целевой функции. Теоретически возможен случай, когда последовательность вырожденных опорных планов начинает повторяться — зацикливается. Для предотвращения зацикливания разработаны специальные приемы. Однако мы их не рассматриваем, так как пока что нет сведений о том, чтобы зацикливание встретилось где-либо в решении практических задач.

Вычислительная схема симплекс-метода для задачи в базисной форме. *Задачей в базисной форме* называется каноническая задача линейного программирования

$\max\{z=\mathbf{cx}\,|\,A\mathbf{x}=\mathbf{b},\ \mathbf{x}\geqslant 0\}$ *cm* уравнениями, *n* неизвестными и неотрицательным вектором $\mathbf{b}$, разрешенная относительно *m* переменных, т. е. имеющая единичный базис. Условия такой задачи удобно задать в виде так называемой *симплекс-таблицы* (табл. 1.6).

Таблица 1.6

$\mathbf{c}_B$	$\mathbf{x}_B$	$\mathbf{b}$	c_1	c_2	...	c_n
			$\mathbf{a}^1$	$\mathbf{a}^2$	...	$\mathbf{a}^n$
c_1	x_1	b_1	a_{11}	a_{12}	...	a_{1n}
c_2	x_2	b_2	a_{21}	a_{22}	...	a_{2n}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
c_m	x_m	b_m	a_{m_1}	a_{m_2}	...	a_{mn}
	Δ	z_0	Δ_1	Δ_2	...	Δ_n

Над обозначениями векторов $\mathbf{a}^1$, $\mathbf{a}^2$, ..., $\mathbf{a}^n$ проставлены коэффициенты целевой функции задачи. В процессе решения в столбце $\mathbf{x}_B$ находится список базисных переменных, в столбце $\mathbf{c}_B$ — соответствующие им коэффициенты целевой функции, в столбце $\mathbf{b}$ — значения базисных переменных, в столбцах $\mathbf{a}^1$, $\mathbf{a}^2$, ..., $\mathbf{a}^n$ — коэффициенты a_{ij} матрицы условий, в оценочной строке Δ — значение целевой функции z_0 и приведенные коэффициенты Δ_j при неизвестных линейной формы.

При заполнении элементов Δ-строки исходной таблицы используются формулы (1.33), (1.34), (1.36), при помощи которых целевая функция приводится к базису, в котором представлена система ограничений.

Замечание 1.4. Если целевая функция выражена только через свободные переменные, т. е. уже приведена к единичному базису, и, кроме того, ее коэффициенты при базисных переменных равны нулю, то элементы Δ-строки определяются особенно просто. Из формул (1.33), (1.34), (1.36) видно, что они принимают значения $z_0=0$, $\Delta_j=0$ для базисных переменных и $\Delta_j=-c_j$ для свободных переменных.

Опишем алгоритм симплекс-метода.

1. Просматриваем знаки всех коэффициентов Δ_j оценочной строки. Если все $\Delta_j\geqslant 0$, то задача решена: допустимое базисное решение (1.32) оптимально, $\max z=z_0$. Если не все $\Delta_j\geqslant 0$, переходим к шагу 2.

2. Среди значений $\Delta_j<0$ находим наибольшее по абсолютной величине, и соответствующий ему столбец выбираем в качестве ведущего. Пусть это будет столбец с номером s. Если в ведущем столбце элементы $a_{is}\leqslant 0$ для всех $i=1, \ldots, m$, имеем случай 2: целевая функция не ограничена, $\max z=\infty$. Решение окончено. Если не все $a_{is}\leqslant 0$, то переходим к шагу 3.

3. Для каждого элемента $a_{is}>0$ ведущего столбца находим отношение b_i/a_{is}, выбираем наименьшее и называем строку, где оно достигается, ведущей. Пусть это будет строка с номером r. Элемент a_{rs} на пересечении ведущего столбца и ведущей строки будет ведущим.

4. Выполняем жорданово преобразование таблицы с ведущим элементом a_{rs} по формулам (1.29), (1.30), (1.42), (1.44) и переходим к шагу 1.

Последовательность операций 1—4 называется *итерацией симплекс-метода.*

Замечание 1.5. Для преобразования Δ-строки можно использовать вместо формул (1.42) и (1.44) формулы (1.33), (1.34), (1.36), что полезно при организации контроля вычислений.

Пример 1.8. Решить симплекс-методом задачу (1.7), (1.8).

Введением дополнительных неотрицательных переменных x_3, x_4, x_5 приведем задачу к каноническому виду

$$\max z=3x_1+2x_2;\ x_1+3x_2+x_3=270;\ 4x_1+6x_2+x_4=600;$$
$$3x_1+x_2+x_5=240;\ x_1\geqslant 0,\ x_2\geqslant 0,\ x_3\geqslant 0,\ x_4\geqslant 0,\ x_5\geqslant 0.$$

Получили задачу в базисной форме, где переменные x_3, x_4, x_5 базисные, x_1, x_2 свободные. В соответствии с постановкой задачи (1.7), (1.8) дополнительные переменные x_3, x_4, x_5 по своему экономическому смыслу представляют предполагаемый избыточный объем ресурсов соответственно первого, второго и третьего видов. Занесем коэффициенты целевой функции и матрицы условий задачи в табл. 1.7. Воспользовавшись замечанием 1.4 или применив формулы (1.33), (1.34), (1.36), заполним Δ-строку. Поскольку базисные переменные x_3, x_4, x_5 входят в целевую функцию с коэффициентами 0, естественно, что начальное базисное решение $x_3=270$, $x_4=600$, $x_5=240$ связано с нулевым значением целевой функции. Отрицательные оценки $\Delta_1=-3$, $\Delta_2=-2$ свободных переменных в оценочной строке указывают, что данное базисное решение может быть улучшено.

Выполняя шаг 2 алгоритма, устанавливаем, что наибольшая по абсолютной величине отрицательная оценка $\Delta_1=-3$ находится в первом столбце таблицы, который становится ведущим. Для увеличения целевой функции надо ввести в базис свободную переменную x_1 и вывести из базиса одну из базисных переменных x_3, x_4 или x_5.

На шаге 3 находим отношения 270/1, 600/4, 240/3 элементов вектора b к положительным элементам ведущего столбца и по наименьшему из этих отношений выбираем ведущий элемент $a_{31}=3$, взятый в рамку.

Таблица 1.7

c_B	x_B	b	3	2	0	0	0
			a^1	a^2	a^3	a^4	a^5
0	x_3	270	1	3	1	0	0
0	x_4	600	4	6	0	1	0
0	x_5	240	[3]	1	0	0	1
	Δ	0	−3	−2	0	0	0
0	x_3	190	0	8/3	1	0	−1/3
0	x_4	280	0	[14/3]	0	1	−4/3
3	x_1	80	1	1/3	0	0	1/3
	Δ	240	0	−1	0	0	1
0	x_3	30	0	0	1	−4/7	3/7
2	x_2	60	0	1	0	3/14	2/7
3	x_1	60	1	0	0	−1/14	3/7
	Δ	300	0	0	0	3/14	5/7

В соответствии с шагом 4 по формулам (1.29), (1.30), (1.42), (1.44) выполняем жорданово преобразование расширенной матрицы, включающей оценочную Δ-строку. Этим заканчивается первая итерация симплекс-метода, в результате чего из базиса выведена переменная x_5 и введена переменная x_1.

Новое базисное решение $x_3=190$, $x_4=280$, $x_1=80$, связанное со значением целевой функции $z=240$, и преобразованные элементы матрицы условий приведены в средней части табл. 1.7. Здесь имеется единственная отрицательная оценка $\Delta_2=-1$. Выполняя вторую итерацию с ведущим элементом $a_{22}=14/3$, получаем таблицу, в которой все коэффициенты оценочной строки неотрицательны. Оптимальное решение $(x_1, x_2, x_3)=(60, 60, 30)$, $z=300$ показывает, что при одинаковых интенсивностях использования первого и второго технологических способов, равных 60 единицам времени, будет получен доход в 300 единиц. При этом первый ресурс имеется в избытке $x_3=30$ единиц, второй и третий используются полностью: $x_4=x_5=0$.

Пример 1.9. Решить задачу

$$\max z=4x_1+x_2+x_3-3x_4;\ 2x_1-x_2+x_3=6;$$

$$x_1-x_2+x_4=8;\ x_1+x_2+x_5=18;$$

$$x_j\geqslant 0,\ j=1, \ldots, 5.$$

Данная задача записана в базисной форме, но в целевую функцию входят как свободные переменные x_1, x_2, так и базисные x_3, x_4. Следовательно, при построении симплекс-таблицы надо выражение для целевой функции привести к тому же единичному базису, что и у матрицы условий, т. е. исключить базисные переменные и выразить целевую функцию через свободные переменные x_1, x_2. Сделаем это в табл. 1.8 с помощью формул (1.33), (1.34), (1.36).

Таблица 1.8

c_B	x_B	b	4	1	1	−3	0
			a^1	a^2	a^3	a^4	a^5
1	x_3	6	[2]	−1	1	0	0
−3	x_4	8	1	−1	0	1	0
0	x_5	18	1	1	0	0	1
	Δ	−18	−5	1	0	0	0
4	x_1	3	1	−1/2	1/2	0	0
−3	x_4	5	0	−1/2	−1/2	1	0
0	x_5	15	0	[3/2]	−1/2	0	1
	Δ	−3	0	−3/2	5/2	0	0
4	x_1	8	1	0	1/3	0	1/3
−3	x_4	10	0	0	−2/3	1	1/3
1	x_2	10	0	1	−1/3	0	2/3
	Δ	12	0	0	2	0	1

Оценочная строка получена после заполнения верхней части табл. 1.8 в результате следующих вычислений:

$$z_0 = \sum_{i=1}^{3} c_i b_i = 1 \cdot 6 - 3 \cdot 8 + 0 \cdot 18 = -18;$$

$$z_1 = \sum_{i=1}^{3} c_i a_{i1} = 1 \cdot 2 - 3 \cdot 1 + 0 \cdot 1 = -1,\ \Delta_1 = z_1 - c_1 = -1 - 4 = -5;$$

$$z_2 = \sum_{i=1}^{3} c_i a_{i2} = 1(-1) - 3(-1) + 0 \cdot 1 = 2,\ \Delta_2 = z_2 - c_2 = 2 - 1 = 1;$$

$$z_3 = \sum_{i=1}^{3} c_i a_{i3} = 1 \cdot 1 - 3 \cdot 0 + 0 \cdot 0 = 1,\ \Delta_3 = z_3 - c_3 = 1 - 1 = 0 \text{ и т. д.}$$

После двух элементарных преобразований с ведущими элементами 2 и 3/2, взятыми в рамку, получаем план $x = (x_1, x_2, x_3, x_4, x_5) = (8, 10, 0, 10, 0)$, $z = 12$, при котором все коэффициенты в оценочной строке неотрицательны, $\Delta j \geqslant 0$, $j = 1, \ldots, 5$. Следовательно, этот план оптимален.

1.4. ПОСТРОЕНИЕ НАЧАЛЬНОГО ОПОРНОГО ПЛАНА

Частный случай. Вычислительная схема симплекс-метода применима лишь к задачам в базисной форме, т. е. к задачам, для которых известен начальный опорный

план и соответствующий ему базис. До сих пор мы считали начальный опорный план известным, не рассматривая вопрос о том, каким образом он получен. Хотя в общем случае нахождение начального опорного плана составляет самостоятельную задачу, иногда его построение не вызывает затруднений. Например, если матрица A содержит единичную подматрицу порядка m и вектор ограничений $\mathbf{b} \geqslant \mathbf{0}$, то столбцы этой подматрицы составляют допустимый базис задачи. Этот случай, в частности, имеет место при приведении стандартной задачи к канонической. Напомним, что переход от системы неравенств стандартной задачи

$$\max z = \sum_{j=1}^{n} c_j x_j; \quad \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i, \quad i = 1, \ldots, m; \quad x_j \geqslant 0,$$
$$j = 1, \ldots, n$$

к эквивалентной системе уравнений канонической задачи

$$\max z = \sum_{j=1}^{n} c_j x_j; \quad \sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i, \quad i = 1, \ldots m; \quad x_j \geqslant 0,$$
$$j = 1, \ldots, n + m$$

осуществляется введением дополнительных неотрицательных переменных $x_{n+i} \geqslant 0$, $i = 1, \ldots, m$, которые входят в целевую функцию с нулевыми коэффициентами. Очевидно, в качестве начального опорного плана такой задачи можно рассматривать план

$$x_j = 0, \quad \text{если } j = 1, \ldots, n;$$
$$x_{n+i} = b_i, \quad \text{если } i = 1, \ldots, m.$$

Этот план является допустимым: легко видеть, что он удовлетворяет всем ограничениям задачи. Далее, векторы условий, соответствующие положительным компонентам плана, линейно независимы, так как они составляют полный набор единичных векторов — линейно независимую систему. Следовательно, план $\mathbf{x} = (0, 0, \ldots, 0, b_1, b_2, \ldots, b_m)$, где нулевыми являются первые n координат, опорный.

В общем случае приходится использовать специальные приемы приведения канонической задачи к базисной форме. С вычислительной точки зрения важно при этом, что приведение канонической задачи к задаче в базисной форме осуществляется включением только новых пере-

менных и не требует добавления новых ограничений. Установлено, что оценка количества итераций симплекс-метода при решении задач с m ограничениями находится в пределах между m и $3m$ и мало зависит от числа переменных n. Поскольку число ограничений задачи, приведенной к базисной форме, не меняется, объем последующей вычислительной работы не увеличивается.

Метод минимизации невязок. Для отыскания опорного плана канонической задачи

$$\max z = \sum_{j=1}^{n} c_j x_j; \tag{1.45}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \ i = 1, \ ..., \ m; \tag{1.46}$$

$$x_j \geqslant 0, \quad j = 1, ..., n \tag{1.47}$$

в общем случае может быть использовано решение следующей вспомогательной задачи с $n+m$ переменными:

$$\max w = -\sum_{i=1}^{m} x_{n+i}; \tag{1.48}$$

$$\sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i, \ i = 1, \ ..., \ m; \tag{1.49}$$

$$x_j \geqslant 0, \quad j = 1, ..., n+m. \tag{1.50}$$

Она получена путем введения m искусственных переменных или невязок $x_{n+i} \geqslant 0$, $i = 1, ..., m$, с требованием минимизации их суммы. Для решения этой вспомогательной задачи уже можно применять симплекс-метод, так как имеется исходный опорный план:

$$x_j = 0, \text{ если } j = 1, ..., n;$$
$$x_{n+i} = b_i, \text{ если } i = 1, ..., m.$$

Поскольку целевая функция вспомогательной задачи ограничена сверху числом 0, то случай 2 симплекс-метода, связанный с неограниченностью целевой функции, здесь исключается. Симплекс-метод через конечное число итераций даст оптимальный план этой задачи, причем для значения целевой функции возможны два исхода.

1. Оптимальное значение целевой функции задачи (1.48) — (1.50) равно нулю. Если max $w=0$, то исходная задача (1.45) — (1.47) допустима. Действительно, если max $w=0$, то в оптимальном плане $\mathbf{x}=(x_1, x_2, ..., x_n, x_{n+1}, ..., x_{n+m})$ все $x_{n+i}=0$, $i=1, ..., m$. Поскольку вектор $\mathbf{x}=(x_1, x_2, ..., x_n, 0, ...,0)$ удовлетворяет условиям (1.49), (1.50) вспомогательной задачи, то вектор $\mathbf{x}=(x_1, x_2, ..., x_n)$ удовлетворяет условиям (1.46), (1.47) исходной задачи. Следовательно, вектор $\mathbf{x}$ — план исходной задачи, задача допустима.

2. Оптимальное значение целевой функции отрицательно. Если max $w<0$, то исходная задача (1.45) — (1.47) недопустима. Действительно, если бы система (1.46) исходной задачи имела хотя бы одно допустимое решение, то такое решение должно было бы совпадать с допустимым решением системы (1.49) вспомогательной задачи при дополнительных условиях $x_{n+1}=0$, $x_{n+2}=0$, ..., $x_{n+m}=0$, влекущих за собой равенство $w=0$. Но это противоречит предположению, что max $w<0$.

Таким образом, решение вспомогательной задачи (1.48) — (1.50) или приводит к некоторому опорному плану задачи (1.45) — (1.47), или устанавливает недопустимость исходной задачи (1.45) — (1.47). Отметим, что симплекс-таблица, получаемая на последней итерации решения вспомогательной задачи, может быть использована при решении исходной задачи. Для этого надо отбросить столбцы, соответствующие искусственным переменным x_{n+1}, x_{n+2}, ..., x_{n+m}, и заменить нулевые элементы оценочной строки элементами, которые вычисляют по формулам (1.34), (1.36).

Таким образом, при методе минимизации невязок процесс решения задачи разбивается на два этапа. На первом этапе из вспомогательной задачи определяется начальный опорный план, который на втором этапе используется для отыскания оптимального плана.

Метод искусственного базиса. В данном методе оба этапа объединены в один. Вместо исходной канонической задачи (1.45) — (1.47) будем рассматривать расширенную задачу с $n+m$ переменными, полученную введением в систему ограничений (1.46) *искусственных переменных* $x_{n+i}\geqslant 0$, $i=1, ..., m$. Эти переменные включаются в целевую функцию с одинаковыми отрицательными коэффициентами $-M$, где M — сколь угодно большое положительное число:

$$\max z = \sum_{j=1}^{n} c_j x_j - M \sum_{i=1}^{m} x_{n+i}; \tag{1.51}$$

$$\sum_{j=1}^{n} a_{ij} x_j + x_{n+i} = b_i, \quad i = 1, \ldots, m; \tag{1.52}$$

$$x_j \geqslant 0, \quad j = 1, \ldots, n+m. \tag{1.53}$$

Расширенная задача (1.51) — (1.53) иногда называется *M-задачей*, а метод искусственного базиса — *M-методом*. Единичные векторы-столбцы $\mathbf{a}^{n+1}$, $\mathbf{a}^{n+2}$, ..., $\mathbf{a}^{n+m}$ матрицы условий *M*-задачи, соответствующие искусственным переменным, составляют *искусственный единичный базис*. Ему соответствуют очевидный начальный опорный план $\mathbf{x} = (0, 0, \ldots, 0, b_1, b_2, \ldots, b_m)$, где нулевыми являются первые n координат, и значение целевой функции $z = -M \sum_{i=1}^{m} x_{n+i}$. Введение в целевую функцию определенных выше коэффициентов $-M$ при искусственных переменных эквивалентно введению штрафа за включение в опорный план переменных x_{n+i}, $i = 1, \ldots, m$. Числа $-M$, по абсолютной величине значительно превосходящие остальные коэффициенты целевой функции, позволяют выводить из базиса искусственные переменные и вводить в базис переменные исходной задачи.

Имея начальный опорный план, можно применить симплекс-метод для отыскания оптимального плана расширенной задачи (1.51) — (1.53).

Теорема 1.5. *Если в оптимальном плане* $\tilde{\mathbf{x}} = (x_1, x_2, \ldots, x_n, 0, \ldots, 0)$ *M-задачи искусственные переменные* $x_{n+i} = 0$, $i = 1, \ldots, m$, *то план* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ *является оптимальным планом исходной задачи.*

◆ Отметим, что планы $\tilde{\mathbf{x}}$ и $\mathbf{x}$ отличаются m последними координатами, равными нулю. Следовательно, если вектор $\tilde{\mathbf{x}}$ удовлетворяет ограничениям (1.52), (1.53) расширенной задачи, то вектор $\mathbf{x}$ удовлетворяет ограничениям (1.46), (1.47) исходной задачи, т. е. является ее планом. При этом из совпадения отличных от нуля координат планов $\tilde{\mathbf{x}}$ и $\mathbf{x}$ следует равенство значений целевых функций расширенной и исходной задач: $z(\tilde{\mathbf{x}}) = z(\mathbf{x})$.

Докажем, что $\mathbf{x}$ — оптимальный план исходной задачи. Предположим противное. Тогда существует план $\mathbf{x}^* = (x_1^*,$

$x_2^*, \ldots, x_n^*)$ исходной задачи, такой, что $z(\mathbf{x}^*) > z(\mathbf{x})$, и план расширенной задачи $\widetilde{\mathbf{x}}^* = (x_1^*, x_2^*, \ldots, x_n^*, 0, \ldots, 0)$, такой, что $z(\widetilde{\mathbf{x}}^*) = z(\mathbf{x}^*) > z(\mathbf{x}) = z(\widetilde{\mathbf{x}})$, т. е. $z(\widetilde{\mathbf{x}}^*) > z(\widetilde{\mathbf{x}})$. Но это противоречит тому, что $\widetilde{\mathbf{x}}$ является оптимальным планом M-задачи. ◆

Таким образом, если в результате применения симплекс-метода к расширенной задаче получен оптимальный план, в котором все искусственные переменные $x_{n+i}=0$, то его первые n координат дают оптимальный план исходной задачи. Можно также доказать следующее: 1) если в оптимальном плане хотя бы одна из искусственных переменных положительна, то исходная задача не имеет допустимых планов — ее условия несовместны; 2) если M-задача не имеет решения, то и исходная задача неразрешима. Итак, при решении M-задачи симплекс-методом или получается оптимальное решение исходной задачи, или устанавливается ее неразрешимость.

В заключение отметим, что при применении M-метода к исходной задаче может встретиться один из следующих трех случаев.

1. Матрица A содержит единичную подматрицу порядка m, т. е. m единичных векторов условий. В этом случае задача имеет начальный опорный план и введение векторов искусственного базиса не требуется.

2. Матрица A содержит $k<m$ различных единичных векторов. В этом случае при составлении M-задачи вводится $m-k$ искусственных переменных.

3. Матрица условий задачи не содержит единичных векторов. В этом случае в каждое уравнение вводится искусственная переменная.

Пример 1.10. Решить M-методом задачу

$$\max\ z=8x_1+2x_2-2x_3-5x_4-6x_5;$$

$$x_1+\ x_2-\ x_3-\ x_4\ \ \ =16;$$

$$4x_1-2x_2+2x_3-\ \ \ x_5=20;$$

$$x_1\geqslant 0,\ x_2\geqslant 0,\ x_3\geqslant 0,\ x_4\geqslant 0,\ x_5\geqslant 0.$$

Система ограничений задачи не содержит базисных переменных, поэтому введем в уравнения искусственные переменные x_6, x_7, включив их в целевую функцию с коэффициентом $-M$, значение которого можно заранее не фиксировать. *Получим задачу*

$$\max\ z=8x_1+2x_2-2x_3-5x_4-6x_5-Mx_6-Mx_7;$$

$$x_1+x_2-x_3-x_4+x_6=16;$$

$$4x_1-2x_2+2x_3-x_5+x_7=20;$$

$$x_j\geqslant 0,\ j=1,\ \ldots,\ 7,$$

решение которой приведено в табл. 1.9. Отметим, что после того как искусственная переменная была выведена из базисных переменных, в силу выбора коэффициента $-M$ она уже не может больше попасть в число базисных переменных. Поэтому если не требуется получение обратной матрицы, то в последующих симплекс-таблицах столбец, соответствующий этой переменной, можно исключить.

Таблица 1.9

c_B	x_B	b	8	2	−2	−5	−6	$-M$	$-M$
			a^1	a^2	a^3	a^4	a^5	a^6	a^7
$-M$	x_6	16	1	1	−1	−1	0	1	0
$-M$	x_7	20	[4]	−2	2	0	−1	0	1
	Δ	$-36M$	$-5M$	M	$-M$	M	M	0	0
			−8	−2	+2	+5	+6		
$-M$	x_6	11	0	[3/2]	−3/2	−1	1/4	1	
8	x_1	5	1	−1/2	1/2	0	−1/4	0	
	Δ	$-11M$	0	$-3/2M$	$3/2M$		$-1/4M$	0	
		+40		−6	+6	+5	+4		
2	x_2	22/3	0	1	−1	−2/3	1/6		
8	x_1	26/3	1	0	0	−1/3	−1/6		
	Δ	84	0	0	0	1	5		

Оптимальный план M-задачи следующий: $x_1=26/3$, $x_2=22/3$, $x_3=x_4=x_5=x_6=x_7=0$, при этом искусственные переменные x_6, x_7 равны нулю. Следовательно, план $x_1=26/3$, $x_2=22/3$, $x_3=x_4=x_5=0$ является оптимальным решением исходной канонической задачи.

1.5. ДВОЙСТВЕННОСТЬ В ЛИНЕЙНОМ ПРОГРАММИРОВАНИИ

Двойственные задачи. В теории линейного программирования установлено, что каждой задаче линейного программирования может быть поставлена в соответствие другая вполне определенная задача линейного программирования, такая, что при решении одной из них одновременно решается и другая. Эти задачи были на-

званы *парой взаимодвойственных* (или *взаимно сопряженных*) *задач*. Результаты исследований, связанных с двойственностью, составили значительную часть теории линейного программирования. На них базируются некоторые эффективные методы решения задач линейного программирования.

Рассмотрим стандартную задачу линейного программирования:

$$\left.\begin{array}{l} \max z = c_1x_1 + c_2x_2 + \ldots + c_nx_n; \\ a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leqslant b_1; \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leqslant b_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leqslant b_m; \\ x_1 \geqslant 0,\ x_2 \geqslant 0,\ \ldots,\ x_n \geqslant 0 \end{array}\right\} \quad (1.54)$$

или в матричной записи

$$\max \mathbf{cx};\ A\mathbf{x} \leqslant \mathbf{b};\ \mathbf{x} \geqslant 0. \quad (1.55)$$

Введя m-мерный вектор-строку $\mathbf{y}$, построим стандартную задачу вида

$$\left.\begin{array}{l} \min w = b_1y_1 + b_2y_2 + \ldots + b_my_m; \\ a_{11}y_1 + a_{21}y_2 + \ldots + a_{m1}y_m \geqslant c_1; \\ a_{12}y_1 + a_{22}y_2 + \ldots + a_{m2}y_m \geqslant c_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{1n}y_1 + a_{2n}y_2 + \ldots + a_m\ y_m \geqslant c_n; \\ y_1 \geqslant 0,\ y_2 \geqslant 0,\ \ldots,\ y_m \geqslant 0 \end{array}\right\} \quad (1.56)$$

или в матричной записи

$$\min \mathbf{yb};\ \mathbf{y}A \geqslant \mathbf{c};\ \mathbf{y} \geqslant 0. \quad (1.57)$$

Задача (1.56) называется *двойственной* задаче (1.54). Соответственно задача (1.54) называется *прямой* по отношению к задаче (1.56).

Из сопоставления задач (1.54) и (1.56) определяются правила, в соответствии с которыми одна стандартная задача (прямая) преобразуется в другую (двойственную). Переменных y_i в задаче (1.56) столько, сколько ограничений в матрице A задачи (1.54). Матрица условий в задаче (1.56) — транспонированная матрица условий задачи (1.54). Задача максимизации переходит в задачу минимизации, ограничения-неравенства вида $\leqslant$ заменяются ограничениями вида $\geqslant$. Вектор с коэффициентов целевой функции прямой задачи становится век-

тором ограничений двойственной задачи, а вектор $\mathbf{b}$ ограничений задачи (1.54) становится вектором коэффициентов целевой функции двойственной задачи (1.56). Обратим внимание на то, что если одна из двойственных задач стандартная, то и другая стандартная. Система ограничений в обоих случаях задана неравенствами. Поэтому задачи (1.54) и (1.56) называются *симметричными*. Связь между ограничениями симметричных задач удобно отразить с помощью следующей схемы:

$$\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i,\ i=1,\ \ldots,\ m \longleftrightarrow y_i \geqslant 0, \qquad i=1,\ \ldots,\ m;$$

$$x_j \geqslant 0,\ j=1,\ \ldots,\ n \longleftrightarrow \sum_{i=1}^{m} a_{ij}y_i \geqslant c_j,\ j=1,\ \ldots,\ n,$$

откуда видно, что каждому ограничению одной задачи сопоставлена переменная с тем же номером другой задачи; каждой переменной одной задачи сопоставлено ограничение с тем же номером другой задачи.

Рассмотрим теперь каноническую задачу

$$\max z=\mathbf{cx};\ A\mathbf{x}=\mathbf{b};\ \mathbf{x}\geqslant 0. \qquad (1.58)$$

Описанные правила построения двойственной задачи можно применить к задаче (1.58), если предварительно записать ее в виде стандартной задачи. В итоге задача, двойственная канонической, имеет вид

$$\min w=\mathbf{yb};\ \mathbf{y}A\geqslant\mathbf{c}, \qquad (1.59)$$

причем на вектор $\mathbf{y}$ не накладывается условие неотрицательности.

Можно также проверить, что задача, двойственная (1.59), совпадает с (1.58), а двойственная (1.56), совпадает с (1.54). Поэтому прямую и двойственную задачи называют также *взаимодвойственными*. Взаимодвойственные задачи (1.58) и (1.59) называются *несимметричными*, так как в прямой задаче система ограничений задана равенствами, а в двойственной — неравенствами, в прямой задаче все переменные неотрицательные, в двойственной могут быть отрицательными.

Построим пример взаимодвойственных задач. В качестве прямой рассмотрим задачу производственного планирования. Напомним ее содержание. Предприятие имеет m видов ресурсов и выпускает n видов продукции. На производство единицы j-й продукции расходуется

a_{ij} единиц i-го ресурса. Запас i-го ресурса составляет b_i единиц, $i=1, \ldots, m$, доход от реализации единицы j-й продукции равен c_j единиц, $j=1, \ldots, n$. Требуется найти план $\mathbf{x}=(x_1, x_2, \ldots, x_n)$ выпуска продукции, при котором ее суммарная стоимость максимальна. В этих обозначениях получаем математическую постановку исходной задачи в виде модели (1.54).

Экономическую интерпретацию двойственной задачи (1.56) можно дать следующим образом. Пусть заданы величины b_i, a_{ij}, c_j, $i=1, \ldots, m$, $j=1, \ldots, n$, того же смысла, что и в исходной задаче. И пусть, кроме того, введено еще одно условие: сырье можно направить или на изготовление продукции, или на продажу другому предприятию. Спрашивается, какую минимальную цену надо установить за единицу каждого вида сырья $i=1, \ldots, m$ при условии, что доход от реализации всех его запасов должен быть не меньше дохода от реализации всей продукции, которая может быть выпущена из этого сырья.

Обозначим через $y_i \geqslant 0$ искомую цену единицы сырья i-го вида. Доход, который можно было бы получить от продажи сырья, необходимого для изготовления единицы продукции вида j, равен

$$\sum_{i=1}^{m} a_{ij} y_i, \quad j = 1, \ldots, n,$$

а доход от реализации всех запасов составит величину $\sum_{i=1}^{m} b_i y_i$.

Для того чтобы продажа сырья была не менее выгодна, чем реализация изготовленной из него продукции, должно выполняться неравенство

$$\sum_{i=1}^{m} a_{ij} y_i \geqslant c_j, \quad j = 1, \ldots, n.$$

Любая система цен $y_i \geqslant 0$, установленных с учетом этого условия, удовлетворяет интересам предприятия-продавца. Естественно, что учет интересов предприятия-покупателя требует выбора такой системы цен, которая минимизирует суммарную стоимость сырья $\sum_i b_i y_i$. В итоге постановка двойственной задачи принимает вид задачи (1.56).

Пример 1.11. Для данной задачи

$$\min\ w=y_1+2y_2;\ \ 2y_1+y_2\geqslant 3;\ \ 2y_1-7y_2\leqslant 1;$$
$$2y_1+3y_2\geqslant 6;\ \ y_1\geqslant 0,\ \ y_2\geqslant 0$$

построить симметричную.

Умножив второе ограничение на -1, получим задачу с однотипными ограничениями-неравенствами:

$$\min\ w=y_1+2y_2;\ \ 2y_1+y_2\leqslant 3;$$
$$-2y_1+7y_2\geqslant -1;\ \ 2y_1+3y_2\geqslant 6;\ \ y_1\geqslant 0,\ y_2\geqslant 0.$$

Симметричная ей задача имеет вид

$$\max\ z=3x_1-x_2+6x_3;\ \ 2x_1-2x_2+2x_3\leqslant 1;$$
$$x_1+7x_2+3x_3\leqslant 2;\ \ x_1\geqslant 0,\ \ x_2\geqslant 0,\ \ x_3\geqslant 0.$$

Теоремы двойственности. Приступим к изучению связей между прямой и двойственной задачами. Для определенности будем формулировать основные утверждения применительно к симметричной паре двойственных задач (1.55) и (1.57). Определим множества

$$X=\{\mathbf{x}\,|\,A\mathbf{x}\leqslant\mathbf{b},\ \mathbf{x}\geqslant 0\};\ Y=\{\mathbf{y}\,|\,\mathbf{y}A\geqslant\mathbf{c},\ \mathbf{y}\geqslant 0\}.$$

Лемма 1.1. *Для любых допустимых планов* $\mathbf{x}\in X$, $\mathbf{y}\in Y$ *прямой и двойственной задач справедливо неравенство*

$$\mathbf{cx}\leqslant\mathbf{yb}. \tag{1.60}$$

◆ Так как $\mathbf{x}\in X$, $\mathbf{y}\in Y$, то $A\mathbf{x}\leqslant\mathbf{b}$, $\mathbf{y}\geqslant 0$. Отсюда следует, что $\mathbf{y}A\mathbf{x}\leqslant\mathbf{yb}$. Аналогично из того, что $\mathbf{y}A\geqslant\mathbf{c}$, $\mathbf{x}\geqslant 0$, следует $\mathbf{y}A\mathbf{x}\geqslant\mathbf{cx}$. Таким образом, $\mathbf{cx}\leqslant\mathbf{y}A\mathbf{x}\leqslant\mathbf{yb}$, и лемма доказана. ◆

Доказательство для несимметричной пары двойственных задач проводится аналогично. Неравенство (1.60) называют *основным неравенством* теории двойственности. Его экономический смысл в том, что при любом допустимом плане производства $\mathbf{x}$ общая стоимость всего продукта не больше суммарной оценки ресурсов, отвечающей любому допустимому вектору оценок $\mathbf{y}$.

Лемма 1.2. *Если для некоторых допустимых планов* $\mathbf{x}^*\in X$, $\mathbf{y}^*\in Y$ *пары двойственных задач выполняется равенство*

$$\mathbf{cx}^*=\mathbf{y}^*\mathbf{b}, \tag{1.61}$$

то векторы $\mathbf{x}^*$ *и* $\mathbf{y}^*$ *являются оптимальными планами соответствующих прямой и двойственной задач.*

◆ Предварительно напомним, что вектор $\mathbf{x}^*$ оптимален тогда и только тогда, когда для любого допустимого вектора $\mathbf{x}$ имеет место неравенство $\mathbf{cx} \leqslant \mathbf{cx}^*$. Пусть $\mathbf{x}$ — допустимое решение прямой задачи. Тогда в соответствии с леммой 1.1 и условием (1.61) имеем $\mathbf{cx} \leqslant \mathbf{y}^*\mathbf{b} = \mathbf{cx}^*$. Оптимальность вектора $\mathbf{y}^*$ доказывается аналогично. ◆

Лемма 1.2 формулирует критерий оптимальности для задач линейного программирования. Согласно этой лемме, если среди допустимых решений найдутся векторы $\mathbf{x}^*$ и $\mathbf{y}^*$, удовлетворяющие условию (1.61), то они будут оптимальными решениями соответствующих задач. С экономической точки зрения это означает, что планы производства и оценки ресурсов являются оптимальными, если цена всей продукции и суммарная оценка ресурсов совпадают.

Первая теорема двойственности (теорема существования). *Если прямая и двойственная задачи допустимы, то они имеют оптимальные решения, причем значения их целевых функций равны.*

Данная теорема представляет основной результат теории линейного программирования. Среди различных методов ее доказательства есть методы, использующие свойства решения задачи линейного программирования и особенности процедуры симплекс-метода. Доказательства теоремы имеются в учебных пособиях (см. список литературы).

Вторая теорема двойственности (теорема о равновесии, теорема о дополняющей нежесткости). *Допустимые решения двойственных задач* $\mathbf{x} \in X$, $\mathbf{y} \in Y$ *оптимальны тогда и только тогда, когда выполняются условия:* 1) $(\mathbf{y}A - \mathbf{c})\,\mathbf{x} = 0$; 2) $\mathbf{y}(\mathbf{b} - A\mathbf{x}) = 0$.

Условия 1 и 2 называются *условиями дополняющей нежесткости.* Приведем доказательство для случая, когда прямая задача является стандартной задачей линейного программирования.

◆ Как показано при доказательстве леммы (1.1), для любых $\mathbf{x} \in X$, $\mathbf{y} \in Y$ имеет место соотношение

$$\mathbf{cx} \leqslant \mathbf{y}A\mathbf{x} \leqslant \mathbf{yb}. \qquad (1.62)$$

Пусть условия 1 и 2 выполняются. Тогда в выражении (1.62) оба неравенства становятся равенствами, откуда следует $\mathbf{cx} = \mathbf{yb}$ и в соответствии с леммой 1.2 $\mathbf{x}$ и $\mathbf{y}$ — оптимальные решения прямой и двойственной задач.

Пусть какое-либо условие не выполняется, наприме
условие 2. В силу неотрицательности выражения в скоб
ках и вектора y оно будет иметь вид $\mathbf{y}(\mathbf{b}-A\mathbf{x})>0$, отку
да получаем $\mathbf{yb}>\mathbf{y}A\mathbf{x}$ и, следовательно, $\mathbf{cx}<\mathbf{yb}$. Отсюд
по крайней мере одно из решений x, y не является опти
мальным для соответствующей ему задачи. Аналогичн
рассматривается невыполнение условия 1. ◆

Таким образом, оптимальные решения пары взаимо
двойственных задач обладают свойством ортогонально
сти: если j-я компонента оптимального вектора x поло
жительна, то j-е ограничение двойственной задачи долж
но выполняться как равенство. Аналогично, есл
положительна j-я компонента оптимального вектора y, т
должно выполняться как равенство j-е ограничение пря
мой задачи. Дадим экономическую интерпретацию усло
виям дополняющей нежесткости.

Согласно условию 1, если некоторый продукт j вхо
дит в оптимальный план производства, $x_j>0$, то пр
оптимальной системе цен двойственной задачи затрат
ресурсов на его изготовление совпадают со стоимость
этого продукта.

По условию 2, если в оптимальной системе цен ка
кой-то ресурс j получает отличную от нуля цену $y_j>0$
то в соответствии с оптимальным планом производств
исходной задачи этот ресурс будет израсходован пол
ностью. Обращаясь к системам ограничений прямо
(1.54) и двойственной (1.56) задач, можно дополнит
интерпретацию условий 1 и 2 следующим образом. И
ограничений (1.56) и условия 1 получаем: если затрат
ресурсов на выпуск какого-либо продукта j превышаю
его стоимость, то этот продукт не производится, $x_j=0$
Из ограничений (1.54) и условия 2 получаем: если ка
кой-то ресурс i расходуется не полностью, то его цен
$y_i=0$.

Таким образом, каждому ресурсу можно присвоит
оценку, являющуюся характеристикой его дефицитност
Ресурсы, используемые в производстве, не равнозначны
Те из них, которые при оптимальном плане производст
ва используются не полностью, получают нулевую оцен
ку. Увеличение или уменьшение запасов таких ресурсо
не отражается на величине целевой функции и, следова
тельно, не влияет на показатель качества производст
венного плана. Если же оценка i-го ресурса $y_i>0$, т
увеличение его использования на единицу означает улуч

шение показателя качества работы — значения целевой функции — на y_i единиц.

В заключение приведем без вывода некоторые следствия из теорем существования и равновесия.

Следствие 1.1. *Если одна из двойственных задач имеет оптимальное решение, то имеет оптимальное решение и другая.*

Следствие 1.2. *Если одна из двойственных задач допустима, а вторая недопустима, то целевая функция первой задачи не ограничена.*

Тесная связь, существующая между взаимодвойственными задачами, проявляется и в том, что при решении одной из них одновременно решается и другая. Действительно, нетрудно заметить, что симплекс-таблица с условиями исходной задачи включает и условия двойственной задачи. Так, строки этой таблицы содержат левые части ограничений исходной задачи, столбцы — левые части ограничений двойственной задачи. Элементы столбца b представляют коэффициенты вектора ограничений прямой задачи и одновременно оценки плана двойственной задачи. Элементы строки Δ дают оценки плана исходной и одновременно коэффициенты вектора ограничений двойственной задач. Поэтому при решении двойственной задачи можно не строить специальную симплекс-таблицу, а решать ее по таблице, где записаны условия исходной задачи. Их оптимальные решения отыскиваются одновременно. Компоненты оптимального плана двойственной задачи находятся в клетках оценочной строки, соответствующих начальному единичному базису прямой задачи.

Пример 1.12. Найти решение пары взаимодвойственных задач из примера 1.11.

Вводя в неравенства прямой задачи дополнительные переменные x_4, $x_5 \geqslant 0$, получаем каноническую задачу с единичным базисом:

$$\max z = 3x_1 - x_2 + 6x_3; \quad 2x_1 - 2x_2 + 2x_3 + x_4 = 1;$$
$$x_1 + 7x_2 + 3x_3 + x_5 = 2; \quad x_j \geqslant 0, \ j = 1, \ldots, 5.$$

Записываем ее условия в симплекс-таблицу (табл. 1.10) и после двух итераций получаем решение прямой задачи: x = (0, 1/20, 11/20, 0, 0), $z_{max} = 13/4$. Решением двойственной задачи будет план y = (9/4, 1/2), $w_{min} = 13/4$.

Двойственный симплекс-метод. Симплекс-метод применяется для решения задач с неотрицательными коэффициентами b_i вектора ограничений и произвольными по

Таблица 1.

c_B	x_B	b	3	−1	6	0	0
			a^1	a^2	a^3	a^4	a^5
0	x_4	1	2	−2	[2]	1	0
0	x_5	2	1	7	3	0	1
	Δ	0	−3	1	−6	0	0
6	x_3	1/2	1	−1	1	1/2	0
0	x_5	1/2	−2	[10]	0	−3/2	1
	Δ	3	3	−5	0	3	0
6	x_3	11/20	4/5	0	1	7/20	1/1
−1	x_2	1/20	−1/5	1	0	−3/20	1/1
	Δ	13/4	2	0	0	9/4	1/2

знаку приведенными коэффициентами целевой функц
Δ_j. Иногда бывает легче найти базис, удовлетворяющ
признаку оптимальности (все $\Delta_j \geqslant 0$), но не удовлетвор
ющий критерию допустимости (так как не все $b_i \geqslant 0$
Вариант симплекс-метода, применяемый для решен
таких задач, называется *двойственным симплекс-ме
дом.* С его помощью решаются задачи линейного пр
граммирования

$$\max \mathbf{cx};\ A\mathbf{x}=\mathbf{b};\ \mathbf{x}\geqslant 0,$$

где матрица A содержит единичный базис и все при
денные коэффициенты целевой функции $\Delta_j \geqslant 0$, j
$=1, \ldots, n$. При этом условие $b_i \geqslant 0$, $i=1, \ldots, m$, не треб
ется. Определенную таким образом задачу будем наз
вать *задачей в двойственной базисной форме.*

Обычный симплекс-метод, применяемый к задаче
базисной форме, приводит к последовательности эквив
лентных задач с возрастающим значением целевой фун
ции и неотрицательными значениями b_i, так что кажд
базисное решение является допустимым. Двойственн
симплекс-метод, применяемый к задаче в двойственн
базисной форме, приводит к последовательности задач
убывающим значением целевой функции, неотрицател
ными коэффициентами Δ_j, $j=1, \ldots, n$, и значениями
любого знака. Преобразования задач выполняются
тех пор, пока не будет установлено, что исходная зада
не имеет допустимого решения или будет получена зад

ча с допустимым базисным решением (все $b_i \geqslant 0$), которое одновременно и оптимально.

Эти особенности алгоритмов находят отражение и в названиях, которые применяют к ним. Обычный симплекс-метод называют *методом последовательного улучшения планов*, а двойственный — *методом последовательного уточнения оценок*.

Перейдем к обоснованию алгоритма. Рассмотрим задачу в двойственной базисной форме. В отношении ее возможен один из трех случаев.

Случай 1. Все координаты вектора ограничений неотрицательны, $b_i \geqslant 0$, $i = 1, \ldots, m$. Тогда они дают не только допустимый, но и оптимальный план задачи, поскольку по предположению все оценки $\Delta_j \geqslant 0$.

Случай 2. Имеется строка $i \in 1, \ldots, m$, такая, что $b_i < 0$ и $a_{ij} \geqslant 0$ для всех $j = 1, \ldots, n$. Тогда задача неразрешима в силу несовместности ограничений.

Действительно, для любого неотрицательного вектора $x = (x_1, x_2, \ldots, x_n)$ при всех $a_{ij} \geqslant 0$ имеем $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geqslant 0$. Поэтому ни один из векторов $x \geqslant 0$ не может удовлетворить i-е ограничение $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i$, где $b_i < 0$.

Случай 3. Имеется строка $r \in 1, \ldots, m$, такая, что $b_r < 0$ и $a_{rj} < 0$ хотя бы для одного $j \in 1, \ldots, n$. Пусть $s \in 1, \ldots, n$ таково, что $a_{rs} < 0$ и

$$\Delta_s / a_{rs} = \max_{a_{rj} < 0} (\Delta_j / a_{rj}).$$

Тогда жорданово преобразование с ведущим элементом a_{rs} приводит к эквивалентной задаче, в которой, во-первых, оценки $\Delta_j \geqslant 0$, $j = 1, \ldots, n$, а во-вторых, значение целевой функции $z' \leqslant z$, и если $\Delta_s \neq 0$, то $z' < z$.

◆ В соответствии с формулами жорданова преобразования (1.42), (1.44) элементы оценочной строки эквивалентной задачи

$$\Delta'_j = \Delta_j - (\Delta_s / a_{rs})\, a_{rj}, \quad j = 1, \ldots, n,$$

$$z' = z - (\Delta_s / a_{rs})\, b_r.$$

При расчете коэффициента Δ'_j возможны два случая: $a_{rj} \geqslant 0$ и $a_{rj} < 0$. Если $a_{rj} \geqslant 0$, то $-(\Delta_s / a_{rs})\, a_{rj} \geqslant 0$, так как $\Delta_s \geqslant 0$, и по определению ведущий элемент $a_{rs} < 0$. Отсюда $\Delta'_j \geqslant \Delta_j \geqslant 0$. Если $a_{rj} < 0$, то в силу выбора s

имеем $\Delta_j/a_{rj} \leqslant \Delta_s/a_{rs}$. Умножая обе части этого неравенства на $a_{rj} < 0$, получаем $\Delta_j \geqslant (\Delta_s/a_{rs})\, a_{rj}$. Отсюда $\Delta'_j = \Delta_j - (\Delta_s/a_{rs})\, a_{rj} \geqslant 0$. Неотрицательность оценок Δ'_j доказана.

Рассмотрим выражение для целевой функции z'. Так как $\Delta_s \geqslant 0$, $a_{rs} < 0$, $b_r < 0$, имеем $(\Delta_s/a_{rs})\, b_r \geqslant 0$. Отсюда $z' = z - (\Delta_s/a_{rs})\, b_r \leqslant z$. Если $\Delta_s \neq 0$, то $(\Delta_s/a_{rs}) \times b_r > 0$, и поэтому $z' < z$. ◆

Вычислительная схема двойственного симплекс-метода. Данный метод, применяемый к задаче в двойственной базисной форме, состоит из следующих шагов, повторяющихся до тех пор, пока в ходе жордановых преобразований не будет установлено соответствие очередной эквивалентной задачи приведенным случаям 1 или 2.

1. Проверяем знаки коэффициентов ограничений b_i. Если все $b_i \geqslant 0$, $i = 1, \ldots, m$, то имеет место случай 1. Базисное решение и значение целевой функции, записанные в столбце свободных членов $\mathbf{b}$, дают оптимальное решение исходной задачи. Если не все $b_i \geqslant 0$, переходим к шагу 2.

2. Среди отрицательных коэффициентов b_i выбираем коэффициент b_r, наибольший по абсолютной величине, и строку r называем ведущей.

3. В ведущей строке проверяем знаки всех коэффициентов a_{rj}, $j = 1, \ldots, n$. Если все $a_{rj} \geqslant 0$, имеет место случай 2. Решение окончено, задача неразрешима. Если найдется хотя бы один коэффициент $a_{rj} < 0$, имеет место случай 3. Переходим к шагу 4.

4. Среди отрицательных коэффициентов a_{rj} ведущей строки выбираем элемент a_{rs}, для которого

$$\Delta_s/a_{rs} = \max_{a_{rj}<0} (\Delta_j/a_{rj}),$$

и называем его ведущим.

5. Выполняем жорданово преобразование симплекс-таблицы с ведущим элементом a_{rs} и переходим к шагу 1.

Оптимальный план прямой задачи задается списком базисных переменных и их значениями в столбце $\mathbf{b}$, двойственной задачи — переменными двойственной задачи, сопоставленными базисным столбцам исходной задачи, и их значениями в оценочной строке Δ.

Задачи в двойственной базисной форме, к которым можно непосредственно применить изложенный алгоритм, обычно возникают двумя путями. В первом случае

это задачи минимизации вида $\min w=\mathbf{cx}$; $A\mathbf{x}\geqslant\mathbf{0}$; $\mathbf{x}\geqslant\mathbf{0}$, где все коэффициенты $c_j\geqslant 0$. Такая задача может быть сведена к эквивалентной задаче в двойственной базисной форме заменой минимизации целевой функции w максимизацией функции $z=-w$, умножением на -1 обеих частей всех неравенств вида $\geqslant$ и введением дополнительных переменных для построения единичного базиса. Другой случай имеет место, когда в систему ограничений задачи, для которой найдено оптимальное решение, вводятся дополнительные условия, делающие это решение недопустимым. Тогда использование двойственного симплекс-метода позволяет получить новое оптимальное решение, при этом не надо проводить все расчеты сначала. Такое применение двойственного симплекс-метода имеет место, в частности, при рассмотрении целочисленных задач. Наконец, двойственный симплекс-метод удобно использовать для решения задач, которые обладают единичным базисом, но не относятся к задачам в базисной или двойственной базисной форме, так как имеют отрицательные коэффициенты среди элементов вектора $\mathbf{b}$ и строки Δ одновременно. Решение такой задачи состоит из двух этапов: сначала с помощью двойственного симплекс-метода исключаются все $x_i<0$, затем оптимальный план находится обычным симплекс-методом. Следует только на первом этапе изменить шаг 4 алгоритма следующим образом.

4′. Среди отрицательных коэффициентов a_{rj} ведущей строки выбрать элемент a_{rs}, для которого

$$b_r/a_{rs}=\max_{a_{rj}<0}(b_r/a_{rj}).$$

Пример 1.13. Решить задачу

$$\min w=x_1+2x_2;\quad 2x_1+x_2\geqslant 3;\quad 2x_1-7x_2\leqslant 1;$$
$$2x_1+3x_2\geqslant 6;\quad x_1\geqslant 0,\quad x_2\geqslant 0.$$

Перейдем к эквивалентной задаче максимизации. Для этого поменяем знаки коэффициентов целевой функции, умножим на -1 первое и третье неравенства и введем дополнительные переменные x_3, x_4, x_5. В результате получаем задачу

$$\max z=-x_1-2x_2;\quad -2x_1-x_2+x_3=-3;$$
$$2x_1-7x_2+x_4=1;\quad -2x_1-3x_2+x_5=-6;\quad x_j\geqslant 0,\quad j=1,\ldots,5.$$

В исходной части симплекс-таблицы (табл. I.11) все элементы Δ_j оценочной строки неотрицательны. В соответствии с шагами 1, 2 двойственного симплекс-метода выбираем ведущую строку $r=3$. Выполняя шаг 4, находим отношения коэффициентов оценочной строки к отрицательным элементам ведущей строки, равные $1/-2$ и

Таблица 1.11

c_B	x_B	b	−1	−2	0	0	0
			a^1	a^2	a^3	a^4	a^5
0	x_3	−3	−2	−1	1	0	0
0	x_4	1	2	−7	0	1	0
0	x_5	−6	**−2**	−3	0	0	1
	Δ	0	1	2	0	0	0
0	x_3	3	0	2	1	0	1
0	x_4	−5	0	**−10**	0	1	1
−1	x_1	3	1	3/2	0	0	−1/2
	Δ	−3	0	1/2	0	0	1/2
0	x_3	2	0	0	1	1/5	−4/5
−2	x_2	1/2	0	1	0	−1/10	−1/10
−1	x_1	9/4	1	0	0	3/20	−7/20
	Δ	−13/4	0	0	0	1/20	11/20

2/−3. Большее из этих чисел указывает ведущий элемент $a_{31} = -2$. Выполнив жорданово преобразование, на следующей итерации по тем же правилам находим ведущий элемент $a_{22} = -10$.

После двух итераций двойственного симплекс-метода получаем оптимальный план $x = (9/4, 1/2, 2, 0, 0)$, $z_{max} = -13/4$. Это дает решение исходной задачи в виде $x = (9/4, 1/2)$, $w_{min} = 13/4$.

Пример 1.14. Решить задачу

$$\max\ z = 8x_1 + 2x_2 - 5x_3; \quad -x_1 + 2x_2 + x_3 \geqslant 6;$$

$$x_1 - 2x_2 + 2x_3 \geqslant 3; \quad 2x_1 + x_2 - x_3 \leqslant 2; \quad x_1 \geqslant 0,\ x_2 \geqslant 0,\ x_3 \geqslant 0.$$

С помощью дополнительных переменных x_4, x_5, $x_6 \geqslant 0$ переходим от ограничений-неравенств к ограничениям-равенствам, и после умножения первого и второго ограничений на -1 окончательно получаем

$$\max\ z = 8x_1 + 2x_2 - 5x_3; \quad x_1 - 2x_2 - x_3 + x_4 = -6;$$

$$-x_1 + 2x_2 - 2x_3 + x_5 = -3; \quad 2x_1 + x_2 - x_3 + x_6 = 2; \quad x_j \geqslant 0,\ j = 1, \ldots, 6.$$

Заполняем симплекс-таблицу (табл. 1.12) и в соответствии с шагами 1 и 2 двойственного алгоритма выбираем ведущую строку $r = 1$. Выполняя шаг 4′, сравниваем отношения $-6/-2$ и $-6/-1$. Поскольку второе отношение больше, выбираем ведущий столбец $s = 3$. Выполняем жорданово преобразование с ведущим элементом $a_{13} = -1$ и получаем эквивалентную задачу с неотрицательными значениями b_i. Проведя еще две итерации по правилам обычного симплекс-метода, получаем оптимальный план $x = (7/5, 11/5, 3, 0, 0, 0)$, $z_{max} = 3/5$.

Таблица 1.12

c_B	x_B	b	8	2	−5	0	0	0
			a^1	a^2	a^3	a^4	a^5	a^6
0	x_4	−6	1	−2	[−1]	1	0	0
0	x_5	−3	−1	2	−2	0	1	0
0	x_6	2	2	1	−1	0	0	1
	Δ	0	−8	−2	5	0	0	0
−5	x_3	6	−1	2	1	−1	0	0
0	x_5	9	−3	[6]	0	−2	1	0
0	x_6	8	1	3	0	−1	0	1
	Δ	−30	−3	−12	0	5	0	0
−5	x_3	3	0	0	1	−1/3	−1/3	0
2	x_2	3/2	−1/2	1	0	−1/3	1/6	0
0	x_6	7/2	[5/2]	0	0	0	−1/2	1
	Δ	−12	−9	0	0	1	2	0
−5	x_3	3	0	0	1	−1/3	−1/3	0
2	x_2	11/5	0	1	0	−1/3	1/15	1/5
8	x_1	7/5	1	0	0	0	−1/5	2/5
	Δ	3/5	0	0	0	1	1/5	18/5

Контрольные вопросы и упражнения

1. Дайте определение задач математического программирования, линейного программирования. Назовите эквивалентные формы задачи линейного программирования, опишите преобразования, необходимые для перехода от одной формы к другой.

2. Что называется областью допустимых решений задачи линейного программирования, что она представляет собой геометрически? В каких случаях задача линейного программирования имеет решение и когда она неразрешима? На чем основывается графический метод решения задач линейного программирования? Когда он применим?

3. Какое решение системы линейных алгебраических уравнений называется базисным? Когда оно является вырожденным?

4. Дайте определение опорного плана и его базиса, поясните их геометрический смысл и основную идею численных методов решения задач линейного программирования.

5. Опишите методы построения исходного опорного плана.

6. Сформулируйте критерий оптимальности решения задачи на максимум, дайте общую схему отыскания оптимального плана симплекс-методом.

7. Как выбирается переменная, выводимая из базиса, вводимая в базис; какой элемент называется ведущим?

8. Опишите преобразования, приводящие задачу к новому базису. Как устанавливается неразрешимость задачи?

9. Как строится задача, двойственная задаче в стандартной форме, в канонической форме? Какие задачи называются симметричными, несимметричными?

10. Сформулируйте теоремы двойственности, дайте их экономическую интерпретацию.

11. Опишите вычислительную схему двойственного симплекс-метода.

12. Решите следующие задачи:

а) $\max z = 4x_1 + 3x_2 + 6x_3$; $2x_1 + x_2 + x_3 \leqslant 40$; $4x_1 + 9x_3 \leqslant 100$;

$3x_2 + 5x_3 \leqslant 30$; $2x_1 + x_2 + 5x_3 \leqslant 60$; $x_1,\ x_2,\ x_3 \geqslant 0$;

б) $\max z = 20x_1 + 10x_2 + 9x_3 + 10x_4$; $7x_1 + 7x_2 + 3x_3 \leqslant 60$; $3x_2 + x_3 + 3x_4 \leqslant 35$;

$8x_1 + 4x_3 + x_4 \leqslant 75$; $5x_1 + 2x_2 + 3x_4 \leqslant 50$; $x_1,\ x_2,\ x_3,\ x_4 \geqslant 0$.

2. ЗАДАЧИ ТРАНСПОРТНОГО ТИПА

2.1. ТРАНСПОРТНАЯ ЗАДАЧА В МАТРИЧНОЙ ПОСТАНОВКЕ

Математическая модель. В данной главе рассматриваются задачи линейного программирования, выделяемые в отдельный класс благодаря особенностям их системы ограничений. Специальный вид ограничений позволяет существенно упростить реализацию общей схемы симплекс-метода, а иногда предложить и другие эффективные алгоритмы, не связанные с симплекс-методом. Начнем с формулировки *классической транспортной задачи*, или *транспортной задачи в матричной постановке*.

Пусть имеется m пунктов производства некоторого однородного продукта (сталь, нефть, уголь и т. д.) и n пунктов его потребления. Известны запасы a_i продукта в каждом пункте-поставщике $i=1, \ldots, m$, спрос b_j в каждом пункте-потребителе $j=1, \ldots, n$ и расходы c_{ij} на перевозку единицы продукта из пункта i в пункт j. Будем полагать, что суммарный запас равен суммарному спросу и $a_i>0$, $b_j>0$, $c_{ij}\geqslant 0$ для всех $i=1, \ldots, m$, $j=1, \ldots, n$.

Таблица 2.

i \ j	1	2	...	n	a_i
1	c_{11}	c_{12}	...	c_{1n}	a_1
2	c_{21}	c_{22}	...	c_{2n}	a_2
⋮	⋮	⋮	⋮	⋮	⋮
m	c_{m1}	c_{m2}	...	c_{mn}	a_m
b_j	b_1	b_2	...	b_n	

Требуется составить план перевозок продукта (или закрепления поставщиков за потребителями), при котором запасы каждого поставщика были бы вывезены, спрос каждого потребителя удовлетворен и общие транспортные расходы были бы минимальными. Условия транспортной задачи обычно записывают в таблицах (матрицах), имеющих вид табл. 2.1.

Составим математическую модель задачи. Обозначим через x_{ij} искомый объем перевозки от поставщика i к потребителю j и будем рассматривать переменные x_{ij}, задающие план перевозок, как компоненты *матрицы перевозок* X размеров $m \times n$

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}. \tag{2.1}$$

Затраты, связанные с некоторой перевозкой x_{ij}, составят величину $c_{ij}x_{ij}$, а общая стоимость перевозок z от всех поставщиков ко всем потребителям определится равенством

$$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij}. \tag{2.2}$$

В соответствии с постановкой задачи план перевозок должен быть составлен так, чтобы вывоз от каждого поставщика равнялся его объему производства:

$$\sum_{j=1}^{n} x_{ij} = a_i, \; i = 1, \dots, m, \tag{2.3}$$

а общие поставки любому потребителю удовлетворяли его спрос:

$$\sum_{i=1}^{m} x_{ij} = b_j, \; j = 1, \dots, n. \tag{2.4}$$

Из физического смысла переменных следуют условия их неотрицательности

$$x_{ij} \geqslant 0, \; i = 1, \dots, m, \; j = 1, \dots, n. \tag{2.5}$$

В итоге получаем формулировку транспортной задачи: *найти значения переменных x_{ij}, удовлетворяющие условиям (2.3) — (2.5) и минимизирующие целевую функцию (2.2)*. Очевидно, это каноническая задача линейного про-

граммирования. В ней число переменных равно mn, число ограничений-равенств — $m+n$.

Нетрудно видеть, что левые части уравнений (2.3) образованы строчными, а уравнений (2.4) — столбцовыми элементами матрицы перевозок (2.1). В соответствии с условиями (2.3) и (2.4) сумма элементов i-й строки матрицы X должна быть равна a_i, а сумма элементов j-го столбца — b_j. В дальнейшем будем называть уравнения (2.3) *строчными*, (2.4) — *столбцовыми*.

Для проверки условий совместности системы (2.3), (2.4) проведем суммирование уравнений (2.3) по индексу i, а (2.4) — по j. Получаем равенства

$$\sum_i \sum_j x_{ij} = \sum_i a_i; \quad \sum_j \sum_i x_{ij} = \sum_j b_j,$$

левые части которых отличаются только порядком суммирования. Следовательно, они равны между собой. Тогда будут равны и правые части

$$\sum_i a_i = \sum_j b_j. \tag{2.6}$$

Условие (2.6) является *условием совместности* системы ограничений (2.3) — (2.4). Оно выражает требование *баланса* между суммарными запасами и суммарными потребностями.

Открытая задача. Транспортную задачу, для которой выполняется условие баланса (2.6), называют *закрытой задачей*. Если же это условие нарушено, то говорят, что задача *открытая*. Здесь возможны два случая: 1) суммарные запасы превышают суммарный спрос; 2) суммарный спрос больше суммарных запасов. В первом случае после удовлетворения спроса всех потребителей у некоторых поставщиков останется невывезенный продукт, во втором случае поставки для некоторых потребителей будут меньше их потребности.

При построении модели в первом случае для совместности условий строчные ограничения должны быть записаны как ограничения-неравенства, допускающие неполный вывоз имеющегося продукта. Модель примет вид

$$\min z = \sum_i \sum_j c_{ij} x_{ij};$$

$$\sum_j x_{ij} \leqslant a_i, \; i = 1, \ldots, m;$$

$$\sum_i x_{ij} = b_j,\ j = 1,\ \ldots,\ n;$$

$$x_{ij} \geqslant 0,\ i = 1,\ \ldots,\ m,\ j = 1,\ \ldots,\ n.$$

Аналогично во втором случае неравенствами должн быть выражены столбцовые ограничения, допускающ неполное удовлетворение спроса.

Открытая модель легко сводится к эквивалентной закрытой модели. Так, в первом случае введем фиктивны потребитель с величиной спроса $\sum_i a_i - \sum_j b_j$ и установ стоимости перевозок от каждого поставщика к фиктивн му потребителю равными нулю. В результате получим з крытую транспортную задачу, решение которой будет р шением исходной открытой задачи. Действительно, в л бом плане данной закрытой задачи спрос всех реальны потребителей будет удовлетворен полностью, так как спр фиктивного потребителя равен имеющемуся избытку пр дукта. Поэтому совокупность перевозок к реальным п требителям дает план исходной открытой задачи, а знач ния фиктивных перевозок — объемы продукта, остающе ся у соответствующих поставщиков. Поскольку расход на перевозки к фиктивному потребителю равны нулю, м нимум целевой функции в обеих задачах имеет одно то же значение.

Во втором случае эквивалентная закрытая зада строится введением фиктивного поставщика, запас кот рого равен избыточному спросу всех потребителей. Р шением исходной открытой задачи будет совокупнос перевозок от реальных поставщиков ко всем потреби лям, а поставки от фиктивного поставщика укаж объем неудовлетворенного спроса соответствующих п требителей.

Векторная запись. Для связи с общей задачей лине ного программирования иногда будет удобно рассматр вать правые части уравнений (2.3), (2.4) в виде вектор столбца $\mathbf{b}^T = (a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n)$, коэффициен издержек c_{ij} — как mn компонент вектора-строки $\mathbf{c} = (c$ $c_{12}, \ldots, c_{1n}, c_{21}, c_{22}, \ldots, c_{2n}, \ldots, c_{m1}, c_{m2}, \ldots, c_{mn})$, переме ные x_{ij} — как mn компонент вектора-столбца $\mathbf{x}^T$ $= (x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, x_{22}, \ldots, x_{2n}, \ldots, x_{m1}, x_{m2}, \ldots, x_m$ а левые части уравнений (2.3), (2.4) записывать в ви матрицы условий

$$A=\begin{array}{l}\left[\begin{array}{cccccccccc}\overbrace{11\ \ldots\ 1}^{n} & & & \overbrace{00\ \ldots\ 0}^{n} & & & \ldots & \overbrace{00\ \ldots\ 0}^{n}\\ 00\ \ldots\ 0 & & & 11\ \ldots\ 1 & & & \ldots & 00\ \ldots\ 0\\ \ldots & & & \ldots & & & & \ldots\\ 00\ \ldots\ 0 & & & 00\ \ldots\ 0 & & & \ldots & 11\ \ldots\ 1\\ 10\ \ldots\ 0 & & & 10\ \ldots\ 0 & & & \ldots & 10\ \ldots\ 0\\ 01\ \ldots\ 0 & & & 01\ \ldots\ 0 & & & \ldots & 01\ \ldots\ 0\\ \ldots & & & \ldots & & & & \ldots\\ 00\ \ldots\ 1 & & & 00\ \ldots\ 1 & & & \ldots & 00\ \ldots\ 1\end{array}\right]\begin{array}{l}\}m\\ \}n\end{array}\end{array} \tag{2.7}$$

с $m+n$ строками и mn столбцами. Это позволяет записать постановку транспортной задачи в виде модели

$$\min z=\mathbf{cx};\ A\mathbf{x}=\mathbf{b};\ \mathbf{x}\geqslant 0 \tag{2.8}$$

или

$$\min z=\mathbf{cx};\ \sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{a}^{ij}x_{ij}=\mathbf{b};\ x_{ij}\geqslant 0,\ i=1,\ \ldots,\ m,\ j=1,\ \ldots,\ n, \tag{2.9}$$

где $\mathbf{a}^{ij}$ — столбец коэффициентов матрицы A при переменной x_{ij}. Пусть, например, $m=2$, $n=3$. Тогда система ограничений задачи в векторной записи примет вид

$$\begin{bmatrix}1&1&1&0&0&0\\0&0&0&1&1&1\\1&0&0&1&0&0\\0&1&0&0&1&0\\0&0&1&0&0&1\end{bmatrix}\begin{bmatrix}x_{11}\\x_{12}\\x_{13}\\x_{21}\\x_{22}\\x_{23}\end{bmatrix}=\begin{bmatrix}a_1\\a_2\\b_1\\b_2\\b_3\end{bmatrix}. \tag{2.10}$$

Поскольку в системе ограничений каждая переменная x_{ij} встречается всего два раза: один раз — в i-м строчном уравнении (2.3), другой — в j-м столбцовом уравнении (2.4), и оба раза с коэффициентом единица, то любой столбец $\mathbf{a}^{ij}$ имеет вид

$$\mathbf{a}^{ij}=\begin{bmatrix}0\\ \vdots\\ 0\\ 1\\ 0\\ \vdots\\ 0\\ 1\\ 0\\ \vdots\\ 0\end{bmatrix}\begin{array}{l}\\ \\ \\ \uparrow i\\ \\ \\ \\ \uparrow m+j\\ \\ \\ \downarrow m+n\end{array} \tag{2.11}$$

В нем $m+n$ компонент, из которых i-я и $(m+j)$-я равны единице, а остальные — нулю. Используя обозначение $\mathbf{e}^i$ для $(m+n)$-мерного вектора-орта, в котором i-я компонента равна единице, а остальные — нулю, можем записать вектор $\mathbf{a}^{ij}$ в виде суммы

$$\mathbf{a}^{ij}=\mathbf{e}^i+\mathbf{e}^{m+j}. \qquad (2.12)$$

Число векторов-столбцов $\mathbf{a}^{ij}$ матрицы A равно mn и совпадает с числом транспортных связей между m поставщиками и n потребителями, при этом каждый вектор $\mathbf{a}^{ij}$ соответствует связи между поставщиком i и потребителем j.

Формально модель (2.8) ничем не отличается от обычной задачи линейного программирования. В модели (2.9) отличие состоит лишь в записи двойных индексов при переменных x_{ij} и векторах $\mathbf{a}^{ij}$, что позволяет упростить постановку задачи. Таким образом, транспортную задачу можно было бы решать с помощью обычного симплекс-метода как задачу с $m+n$ ограничениями и mn переменными. Однако даже при малых значениях m и n число переменных mn задачи слишком велико, чтобы ее можно было эффективно решать обычным симплекс-методом. В то же время, как отмечалось, векторы условий (2.11) транспортной задачи имеют очень простую структуру. Благодаря этому для реализации общих идей симплекс-метода разработаны вычислительные процедуры, резко сокращающие требования к памяти и быстродействию ЭВМ. Рассмотрим, в частности, как упрощается исследование линейной зависимости векторов-столбцов матрицы A.

Линейная зависимость и циклы. Запишем матрицу перевозок (2.1) в виде *таблицы перевозок* — табл. 2.2. Каждая клетка (i, j) этой таблицы занята переменной x_{ij} и соответствует столбцу $\mathbf{a}^{ij}$ матрицы условий, а совокупность всех клеток соответствует матрице A. Любую последовательность клеток таблицы будем записывать в виде последовательности (i_1, j_1), ..., (i_t, j_t). Рассмотрим, в частности, замкнутые последовательности клеток, или циклы. *Циклом* в транспортной таблице называется такая последовательность клеток (i_1, j_1), ..., (i_t, j_t), что, во-первых, все входящие в нее клетки различны и, во-вторых, две смежные клетки последовательности лежат в одной строке или в одном столбце. При этом клетки (i_t, j_t), (i_1, j_1) также считаются смежными, следовательно, лежат в одной строке или в одном столбце.

Таблица 2.2

i \ j	1	2	...	n	a_i
1	x_{11}	x_{12}	...	x_{1n}	a_1
2	x_{21}	x_{22}	...	x_{2n}	a_2
⋮	⋮	⋮	⋮	⋮	⋮
m	x_{m1}	x_{m2}	...	x_{mn}	a_m
b_j	b_1	b_2	...	b_n	

Таким образом, у любой пары соседних клеток цикла должны быть одинаковыми или первые или вторые индексы. Так, например, последовательность

$$(i_1, j_1), (i_1, j_2), (i_2, j_2), \ldots, (i_p, j_p), (i_p, j_1) \quad (2.13)$$

образует цикл, где переход от клетки (i_1, j_1) к клетке (i_1, j_2) выполняется по строке i_1, а возврат от клетки (i_p, j_1) к клетке (i_1, j_1) осуществляется по столбцу j_1. Примеры циклов приведены на рис. 2.1.

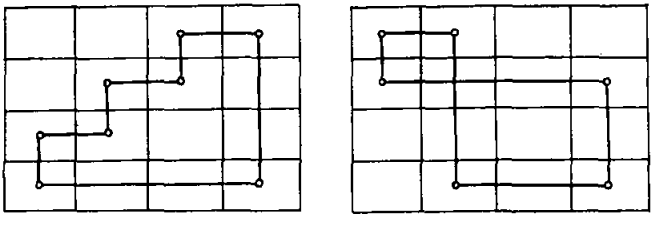

Рис. 2.1.

Рассмотрим совокупность столбцов $\mathbf{a}^{ij}$, для которых соответствующие им клетки (i, j) образуют цикл вида (2.13), и составим линейную комбинацию

$$\mathbf{a}^{i_1 j_1} - \mathbf{a}^{i_1 j_2} + \mathbf{a}^{i_2 j_2} - \ldots + \mathbf{a}^{i_p j_p} - \mathbf{a}^{i_p j_1},$$

в которой коэффициентами являются чередующиеся $+1$ и -1, а последовательность столбцов отвечает последовательности клеток цикла. Заменяя в соответствии с

(2.12) каждый столбец суммой двух векторов-ортов, получаем

$$\mathbf{e}^{i_1} + \mathbf{e}^{m+j_1} - \mathbf{e}^{i_1} - \mathbf{e}^{m+j_2} + \ldots - \mathbf{e}^{i_p} - \mathbf{e}^{m+j_1}.$$

Нетрудно видеть, что сокращение членов с противоположными знаками обращает это выражение в нулевой вектор, следовательно,

$$\mathbf{a}^{i_1 j_1} - \mathbf{a}^{i_1 j_2} + \mathbf{a}^{i_2 j_2} - \ldots + \mathbf{a}^{i_p j_p} - \mathbf{a}^{i_p j_1} = \mathbf{0}.$$

Отсюда совокупность столбцов матрицы A, которым отвечает последовательность клеток, образующих цикл, является линейно зависимой. Перенося вправо все члены линейной комбинации, кроме первого, получим выражение

$$\mathbf{a}^{i_1 j_1} = \mathbf{a}^{i_1 j_2} - \mathbf{a}^{i_2 j_2} + \ldots - \mathbf{a}^{i_p j_p} + \mathbf{a}^{i_p j_1}. \qquad (2.14)$$

Теорема 2.1. *Система векторов-столбцов* $\mathbf{a}^{ij}$ *матрицы* A *линейно зависима тогда и только тогда, когда из совокупности отвечающих им клеток таблицы перевозок можно выделить последовательность, образующую цикл. Столбцы, соответствующие клеткам цикла, удовлетворяют уравнениям (2.14).*

◆ Достаточность доказана рассуждениями, непосредственно предшествующими теореме. Для доказательства необходимости предположим, что имеется некоторая линейно зависимая система векторов-столбцов матрицы A. Соответствующую ей совокупность клеток обозначим через C. Из предположения линейной зависимости столбцов следует, что существует такая совокупность чисел α_{ij}, среди которых не все равны нулю, что

$$\sum_{(i,\ j)\in C} \alpha_{ij}\mathbf{a}^{ij} = \sum_{(i,\ j)\in C} \alpha_{ij}\,(\mathbf{e}^{i} + \mathbf{e}^{m+j}) = \mathbf{0}.$$

В силу линейной независимости векторов-ортов равенство выполняется, если

$$\sum_{(i,\ j)\in C} \alpha_{ij}\mathbf{e}^{i} = \mathbf{0}; \qquad (2.15)$$

$$\sum_{(i,\ j)\in C} \alpha_{ij}\mathbf{e}^{m+j} = \mathbf{0}. \qquad (2.16)$$

Выберем в совокупности C клетку $(i_1,\ j_1)$, такую, что $\alpha_{i_1 j_1} \neq 0$. Тогда из (2.15) вытекает, что должна найтись другая клетка $(i_1,\ j_2) \in C$, $j_2 \neq j_1$, для которой $\alpha_{i_1 j_2} \neq 0$. Далее в силу (2.16) в совокупности C должна быть клет-

ка $(i_2, j_2) \in C$, $i_2 \neq i_1$, такая, что $\alpha_{i_2 j_2} \neq 0$. Продолжая эти рассуждения, будем поочередно двигаться то по строке, то по столбцу, так как каждая выделяемая клетка отличается от непосредственно предшествующей одним индексом. Поскольку число клеток $(i, j) \in C$ конечно, то, двигаясь по взаимно ортогональным направлениям, мы в конце концов возвратимся в начальную клетку, т. е. построим цикл. ◆

Как будет показано, из данной теоремы вытекает простой способ разложения векторов-столбцов матрицы A по векторам базиса.

2.2. ОСОБЕННОСТИ ТРАНСПОРТНОЙ ЗАДАЧИ

Линейная зависимость системы ограничений. Специфический характер уравнений (2.3), (2.4) и соответствующей им матрицы условий (2.7), состоящей из одних нулей и единиц, влечет за собой целый ряд свойств транспортной задачи.

Свойство 1. *Ранг $r(A)$ матрицы условий A равен $m+n-1$.*

Сначала докажем неравенство $r(A) \leqslant m+n-1$. Оно легко проверяется: сумма первых m строк и сумма последних n строк равны одному и тому же вектору, где все компоненты — единицы. Образуя разность этих сумм, получаем нулевой вектор. Отсюда одна из строк матрицы A (любая) является линейной комбинацией остальных и $r(A) \leqslant m+n-1$.

Покажем теперь, что на самом деле $r(A) = m+n-1$. Для этого достаточно доказать, что в матрице A существует $m+n-1$ линейно независимых строк. Удалим из матрицы A одну строку, допустим, первую, а из остальных $m+n-1$ строк образуем линейную комбинацию следующим образом: возьмем строки $i=2, 3, \ldots, m$ с коэффициентами $\alpha_2, \alpha_3, \ldots, \alpha_m$, строки $i=m+1, \ldots, m+n$ с коэффициентами $\beta_1, \beta_2, \ldots, \beta_n$ и поэлементно сложим их. Если в результате будет получен нулевой вектор-строка при коэффициентах α_i и β_j, не всех равных нулю, тогда данная совокупность строк линейно зависима.

Рассмотрим, как при этом формируются компоненты нулевого вектора-строки. Если при записи слагаемых сохранить лишь те слагаемые, где сомножители при коэффициентах α_i и β_j отличны от нуля, то с учетом структуры матрицы A для первых n компонент получим ра-

венства $\beta_1 \cdot 1 = 0$, $\beta_2 \cdot 1 = 0$, ..., $\beta_n \cdot 1 = 0$ или $\beta_1 = \beta_2 = \ldots = \beta_n = 0$. При формировании $(n+1)$-й, $(2n+1)$-й, ..., $[(m-1)n+1]$-й компонент линейной комбинации каждый столбец матрицы (2.7) содержит уже две единицы и результат суммирования принимает вид:

$$\alpha_2 \cdot 1 + \beta_1 \cdot 1 = 0:\ \alpha_3 \cdot 1 + \beta_1 \cdot 1 = 0,\ \ldots,\ \alpha_m \cdot 1 + \beta_1 \cdot 1 = 0.$$

Учитывая, что $\beta_1 = 0$, получаем $\alpha_2 = \alpha_3 = \ldots = \alpha_m = 0$ (аналогично проводятся рассуждения при выборе любых других $m+n-1$ строк матрицы условий). Таким образом, нулевая линейная комбинация любых $m+n-1$ строк матрицы A получается только при всех нулевых коэффициентах комбинации и, следовательно, эти строки образуют линейно независимую систему. Отсюда $r(A) = m+n-1$. Из доказанного вытекает, что базис системы векторов-столбцов матрицы A состоит из $m+n-1$ столбцов, так что невырожденный план задачи содержит $m+n-1$ положительных компонент.

Разрешимость. В отличие от общей задачи линейного программирования вопрос о существовании допустимых и оптимальных планов транспортной задачи решается очень просто.

Свойство 2. *Транспортная задача всегда имеет оптимальный план.*

Покажем, что задача, для которой удовлетворяется условие баланса (2.6), имеет допустимый план. Обозначим $\sum_i a_i = \sum_j b_j = s$. Придадим переменным x_{ij} значения

$$x_{ij} = a_i b_j / s,\ i = 1,\ \ldots,\ m,\ j = 1,\ \ldots,\ n. \qquad (2.16)$$

Тогда $x_{ij} \geqslant 0$ для всех $i = 1, \ldots, m$, $j = 1, \ldots, n$ и

$$\sum_{j=1}^{n} x_{ij} = \frac{a_i}{s} \sum_{j=1}^{n} b_j = \frac{a_i}{s} s = a_i,\ i = 1,\ \ldots,\ m;$$

$$\sum_{i=1}^{m} x_{ij} = \frac{b_j}{s} \sum_{i=1}^{m} a_i = \frac{b_j}{s} s = b_j,\ j = 1,\ \ldots,\ n.$$

Следовательно, переменные x_{ij}, удовлетворяющие всем ограничениям транспортной задачи, образуют ее допустимый план. Отсюда условие баланса (2.6) является необходимым и достаточным для допустимости

транспортной задачи. Этим, однако, еще не решается вопрос о ее разрешимости, так как не всякая допустимая задача имеет оптимальный план: если множество допустимых планов неограниченно, то целевая функция на этом множестве может быть неограниченной.

Ограниченность допустимого многогранника очевидна, так как из вида уравнений (2.3), (2.4) следует $x_{ij} \leqslant \min(a_i, b_j)$ для всех i, j. Следовательно, целевая функция тоже ограничена и транспортная задача всегда имеет оптимальный план.

Допустимые планы. Из общей теории известно, что количество положительных компонент опорного плана не превышает ранг матрицы A. Поскольку для транспортной задачи $r(A)=m+n-1$, допустимый план, заданный соотношением (2.16), не является опорным. Отыскать опорный план можно было бы, конечно, с помощью симплекс-метода. Оказывается, однако, транспортная задача обладает особенностью, упрощающей процедуру отыскания такого плана.

Свойство 3. *В транспортной задаче всегда существуют допустимые планы, содержащие не более* $m+n-1$ *положительных элементов.*

Доказательство носит конструктивный характер. Из него вытекает удобный способ построения допустимых планов, которые, как мы увидим далее, всегда будут опорными.

Доказательство свойства проведем по индукции. При $m=n=1$ допустимый план — решение системы (2.3), (2.4) — определяется сразу: $x_{11}=a_1=b_1$. Число его положительных компонент равно единице и совпадает со значением $m+n-1$. Пусть теперь $m=1$, $n>1$. Тогда искомая матрица X состоит из одной строки $(x_{11}, x_{12}, \ldots, x_{1n})$. Записывая для нее систему столбцовых уравнений (2.4), непосредственно получаем совокупность чисел $x_{11}=b_1$, $x_{12}=b_2$, ..., $x_{1n}=b_n$, удовлетворяющую условиям (2.3) — (2.5). Следовательно, это допустимый план, причем число его положительных компонент опять равно $m+n-1$. Аналогично, если $m>1$, $n=1$, то непосредственно устанавливаем допустимое решение системы (2.3), (2.4) в виде $x_{i1}=a_i$, $i=1, \ldots, m$, с числом положительных компонент, равным $m+n-1$.

Пусть теперь свойство 3 имеет место при решении всех систем вида (2.3), (2.4), если $m+n \leqslant k$. Покажем, что тогда оно выполняется и для систем, где $m+n=k+$

$+1$. Начнем решение с определения значения x_{11}. Здесь возможны три случая.

Случай 1. Если $a_1 < b_1$, то положим $x_{11} = a_1$ и запишем первую строку искомой матрицы (2.1) в виде

$$x_{11} = a_1, \; x_{12} = x_{13} = \ldots = x_{1n} = 0.$$

Тем самым оказывается удовлетворенным первое из уравнений (2.3), и вместо системы (2.3), (2.4) имеем сокращенную систему из $k = m + n - 1$ уравнений:

$$\sum_{j=1}^{n} x_{ij} = a_i, \; i = 2, \ldots, m; \quad \sum_{i=2}^{m} x_{ij} = b'_j, \; j = 1, \ldots, n,$$

где потребности b'_j таковы: $b'_1 = b_1 - a_1$; $b'_2 = b_2$; $b'_3 = b_3$, ..., $b'_n = b_n$, причем $\sum_{i=2}^{m} a_i = \sum_{j=1}^{n} b'_j$.

В новой системе на одно строчное уравнение меньше, чем в исходной. По предположению индукции, мы можем решить сокращенную систему из k уравнений, получив не более $k-1$ ненулевых значений x_{ij}. Поскольку мы уже нашли первую строку матрицы (2.1) с одним ненулевым элементом, то решение исходной системы будет содержать не более $k = m + n - 1$ отличных от нуля элементов.

Случай 2. Если $b_1 < a_1$, то положим $x_{11} = b_1$ и запишем первый столбец матрицы (2.1) в виде $x_{11} = b_1$; $x_{21} = x_{31} = \ldots = x_{n1} = 0$, а оставшийся запас у поставщика 1 определим как $a'_1 = a_1 - b_1 > 0$. Получаем сокращенную систему, имеющую на одно столбцовое уравнение меньше. Как и прежде, рассуждая по индукции, заключаем, что можем решить исходные уравнения (2.3), (2.4) с числом ненулевых компонент x_{ij}, не большим $m + n - 1$.

Случай 3. Если $a_1 = b_1$, то можно положить либо $x_{11} = a_1$, либо $x_{11} = b_1$, и считать, что имеет место первый или второй случай. Для определенности возьмем $x_{11} = a_1$. Тогда после исключения строчного уравнения новое значение спроса первого потребителя будет $b'_1 = b_1 - a_1 = 0$, т. е. в правой части появится нуль и на следующем шаге при решении сокращенной системы в матрице X элемент $x_{21} = 0$. В остальном все проведенные рассуждения сохраняются полностью. Таким образом, свойство 3 доказано.

Целочисленность планов транспортной задачи. Далее будет показано, что рассмотренный метод решения уравнений (2.3), (2.4) приводит к построению допустимых планов, в действительности являющихся опорными. С учетом этого можно отметить одну важную особенность, которой обладают опорные планы транспортной задачи: транспортная задача всегда имеет целочисленные опорные планы, если только все значения a_i, b_j — целые числа. Действительно, рассмотренный при доказательстве свойства 3 метод решения уравнений (2.3), (2.4) не требовал никаких других операций, кроме операций сравнения, сложения и вычитания, которые не могут привести к дробным величинам при целых значениях исходных данных. Что касается значений запасов a_i и потребностей b_j, то они, как правило, задаются целыми числами; в противном случае они могут быть выражены в целых числах умножением на 10 в соответствующей степени.

Поскольку оптимальный план находится среди опорных планов задачи, можно предположить, что свойство целочисленности распространяется и на оптимальные решения. Действительно, имеет место следующее свойство, которое мы приведем без доказательства.

Свойство 4. *Если в транспортной задаче все числа a_i, b_j целые, то она имеет оптимальный целочисленный план.*

Это свойство важно прежде всего с вычислительной точки зрения, так как позволяет вести программирование в целых числах, что экономит память машины, повышает ее быстродействие при обработке таких программ и исключает ошибки округления.

2.3. ОПОРНЫЕ ПЛАНЫ

Построение опорных планов. Из доказательства свойства 3 транспортной задачи вытекает несложный способ построения допустимых планов, имеющих не более $m+n-1$ положительных компонент. Он называется *методом северо-западного угла.* Вначале дадим его описание, а затем докажем опорность получаемых с его помощью планов.

Будем строить допустимый план в таблице перевозок, имеющей вид табл. 2.2, где каждой переменной x_{ij} сопоставлена соответствующая клетка (i, j). Вначале

полагаем все переменные x_{ij} равными нулю. После того как определено значение какой-либо переменной x_{ij}, оно заносится в соответствующую клетку (i, j), которая будет называться *занятой*. Для начала вычислений выбираем одну из переменных и переходим к шагу 1 алгоритма.

1. Выбранной переменной x_{ij} присваиваем значение $x_{ij}=\min(a_i, b_j)$, где a_i и b_j — текущие значения запаса и спроса.

2. Если $x_{ij}=a_i<b_j$, так что удовлетворяется i-е строчное уравнение, то исключаем из дальнейшего рассмотрения i-ю строку и устанавливаем текущее значение b_j равным b_j-x_{ij}. Переходим к шагу 5.

3. Если $x_{ij}=b_j<a_i$, так что удовлетворяется j-е столбцовое ограничение, исключаем из дальнейшего рассмотрения j-й столбец и текущее значение a_i определяем как a_i-x_{ij}. Переходим к шагу 5.

4. Если $x_{ij}=a_i=b_j$, так что удовлетворяются оба ограничения, полагаем, что данным значением x_{ij} удовлетворяется только одно из них, и исключаем соответствующую ему строку (столбец). В другом ограничении выбираем одну из переменных, не исключенных ранее, полагаем ее равной нулю и также исключаем соответствующий этому ограничению столбец (строку). Переходим к шагу 5.

5. Если присвоены значения менее чем $m+n-1$ переменным, выбираем одну из неисключенных переменных и возвращаемся к шагу 1. Если определены значения $m+n-1$ переменных, алгоритм закончен.

С учетом того, что ранг матрицы A равен $m+n-1$, полученный допустимый план будет невырожден, если все его $m+n-1$ компонент положительны, и вырожден в противном случае. Отметим, что для выбора переменной x_{ij} на шаге 5 могут быть использованы различные способы. Если алгоритм начинается с верхней левой клетки и всякий раз после исключения столбца (строки) в оставшейся подматрице снова берется верхняя левая клетка, имеем собственно метод северо-западного угла, давший название всей процедуре. Выбирая для начала верхнюю правую клетку и поступая так же при рассмотрении каждой новой подматрицы, получим аналогичную процедуру, называемую *методом северо-восточного угла*. В первом случае процесс развивается по главной диагонали таблицы от клетки (1, 1) к клетке (m, n), во

втором — по побочной диагонали от клетки $(1, n)$ к клетке $(m, 1)$. Эти методы называют также *диагональными*.

Существуют методы, связывающие выбор переменной x_{ij} с величиной издержек c_{ij}. Так, в *методе минимального элемента* в начале процедуры и затем на шаге 5 выбирается та из неисключенных переменных x_{ij}, которой соответствует наименьший коэффициент c_{ij}. Его разновидностями являются *метод минимума по строке* и *метод минимума по столбцу*. Здесь минимальный коэффициент c_{ij} выбирается не из всей матрицы или очередной подматрицы, а в первой из невычеркнутых строк или в первом из невычеркнутых столбцов. Можно было бы ожидать, что в смысле близости к оптимуму допустимые планы, построенные с учетом затрат, будут лучше планов, полученных диагональными методами. Однако на практике это не всегда так, поэтому в дальнейшем будем использовать самый простой вариант общей процедуры — метод северо-западного угла.

Теорема 2.2. *План $X=(x_{ij})$, построенный по методу северо-западного угла, является опорным.*

◆ Напомним, что план $X=(x_{ij})$ является опорным, если соответствующая его положительным компонентам система векторов-столбцов $\mathbf{a}^{ij}$ матрицы A линейно независима. Поэтому, согласно теореме 2.1, достаточно показать, что никакое подмножество клеток (i, j), взятое из совокупности $m+n-1$ клеток, заполненных методом северо-западного угла, не образует цикл. Доказательство проведем от противного.

Предположим, что в совокупности занятых клеток можно выбрать последовательность $(i_1, j_1), \ldots, (i_t, j_t)$, образующую цикл. Пусть (i_k, j_k) — клетка, заполненная раньше всех остальных клеток этого цикла. Допустим, что из дальнейшего рассмотрения была исключена строка i_k. Тогда никакая другая клетка рассматриваемой последовательности не может быть расположена в этой строке. Если был исключен столбец j_k, то в последовательности не может быть ни одной другой клетки из этого столбца. И в том и в другом случаях из клеток рассматриваемой последовательности нельзя построить цикл. ◆

Так как переменные x_{ij}, имеющие в опорном плане отличные от нуля значения, называются базисными, то совокупность занятых клеток, полученных методом северо-западного угла, отвечает совокупности базисных

переменных. Вместе с тем каждая занятая клетка (i, j) соответствует одному из линейно независимых векторов a^{ij} базиса данного опорного плана. Поэтому множество $B=\{(i, j\}$, содержащее все двойные индексы, отвечающие занятым клеткам, можно в зависимости от обстоятельств считать множеством либо всех базисных переменных, либо всех векторов базиса. И в том и в другом случае его удобно называть просто базисом.

Пример 2.1. Найти методом северо-западного угла опорный план задачи, где $m=3$, $n=4$, а запасы и спрос заданы векторами (7, 9, 13) и (10, 7, 4, 8).

Таблица 2.3

i \ j	1	2	3	4	a_i
1	x_{11}	x_{12}	x_{13}	x_{14}	7
2	x_{21}	x_{22}	x_{23}	x_{24}	9
3	x_{31}	x_{32}	x_{33}	x_{34}	13
bi	10	7	4	8	

Составим систему уравнений (2.3), (2.4), которая для данного случая примет вид табл. 2.3. Уравнениям (2.3) отвечают строки этой таблицы, а уравнениям (2.4) — столбцы. Сравнивая $a_1=7$ и $b_1=10$, полагаем $x_{11}=a_1=7$, исключаем первую строку и получаем в табл. 2.4 сокращенную систему, где b_1 уменьшено до $b_1=10-7=3$.

Таблица 2.4

i \ j	1	2	3	4	a_i
2	x_{21}	x_{22}	x_{23}	x_{24}	9
3	x_{31}	x_{32}	x_{33}	x_{34}	13
bi	3	7	4	8	

Для определения значения x_{21} сравниваем в табл. 2.4 $a_2=9$ и $b_1=3$ и полагаем $x_{21}=3$. Исключаем первый столбец табл. 2.4, уменьшаем $a_2=9$ до $a_2=9-3=6$ и получаем сокращенную систему в виде табл. 2.5.

Таблица 2.5

i \ j	2	3	4	a_i
2	x_{22}	x_{23}	x_{24}	6
3	x_{32}	x_{33}	x_{34}	13
b_j	7	4	8	

Положив $x_{22}=6$ и исключив первую строку, получим табл. 2.6, откуда непосредственно следует $x_{32}=1$, $x_{33}=4$, $x_{34}=8$.

Таблица 2.6

i \ j	2	3	4	a_i
3	x_{32}	x_{33}	x_{34}	13
b_j	1	4	8	

Таким образом, получен допустимый план, представленный в табл. 2.7, где пустые клетки соответствуют переменным $x_{ij}=0$. Практически решение в виде табл. 2.7 строится без промежуточных табл. 2.3—2.6, которые здесь приведены для пояснения метода.

Таблица 2.7

i \ j	1	2	3	4	a_i
1	7				7
2	3	6			9
3		1	4	8	13
b_j	10	7	4	8	

Для того чтобы при $m=3$, $n=4$ опорный план был невырожденным, необходимо, чтобы число его ненулевых компонент было равно $m+n-1=6$. План, представленный в табл. 2.7, имеет шесть отличных от нуля компонент и, следовательно, является невырожденным.

Базис данного опорного плана может быть задан в виде $B = \{a^{11}, a^{21}, a^{22}, a^{32}, a^{33}, a^{34}\}$ как совокупность составляющих его векторов-столбцов матрицы A или в виде $B = \{(1,1), (2,1), (2,2), (3,2), (3,3), (3,4)\}$ как множество двойных индексов базисных переменных — занятых клеток табл. 2.7.

Таблица 2.8

i \ j	1	2	3	4	a_i
1				7	7
2		4	4	1	9
3	10	3			13
b_j	10	7	4	8	

Другой вариант опорного плана показан в табл. 2.8. Он получен методом северо-восточного угла, при котором построение начинается с переменной x_{14}, лежащей в верхнем правом углу матрицы, и продолжается влево, вниз вдоль побочной диагонали. Если из каких-либо соображений желателен допустимый план, включающий отличную от нуля перевозку между определенной парой пунктов i и j, процесс следует начать с переменной x_{ij}. Пусть, например, для переменной x_{12}, которая в обоих вариантах допустимого плана равна нулю, требуется обеспечить условие $x_{12} > 0$. Начнем заполнение табл. 2.9 с клетки (1,2). Поскольку $a_1 = b_2 = 7$, полагаем $x_{12} = a_1 = 7$, а $b_2' = b_2 - a_1 = 0$ и исключаем первую строку.

Таблица 2.9

i \ j	1	2	3	4	a_i
1		7			7
2		0	4	5	9
3	10			3	13
b_j	10	7	4	8	

Далее перевозкой $x_{22} = 0$ удовлетворяем спрос $b_2' = 0$ второго потребителя и исключаем второй столбец. Затем определяем элемент $x_{23} = \min(a_2 = 9,\ b_3 = 4) = 4$ и удаляем столбец 3, сохранив строку 2 при величине запаса $a_2' = 9 - 4 = 5$. Продолжая аналогич-

но, получаем $x_{24}=5$, $x_{31}=10$, $x_{34}=3$. Очевидно, допустимый план в табл. 2.9 вырожденный, так как число его положительных компонент, равное 5, меньше, чем $m+n-1=6$.

Разложение столбцов матрицы A по базису опорного плана. План, полученный методом северо-западного угла, является опорным по построению. План, содержащий более $m+n-1$ положительных компонент, не является опорным в силу свойства 1 транспортной задачи. Как проверить, является ли опорным произвольный допустимый план, имеющий не более $m+n-1$ положительных компонент? Объединяя результаты теорем 2.1 и 2.2, получим простой признак опорности допустимого плана: *план является опорным, если из совокупности занятых клеток таблицы перевозок нельзя выделить последовательность, образующую цикл.*

Таким образом, если в совокупности занятых клеток найдется последовательность, образующая цикл, то X — неопорный план. В противном случае X — опорный план, и система векторов-столбцов матрицы A, отвечающих совокупности занятых клеток таблицы перевозок, образует линейно независимую систему — базис опорного плана X. Любой другой вектор $\mathbf{a}^{ij}$ может быть разложен по векторам этого базиса следующим образом. В соответствии с теоремой 2.1 надо присоединить вектор $\mathbf{a}^{ij}$ к базису, найти цикл, замыкающийся на клетке (i, j), и взять коэффициенты разложения в соответствии с уравнениями (2.14). Правило выбора коэффициентов удобно описать, пронумеровав последовательно все клетки цикла так, что первый номер присвоен клетке (i, j). Тогда в соответствии с формулой (2.14) коэффициенты разложения некоторого столбца матрицы A по векторам-столбцам базиса равны $+1$ при векторах, соответствующих четным клеткам, равны -1 при векторах, соответствующих нечетным клеткам цикла, и равны нулю при всех остальных векторах.

Применительно к операциям симплекс-метода это правило представляет способ определения коэффициентов матрицы при переходе от одного базиса к другому. Таким образом, при всех преобразованиях матрицы A транспортной задачи ее коэффициентами являются только числа 0, $+1$ или -1.

Непосредственной проверкой легко убедиться, что планы, представленные в табл. 2.7—2.9, удовлетворяют приведенному признаку: ни в одной из таблиц нельзя

построить цикл, проходящий по занятым клеткам. Следовательно, эти планы опорные. Полезно, однако, иметь способ, позволяющий формальным образом устанавливать наличие циклов на выделенном множестве клеток.

Для построения такого способа можно воспользоваться тем, что любая клетка (i, j), входящая в цикл, должна иметь соседнюю клетку как в той же строке i, так и в том же столбце j. Поэтому если в таблице найдется столбец (строка), имеющий менее двух занятых клеток, то такой столбец (строку) можно исключить из таблицы, не разрушая цикла, если он есть. Отсюда получаем следующий способ выделения цикла, называемый *методом вычеркивания*.

Просматриваем все строки таблицы и вычеркиваем те из них, которые содержат менее двух занятых клеток. В оставшейся подматрице просматриваем все столбцы, производим вычеркивание по тому же правилу и переходим к новой подматрице. В ней вновь просматриваем и вычеркиваем, если возможно, строки и т. д. Процесс вычеркивания заканчивается одним из двух случаев: 1) все строки и столбцы таблицы вычеркнуты; 2) осталась подматрица, в каждой строке и каждом столбце которой имеются по крайней мере две занятые клетки. В первом случае из занятых клеток исходной таблицы нельзя выделить последовательность, образующую цикл, во втором не вычеркнутые занятые клетки образуют цикл.

Пример 2.2. В табл. 2.10 и 2.11 представлены два плана одной и той же задачи. Проверить, являются ли они опорными.

Таблица 20.1

i \ j	1	2	3	4	a_i
1		15	5		30
2	15			5	20
3			20		20
4			10	20	30
b_i	15	15	45	25	

Таблица 2.11

i \ j	1	2	3	4	a_i
1		15	15		30
2	10			10	20
3			20		20
4	5		10	15	30
b_j	15	15	45	25	

Проверка баланса по строкам и столбцам показывает, что в обоих планах весь запас каждого поставщика вывозится, весь спрос каждого потребителя удовлетворяется. Другими словами, строчные и столбцовые уравнения системы (2.3), (2.4) удовлетворяются. Следовательно, оба плана допустимы.

Применим метод вычеркивания к плану, приведенному в табл. 2.10. При первом просмотре вычеркиваем строку 3, при втором — столбцы 1 и 2, при третьем — строки 1 и 2, при четвертом — столбцы 3 и 4. В результате просмотра все строки и столбцы вычеркнуты, следовательно, данный план опорный. Применяя те же операции к плану, содержащемуся в табл. 2.11, при первом просмотре вычеркиваем строку 3, при втором — столбец 2, при третьем — строку 1, при четвертом — столбец 3. Больше вычеркиваний сделать нельзя, так как нет строк или столбцов, содержащих менее двух занятых клеток. Оставшиеся невычеркнутыми занятые клетки (2,1), (2,4), (4,4), (4,1) образуют цикл. Следовательно, данный допустимый план не является опорным, а векторы a^{21}, a^{24}, a^{44}, a^{41} образуют линейно зависимую систему. В этом легко убедиться, составив из них линейную комбинацию с чередующимися коэффициентами $+1$ и -1 и представив для наглядности каждый вектор a^{ij} в виде суммы соответствующих векторов-ортов e^i и e^{m+j}. Получаем

$$a^{21}-a^{24}+a^{44}-a^{41}=(e^2+e^5)-(e^2+e^8)+(e^4+e^8)-(e^4+e^5)=0.$$

Поскольку линейная комбинация получена при коэффициентах, отличных от нуля, составляющие ее векторы-столбцы a^{21}, a^{24}, a^{44}, a^{41} действительно линейно зависимы.

2.4. МЕТОД ПОТЕНЦИАЛОВ

Двойственная задача. Решение транспортной задачи симплекс-методом, как и любой задачи линейного программирования, состоит из двух этапов. На первом этапе отыскивается некоторый начальный опорный план, на втором осуществляется итерационный процесс его улучшения. Содержанием каждой итерации является проверка имеющегося плана на оптимальность, и в слу-

чае его неоптимальности переход к новому опорному плану с меньшим значением целевой функции. Мы видели, что специфика транспортной задачи приводит к существенному упрощению первого этапа, а именно: если в общей задаче линейного программирования построение начального опорного плана выполняется с помощью той же процедуры симплекс-метода, которая применяется и на втором этапе, то в транспортной задаче опорные планы строятся элементарными способами, например методом северо-западного угла. Для этапа проверки оптимальности и перехода к новым опорным планам в транспортной задаче также разработан целый ряд алгоритмов, более простых и удобных по сравнению с общей процедурой симплекс-метода, а иногда и вообще не связанных с нею. Излагаемый здесь метод потенциалов является разновидностью модифицированного симплекс-метода, приспособленного к особенностям транспортной задачи.

Если рассматривать задачу (2.8) как прямую, то в соответствии с теорией двойственности ей можно сопоставить следующую двойственную задачу:

$$\max w = \mathbf{yb};\ \mathbf{y}A \leqslant \mathbf{c}. \tag{2.17}$$

Отметим, что компоненты вектора $\mathbf{y}$ не ограничены по знаку, потому что прямая задача (2.8) каноническая. Неравенства двойственной задачи становятся нагляднее, если представить двойственный вектор $\mathbf{y}$ в виде $\mathbf{y}=(u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n)$, где первые m компонент u_i соответствуют строчным уравнениям (2.3), последующие n компонент v_j — столбцовым уравнениям (2.4). Число u_i принято называть *потенциалом поставщика* i, число v_j — *потенциалом потребителя* j. Если в задаче (2.17) записать векторы $\mathbf{y}$, $\mathbf{b}$, $\mathbf{c}$ и матрицу A покомпонентно, то с учетом того, что каждый столбец матрицы A содержит всего две единицы, одна из которых соответствует строчному, другая — столбцовому уравнению прямой задачи, получим двойственную задачу в виде

$$\max w = \sum_{i}^{m} a_i u_i + \sum_{j}^{n} b_j v_j; \tag{2.18}$$

$$u_i + v_j \leqslant c_{ij},\ i=1, \ldots, m,\ j=1, \ldots, n. \tag{2.19}$$

Например, для задачи (2.10) двойственной будет задача

$$\max w = a_1u_1 + a_2u_2 + b_1v_1 + b_2v_2 + b_3v_3;$$
$$u_1 + v_1 \leqslant c_{11}; \quad u_2 + v_1 \leqslant c_{21};$$
$$u_1 + v_2 \leqslant c_{12}; \quad u_2 + v_2 \leqslant c_{22};$$
$$u_1 + v_3 \leqslant c_{13}: \quad u_2 + v_3 \leqslant c_{23}.$$

Из постановки двойственной задачи (2.18), (2.19) видно, что увеличение потенциалов u_i и v_j приводит к возрастанию целевой функции (2.18), так как по предположению $a_i > 0$, $b_j > 0$ для всех i, j. Однако неравенства (2.19) ограничивают рост целевой функции. Согласно им, сумма потенциалов поставщика и потребителя не должна превышать расходы на перевозку между ними единицы продукта. Более строго, из второй теоремы двойственности следует, что оптимальный план прямой задачи может включать ненулевое значение перевозки x_{ij} только в том случае, если сумма потенциалов поставщика i и потребителя j равна величине расходов на перевозку между ними единицы продукта. Применяя этот результат к паре задач (2.2) — (2.5) и (2.18), (2.19), получаем следующий признак оптимальности плана транспортной задачи: *если план $X = (x_{ij})$ оптимален, то ему соответствует система из $m+n$ чисел u_i и v_j, удовлетворяющих условиям:*

$$u_i + v_j = c_{ij} \text{ для } x_{ij} > 0; \tag{2.20}$$
$$u_i + v_j \leqslant c_{ij} \text{ для } x_{ij} = 0. \tag{2.21}$$

Введем для переменных x_{ij} оценки

$$\Delta_{ij} = u_i + v_j - c_{ij}, \tag{2.22}$$

которые являются приведенными коэффициентами целевой функции, эквивалентными оценкам $\Delta_j = z_j - c_j$ симплекс-метода. Тогда из (2.20), (2.21) получаем признак оптимальности в следующем виде: *план $X = (x_{ij})$ оптимален, если все его оценки Δ_{ij} вида (2.22) неположительны.*

Проверка плана на оптимальность. Пусть $X = (x_{ij})$ — невырожденный опорный план с $m + n - 1$ положительными компонентами x_{ij} и базисом $B = \{(i, j)\}$, который задан множеством двойных индексов при этих переменных. В соответствии с признаком оптимальности для проверки данного плана на оптимальность необходимо построить систему из $m + n$ двойственных переменных u_i, v_j, $i =$

$= 1, \ldots, m, j = 1, \ldots, n$, удовлетворяющую условию (2.20) для всех базисных переменных $(i, j) \in B$, и проверить выполнение условия (2.21) для всех остальных $(i, j) \overline{\in} B$.

При проверке условия (2.21) для небазисных переменных могут встретиться два случая: 1) $u_i + v_j \leqslant c_{ij}$ для всех $(i, j) \overline{\in} B$; 2) $u_k + v_l > c_{kl}$ для некоторых $(k, l) \in B$.

В первом случае все переменные u_i, v_j удовлетворяют условиям оптимальности (2.20), (2.21), следовательно, план X оптимален.

Во втором случае условия оптимальности не выполняются и целевая функция прямой задачи может быть уменьшена введением одной из переменных x_{kl} в базис.

Рассмотрим теперь, как строится система двойственных переменных. Поскольку она должна удовлетворять условию (2.20), то искомые потенциалы определятся из решения системы

$$u_i + v_j = c_{ij}, \ (i, j) \in B \tag{2.23}$$

с $m+n-1$ уравнениями и $m+n$ переменными, где c_{ij} — коэффициенты транспортных расходов для занятых клеток.

Поскольку в данной неопределенной системе число уравнений на единицу меньше, чем число неизвестных, значение одной из переменных может быть выбрано произвольно. Положим для определенности $u_1 = 0$. После этого немедленно определяется значение той переменной v_j, которая входит в то же уравнение, что и u_1. Далее процесс продолжается аналогично и все переменные u_i, v_j определяются последовательно одна за другой.

Пример 2.3. Проверить оптимальность опорного плана, представленного в табл. 2.7, если матрица транспортных расходов

$$C = \begin{bmatrix} 5 & 14 & 7 & 8 \\ 9 & 4 & 3 & 3 \\ 8 & 18 & 5 & 10 \end{bmatrix}.$$

Перенесем опорный план из табл. 2.7 в табл. 2.12, где в верхнем правом углу каждой клетки (i, j) помещен соответствующий коэффициент c_{ij}, а компоненты опорного плана взяты в рамку.

Базис данного опорного плана, заданный индексами занятых клеток, имеет вид $B = \{(1,1), (2,1), (2,2), (3,2), (3,3) (3,4)\}$. Составляем соответствующую опорному плану систему уравнений вида (2.23): $u_1 + v_1 = 5$; $u_2 + v_1 = 9$; $u_2 + v_2 = 4$; $u_3 + v_2 = 18$; $u_3 + v_3 = 5$; $u_3 + v_4 = 10$. Имеем неопределенную систему с шестью уравнениями и семью неизвестными. Произвольно полагаем $u_1 = 0$ и последователь-

Таблица 2.12

i \ j	1	2	3	4	a_i	u_i
1	[5] 7	[14] −14	[7] −20	[8] −16	7	0
2	[9] 3	[4] 6	[3] −12	[3] −7	9	4
3	[8] 15 +	[18] 1	[5] 4	[10] 8	13	18
b_j	10	7	4	8		
v_j	5	0	−13	−8		

ной подстановкой определяем значения остальных неизвестных: $v_1=5$; $u_2=4$; $v_2=0$; $u_3=18$; $v_3=-13$; $v_4=-8$. Значения потенциалов u_i, v_j представлены в последнем столбце и нижней строке табл. 2.12. При ручном счете их обычно получают прямо по таблице, не выписывая в явном виде уравнения (2.23).

Итак, система потенциалов построена. Условие (2.20) признака оптимальности для базисных переменных удовлетворяется по построению. Осталось проверить выполнение признака оптимальности для небазисных переменных x_{12}, x_{13}, x_{14}, x_{23}, x_{24}, x_{31} или для клеток (1,2), (1,3), (1,4), (2,3), (2,4), (3,1). В табл. 2.12 в нижнем левом углу каждой небазисной клетки записана оценка Δ_{ij}, вычисленная по формуле (2.22). Покажем, например, как вычислена оценка Δ_{23}: $u_2+v_3=4+(-13)=-9$, $\Delta_{23}=(u_2+v_3)-c_{23}=-9-3=-12$. Вычисления остальных оценок выполнены непосредственно в таблице.

Анализ коэффициентов Δ_{ij}, $(i, j)\in \overline{B}$, показывает, что данный опорный план неоптимален, так как среди оценок небазисных переменных имеется положительная оценка $\Delta_{31}=15$ (отмечена плюсом). Следовательно, введением переменной x_{31} в базис план может быть улучшен. Отметим, что значение целевой функции для рассматриваемого опорного плана составляет величину

$$z=\sum_{(i,\ j)\in B} c_{ij}x_{ij}=5\cdot 7+9\cdot 3+4\cdot 6+18\cdot 1+5\cdot 4+10\cdot 8=204.$$

То же значение имеет и целевая функция двойственной задачи:

$$w=\sum_{i=1}^{m} a_iu_i+\sum_{j=1}^{n} b_jv_j=7\cdot 0+9\cdot 4+13\cdot 18+10\cdot 5+7\cdot 0+$$

$$+4(-13)+8(-8)=36+234+50-52-64=204.$$

Построение нового опорного плана. Пусть имеется опорный план $X=(x_{ij})$ транспортной задачи, для которого определены все приведенные коэффициенты целевой функции Δ_{ij} вида (2.22). Рассмотрим операции симплекс-метода, выполняемые при оценке оптимальности данного плана и переходе в случае необходимости к новому опорному плану. Поскольку неограниченность целевой функции в транспортной задаче исключена, остается рассмотреть случаи 1 и 3 симплекс-метода (см. § 1.3).

Случай 1. Если каждый коэффициент Δ_{ij} неположителен, опорный план оптимален. Напомним, что транспортная задача решается на минимум целевой функции, отсюда и переход от признака оптимальности $\Delta_j \geqslant 0$ к условию $\Delta_{ij} \leqslant 0$.

Случай 3. Если некоторые из коэффициентов Δ_{ij} положительны, необходимо выбрать ведущий элемент и выполнить шаг преобразования матрицы A.

Выберем столбец (i, j), для которого коэффициент Δ_{ij} максимален среди всех $\Delta_{ij}>0$. Переменная x_{ij} войдет в базис. Для определения выводимой из базиса переменной в общей схеме симплекс-метода вычисляются отношения правых частей к положительным коэффициентам столбца $\mathbf{a}^{ij}$. Выводится из базиса переменная, соответствующая строке, в которой это отношение минимально. Поскольку рассматривается случай 3, то в столбце $\mathbf{a}^{ij}$ обязательно имеется хотя бы один положительный элемент. Как мы знаем, в процессе преобразования элементы матрицы A принимают только одно из трех значений: 0, $+1$ или -1. Следовательно, положительными элементами столбца $\mathbf{a}^{ij}$ могут быть лишь элементы $+1$. Отсюда выбор переменной, удаляемой из базиса, становится очень простым: в разложении (2.14) столбца $\mathbf{a}^{ij}$ по векторам базиса надо просмотреть векторы, входящие с коэффициентом $+1$, и вывести вектор, для которого значение базисной переменной минимально.

В таблице перевозок описанные операции выполняются путем построения цикла, который начинается в клетке (i, j), отвечающей вводимой в базис переменной, и проходит через базисные клетки. В соответствии с правилом разложения вектора по базису приписываем вводимой клетке (i, j) номер один и затем последовательно присваиваем номера всем клеткам цикла. Векторы базиса, соответствующие четным клеткам цикла, входят в разложе-

ние с коэффициентами $+1$, нечетным — с коэффициентами -1. Следовательно, из базиса удаляется переменная, допустим x_{kl}, соответствующая четной клетке цикла, в которой стоит величина перевозки $x_{kl}=\theta$, минимальная по сравнению со всеми остальными перевозками в четных клетках. Найдя такую клетку (k, l), преобразование опорного плана выполняем следующим образом. В клетку (i, j), соответствующую вводимой в базис переменной, заносим перевозку θ; далее, двигаясь по циклу, уменьшаем перевозки на величину θ в четных клетках и увеличиваем на величину θ в нечетных. Клетку (k, l), соответствующую выводимой из базиса переменной x_{kl}, оставляем незанятой.

Если минимальная перевозка θ встречается в нескольких четных клетках, то незанятой оставляем только одну, а в остальных записываем нули. Это соответствует случаю вырожденного опорного плана. При появлении вырожденных планов могут встретиться итерации, когда из базиса выводится переменная с перевозкой $x_{kl}=\theta=0$, так что в результате такого преобразования целевая функция не улучшается. Теоретически в случае вырожденных планов не исключено и зацикливание, но на практике оно не встречается. Описанные преобразования повторяются до тех пор, пока не встретится случай 1, т. е. не будет выполнен признак оптимальности.

Пример 2.4. Проведем преобразования опорного плана, представленного в табл. 2.12, до получения оптимального решения. Как было показано, среди приведенных коэффициентов целевой функции имеется положительный $\Delta_{31}=15$. Цикл, возникающий после добавления клетки (3.1) к системе базисных клеток рассматриваемого опорного плана, показан в табл. 2.12 штриховыми линиями.

Начиная обход цикла с вводимой в базис клетки (3,1), которая получает номер один, определяем четные клетки (3,2) и (2,1) и находим $\Theta=\min\{x_{32}, x_{21}\}=\min\{1, 3\}=1$. Обходя цикл, прибавляем $\Theta=1$ к значениям переменных в нечетных клетках (3,1) и (2,2) и вычитаем из элементов четных клеток (3,2) и (2,1). Преобразованный опорный план представлен в табл. 2.13. Связанные с ним транспортные расходы составляют величину $z=5\cdot 7+9\cdot 2+8\cdot 1+4\cdot 7+5\cdot 4+10\cdot 8=35+18+8+28+20+80=189$. Расчет потенциалов u_i, v_j и оценок Δ_{ij} небазисных клеток выполняем прямо в таблице. План неоптимален, так как имеются две положительные оценки: $\Delta_{23}=3$, $\Delta_{24}=8$. С учетом того, что $\Delta_{24}>\Delta_{23}$, вводим в базис переменную x_{24}. Цикл пересчета показан в табл. 2.13 штриховыми линиями. В четных клетках находятся перевозки $x_{21}=2$, $x_{34}=8$, следовательно, $\Theta=2$. Выполнив перераспределение перевозок по этому циклу, получаем опорный план, представленный в табл. 2.14. Этот план оптимален, так как все его оценки $\Delta_{ij}\leqslant 0$. Соответствующее ему значение целевой функции $z=5\cdot 7+8\cdot 3+4\cdot 7+5\cdot 4+3\cdot 2+10\cdot 6=35+24+28+20+6+60=173$.

Таблица 2.1

i \ j	1	2	3	4	a_i	u_i
1	[7] 5	[14] −14	[7] −5	[8] −1	7	0
2	[2] 9	[7] 4	3 [3] 8	+ [3]	9	4
3	[1] 8	−15 18	[4] 5	[8] 10	13	3
b_j	10	7	4	8		
v_j	5	0	2	7		

Таблица 2.

i \ j	1	2	3	4	a_i	u_i
1	[7] 5	−6 14	−5 7	−1 8	7	0
2	−8 9	[7] 4	−5 3	[2] 3	9	−4
3	[3] 8	−7 18	[4] 5	[6] 10	13	3
b_j	10	7	4	8		
v_j	5	8	2	7		

Вычислительная схема. Рассмотрим последовательность действий, выполняемых при решении транспортной задачи методом потенциалов. Начальный опорный план считаем известным — он всегда может быть найден, например, методом северо-западного угла.

1. Из решения системы (2.23) находим значения двойственных переменных u_i, v_j, $i=1, \ldots, m$, $j=1, \ldots, n$.

2. По формуле (2.22) вычисляем оценки Δ_{ij} для всех переменных x_{ij}, $i=1, ..., m$, $j=1, ..., n$. Если все оценки неположительны, план оптимален. Если среди чисел Δ_{ij} имеются положительные, переходим к шагу 3.

3. Пусть наибольшая оценка отвечает небазисной переменной x_{kl}. Тогда в таблице перевозок строим цикл, включающий клетку (k, l), и определяем число θ — наименьшее из всех чисел, стоящих в четных клетках цикла. Это число заносим в клетку (k, l), далее прибавляем его ко всем элементам в нечетных клетках и вычитаем из всех чисел, стоящих в четных клетках. Все остальные элементы таблицы перевозок оставляем без изменения. С новым опорным планом возвращаемся к шагу 1.

Рассмотренный метод представляет собой вариант симплекс-метода, в котором отсутствует случай 2. Следовательно, он позволяет отыскать оптимальный план за конечное число шагов. При использовании данного метода зацикливание никогда не встречается на практике, поэтому специальные приемы для его предотвращения не требуются.

2.5. ЗАДАЧИ О МАКСИМАЛЬНОМ ПОТОКЕ

Задача о максимальном потоке в матричной постановке. При рассмотрении транспортной задачи считалось, что каждый поставщик связан с каждым потребителем. Предположим, что каждый поставщик связан лишь с некоторыми потребителями. Тогда условие баланса уже недостаточно для разрешимости задачи в форме (2.2) — (2.5). Рассмотрим, например, рис. 2.2, где показаны два поставщика и три потребителя, связанные четырьмя коммуникациями, причем суммарный запас равен суммарному спросу и составляет 60 единиц продукта. Хотя условие баланса выполняется, при данном составе коммуникаций общий объем перевозимого продукта не может превысить 50 единиц. Следовательно, в подобных случаях имеет место самостоятельная задача определения максимального объема продукта (максимального потока), который может быть перевезен по всем коммуника-

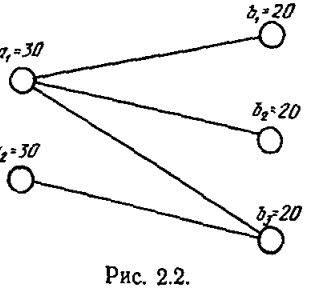

Рис. 2.2.

циям за данное время. Ее математическая постановка очевидна:

$$\left.\begin{aligned} &\max z = \sum_i \sum_j x_{ij}; \\ &\sum_j x_{ij} \leqslant a_i,\ i = 1,\ \ldots.\ m; \\ &\sum_i x_{ij} \leqslant b_j,\ j = 1,\ \ldots,\ n; \\ &x_{ij} \geqslant 0,\ (i,\ j) \in U. \end{aligned}\right\} \qquad (2.24)$$

Здесь x_{ij} — искомая величина перевозки по коммуникации $(i,\ j)$; U — множество всех имеющихся коммуникаций. Как и в транспортной задаче (2.2) — (2.5), искомые переменные можно записать в виде матрицы $X = (x_{ij})$. Поэтому задача (2.24) называется *задачей о максимальном потоке в матричной постановке.*

С практической точки зрения большее значение имеет задача о максимальном потоке в *сетевой* постановке. Для упрощения ее описания и изучения полезно обратиться к графическому представлению условий задачи. Будем изображать пункты-поставщики и пункты-отправители точками, а соединяющие их коммуникации — линиями. Поскольку для объектов такого рода в математике разработана соответствующая терминология и создана теория, называемая теорией графов, ознакомимся с основными ее понятиями.

Графы и сети. *Графом* называется совокупность точек *(вершин)* и соединяющих их линий *(ребер)*. Будем обозначать множество вершин через I, ребер — через U и граф — через $(I,\ U)$. Поскольку каждое ребро соединяет две вершины, элементы $u \in U$ удобно задавать указанием соответствующей пары $(i,\ j)$ элементов множества I. Если вершины i и j связаны ребром $u = (i,\ j)$, то говорят, что ребро u *инцидентно* вершинам i и j, которые в свою очередь инцидентны ребру u и *смежны* между собой. Два ребра, имеющие общую вершину, также называются *смежными.*

Цепью в графе называется такая последовательность ребер, в которой конец одного ребра служит началом другого и каждое ребро встречается не более одного раза. Цепь может быть задана также перечислением вершин, через которые она проходит. Замкнутая цепь называется *циклом*; цепь, у которой все вершины различны, называется *простой.* Граф будет *связным*, если любая пара его

вершин соединена цепью, и *несвязным* в противном случае. Для большей ясности отметим, что наши определения относятся к *неориентированным* графам, т. е. к графам, где существенно наличие связи между вершинами i и j и несущественна ее ориентация. В этом случае ребро, инцидентное вершинам i и j, может быть задано в виде или $u=(i, j)$ или $u=(j, i)$. В тех случаях, когда для ребра $u=(i, j)$ существенным является порядок расположения его концов, говорят, что ребро *и ориентировано*. Ориентированные ребра называются *дугами* и при графическом изображении снабжаются стрелками, идущими от начала к концу. Ориентированную цепь называют *путем*.

Граф, элементам которого поставлены в соответствие некоторые характеристики или параметры, назовем *сетью*, а сами характеристики — *функциями*. Функции могут быть заданы на таких элементах, как вершины, дуги, подмножества вершин и дуг.

На вершинах сети определим функцию *интенсивности*. Это такая функция, которая каждой вершине $i \in I$ сопоставляет некоторое число d_i. Те вершины, для которых $d_i > 0$, называются *источниками*, вершины, для которых $d_i < 0$, — *стоками*, остальные будут *нейтральными*. Так, на транспортной сети источниками являются пункты-поставщики, стоками — пункты-получатели, нейтральными — пункты, где отсутствует как производство, так и потребление рассматриваемого продукта.

На дугах могут быть заданы различные функции, одна из них — пропускная способность. Функция пропускной способности ставит в соответствие каждой дуге (i, j) графа (I, U) неотрицательное целое число r_{ij}, называемое *пропускной способностью дуги*. В транспортных сетях пропускная способность дуги означает максимальное количество продукта, которое соответствующая коммуникация может пропустить за единицу времени.

Задача о максимальном потоке на сети. В сети (I, U) с одним источником s и одним стоком t и с заданной функцией пропускной способности может быть введена функция потока и сформулирована задача о максимальном потоке.

Потоком в сети называется функция, сопоставляющая каждой дуге (i, j) целое число x_{ij} и обладающая свойствами:

$$0 \leqslant x_{ij} \leqslant r_{ij}, \qquad (i,\ j) \in U; \tag{2.25}$$

$$\sum_i x_{ik} - \sum_j x_{kj} = 0, \quad k \neq s,t, \quad k \in I; \tag{2.26}$$

$$\sum_j x_{sj} = \sum_i x_{it} = v, \quad i,j \in I. \tag{2.27}$$

Для транспортной сети поток x_{ij} по дуге $(i,\ j)$ означает количество продукта, проходящее через эту дугу в единицу времени. Условия (2.25) означают, что поток по каждой дуге неотрицателен и не превышает ее пропускной способности, а (2.26) — что количество продукта, поступающего в любую нейтральную вершину, равно количеству продукта, вытекающего из нее. Следовательно, общее количество продукта, вытекающего из источника, совпадает с общим количеством продукта, поступающего в сток, что и отражается в (2.27). Линейная форма v в (2.27) есть величина потока в сети.

Одна из основных задач на сетях состоит в определении величины наибольшего потока, допустимого в сети при заданной функции пропускной способности. Математически она формулируется так: найти значения переменных x_{ij}, максимизирующие линейную форму (2.27) при ограничениях (2.25), (2.26). При решении задачи о максимальном потоке важную роль играет понятие разреза.

Разрез. Дадим определение разреза для сети $(I,\ U)$ с одним источником s и одним стоком t. Если разбить множество всех вершин I на два непересекающихся подмножества R и R', такие, что $R \cup R' = I$, $R \cap R' = \varnothing$, причем $s \in R$, $t \in R'$, то *разрезом*, отделяющим источник от стока, называется множество $(R,\ R')$. Иначе, разрез составляют все те и только те дуги, которые исходят из вершин $i \in R$ и заканчиваются в вершинах $j \in R'$. При этом источник и сток находятся в различных подмножествах вершин, $s \in R$, $t \in R'$.

Сумма пропускных способностей дуг разреза называется его пропускной способностью или просто *величиной разреза* и обозначается как

$$r(R,\ R') = \sum_{\substack{i \in R \\ j \in R'}} r_{ij} = \sum_{(i,\ j) \in (R,\ R')} r_{ij}.$$

Разрез сети, имеющий наименьшую пропускную способность, называется *минимальным*.

Рассмотрим, например, сеть, представленную на рис.2.3 (числа при дугах указывают их пропускные способности). Будем считать вершину 1 источником, вершину 5 — стоком. В данной сети можно построить восемь разрезов. Назовем некоторые из них. Пусть $R_1 = \{1, 2\}$. Тогда $\bar{R}_1 = \{3, 4, 5\}$ и $(R_1, \bar{R}_1) = \{(1, 3), (1, 4), (2, 3), (2, 5)\}$. Пусть $R_2 = \{1, 3, 4\}$. Тогда $\bar{R}_2 = \{2, 5\}$ и $(R_2, \bar{R}_2) = \{(1, 2), (3, 5), (4, 5)\}$. Пропускные способности разрезов $(R_1, \bar{R}_1)$ и $(R_2, \bar{R}_2)$ равны: $r(R_1, \bar{R}_1) = r(1, 3) + r(1, 4) + r(2, 3) + r(2, 5) = 7 + 4 + 2 + 6 = 19$; $r(R_2, \bar{R}_2) = r(1, 2) + r(3, 5) + r(4, 5) = 5 + 3 + 9 = 17$. Легко убедиться, что минимальным в этой сети является разрез $(R, R') = \{(1, 4), (2, 5), (3, 4), (3, 5)\}$ с пропускной способностью $r = 15$.

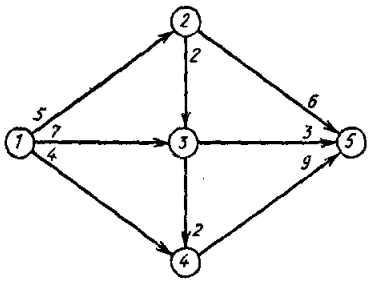

Рис. 2.3.

Отметим важное свойство разреза, связывающее величину допустимого потока в сети с его пропускной способностью. Возьмем произвольный разрез (R, R'). Какой бы путь от s к t мы ни рассматривали, хотя бы одна его дуга будет входить в разрез (R, R'). Это очевидно, так как s и t принадлежат различным множествам R и R'. Следовательно, если удалить все дуги какого-либо разреза, то в сети не останется ни одного пути, ведущего из s в t. Иначе, *любой разрез блокирует все пути из источника в сток*. Отсюда поток v, текущий в сети от s к t по всем возможным путям, равен величине потока через любой разрез:

$$v = \sum_{(i,\ j) \in (R,\ R')} x_{ij} = x(R,\ R').$$

Применяя ко всем дугам разреза условие (2.25), получаем

$$v = x(R, R') \leqslant r(R, R'), \qquad (2.28)$$

т. е. *поток в сети не превышает пропускную способность любого разреза.*

Теперь можно предположить, что если удастся построить такой поток $\{x_{ij}\}$, величина которого v окажется

равной пропускной способности некоторого разрез
(R, R'), $v=r(R, R')$, то этот поток будет максимален,
разрез (R, R') минимален.

**Теорема о максимальном потоке и минимальном раз
резе (теорема Форда — Фалкерсона).** Простые рассуж
дения показывают, что величина потока в сети не превы
шает пропускной способности любого ее разреза. Тепер
сформулируем теорему, которая устанавливает равенст
во между максимальным потоком и минимальным раз
резом.

Теорема 2.3. *Для любой сети с одним источником*
и одним стоком t величина максимального потока из
в t равна пропускной способности минимального разре
за, отделяющего s от t.

◆ Пусть v — величина максимального потока в сети
(I, U). Необходимо доказать, что среди всех разрезов
этой сети всегда найдется такой разрез (R, R'), что усло
вие (2.28) выполняется как равенство, т. е. $v=r(R, R')$.

Покажем, что такой разрез можно построить, если
определить множество вершин R следующим образом

$$\left.\begin{array}{l} s\in R; \\ \text{если } i\in R \text{ и } x_{ij}<r_{ij}, \text{ то } j\in R; \\ \text{если } i\in R \text{ и } x_{ji}>0, \text{ то } j\in R. \end{array}\right\} \qquad (2.29)$$

Будем говорить, что дуга (i, j) *насыщена*, если $x_{ij}=$
$=r_{ij}$. Дугу, входящую в некоторую цепь, будем называть
прямой, если ее направление совпадает с направлением
обхода вершин, и *обратной* в противном случае. Как сле
дует из правила (2.29), вершины, составляющие множе
ство R, определяются одна за другой, начиная с источ
ника s. Некоторая вершина j включается в множество
R, если существует цепь от s к j, прямые дуги которой
не насыщены, а поток на обратных дугах отличен от
нуля.

Итак, надо показать, что, во-первых, применение пра
вила (2.29) приводит к построению разреза, и во-вторых,
для этого разреза имеет место равенство $v=r(R, R')$.

Докажем первое. Разрез будет построен в том случае,
если в результате применения правила (2.29) $t\overline{\in}R$. Пред
положим противное, т. е. $t\in R$. Тогда из способа построе
ния множества R следует, что найдется цепь из s в t, та
кая, что для всех прямых дуг этой цепи поток меньше
пропускной способности, а для всех обратных — больше
нуля. Пусть ε_1 — минимум разности $r-x$, взятый по всем

прямым дугам этой цепи, а ε_2 — минимум x, взятый по всем ее обратным дугам, и пусть $\varepsilon = \min(\varepsilon_1, \varepsilon_2)$. Изменим теперь поток $\{x_{ij}\}$ следующим образом: увеличим x на величину ε на всех прямых дугах и уменьшим на ε на всех обратных дугах этой цепи. Нетрудно видеть, что величина нового потока станет равной $v+\varepsilon$, что противоречит предположению о максимальности v. Отсюда следует, что $t \overline{\in} R$ и множество (R, R') есть разрез, отделяющий s от t.

Теперь покажем, что для построенного разреза имеет место $v = r(R, R')$. Из правила (2.29) построения множества R следует, что если $i \in R$, $j \in R'$, то должно быть $x_{ij} = r_{ij}$, иначе вершина j входила бы в R. Поэтому $\sum\limits_{(i,\ j) \in (R,\ R')} x_{ij} = \sum\limits_{(i,\ j) \in (R,\ R')} r_{ij}$ или $x(R, R') = v = r(R, R')$ и в силу условия (2.28) разрез (R, R') минимален. ◆

Алгоритм Форда—Фалкерсона. Теорема о максимальном потоке и минимальном разрезе указывает критерий оптимальности потока: следует искать разрез, пропускная способность которого минимальна; она и будет равна величине потока. Из доказательства теоремы следует алгоритм, а именно: *начав с некоторого потока* величины v (например, с нулевого), определяем по правилу (2.29) множество вершин R, которые могут быть достигнуты из s по ненасыщенным цепям. Если $t \in R$, то от s к t имеется ненасыщенная цепь, и поток можно увеличить. Если же $t \overline{\in} R$, то данный поток максимален, а разрез (R, R') минимален.

Удобный способ реализации алгоритма состоит в расстановке *на вершинах сети пометок*, с помощью которых поток может быть увеличен. Пометка вершины j имеет вид (i^+, ε_j) или (i^-, ε_j), где $i \in I$ и означает номер предшествующей вершины ненасыщенной цепи от s к t, а ε_j — целое число или ∞, показывающее величину возможного прироста потока вдоль цепи от источника s до данной вершины j. Опишем алгоритм.

1. Пометим вершину s меткой $(0, \infty)$. Далее расстановка меток осуществляется в соответствии с правилом (2.29). Выберем любую помеченную вершину i (первоначально это источник s). Рассмотрим все непомеченные вершины j, связанные дугами с вершиной i. Тем вершинам j, для которых $x_{ij} < r_{ij}$, припишем метку (i^+, ε_j), где $\varepsilon_j = \min(\varepsilon_i, r_{ij} - x_{ij})$. Тем вершинам j, для которых

$x_{ji}>0$, припишем метку (i^-, ε_j), где $\varepsilon_j=\min(\varepsilon_i, x_{ji})$. Затем рассмотрим следующую помеченную вершину и поступим аналогично. Этот общий шаг повторяем до тех пор, пока не пометим вершину t, или до тех пор, пока нельзя будет сделать новых пометок.

2. Вершина t получает пометку (j^+, ε_t) или (j^-, ε_t). Если пометка вида (j^+, ε_t), заменяем x_{jt} на $x_{jt}+\varepsilon_t$, если же пометка вида (j^-, ε_t), то заменяем x_{tj} на $x_{tj}-\varepsilon_t$. Затем переходим к вершине j и поступаем аналогично: если ее пометка вида (i^+, ε_j), то заменяем x_{ij} на $x_{ij}+\varepsilon_t$, если ее пометка вида (i^-, ε_j), то заменяем x_{ji} на $x_{ji}-\varepsilon_t$ и переходим к вершине i. Так поступаем, пока не достигнем источника s. После этого стираем все пометки и вновь переходим к шагу 1.

Таким образом, процесс расстановки пометок представляет собой поиск цепи из s в t, на которой поток может быть увеличен. Если в результате шага 1 сток оказался помеченным, то вдоль найденной цепи можно увеличить поток. Если же шаг 1 закончился, а сток остался непомеченным, то достигнутый поток максимален. В соответствии с теоремой о максимальном потоке и минимальном разрезе множество дуг, ведущих из помеченных вершин в непомеченные, образует минимальный разрез.

Из алгоритма следует, что при начальном целочисленном потоке и целочисленных пропускных способностях дуг величина ε_t, определяющая приращение потока, представляет минимум двух целых чисел. Поэтому поток, увеличенный на ε_t, вновь будет целочисленным. И так как на каждом шаге поток возрастает по крайней мере на единицу, то алгоритм конечен, а максимальный поток целочислен. Сформулируем этот вывод в виде следующего свойства потока: *если пропускные способности всех дуг сети целочисленны, то на ней существует целочисленный максимальный поток.*

Пример 2.5. Для сети, заданной на рис. 2.4, *а*, найти максимальный поток из вершины *1* в вершину *6*. Каждое ребро представляет две противоположно направленные дуги. Числа, стоящие при ребрах, означают их пропускные способности, одинаковые в обоих направлениях.

Решение можно начинать с произвольного начального потока, в том числе с нулевого. Обычно не составляет труда построить некоторый допустимый поток, отличный от нуля. Это ускоряет решение. На рис. 2.4,*б* построен такой начальный поток величиной 12 единиц. Стрелками показано направление потока, а числа, стоящие рядом с пропускными способностями, показывают его величину. Пометки при вершинах получены в результате первого шага.

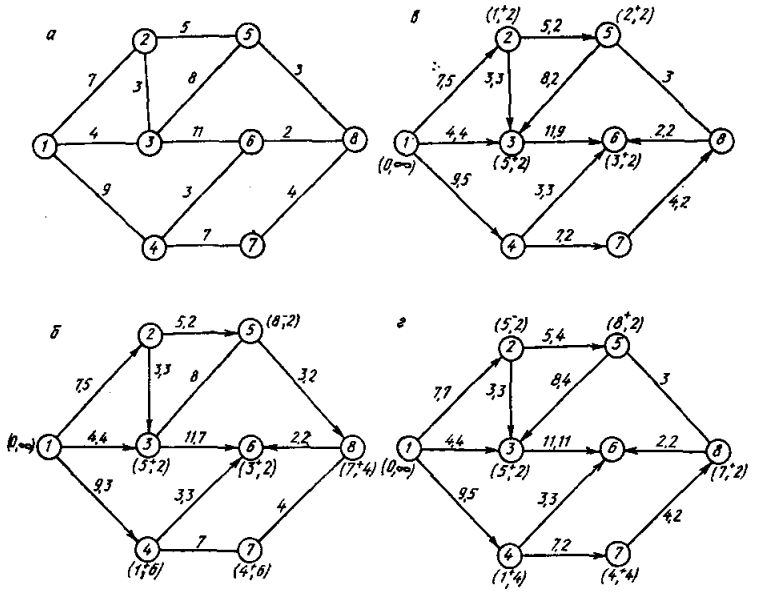

Рис. 2.4.

Рассмотрим, например, как получена пометка вершины *5*. Исходя из источника, последовательно пометим вершины 4; 7; 8. Находясь в вершине *8*, мы не можем пометить вершину *6*, так как прямая дуга (8,6) насыщена. Зато вершина *5* может быть помечена по обратной дуге (5,8) меткой $(8^-, \min(4,2)) = (8^-, 2)$. Исходя из вершины *5*, пометим вершину *3* меткой $(5^+, \min(2,8-0)) = (5^+,2)$. После этого вершина *6* получает пометку $(3^+, \min(2,11-7)) = (3^+,2)$ и первый шаг закончен. Переходя ко второму шагу, увеличиваем поток на величину $\varepsilon_6 = 2$. Для этого, *двигаясь по выделенной цепи* $s = (1, 4, 7, 8, 5, 3, 6)$, увеличим поток на 2 единицы на всех прямых дугах и уменьшим на столько же на обратной дуге (5,8). Новый поток величины *14* показан на рис. 2.4, *в*.

Выполняем расстановку пометок при данном потоке. На первом шаге метим вершины *2*; *5*; *3*; *6* и на втором шаге увеличиваем поток вдоль цепи $s = (1, 2, 5, 3, 6)$ на две единицы. Новый поток величины 16 показан на рис. 2.4, *г*.

Выполняя первый шаг при потоке, представленном на рис. 2.4, *г*, видим, что сток пометить не удается. Следовательно, максимальная величина потока достигнута и равна 16. Помеченные вершины образуют множество $R = \{1, 2, 3, 4, 5, 7, 8\}$. Отсюда находим минимальный разрез $(R, R') = \{(3,6), (4,6), (8,6)\}$. Его пропускная способность $r(R, R') = 11+3+2 = 16$ и равна величине максимального потока.

Обычно при решении задачи достаточно сделать один рисунок, на котором перед каждым новым шагом стирают все предыдущие пометки.

2.6 ЗАДАЧА О НАЗНАЧЕНИЯХ

Постановка задачи. Транспортная задача выделяется в самостоятельный класс задач линейного программирования, так как для ее решения имеются специальные методы, эффективность которых гораздо выше по сравнению с методами решения общей задачи линейного программирования. В свою очередь известны специальные типы транспортной задачи, для решения которых существуют более эффективные алгоритмы, чем для общей транспортной задачи.

Рассмотрим, например, транспортную задачу, в которой объемы как запасов a_i, так и потребностей b_j в каждом из пунктов i, j одинаковы и равны 1. Тогда из условия (2.6) следует $m=n$ и модель задачи принимает вид

$$\left.\begin{aligned} &\min z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij};\\ &\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,\ \ldots,\ n;\\ &\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,\ \ldots,\ n;\\ &x_{ij} \geqslant 0,\ i = 1,\ \ldots,\ n,\ j = 1,\ \ldots,\ n. \end{aligned}\right\} \qquad (2.30)$$

Транспортная задача в постановке (2.30) называется задачей о назначениях. Как и любая транспортная задача, она имеет решение в целых числах, причем ограничения позволяют переменным x_{ij} принимать только значения 0 или 1. Название задачи связано со следующей экономической интерпретацией ее условий.

Имеется n различных работ, которые могут быть поручены n различным исполнителям. Назначение исполнителя i на работу j связано с затратами c_{ij}. Каждый исполнитель должен быть назначен на одну и только одну работу, так, чтобы суммарные затраты на выполнение всех работ были минимальны. Введя переменные x_{ij}, принимающие значения: $x_{ij}=1$, если исполнитель i назначается на работу j, и $x_{ij}=0$ в противном случае, можно записать постановку задачи в виде модели (2.30), дополненной требованием $x_{ij}=0$ или 1. Однако в силу целочисленности решений транспортной задачи включение последнего условия излишне, и, следовательно, модель (2.30) полностью описывает постановку задачи. Ее реше-

ние можно представить в виде *матрицы назначений* $X=(x_{ij})$, в каждой строке и каждом столбце которой имеется один элемент $x_{ij}=1$, а все остальные $x_{ij}=0$. Таким образом, любое допустимое решение задачи о назначениях всегда вырождено, так как число положительных компонент $x_{ij}=1$ в нем равно n, а ранг матрицы условий, как и в обычной транспортной задаче, равен $m+n-1=2n-1$. Отметим, что условие $m=n$ не снижает общности постановки. Если $m<n$, введем фиктивных исполнителей $i=m+1, \ldots, n$ с расходами $c_{ij}=0$ для всех $i>m$, а если $m>n$, введем фиктивные работы $j=n+1, \ldots, m$ при $c_{ij}=0$ для всех $j>n$.

Двойственная задача. Эквивалентные преобразования. Имея двойственную пару (2.2) — (2.5) и (2.18), (2.19) для транспортной задачи, легко построить двойственную пару для задачи о назначениях. По аналогии с (2.18), (2.19) задача, двойственная к (2.30) при условии $a_i=b_j=1$, запишется следующим образом:

$$\max w=\sum_i u_i+\sum_j v_j; \tag{2.31}$$

$$u_i+v_j\leqslant c_{ij},\ i,\ j=1,\ \ldots,\ n. \tag{2.32}$$

Сохраняет силу и критерий оптимальности (2.20), (2.21), которому можно придать следующий вид. Пусть X — допустимое решение прямой задачи, $\mathbf{y}=(\mathbf{u},\ \mathbf{v})$ — допустимое решение двойственной задачи. Тогда решение X оптимально в прямой задаче, а $\mathbf{y}$ — в двойственной при условии

$$x_{ij}>0, \text{ если только } u_i+v_j=c_{ij}. \tag{2.33}$$

Иначе, если вектор $(\mathbf{u},\ \mathbf{v})$ оптимален в двойственной задаче, то исполнитель i может быть назначен на работу j, если только

$$c_{ij}-u_i-v_j=0. \tag{2.34}$$

Для отыскания системы двойственных переменных u_i, v_j, удовлетворяющих условию (2.34), удобно использовать преобразования матрицы затрат C, состоящие в вычитании из всех столбцов одного и того же вектора-столбца $\mathbf{u}=(u_i)_n$ и из всех строк одного и того же вектора-строки $\mathbf{v}=(v_j)_n$. Получаемая в результате матрица A с элементами $a_{ij}=c_{ij}-u_i-v_j$ называется эквивалентной матрице C, и наоборот, матрица C эквивалентна матрице A.

Теорема 2.4. *Оптимальные назначения двух задач с эквивалентными матрицами совпадают.*

◆ Пусть матрицы C и A эквивалентны и пусть оптимальное назначение для матрицы C задается парами индексов вида $(1, j_1)$, $(2, j_2)$, ..., (n, j_n), указывающими, что первый исполнитель получает работу j_1, второй — j_2 и т. д. Допустим, что для задачи с матрицей A оптимальным является другое назначение, заданное, например, парами индексов $(1, j_1')$, $(2, j_2')$, ..., (n, j_n'). Тогда $a_{1j_1'} + a_{2j_2'} + \ldots + a_{nj_n'} < a_{1j_1} + a_{2j_2} + \ldots + a_{nj_n}$. Так как по условию $a_{ij} = c_{ij} - u_i - v_j$, то можем записать $(c_{1j_1'} - u_1 - v_{j_1'}) + (c_{2j_2'} - u_2 - v_{j_2'}) + \ldots + (c_{nj_n'} - u_n - v_{j_n'}) < (c_{1j_1} - u_1 - v_{j1}) + (c_{2j_2} - u_2 - v_{j2}) + \ldots + (c_{nj_n} - u_n - v_{j_n})$. Очевидно, что $v_{j_1'} + v_{j_2'} + \ldots + v_{j_n'} = v_{j_1} + v_{j_2} + \ldots + v_{j_n}$, так как это сумма одних и тех же чисел, отличающаяся только порядком суммирования. Поэтому неравенство принимает вид $c_{1j_1'} + c_{2j_2'} + \ldots + c_{nj_n'} < c_{1j_1} + c_{2j_2} + \ldots + c_{nj_n}$, что невозможно, так как по предположению $(1, j_1)$, $(2, j_2)$, ..., (n, j_n) оптимально для матрицы C. Следовательно, для задачи с матрицей A оптимальным будет то же назначение, что и для задачи с матрицей C. ◆

Следовательно, для решения задачи о назначениях матрицу затрат (c_{ij}) можно преобразовывать к некоторой эквивалентной, более удобной для отыскания решения матрице. В частности, числа u_i и v_j могут быть подобраны так, чтобы в матрице A часть элементов обратилась в нули. Тогда элементы этой матрицы разбиваются на два класса: нули и не нули.

Пример 2.6. Пусть имеются матрица C, вектор-столбец $\mathbf{u}$ и вектор-строка $\mathbf{v}$ вида

$$C = \begin{bmatrix} 4 & 1 & 3 & 6 \\ 3 & 4 & 8 & 4 \\ 2 & 7 & 5 & 3 \\ 5 & 6 & 2 & 4 \end{bmatrix}; \quad \mathbf{u} = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix}; \quad \mathbf{v} = (0,\ 0,\ 1,\ 1).$$

Вычитая из каждого столбца матрицы C вектор-столбец $\mathbf{u}$, получим матрицу A^1; затем, вычитая из каждой строки матрицы A^1 вектор-строку $\mathbf{v}$, получим матрицу A:

$$A^1 = \begin{bmatrix} 3 & 0 & 2 & 5 \\ 0 & 1 & 5 & 1 \\ 0 & 5 & 3 & 1 \\ 4 & 5 & 1 & 3 \end{bmatrix}; \quad A = \begin{bmatrix} 3 & 0 & 1 & 4 \\ 0 & 1 & 4 & 0 \\ 0 & 5 & 2 & 0 \\ 4 & 5 & 0 & 2 \end{bmatrix}.$$

На матрице A оптимальное распределение исполнителей по работам отыскивается непосредственно. Для этого достаточно выбрать в каждой строке нуль, не стоящий в одном и том же столбце с другим выбранным нулем. В результате получаем назначение (1, 2), (2, 1), (3, 4), (4, 3) с суммарными затратами, равными нулю. Оно допустимо, так как из каждой строки и каждого столбца выбрано по одному элементу, и оптимально, так как сумма неотрицательных чисел не может быть меньше нуля. На матрице C ему отвечает величина затрат $\sum_i \sum_j c_{ij}x_{ij} = 1 + 3 + 3 + 2 = 9$. Отметим, что это решение не единственное: имеется еще одно распределение исполнителей по работам (1, 2), (2, 4), (3, 1), (4, 3), которое дает ту же величину затрат $1 + 4 + 2 + 2 = 9$. При этом сумма компонент векторов $\mathbf{u}$ и $\mathbf{v}$ равна этому же числу: $\sum_i u_i + \sum_j v_j = 7 + 2 = 9$.

Упрощенная задача. При решении задачи (2.30) широко используются методы, основанные на последовательном решении более простой задачи в следующей постановке.

Пусть каждый исполнитель может выполнять некоторые, но не все работы, и при распределении по работам учитывается только, пригоден исполнитель к данной работе или нет, а оценки затрат отсутствуют. Всего имеется m исполнителей и n работ. Требуется распределить исполнителей по работам так, чтобы общее число назначений было максимальным, и при этом на одну работу не назначалось более одного исполнителя, а одна работа поручалась только одному исполнителю.

Введем матрицу $Q = (q_{ij})_{m \times n}$, элементы которой могут принимать одно из двух значений: $q_{ij} = 1$, если исполнитель i пригоден к работе j, и $q_{ij} = 0$ в противном случае. Будем называть матрицу Q *матрицей квалификаций*; ее строки и столбцы — *рядами*; клетки, содержащие единицы, — *допустимыми*, а нули — *недопустимыми*. Таким образом, по условию все элементы матрицы квалификаций разбиты на два класса: допустимые и недопустимые клетки. Совокупность клеток назовем *независимой*, если никакие две клетки из этой совокупности не лежат в одном ряду. Тогда *упрощенная задача состоит в том, чтобы в матрице квалификаций найти максимальное число независимых допустимых клеток.*

Задача допускает простую и удобную постановку с помощью графов специального вида, называемых *двудольными*. В двудольном графе $G(I, U) = G(K, P; U)$ множество вершин I разбито на два непересекающихся

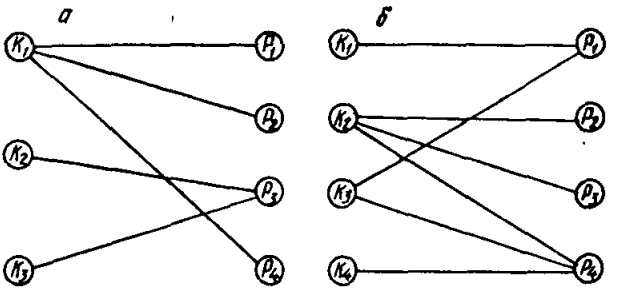

Рис. 2.5.

подмножества K и P, $K \cup P = I$, $K \cap P = \varnothing$, ребра связывают вершины множества K с вершинами множества P, а вершины одного и того же множества не связаны между собой (рис. 2.5).

Если сопоставить вершины множества K исполнителям, множества P — работам и соединить вершину k_i с вершиной p_j в том и только том случае, если исполнитель k_i пригоден к работе p_j, то *задача состоит в том, чтобы найти в двудольном графе (K, P; U) максимальное число дуг, попарно не имеющих общих вершин.*

Покажем, как можно определить верхнюю границу числа дуг, попарно не имеющих общих вершин. Введем для двудольного графа $(K, P; U)$ понятие *рассекающего* множества вершин: это такое множество вершин, при удалении которых исчезают все дуги графа. Рассмотрим, например, рис. 2.5, *а*. Поскольку дуга исчезает при удалении любой из ее концевых вершин, рассекающим будет как множество K, состоящее из трех вершин, так и множество P, включающее четыре вершины. Можно, однако, найти рассекающее множество $\{k_1, p_3\}$, состоящее из двух вершин. Оно минимально, поскольку в данном графе не существует рассекающего множества с меньшим числом вершин. Непосредственной проверкой можно убедиться, что максимальное число назначений на этом графе также равно двум. Действительно, исполнитель k_1 может быть назначен только на одну из подходящих ему работ p_1, p_2 или p_4, а работа p_3 будет поручена только одному из соответствующих ей исполнителей k_2, k_3. Аналогично на графе, показанном на рис. 2.5, *б*, минимальное рассекающее множество состоит из трех вершин k_2, p_1, p_4 и максимальное число назначений также равно

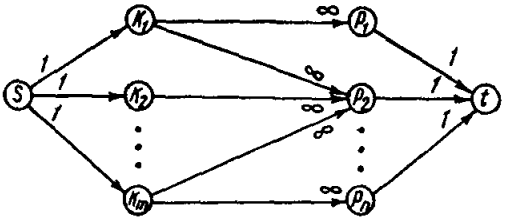

Рис. 2.6.

трем. В обоих случаях замечаем, что максимальное число дуг, попарно не имеющих общих вершин, совпадает с числом вершин минимального рассекающего множества рассматриваемого графа. Существует теорема, распространяющая этот вывод на общий случай.

Теорема 2.6. (теорема Кенига — Эгервари). *Максимальное число дуг двудольного графа, попарно не имеющих общих вершин, равно числу вершин минимального рассекающего множества.*

Теорема может быть доказана различными способами. Удобнее всего сделать это с помощью теоремы о максимальном потоке и минимальном разрезе, хотя исторически последняя появилась значительно позже теоремы Кенига—Эгервари.

◆ Пусть дан двудольный граф $(K, P; U)$, $K=\{k_i\}$, $i=1, ..., m$; $P=\{p_j\}$, $j=1, ..., n$. Построим фиктивную сеть (рис. 2.6), присоединив к нему источник s, сток t, дуги (s, k_i), (p_j, t) и задав функцию пропускных способностей по правилам:

$$r(s, k_i)=1,\ k_i\in K;\ r(p_j, t)=1,\ p_j\in P;\ r(k_i, p_j)=\infty,\ (k_i, p_j)\in U.$$

Положим, что в данной сети течет максимальный целочисленный поток величины v. Тогда в исходном графе $(K, P; U)$ найдется ровно v дуг (k_i, p_j), по которым проходит единичный поток, причем они попарно не имеют общих вершин. Действительно, если по дуге (k_i, p_j) проходит единичный поток, то он насыщает дуги (s, k_i), (p_j, t), поэтому ни при вершине k_i, ни при вершине p_j нет больше дуг исходного графа с потоком, отличным от нуля.

Покажем теперь, что v — число вершин минимального рассекающего множества. По теореме Форда — Фалкерсона максимальному потоку величины v соответствует минимальный разрез той же величины v. Поскольку про-

пускная способность данного разреза конечна, то в нем отсутствуют дуги вида (k_i, p_j). Следовательно, минимальный разрез содержит лишь дуги вида (s, k_i) и (p_j, t), число которых равно v. Каждой дуге такого разреза соответствует одна вершина исходного графа, принадлежащая множеству либо K либо P, следовательно, число таких вершин также равно v, а их совокупность представляет минимальное рассекающее множество.

Для практического применения удобно переформулировать теорему Кенига — Эгервари в терминах матриц квалификации: *максимальное число независимых допустимых клеток равно минимальному числу рядов, в которых содержатся все допустимые клетки.*

Алгоритм для упрощенной задачи. Из доказательства теоремы Кенига — Эгервари вытекает алгоритм, представляющий по существу модификацию алгоритма Форда — Фалкерсона для задачи о максимальном потоке в табличной форме. Алгоритм состоит из начального шага, на котором определяется начальное допустимое решение, и общего шага, на котором выполняется итерационный процесс его улучшения.

Начальный шаг. Найдем начальное допустимое распределение исполнителей по работам методом, аналогичным методу северо-западного угла: рассматривая первую строку, назначим исполнителя k_1 на работу с наименьшим номером, для которой он пригоден; переходя ко второй строке, назначим исполнителя k_2 на подходящую ему работу, которая еще не занята и имеет наименьший номер, и т. д. При каждом назначении делаем отметку (ставим, например, звездочку) в соответствующей клетке. Закончив просмотр всех строк, переходим к общему шагу.

Общий шаг. Пометим черточкой все строки, не содержащие звездочки. Затем возьмем какую-либо помеченную строку, например i-ю, и просмотрим ее, отыскивая допустимые невыделенные клетки. Столбцы, соответствующие таким клеткам, пометим номером просматриваемой строки, в данном случае i. Будем повторять это до тех пор, пока не просмотрим все помеченные строки. Столбец, уже получивший какую-нибудь пометку, больше не помечается.

Теперь выберем любой помеченный столбец, например j-й, и просмотрим его, отыскивая звездочку. Если она будет найдена, пометим строку, в которой стоит эта

звездочка, номером просматриваемого столбца, в данном случае j. Снова выберем какой-либо помеченный непросмотренный столбец и повторим эту операцию. После того как будут просмотрены все помеченные столбцы, переходим к просмотру вновь отмеченных строк, отыскивая в них допустимые невыделенные клетки, и т. д. Продолжаем этот процесс, попеременно просматривая строки и столбцы, пока не достигнем следующего: а) либо будет помечен столбец, не содержащий звездочки; б) либо приписать никаких пометок больше нельзя и все помеченные столбцы содержат звездочки.

В случае «а» общее число звездочек (назначений) увеличиваем следующим образом. В помеченном столбце, не содержащем звездочки, поставим звездочку в клетке, указываемой пометкой этого столбца. Затем в строке, где проставлена новая звездочка, уберем прежнюю из клетки, указываемой пометкой этой строки. В столбце, в котором находится удаляемая звездочка, перейдем к клетке, указываемой его пометкой, и поставим звездочку, и т. д.

В конце концов придем к одной из строк, помеченных черточкой, и общее число звездочек в таблице возрастет на 1. После этого сотрем пометки, повторим процесс расстановки пометок и вновь переместим звездочки, продолжая процесс до тех пор, пока не придем к случаю «б», в котором оптимальное закрепление достигнуто. Минимальное число рядов, содержащих все допустимые клетки, состоит из непомеченных строк и помеченных столбцов.

Пример 2.7. Найти максимальное число назначений в матрице квалификаций, заданной в табл. 2.15. Допустимыми являются клетки, где проставлены единицы, звездочками отмечено начальное распределение, полученное на начальном шаге алгоритма. В строках 6 и 7 звездочек нет. Пометим эти строки черточкой, затем последовательно просмотрим их, отыскивая допустимые неотмеченные клетки. Просматривая строку 6, отмечаем пометкой 6 столбцы 1 и 6. Строка 7 дает пометку столбцу 3. Теперь все помеченные строки просмотрены. Переходим к просмотру помеченных столбцов.

Исходя из столбца 1, помечаем строку 2; исходя из столбца 3, помечаем строку 5; исходя из столбца 6, помечаем строку 3. Переключаясь на просмотр строк, получаем только одну дополнительную пометку: строка 2 дает пометку столбцу 8.

Переходим к просмотру столбцов. Вновь помеченный столбец 8 не содержит звездочки, следовательно, можно увеличить число назначений. Для этого, переходя к табл. 2.16, поставим звездочку в клетке 2 столбца 8, в строке 2 удалим звездочку из столбца 1 и в столбце 1 поставим звездочку в строке 6. Поскольку она помечена черточ-

Таблица 2.15

k_i \ p_j	1	2	3	4	5	6	7	8	
1		1*			1		1		
2	1*		1					1	1
3	1					1*		1	6
4				1*			1		
5	1		1*			1			3
6	1					1			—
7			1			1			—
8		1		1	1*		1		
	6		7			6		2	

Таблица 2.16

k_i \ p_j	1	2	3	4	5	6	7	8	
1		1*			1		1		
2	1		1					1*	8
3	1					1*		1	6
4				1*			1		
5	1		1*			1			3
6	1*					1			1
7			1			1			—
8		1		1	1*		1		
	3		7			7		3	

кой, перемещения на этом прекращаем, а общее число назначений увеличивается на одно.

В новой матрице назначений строка 7 не содержит звездочки. Помечаем ее черточкой и метим числом 7 столбцы 3 и 6. Переходим к просмотру столбцов. Столбец 3 дает пометку строке 5, столбец 6 — строке 3. Просматривая вновь помеченные строки 3 и 5, отмечаем номером просматриваемой строки 3 столбцы 1 и 8; строка 5 пометок не дает, так как все ее допустимые клетки находятся в помеченных столбцах. Просматривая вновь помеченные столбцы 1 и 8, находим в них звездочки и даем пометки 1 и 8 строкам 6 и 2, в которых эти звездочки стоят.

Переключаемся на просмотр строк. Видим, что ни одной пометки больше сделать нельзя. При этом все помеченные столбцы содержат звездочки. Следовательно, задача о назначениях решена: *найдено минимальное число рядов, содержащих максимальное число независимых допустимых клеток.* Легко убедиться, что эти ряды составляют непомеченные строки 1, 4, 8 и помеченные столбцы 1, 3, 6, 8. Если их зачеркнуть, то будут зачеркнуты все допустимые клетки (некоторые, может быть, дважды — по столбцу и по строке).

Таким образом, минимальное число рядов, покрывающих все допустимые клетки, равно семи. Следовательно, в соответствии с теоремой Кенига — Эгервари, максимальное число назначений тоже равно семи. Действительно, в табл. 2.16 имеется семь отмеченных звездочками клеток, указывающих, что первый исполнитель назначен на вторую работу, второй — на восьмую, третий — на шестую и т. д. Заметим, что это решение не единственное.

Венгерский метод для задачи о назначениях. Предшествующими рассуждениями подготовлен переход к обоснованию алгоритма, который сводит решение задачи о назначениях к решению последовательности упрощенных задач о назначениях.

Итак, пусть имеется некоторое допустимое решение $u_1, \ldots, u_n, v_1, \ldots, v_n$ двойственной задачи (2.31), (2.32). Найдем в матрице C исходной задачи (2.30) все те клетки (i, j), для которых в условии (2.32) двойственной задачи имеет место равенство, и назовем их допустимыми. Рассмотрим упрощенную задачу о назначениях для n исполнителей и n работ, где исполнитель k_i может быть назначен на работу p_j в том и только том случае, если (i, j) — допустимая клетка. Здесь возможны два случая: 1) назначения получают все n исполнителей; 2) число назначений t меньше числа исполнителей n.

В первом случае задача о назначениях решена. Действительно, признак оптимальности (2.33) выполнен и переменные $x_{ij}=0$ или 1 удовлетворяют ограничениям:

$$\sum_{j=1}^{n} x_{ij} = 1; \quad \sum_{i=1}^{n} x_{ij} = 1; \quad x_{ij} \geqslant 0$$

задачи (2.30). Отсюда решение $X=(x_{ij})$ оптимально для прямой задачи, а вектор $(\mathbf{u}, \mathbf{v})$ оптимален для двойственной задачи.

Во втором случае, согласно теореме Кенига — Эгервари, минимальное число рядов, содержащих все допустимые клетки, равно t, причем $t<n$. Пусть оно состоит из r строк, образующих множество M, и q столбцов, образующих множество N. Найдем число

$$h=\min_{i\overline{\in}M,\ j\overline{\in}N}[c_{ij}-(u_i+v_j)]. \tag{2.35}$$

Определим новые двойственные переменные u_i' и v_j':

$$u_i'=\begin{cases}u_i+h & \text{при } i\overline{\in}M;\\ u_i & \text{при } i\in M;\end{cases} \tag{2.36}$$

$$v_j'=\begin{cases}v_j & \text{при } j\overline{\in}N;\\ v_j-h & \text{при } j\in N.\end{cases} \tag{2.37}$$

Докажем, что числа u_i' и v_j' удовлетворяют условию (2.32) и, следовательно, допустимы в двойственной задаче. Действительно, рассматривая (2.36) и (2.37), находим: если $i\in M$, то $u_i'+v_j'\leqslant u_i+v_j\leqslant c_{ij}$; если $i\overline{\in}M$, $j\in N$, то $u_i'+v_j'=u_i+v_j\leqslant c_{ij}$; если $i\overline{\in}M$, $j\overline{\in}N$, то $u_i'+v_j=u_i+v_j+h\leqslant c_{ij}$ в силу (2.35).

Следовательно, переменные u_i' и v_j' допустимы. Подставим их выражения в целевую функцию (2.31):

$$\sum_i u_i'+\sum_j v_j'=\sum_i u_i+(n-r)h+\sum_j v_j-qh=\sum_i u_i+\sum_j v_j+h[n-(r+q)]. \tag{2.38}$$

Здесь $r+q$ — число назначений. По предположению $r+q=t<n$, значит, последнее слагаемое в (2.38) больше нуля и целевая функция с переменными u_i', v_j' больше, чем с переменными u_i, v_j.

Следовательно, если в упрощенной задаче получают назначения не все n исполнителей, то целевая функция двойственной задачи может быть увеличена. В соответствии с выражениями (2.35) — (2.37) можно перейти к новым значениям двойственных переменных, которые, удовлетворяя условиям (2.32), обеспечивают большее значение целевой функции (2.31). Процесс будет продолжаться до тех пор, пока при некоторой совокупности

двойственных переменных (u_i, v_j) не будет выполнено условие оптимальности (2.33), т. е. до тех пор, пока решение упрощенной задачи, где допустимыми являются клетки, для которых выполняется соотношение (2.34), не закончится назначением всех n исполнителей. Таким образом, это по существу вариант двойственного симплекс-метода, или метода последовательного сокращения невязок, применительно к частному случаю транспортной задачи. Метод конечен, так как всякий переход к новым двойственным переменным при целочисленной матрице C приводит к увеличению двойственной целевой функции (2.31), по крайней мере, на единицу, а ее величина, как нетрудно показать, ограничена сверху.

Вычислительная схема. Алгоритм состоит из начального и общего шагов.

Начальный шаг. Выполняем эквивалентное преобразование матрицы C, выбирая векторы $\boldsymbol{u}$ и $\mathbf{v}$ так, чтобы получить матрицу, в каждой строке и каждом столбце которой имеется хотя бы один нуль. Для этого просмотрим каждую строку матрицы C, найдем в ней наименьший элемент и вычтем его из элементов этой строки. Полученная матрица называется *приведенной по строкам*. В этой матрице просмотрим все столбцы, найдем в каждом из них минимальный элемент и вычтем его из элементов просматриваемого столбца. Получим матрицу, *приведенную по строкам и столбцам*. Клетку, содержащую нуль, будем считать допустимой, и с нулями будем обращаться так, как с единицами в упрощенной задаче.

Общий шаг. 1. Решаем упрощенную задачу о назначении. При этом возможны два случая: 1) назначения получают все n исполнителей; 2) назначения получают не все n исполнителей. В первом случае исходная задача решена, во втором переходим к шагу 2.

2. Зачеркнем каждую непомеченную строку и каждый помеченный столбец. Рассмотрим часть матрицы, состоящую из незачеркнутых элементов, и возьмем наименьшее число в ней. Вычтем это число из всех незачеркнутых элементов и прибавим к дважды зачеркнутым элементам (т. е. к элементам, стоящим на пересечении зачеркнутой строки с зачеркнутым столбцом). Переходим к шагу 1.

Пример 2.8. Найти оптимальное распределение исполнителей по работам при матрице затрат, заданной в табл. 2.17.

Таблица 2.17

k_i \ p_j	1	2	3	4	5	6
1	87	62	64	43	72	57
2	52	68	94	63	76	83
3	77	48	64	54	59	68
4	58	49	71	42	66	89
5	86	79	75	81	59	70
6	78	39	58	51	55	64

После приведения матрицы затрат по столбцам получаем табл. 2.18.

Таблица 2.18

k_i \ p_j	1	2	3	4	5	6
1	35	23	6	1	17	0
2	0	29	36	21	21	26
3	25	9	6	12	4	11
4	6	10	13	0	11	32
5	34	40	17	39	4	13
6	26	0	0	9	0	7

Приведя ее по строкам, получаем табл. 2.19, где в каждой строке и каждом столбце имеется хотя бы один нуль. Рассматриваем клетки с нулями как допустимые и решаем на них упрощенную задачу о назначениях. Назначения получают пять исполнителей, а минимальное множество рядов, содержащих все допустимые клетки, составляют непомеченные строки 1, 2, 4, 6 и помеченный столбец 5. Зачеркнув их, находим среди оставшихся незачеркнутыми в табл. 2.19 элементов наименьший, равный двум. Вычитаем его из всех незачеркнутых элементов и прибавляем ко всем дважды зачеркнутым. Получаем табл. 2.20 с восемью нулями, в которой назначения имеют уже все исполнители.

Таблица 2.19

k_i \ p_j	1	2	3	4	5	6	
1	35	23	6	1	17	0*	
2	0*	29	36	21	21	26	
3	21	5	2	8	0*	7	5
4	6	10	13	0*	11	32	
5	30	36	13	35	0	9	—
6	26	0*	0	9	0	7	
					5		

Таблица 2.20

k_i \ p_j	1	2	3	4	5	6
1	35	23	6	1	19	0*
2	0*	29	36	21	23	26
3	19	3	0*	6	0	5
4	6	10	13	0*	13	32
5	28	34	11	33	0*	7
6	26	0*	0	9	2	7

Запишем решение в виде совокупности двойных индексов (1,6), (2,1), (3,3), (4,4), (5,5), (6,2), где первый индекс — номер исполнителя, второй — номер поручаемой ему работы. Выбирая из соответствующих клеток исходной табл. 2.17 коэффициенты затрат и суммируя их, получаем значение целевой функции, соответствующее данному решению: $z=57+52+64+42+59+39=313$.

Контрольные вопросы и упражнения

1. Дайте постановку транспортной задачи и запишите ее математическую модель. Как формулируется условие допустимости транспортной задачи?

2. Какая модель называется закрытой; какая открытой? Как приводится открытая модель к закрытой?

3. Назовите основные особенности транспортной задачи.

4. Каким методом строится начальный опорный план? Сколько положительных компонент содержит невырожденный опорный план и почему? Как проверяется опорность допустимого плана?

5. Сформулируйте двойственную задачу, дайте признак оптимальности транспортной задачи.

6. Опишите вычислительную схему метода потенциалов.

7. Решите транспортную задачу, где $m=3$, $n=4$, запасы поставщиков заданы вектором (70, 80, 50), потребности потребителей — вектором (100, 40, 60), а транспортные расходы — одной из матриц следующего вида:

а) $\begin{bmatrix} 14 & 6 & 9 & 21 \\ 5 & 12 & 14 & 17 \\ 8 & 10 & 19 & 7 \end{bmatrix}$; б) $\begin{bmatrix} 8 & 12 & 15 & 7 \\ 11 & 9 & 10 & 13 \\ 4 & 7 & 18 & 6 \end{bmatrix}$; в) $\begin{bmatrix} 10 & 7 & 8 & 11 \\ 8 & 4 & 14 & 9 \\ 5 & 6 & 12 & 15 \end{bmatrix}$.

8. Дайте математическую формулировку задачи о максимальном потоке в матричной и сетевой постановке.

9. Что такое разрез? Что называется величиной разреза?

10. Сформулируйте и докажите теорему Форда — Фалкерсона о максимальном потоке и минимальном разрезе.

11. Какой граф называется двудольным?

12. Дайте математическую постановку задачи о назначениях.

13. Сформулируйте теорему Кенига — Эгервари.

3. ДИСКРЕТНОЕ ПРОГРАММИРОВАНИЕ

3.1. ТИПЫ ЗАДАЧ

Математическая постановка. К задачам *дискретного программирования* относятся задачи математического программирования, в которых требуется найти максимум целевой функции z, определенной на некотором *дискретном множестве* $P = \{p_1, p_2, \ldots, p_n\}$. Для решения задачи надо определить элемент $p^* \in P$ такой, что

$$z(p^*) = \max \{z(p_i) | p_i \in P\}.$$

Частным случаем является задача *линейного целочисленного программирования*, которая формулируется как задача линейного программирования, дополненная требованием целочисленности всех или некоторых переменных:

$$\max z = \mathbf{cx};\ A\mathbf{x} = \mathbf{b};\ \mathbf{x} \geqslant \mathbf{0};\ x_j \text{ — целые},\ j \in J.$$

Если множество индексов J включает все $j = 1, \ldots, n$, то имеем *полностью целочисленную* задачу. В ней элементами множества P являются целочисленные точки многогранника допустимых решений X задачи линейного программирования. Если требование целочисленности распространяется не на все $j = 1, \ldots, n$, то имеем *частично целочисленную* задачу линейного программирования.

В задачах дискретного программирования конечное множество P может быть задано не только линейной системой неравенств и уравнений, но и другими способами, например сетями. К моделям дискретного программирования приводит рассмотрение многих задач календарного планирования (теории расписаний), маршрутизации перевозок, синхронизации конвейера и т. д. Поскольку решение задач дискретного программирования обычно связано с построением и анализом последовательностей (комбинаций) элементов допустимого множества, их называют также *комбинаторными*. Хотя большинство

комбинаторных задач допускает формулировку в виде целочисленной модели, необходимо все же отметить, что линейные неравенства и уравнения плохо приспособлены для представления их комбинаторных свойств. Поэтому в дискретном программировании преимущественно используются алгоритмы, ориентированные на учет специальной структуры множества допустимых решений. Далее рассмотрим ряд моделей целочисленного линейного программирования.

Задачи с неделимостями. Иногда дробные значения переменных противоречат физическому смыслу задачи. Например, в задачах распределения станков, механизмов по видам работ, автомобилей по маршрутам распределяемые объекты по своей природе неделимы и, следовательно, должны быть выражены целыми числами. Целочисленными также являются задачи рационального раскроя материалов, в которых ищется количество листов, раскроенных различными способами. Система ограничений для задач этого типа отличается от ограничений нецелочисленной задачи лишь дополнительным требованием целочисленности всех или некоторых переменных.

В качестве примера рассмотрим несколько вариантов *задачи о рюкзаке.* В первом из них дано n неделимых предметов, для каждого из которых известны стоимость c_j и вес a_j, $j=1, \ldots, n$. Рюкзак вмещает не более чем b единиц груза. С учетом вместимости рюкзака требуется составить набор предметов максимальной ценности. Введя переменные $x_j=1$, если j-й предмет входит в набор, и $x_j=0$ в противном случае, получаем модель

$$\max z = \sum_{j=1}^{n} c_j x_j; \quad \sum_{j=1}^{n} a_j x_j \leqslant b; \quad x_j = 0 \text{ или } 1, \quad j = 1, \ldots, n. \tag{3.1}$$

Отметим, что переменные со значениями 0 или 1 называются *булевыми* или *двоичными.* Задача о рюкзаке имеет много экономических интерпретаций. Одна из них связана с распределением капиталовложений при следующих условиях.

Существует n проектов (способов) использования капиталовложений, объем которых равен b. Для каждого проекта известны расходы a_j и прибыль c_j от его реализации, $j=1, \ldots, n$. Проекты независимы друг от друга, так что общие расходы и прибыль определяются суммированием величин a_j и c_j по выбранным проектам, при

этом наличных ресурсов недостаточно для осуществления всех проектов. Требуется определить набор проектов, обеспечивающих в пределах имеющихся ресурсов максимальную общую прибыль от их реализации. Вводя булевы переменные, принимающие значения $x_j = 1$ или 0 в зависимости от того, выбирается j-й проект или нет, получаем постановку задачи в виде модели (3.1).

Естественное расширение модели (3.1) за счет включения нескольких ограничений приводит к *многомерной, или обобщенной, задаче о рюкзаке*

$$\max z = \sum_{j=1}^{n} c_j x_j; \quad \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i, \; i = 1, \ldots, m; \; x_j = 0 \text{ или } 1, \quad j = 1, \ldots, n. \tag{3.2}$$

Этой моделью, в частности, описывается задача распределения капиталовложений по m временным периодам с бюджетными ограничениями для каждого периода или одноперiодная задача с ограничениями на m видов ресурсов (деньги, материалы, энергия и т. д.).

Еще один вариант задачи о рюкзаке возникает, если все предметы (или проекты в задаче распределения капиталовложений) разбиты на группы, так что из одной группы выбирается не более одного элемента. Тогда модель (3.1) или (3.2) дополняется ограничениями

$$\sum_{j \in N_k} x_j \leqslant 1, \; k = 1, \ldots, p, \tag{3.3}$$

где p — количество групп; N_k — список индексов элементов k-й группы. Следует отметить, что наличие ограничений вида (3.3) способствует сокращению времени счета, поэтому пакеты прикладных программ булева программирования предоставляют пользователю возможность задавать группировку переменных, если даже она не предусмотрена постановкой задачи. Эвристические соображения, основанные на понимании экономического содержания задачи, обычно позволяют сделать это, в результате чего резко снижается время на получение достаточно хорошего начального решения.

Пример 3.1. Рассмотрим целочисленный вариант задачи о диете, которая обычно формулируется следующим образом. Известна минимальная суточная потребность b_i организма в каждом из питательных веществ $i = 1, \ldots, m$. Имеется n продуктов, описываемых двумя числами: c_j — *стоимость единицы j-го продукта*; a_{ij} — содержание i-го питательного вещества в единице j-го продукта, $i = 1, \ldots, m$;

Таблица 3.1

j	c_j	a_{1j}	a_{2j}	a_{3j}	a_{4j}
1	2	3	4	5	6
1	18	236	12.0	6.4	4.3
2	17	338	14.0	11.4	39.3
3	12	181	12.0	11.0	41.0
4	17	280	16.4	7.3	20.3
5	16	185	11.4	2.0	16.7
6	18	220	10.0	3.0	29.0
7	20	232	11.0	3.0	31.0
8	34	214	17.8	7.7	5.2
9	17	232	10.0	3.0	31.0
11	26	378	23.0	10.0	25.0
10	20	344	9.4	23.0	33.0
12	28	505	18.0	23.0	58.0
13	30	398	26.0	12.0	21.0
14	37	511	36.0	16.0	21.0
15	45	659	48.0	16.0	28.0
16	43	642	47.0	22.0	21.0
17	27	349	20.0	9.0	25.0
18	30	452	18.0	12.0	57.0
19	25	351	18.0	10.0	29.0
20	32	452	24.0	14.0	36.0
21	20	254	10.0	23.0	34.0
22	22	306	14.0	7.0	29.0
23	16	250	10.0	22.0	33.0
24	30	546	28.0	20.0	43.0
25	16	317	17.0	9.0	25.0
26	23	314	17.0	8.0	25.0
27	9	127	10.0	3.0	6.0
28	18	177	11.0	7.0	10.0
29	4	58	5.0	0.8	3.0
30	13	257	10.0	11.0	19.0
31	5	53	2.0	3.0	8.0
32	11	153	5.0	14.0	11.0
33	16	200	19.0	3.0	3.0
34	8	52	6.0	0.0	0.0
	b_i =	800	45	25	65

$j=1, \ldots, n$. Требуется определить, какое количество x_j каждого продукта $j=1, \ldots, n$ следует включить в диету, чтобы потребности в питательных веществах были удовлетворены при минимальной стоимости диеты. При этих условиях модель задачи имеет вид

$$\min z = \sum_{j=1}^{n} c_j x_j; \quad \sum_{j=1}^{n} a_{ij} x_j \geqslant b_i, \quad i = 1, \ldots, m; \quad x_j \geqslant 0, \quad j = 1, \ldots, n.$$

Одна из возможных целочисленных моделей возникает, если выбор диеты производится из дискретного множества — набора блюд. Пусть c_j и a_{ij} — соответственно цена j-го блюда и содержание i-го питательного вещества в нем, $j=1, \ldots, n$, $i=1, \ldots, m$. Требуется определить набор блюд, обеспечивающий потребность b_i в каждом веществе i при минимальной цене обеда.

Введем булевы переменные $x_j=1$, если блюдо j входит в набор, и $x_j=0$ в противном случае. Тогда постановка задачи принимает вид многомерной задачи о рюкзаке на минимум целевой функции:

$$\min z = \sum_{j=1}^{n} c_j x_j;\ \sum_{j=1}^{n} a_{ij} x_j \geqslant b_i,\ i = 1, \ldots, m;\ x_j = 0 \text{ или } 1,\ j = 1, \ldots, n. \tag{3.4}$$

Если через N_1, N_2, N_3 обозначить множество индексов соответственно первых, вторых и третьих блюд, то введение ограничений

$$\sum_{j \in N_k} x_j \leqslant 1,\ k = 1,\ 2,\ 3 \tag{3.5}$$

исключает варианты, когда в набор входит несколько блюд одного вида.

Так, модель (3.4) для задачи выбора обеда из 34 блюд (табл. 3.1), где первые блюда обозначены индексами от 1 до 9, вторые — от 10 до 28, третьи — от 29 до 34, требования по калорийности, содержанию жиров, белков и углеводов заданы соответственно значениями $b_1=800$, $b_2=45$, $b_3=25$, $b_4=65$, дает вариант обеда из двух вторых блюд ($x_{24}=x_{25}=1$, остальные $x_j=0$), $z_{\min}=46$. Введение условия (3.5) дает два варианта обеда из трех блюд ($x_2=x_{24}=x_{29}=1$, остальные $x_j=0$, или $x_4=x_{24}=x_{29}=1$, остальные $x_j=0$) со значением $z_{\min}=51$ для каждого.

Задачи размещения производства. Это один из наиболее распространенных видов задач целочисленного программирования. Мы рассмотрим два варианта постановки — с учетом и без учета транспортных расходов.

Пусть имеется m поставщиков однородной продукции с объемом производства у каждого a_i, $i=1, \ldots, m$, и n потребителей с потребностями у каждого b_j, $j=1, \ldots, n$. Через x_{ij} обозначим объем поставки от поставщика i потребителю j, а через c_{ij} — стоимость перевозки единицы продукции от i к j, $i=1, \ldots, m$, $j=1, \ldots, n$. Однако в отличие от обычной транспортной задачи будем предполагать, что рассматриваются не существующие заводы, а проектируемые к строительству. В результате решения задачи надо определить, какие из этих заводов должны быть построены, чтобы суммарные расходы на строительство и транспортировку были минимальны. Пусть f_i —

приведенные затраты на строительство i-го завода мощностью a_i, $i=1, \ldots, m$; y_i — целочисленная переменная, равная 1, если i-й завод строится, и 0 в противном случае.

Учитывая, что поставщиками могут быть только запланированные к строительству заводы, запишем условие, в соответствии с которым объем вывоза с любого завода не превосходит его мощности, в виде $\sum_{j=1}^{n} x_{ij} \leqslant a_i y_i$, $i = 1, \ldots, m$.

Условие удовлетворения потребителей остается таким же, как и в транспортной задаче, так как их состав и потребности b_j считаются заданными: $\sum_{i=1}^{m} x_{ij} = b_j$, $j = 1, \ldots, n$. Для разрешимости задачи, очевидно, должно быть выполнено условие $\sum_{i=1}^{m} a_i \geqslant \sum_{j=1}^{n} b_j$. В итоге математическая модель принимает вид

$$\min z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i;$$

$$a_i y_i - \sum_{j=1}^{n} x_{ij} \geqslant 0, \ i = 1, \ldots, m;$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \ j = 1, \ldots, n;$$

$$x_{ij} \geqslant 0, \ i = 1, \ldots, m, \ j = 1, \ldots, n;$$

$$y_i = 0 \text{ или } 1, \ i = 1, \ldots, m.$$

Это задача смешанного типа: в ней присутствуют непрерывные переменные x_{ij} и целые y_i. Ее целевая функция представляет сумму двух линейных форм, отражающих затраты, связанные с перевозками и строительством. При решении широко применяются приближенные методы, с помощью которых задача решается последовательно на минимум каждого из видов расходов.

Другую постановку получаем в предположении, что транспорные затраты на перевозку сырья и готовой продукции составляют относительно небольшую величину в суммарных затратах на продукцию и не могут влиять на выбор вариантов. Задача принимает вид

$$\min z = \sum_{k=1}^{N} \sum_{j=1}^{m_k} c_{jk} x_{jk};$$

$$\sum_{k=1}^{N} \sum_{j=1}^{m_k} a_{jk}^{i} x_{jk} \geqslant b^{i}, \quad i = 1, \ldots, L;$$

$$\sum_{j=1}^{m_k} x_{jk} \leqslant 1, \qquad k = 1, \ldots, N.$$

$$x_{jk} = 0 \text{ или } 1, \quad j = 1, \ldots, m_k, \; k = 1, \ldots, N.$$

Здесь N —количество предприятий; k — номер предприятия; j — номер варианта строительства или реконструкции предприятия; m_k — количество вариантов строительства или реконструкции предприятия k; i — вид выпускаемой продукции; L — количество видов продукции; a_{jk}^{i} — объем выпуска i-го вида продукции на k-м предприятии при j-м варианте; b^{i} — потребность в i-м виде продукции (*или задание по выпуску i-й продукции*); c_{jk} — приведенные затраты (текущие и капитальные), соответствующие j-му варианту развития k-го предприятия; $x_{jk}=1$, если для k-го предприятия выбирается j-й вариант, и $x_{jk}=0$ в противном случае.

Задачи с альтернативными условиями. Мы рассмотрели случаи, когда задача целочисленного программирования возникает в силу свойств моделируемого процесса, которые требуют, чтобы некоторые или все параметры, описывающие процесс, принимали целочисленные значения. Между тем целочисленная задача может быть сформулирована не только путем постановки задачи линейного программирования с последующим включением требования целочисленности переменных. С ней мы встречаемся, когда исходная математическая модель, не будучи целочисленной, обладает некоторыми специальными свойствами, которые позволяют свести ее к задаче целочисленного программирования. В качестве примера приведем задачи, постановка которых связана с рассмотрением альтернативных условий.

Предположим, разрабатывается проект металлорежущего станка и имеется возможность выбора различных вариантов его производительности, стоимости и т. д. Если таких вариантов два или три, то выбор может быть сделан при непосредственном сравнении технических и технико-экономических показателей различных вариантов. Если же мы проектируем не отдельный объект, а

систему взаимосвязанных объектов, имеющих сложные логические взаимообусловливающие связи (например, одни проектные решения исключают некоторые другие или, наоборот, предполагают наличие некоторых других), и если по всей совокупности проектов существуют общие ограничения по ресурсам, то непосредственное сравнение затруднительно. В этом случае для выбора лучших вариантов проекта может быть построена задача целочисленного программирования.

Пусть, например, в некоторой задаче линейного программирования необходимо отразить дополнительное условие, требующее, чтобы только одна из двух переменных $x_1 \geqslant 0$ и $x_2 \geqslant 0$ могла быть положительной. Введем булеву переменную y и положим, что $y=1$ соответствует условию $x_1 \geqslant 0$, $x_2=0$, а $y=0$ — условию $x_1=0$, $x_2 \geqslant 0$. Нетрудно проверить, что эти условия будут выполнены, если к ограничениям исходной задачи добавить ограничения вида $x_1 \leqslant M_1 y$, $x_2 \leqslant M_2(1-y)$; $y=0$ или 1, где M_1, M_2 — соответственно верхние границы переменных x_1 и x_2.

Рассмотрим более сложный пример, где производственная задача

$$\max z = \sum_{j=1}^{n} c_j x_j; \tag{3.6}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i, \quad i = 1, \ldots, m; \tag{3.7}$$

$$x_j \geqslant 0, \quad j = 1, \ldots, n \tag{3.8}$$

дополняется условием, что если j-е изделие производится, то объем его выпуска x_j должен быть не меньше величины L_j и не больше величины M_j, $j=1, \ldots, n$, т. е. условием вида

$$\text{или } x_j=0 \text{ или } L_j \leqslant x_j \leqslant M_j, \; j=1, \ldots, n. \tag{3.9}$$

Для того чтобы учесть логическое условие (3.9), введем n булевых переменных y_j и составим ограничения

$$x_j \leqslant M_j y_j, \quad j=1, \ldots, n; \quad x_j \geqslant L_j y_j, \quad j=1, \ldots, n; \; y_j=0 \text{ или } 1, \quad j=1, \ldots, n. \tag{3.10}$$

Рассмотрим теперь целочисленную задачу (3.6), (3.7), (3.10) с булевыми переменными и покажем, что она эквивалентна задаче (3.6) — (3.8), дополненной логическим условием (3.9). Для этого нужно показать, что обе зада-

чи имеют одинаковые множества допустимых решений.

Действительно, пусть $\mathbf{x}=(x_1, x_2, \ldots, x_n)$ — решение задачи на допустимом множестве, которое задано ограничениями (3.7) — (3.8) и условием (3.9), и пусть в этом решении некоторая переменная $x_k=0$. Тогда для $y_k=0$ значения x_k, y_k удовлетворяют ограничениям (3.10) и являются, таким образом, решением целочисленной задачи с допустимой областью (3.7), (3.10). Если $L_k \leqslant x_k \leqslant M_k$, то значения x_k, y_k удовлетворяют ограничениям (3.10) при $y_k=1$ и также являются решением целочисленной задачи с областью (3.7), (3.10).

Пусть теперь $\mathbf{x}$, $\mathbf{y}$ — решение целочисленной задачи с допустимой областью (3.7), (3.10). Тогда вектор $\mathbf{x}$ неотрицателен, а переменные y_j равны 0 или 1. В зависимости от значения $y_j=0$ или $y_j=1$ в ограничениях (3.10) выполняется или условие $x_j \leqslant 0$, или $L_j \leqslant x_j \leqslant M_j$, что отвечает требованиям (3.9). Отсюда, $\mathbf{x}$ есть решение задачи (3.7) — (3.8), дополненной ограничениями (3.9). Таким образом, вектор $\mathbf{x}$, допустимый для одной задачи, является допустимым и для другой. Поскольку целевая функция в обоих случаях одна и та же, доказательство эквивалентности задачи (3.6) — (3.8), дополненной логическим условием (3.9), и целочисленной задачи линейного программирования (3.6), (3.7), (3.10) с булевыми переменными завершено. Аналогично могут быть учтены и другие логические условия.

Пример 3.2. Построить целочисленную модель задачи (1.7), (1.8) из примера 1.2 при дополнительном условии

$$\text{или } x_2 \leqslant 30 \quad \text{или} \quad x_2 \geqslant 75. \tag{3.11}$$

Введем булеву переменную y и положим, что $y=0$ отвечает первой альтернативе (т. е. $x_2 \leqslant 30$), $y=1$ — второй (т. е. $x_2 \geqslant 75$). Анализ ограничений (1.8) дает для x_2 значение верхней границы $x_2 \leqslant 90$. Тогда модель задачи, удовлетворяющей условиям (1.8) и (3.11), примет вид

$$\max z = 3x_1 + 2x_2; \quad x_1 + 3x_2 \leqslant 270;$$
$$4x_1 + 6x_2 \leqslant 600, \quad 3x_1 + x_2 \leqslant 240;$$
$$x_2 - 60y \leqslant 30; \quad x_2 - 75y \geqslant 0;$$
$$x_1, x_2 \geqslant 0; \quad y = 0 \text{ или } 1.$$

Действительно, при $y=0$ дополнительные ограничения $x_2 - 60y \leqslant 30$ и $x_2 - 75y \geqslant 0$ обеспечивают переменной x_2 значения $0 \leqslant x_2 \leqslant 30$, что соответствует первой альтернативе. При $y=1$ получаем условие $x_2 \leqslant 90$, не стесняющее выбор x_2, и условие $x_2 \geqslant 75$, выражающее вторую альтернативу.

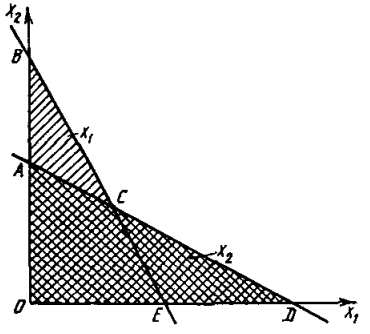

Рис. 3.1.

Задачи с альтернативными ограничениями. Эти задачи встречаются, когда для допустимости переменных достаточно, чтобы они удовлетворяли лишь части всех ограничений, причем заранее нельзя указать, какие именно из ограничений должны быть удовлетворены.

Область допустимых решений в таких задачах невыпукла. Так, если имеются два ограничения и необходимо удовлетворить им обоим, то получим (рис. 3.1) выпуклую область $OACE = X_1 \cap X_2$. Если же достаточно, чтобы переменные удовлетворяли одному из них (либо первому, либо второму), областью допустимых решений становится невыпуклый многоугольник $OBCD = X_1 \cup X_2$.

Прямой путь решения таких задач, в которых из m должны быть выполнены $q < m$ ограничений, состоит в следующем. Отбросим $m-q$ произвольных ограничений и решим получившуюся задачу линейного программирования при оставшихся q ограничениях. Затем перейдем к новому набору ограничений и будем повторять так, пока не переберем все сочетания из m ограничений по q. То из решений, которое доставляет наибольшее значение целевой функции, и будет оптимальным. При этом потребуется решить

$$C_m^q = \frac{m!}{q!\,(m-q)!}$$

различных задач линейного программирования.

Другой путь решения основан на предварительном преобразовании исходной задачи в частично целочисленную с выпуклым многогранником ограничений. Это может быть выполнено следующим образом. Пусть в исходной задаче существует m ограничений вида $\sum_{i=1}^{m} a_{ij}x_j \leqslant b_i$. Положим далее, что для некоторой области $S \supset \bigcup_i X_i$, где X_i — область допустимых векторов x, задаваемая i-м ограничением в положительном октанте, известны числа M_i, $i = 1, \ldots, m$,

представляющие верхние границы функций $f_i(\mathbf{x}) = \sum_{j=1}^{n} a_{ij}x_j$, т. е. $\sum_{j=1}^{n} a_{ij}x_j \leqslant M_i$ для всех $\mathbf{x} \in S$, $i = 1, \ldots, m$. Введем теперь вспомогательные переменные y_i, которые могут принимать значения 0 и 1, и рассмотрим систему ограничений

$$\sum_{j=1}^{n} a_{1j}x_j - M_1y_1 \leqslant b_1;$$

$$\sum_{j=1}^{n} a_{2j}x_j - M_2y_2 \leqslant b_2;$$

. . . . . .

$$\sum_{j=1}^{n} a_{mj}x_j - M_my_m \leqslant b_m;$$

$$\sum_{i=1}^{m} y_i \geqslant m - q;$$

$$y_i = 0 \text{ или } 1,\ i = 1, \ldots, m$$

при целевой функции $z = \sum_{j=1}^{n} c_jx_j$. Дополнительные переменные y_i в целевую функцию не входят.

Покажем, что данная система ограничений отражает условия исходной задачи, в которой требуется, чтобы выполнялось не более q ограничений из m. В самом деле, если в решении $(\mathbf{x}, \mathbf{y})$ данной задачи некоторая переменная y_i принимает значение 0, это значит, что соответствующее ограничение учитывается в решении, так как совпадает с требуемым $\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i$. Если же $y_i = 1$, то i-е ограничение сводится к автоматически выполняющемуся соотношению $\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i + M_i$ и, следовательно, не влияет на выбор переменных. Таким образом, количество ограничений, учтенных при решении задачи, совпадает с количеством вспомогательных переменных y_i, равных нулю. Остальные переменные y_i, равные 1, определяют ограничения, не учитываемые при решении. В соответствии с условием задачи их сумма должна быть не меньше $m - q$.

Из этого становится ясным написание предпоследнего ограничения рассматриваемой системы. Если в решении требуется выполнение ровно q из m ограничений, то неравенство в нем следует заменить равенством, т. е. $\sum_{i=1}^{m} y_i = m - q$.

О решениях целочисленных задач. Рассмотренные постановки отличаются от задач линейного программирования только дополнительным условием целочисленности всех или части переменных. В связи с этим возникает предположение, что целочисленные решения могут быть получены из решений обычных задач линейного программирования или непосредственно, или округлением нецелых значений в ответе. К сожалению, эти предположения редко осуществляются на практике, в связи с чем для целочисленного программирования требуются собственные методы, обычно более сложные, чем методы линейного программирования. Это связано с дискретностью множества допустимых решений целочисленных задач.

Так, в двумерном случае планы целочисленной задачи определяются конечным множеством целых точек, лежащих внутри и на границе многоугольника ограничений обычной задачи линейного программирования (рис. 3.2). Методы линейного программирования осуществляют перебор только крайних точек. Следовательно, они гарантируют отыскание оптимального целочисленного решения, если все крайние точки выпуклого многоугольника (в общем случае — многогранника) целочисленны. Такими свойствами обладает узкий класс задач — транспортная и ряд близких к ней (о назначении, о максимальном потоке, другие задачи на сетях), где целочисленные решения обеспечены всегда, если условия представлены в целых числах.

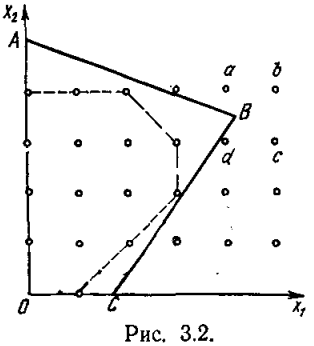

Рис. 3.2.

Если крайние точки многогранника нецелочисленны, то найти оптимальный план целочисленной задачи методами линейного программирования в общем случае нельзя. Не приходится надеяться и на то, что округление решения непрерывной задачи может быть надежным способом приближенного

решения целочисленной задачи. Так, на рис. 3.2 оптимум задачи линейного программирования лежит в точке B. Однако ни одна из ближайших к ней целочисленных точек a, b, c, d не является допустимой. Можно построить и другие примеры, показывающие ограниченность метода округления. Так, решая задачу о рюкзаке

$$\max z = 58x_1 + 74x_2 + 60x_3 + 22x_4; \quad 18x_1 + 24x_2 + 20x_3 + 15x_4 \leqslant 60; \quad 0 \leqslant x_j \leqslant 1 \text{ целые}, \ j = 1, \ldots, 4$$

без условия целочисленности, находим оптимальный план $x_1 = 1$, $x_2 = 1$, $x_3 = 0{,}9$, $x_4 = 0$, $z = 186$. Можно убедиться, что никакие варианты округления переменной x_3 не дают оптимального целочисленного решения, которое имеет вид $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$, $z = 156$ и значительно отличается от оптимального плана нецелочисленной задачи. Таким образом, ограничения целочисленности являются существенной частью задачи, указывающей на необходимость применения специальных методов для ее решения. Их укрупненно можно разделить на три группы: методы отсекающих плоскостей или отсечения; комбинаторные; приближенные.

Основную идею *метода отсечения* поясним на рис. 3.2, где сплошными линиями ограничена допустимая область X обычной задачи линейного программирования, а штриховыми — выпуклая оболочка X' точек этой области, удовлетворяющих условию целочисленности. Все крайние точки указанной оболочки целочисленны, и если решить на ней задачу линейного программирования, то одновременно будет решена и задача целочисленного программирования. Однако задача построения выпуклой оболочки в общем случае не проще исходной целочисленной задачи. Метод отсечения представляет собой процедуру построения оболочки X' за счет последовательного отбрасывания некоторых подмножеств области X, не содержащих целых точек множества X'.

Комбинаторными методами в общем случае называются методы анализа дискретных множеств, основанные на выделении и упорядочении последовательностей (выборок, комбинаций) подмножеств. Их общая идея состоит в замене полного перебора всех подмножеств частичными переборами. Это осуществляется за счет обнаружения и исключения некоторых подмножеств, заведомо не содержащих искомого решения, и сужения области перспективных вариантов.

Методы отсечения и комбинаторные являются точными, т. е. обеспечивают отыскание оптимального целочисленного решения за конечное число шагов. Но даже для не очень больших задач скорость сходимости этих методов оказывается недостаточной при возможностях современных ЭВМ. Поэтому на практике оправдано использование *приближенных методов* решения. Их характерная черта — максимальный учет специфики решаемой задачи, который с одной стороны позволяет упростить поиск решения, а с другой, ограничивает сферу применения метода. Поэтому мы не будем останавливаться на каком-либо из них и рассмотрим методы только первых двух типов.

3.2. МЕТОД ОТСЕЧЕНИЯ

Общая схема. Рассмотрим полностью целочисленную задачу

$$\max z = \mathbf{cx}; \tag{3.12}$$

$$A\mathbf{x} = \mathbf{b}; \tag{3.13}$$

$$\mathbf{x} \geqslant 0; \tag{3.14}$$

$$x_j \text{ — целые, } j = 1, \ldots, n \tag{3.15}$$

и связанную с ней задачу линейного программирования (3.12) — (3.14), которые будем обозначать как задачи I и L соответственно. Предположим, что все коэффициенты a_{ij}, b_i, c_j — целые числа, область допустимых решений задачи L ограничена, задача I имеет допустимое решение. В соответствии с основной идеей метода требование целочисленности заменяется добавлением новых ограничений, последовательно сужающих допустимую область задачи L, но не затрагивающих точек допустимой области задачи I.

Вначале решается задача $L_0 = L$. Если ее решение удовлетворяет условиям целочисленности, то оно является решением и задачи I. В противном случае к ограничениям (3.13), (3.14) задачи L добавляется новое линейное ограничение, называемое *отсечением*, которое переводит задачу L_0 в задачу L_1 с допустимой областью, обладающей двумя свойствами. Эти свойства таковы: во-первых, нецелочисленная оптимальная точка задачи L_0 не принадлежит области L_1; во-вторых, все целочисленные точки L_0 сохраняются в допустимой области L_1.

После этого решается задача L_1. Затем в случае необходимости добавляется еще одно ограничение и решается задача L_2 и т. д.

Геометрически добавление каждого линейного ограничения отвечает построению гиперплоскости, которая отсекает от многогранника решений предыдущей задачи оптимальную точку с дробными координатами, но не затрагивает ни одной из целочисленных точек этого многогранника. Таким образом, вместо задачи I решается последовательность задач линейного программирования $L_0=L$, L_1, L_2, ..., пока решение одной из них не окажется целочисленным. Оно будет одновременно и оптимальным решением задачи I.

Построение отсечения. Опишем алгоритм для полностью целочисленных задач (первый алгоритм Гомори). Предположим, что в оптимальном решении задачи L базисными являются первые m переменных. Тогда имеет место базисная система вида

$$x_i = b_i - \sum_{j=m+1}^{n} a_{ij}x_j, \quad i = 1, \ .., \ m. \tag{3.16}$$

Если положить $x_j=0$ для $j>m$, то получим базисные компоненты оптимального решения $x_i=b_i$, $i=1$, ..., m. Предположим, b_k — нецелое и, кроме того, дробным является по крайней мере один из элементов a_{kj} k-й строки. Представим b_k и все a_{kj} в виде двух составляющих: $b_k=[b_k]+f_k$; $a_{kj}=[a_{kj}]+f_{kj}$, $j=m+1$, ..., n. Здесь скобки [ ] означают целую часть числа; $[b_k]$ — наибольшее целое, не превышающее b_k. Так, для $b_k=15/7$ имеем $[b_k]=[15/7]=2$, $f_k=1/7$, для $b_k=-15/7$ имеем $[b_k]=[-15/7]=-3$, $f_k=6/7$. Для любого числа по определению $0\leqslant f_k<1$.

С учетом введенных обозначений k-е уравнение системы (3.16) примет вид

$$x_k = [b_k] + f_k - \sum_{j=m+1}^{n} [a_{kj}]\, x_j - \sum_{j=m+1}^{n} f_{kj}x_j. \tag{3.17}$$

Разобьем слагаемые в правой части выражения (3.17) на две группы:

$$[b_k] - \sum_{j=m+1}^{n} [a_{kj}]\, x_j; \tag{3.18}$$

$$f_k - \sum_{j=m+1}^{n} f_{kj}x_j. \tag{3.19}$$

Выражение (3.18) при целых значениях x_j всегда целочисленно. Поэтому и выражение (3.19) должно быть целочисленным, как только будет целочисленным x_k в выражении (3.17). Кроме того, из $f_{kj} \geqslant 0$, $x_j \geqslant 0$ следует $\sum_{j=m+1}^{n} f_{kj}x_j \geqslant 0$. Отсюда получаем $f_k - \sum_{j=m+1}^{n} f_{kj}x_j \leqslant f_k < 1$. Теперь, учитывая целочисленность выражения (3.19) при всех целочисленных x_j, заключаем, что ограничение

$$f_k - \sum_{j=m+1}^{n} f_{kj}x_j \leqslant 0 \tag{3.20}$$

не исключает ни одного допустимого целочисленного решения, кроме того, оно не удовлетворяется оптимальным решением задачи L, где $x_j = 0$, $j = m+1, \ldots, n$, так как $f_k > 0$. Отсюда неравенство (3.20), которое для удобства запишем в виде

$$\sum_{j=m+1}^{n} f_{kj}x_j \geqslant f_k, \tag{3.21}$$

является отсечением.

Процесс включения в условия задачи новых ограничений-отсечений строится следующим образом. Вычитая из левой части неравенства (3.21) дополнительную неотрицательную переменную x_{n+1} и умножая обе его части на -1, получаем отсечение в виде уравнения

$$-\sum_{j=m+1}^{n} f_{kj}x_j + x_{n+1} = -f_k$$

с базисной переменной x_{n+1}. Добавив это уравнение к последней симплекс-таблице предыдущей задачи, решим новую задачу с помощью двойственного симплекс-метода. Если решение окажется нецелочисленным, опять используем последнюю симплекс-таблицу для построения нового ограничения, поступая аналогично до тех пор, пока на очередном шаге не будет получено целочисленное решение.

Хотя сходимость метода отсечения доказана, опыт практического применения и результаты экспериментальных расчетов показали его довольно невысокую эффективность, меняющуюся от задачи к задаче трудно предсказуемым образом. В частности, известны примеры

задач, решение которых методом полного перебора всех допустимых значений переменных требует меньше времени, чем методом отсечения.

Пример 3.3. Решить методом отсечения задачу

$$\max z=6x_1+7x_2; \quad 6x_1+4x_2\leqslant 36; \quad 5x_1+6x_2\leqslant 40;$$

$$x_1,\ x_2\geqslant 0;\ x_1,\ x_2 \text{ — целые.}$$

Вводя дополнительные неотрицательные переменные x_3, x_4, строим каноническую задачу линейного программирования

$$\max z=6x_1+7x_2;\ 6x_1+4x_2+x_3=36;$$

$$5x_1+6x_2+x_4=40;\ x_1,\ x_2,\ x_3,\ x_4\geqslant 0$$

и решаем ее без учета требования целочисленности. Из табл. 3.2, где даны результаты последней итерации симплекс-метода, получаем ее оптимальное решение в виде $x_1=3\frac{1}{2}$, $x_2=3\frac{3}{4}$, $z_{\max}=47\frac{1}{4}$.

Таблица 3.2

c_B	x_B	b	6	7	0	0
			a^1	a^2	a^3	a^4
6	x_1	7/2	1	0	6/16	—2/8
7	x_2	15/4	0	1	—5/16	3/8
	Δ	189/4	0	0	1/16	9/8

Поскольку обе переменные x_1, x_2 нецелочисленны, можно построить отсечение (3.21) по любой из них. Первая строка дает отсечение $6/16x_3+6/8x_4\geqslant 1/2$, а после введения дополнительной неотрицательной переменной x_5 получаем: $-6/16x_3-6/8x_4+x_5=-1/2$. Из второй строки находим $11/16x_3+3/8x_4\geqslant 3/4$ или $-11/16x_3-3/8x_4+x_5=-3/4$. Выберем первое из отсечений и допишем его к итоговой симплекс-таблице исходной задачи, после чего она примет вид табл. 3.3 с базисными переменными x_1, x_2, x_5. Здесь Δ-строка перенесена в верхнюю часть таблицы для удобства включения новых ограничений.

Таблица 3.3

c_B	x_B	b	6	7	0	0	0
			a^1	a^2	a^3	a^4	a^5
	Δ	189/4	0	0	1/16	9/8	0
6	x_1	7/2	1	0	6/16	—2/8	0
7	x_2	15/4	0	1	—5/16	3/8	0
0	x_5	—1/2	0	0	**—6/16**	—6/8	1

В соответствии с правилами двойственного симплекс-метода переменная x_3 вводится в базис вместо переменной x_5. Выполнив одну итерацию двойственного симплекс-метода, придем к оптимальному решению, в котором переменная $x_2=4\frac{1}{6}$ еще нецелочисленна (табл. 3.4, верхняя часть): Она дает отсечение вида $1/6x_5 \geqslant 1/6$ или после введения дополнительной переменной $x_6 \geqslant 0$: $-1/6x_5+x_6=-1/6$. Включив это отсечение в табл. 3.4 и выполнив одну итерацию двойственного симплекс-метода, вводим в базис вместо переменной x_6 переменную x_5 и получаем оптимальное решение $x_1=2$, $x_2=5$, $x_3=4$, $x_4=0$, $x_5=0$, $x_6=0$, $z=47$ задачи линейного программирования. Оно является целочисленным и, следовательно, дает оптимальное целочисленное решение исходной задачи: $x_1=2$, $x_2=5$, $z=47$.

Таблица 3.4

c_B	x_B	b	6	7	0	0	0	0
			a^1	a^2	a^3	a^4	a^5	a^6
	Δ	283/6	0	0	0	1	1/6	0
6	x_1	3	1	0	0	—1	1	0
7	x_2	25/6	0	1	0	1	—5/6	0
0	x_3	4/3	0	0	1	2	—8/3	0
0	x_6	—1/6	0	0	0	0	—1/6	1
	Δ	47	0	0	0	1	0	1
6	x_1	2	1	0	0	—1	0	6
7	x_2	5	0	1	0	1	0	—5
0	x_3	4	0	0	1	2	0	—16
0	x_5	1	0	0	0	0	1	—6

Таким образом, для получения целочисленного решения пришлось ввести два отсечения, которые можно записать в виде $x_5 = -1/2 + 6/16x_3 + 6/8x_4$ и $x_6 = -1/6 + 1/6x_5$. Их геометрический смысл будет нагляднее, если выразить переменные x_5 и x_6 через переменные x_1, x_2 исходной задачи. Тогда первое отсечение примет вид $x_5 = -1/2 + 6/16\,(36 - 6x_1 - 4x_2) + 6/8\,(40 - 5x_1 - 6x_2) = 43 - 6x_1 - 6x_2$ или $x_1 + x_2 \leqslant 7\frac{1}{6}$. Второе отсечение выразится как $x_6 = -1/6 + 1/6\,(43 - 6x_1 - 6x_2) = 7 - x_1 - x_2$ или $x_1 + x_2 \leqslant 7$. Первым из этих отсечений была исключена оптимальная нецелочисленная точка $x_1 = 3\frac{1}{2}$, $x_2 = 3\frac{3}{4}$ задачи L_0, вторым — оптимальное нецелочисленное решение $x_1 = 3$, $x_2 = 4\frac{1}{6}$ задачи L_1.

Пример 3.4. Решить методом отсечения задачу

$$\max z = x_1 - 3x_2 + 3x_3;\quad 4x_1 - 3x_2 + x_4 = 2;\quad -3x_1 + 2x_2 + x_3 + x_5 = 3;$$

$$2x_1 + x_2 - x_3 + x_6 = 4;\quad x_j \geqslant 0,\ j = 1, \ldots, 6;\quad x_j \text{ — целые},\ j = 1, \ldots, 6.$$

В табл. 3.5 дано оптимальное решение задачи без учета требований целочисленности.

Таблица 3.5

c_B	x_B	b	1	−3	3	0	0	0
			a^1	a^2	a^3	a^4	a^5	a^6
1	x_1	2/4	1	−3/4	0	1/4	0	0
3	x_3	18/4	0	−1/4	1	3/4	1	0
0	x_6	30/4	0	9/4	0	1/4	1	1
	Δ	14	0	6/4	0	10/4	3	0

Таблица 3.6

c_B	x_B	b	1	−3	3	0	0	0	0
			a^1	a^2	a^3	a^4	a^5	a^6	a^7
	Δ	14	0	6/4	0	10/4	3	0	0
1	x_1	2/4	1	−3/4	0	1/4	0	0	0
3	x_3	18/4	0	−1/4	1	3/4	1	0	0
0	x_6	30/4	0	9/4	0	1/4	1	1	0
0	x_7	−2/4	0	−1/4	0	−1/4	0	0	1
	Δ	11	0	0	0	1	3	0	6
1	x_1	2	1	0	0	1	0	0	−3
3	x_3	5	0	0	1	1	1	0	−1
0	x_6	3	0	0	0	−2	1	1	9
−3	x_2	2	0	1	0	1	0	0	−4

Отсечение, порождаемое первой строкой, имеет вид: $1/4x_2+1/4x_4\geqslant 2/4$. Вводя дополнительную переменную x_7, получаем уравнение $-1/4x_2-1/4x_4+x_7=-2/4$. Допишем его к табл. 3.5, после чего она примет вид табл. 3.6 (верхняя часть) с базисными переменными x_1, x_3, x_6, x_7.

В соответствии с правилами двойственного симплекс-метода переменная x_2 вводится в базис вместо переменной x_7. Выполнив одну итерацию двойственного симплекс-метода, получаем оптимальное решение $x_1=2$, $x_2=2$, $x_3=5$, $x_4=x_5=0$, $x_6=3$, $x_7=0$, $z=11$. Это решение целочисленно, откуда следует, что оно является оптимальным решением и исходной целочисленной задачи.

3.3. МЕТОД ВЕТВЕЙ И ГРАНИЦ

Общие понятия. Рассмотрим метод применительно к полностью целочисленной задаче вида

$$\max z = \mathbf{cx}; \tag{3.22}$$

$$A\mathbf{x}\leqslant\mathbf{b},\ \mathbf{x}\geqslant 0; \tag{3.23}$$

x_j — целые для всех j (3.24)

с m ограничениями и n переменными.

Предположение полной целочисленности не является существенным и служит для упрощения рассуждений. Обозначим задачу (3.22) — (3.24) через I, ее допустимое множество — через P. Будем считать выпуклую область X, заданную условиями $A\mathbf{x} \leqslant \mathbf{b}$, $\mathbf{x} \geqslant \mathbf{0}$, ограниченной. Тогда $P \subset X$ — конечное множество.

Метод ветвей и границ предусматривает замену задачи I последовательностью *подзадач* $I_1, I_2, \ldots, I_n$, заданных на непересекающихся подмножествах $P_1, P_2, \ldots, P_n$, таких, что $P_1 \cup P_2 \cup \ldots \cup P = P$, $P_i \cap P_j = \varnothing$ при $i \neq j$. Совокупность определенных таким образом подмножеств называется *разбиением* множества P, процесс построения подмножеств — *ветвлением*, а задачи $I_1, I_2, \ldots, I_n$ — *задачами-кандидатами* (ЗК).

Задачи-кандидаты I_j заменяются некоторыми вспомогательными задачами, более простыми для решения. Обычный способ построения вспомогательной задачи состоит в ослаблении (релаксации) некоторых условий исходной задачи без изменения ее целевой функции. В итоге вместо ЗК с допустимой областью P_j получаем ее *релаксацию* (ЗКР) с допустимой областью X_j, где $P_j \subset X_j$.

Решение вспомогательной задачи на подмножестве X_j дает *верхнюю границу* целевой функции на подмножестве P_j, т. е. число $\varphi(P_j)$, такое, что $\varphi(P_j) \geqslant z(\mathbf{x})$ для всех $\mathbf{x} \in P_j$. Если при этом оказывается, что элемент $\mathbf{x}^* \in X_j$, на котором достигается верхняя граница $\varphi(P_j)$, таков, что $\mathbf{x}^* \in P_j$, то план $\mathbf{x}^*$ объявляется *решением-претендентом* для задачи I, а значение его целевой функции $z^* = \mathbf{c}\mathbf{x}^*$ — *рекордом*. Если затем в процессе ветвления и оценивания подмножеств отыскивается план $\mathbf{x} \in P$ с бóльшим значением целевой функции, он становится новым планом-претендентом, а связанное с ним значение целевой функции — новым рекордом. Процесс заканчивается, когда имеются некоторое разбиение $\{P_i\}$, $i = 1, \ldots, n$ и план-претендент $\mathbf{x}^* \in P_k$, такой, что

$$z(\mathbf{x}^*) = \varphi(P_k) \geqslant \varphi(P_i), \quad i = 1, \ldots, n. \tag{3.25}$$

В соответствии со смыслом верхних границ этот план оптимален; условие (3.25) — *признак оптимальности плана*.

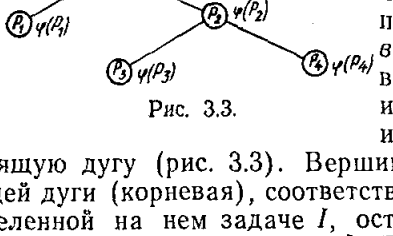

Рис. 3.3.

Процесс ветвления подмножеств может быть представлен в виде *дерева* — специального графа, все вершины которого, за исключением одной, имеют единственную входящую дугу (рис. 3.3). Вершина, не имеющая входящей дуги (корневая), соответствует множеству P и определенной на нем задаче I, остальные вершины — подмножествам P_j и задачам I_j. Предполагается, что дуги ориентированы сверху вниз.

Каждая вновь полученная в процессе ветвления ЗК считается входящей в *список задач-кандидатов*. Она находится в нем, пока не будет *прозондирована* — удалена из списка как неперспективная без дальнейшего ветвления или *разветвлена* — заменена в списке новыми подзадачами для дальнейшего оценивания.

ЗК считается прозондированной, если в результате решения ЗКР выполнен один из следующих трех *критериев зондирования.*

1. ЗКР не имеет допустимого решения. В таком случае следует исключить ЗК из дальнейшего рассмотрения, так как она тоже недопустима.

2. Оптимальное значение целевой функции ЗКР меньше или равно текущему рекорду z^*. В этом случае ни на одном из допустимых решений ЗК не может быть достигнуто значение целевой функции большее, чем имеющийся рекорд; следовательно, ЗК может быть исключена из дальнейшего рассмотрения.

3. Оптимальное решение $\mathbf{x}$ ЗКР целочисленно. В этом случае дальнейшее ветвление ЗК не даст целочисленного решения с бо́льшим, чем $z(\mathbf{x})$, значением целевой функции. Если $z(\mathbf{x}) > z^*$, то $\mathbf{x}$ становится решением-претендентом $\mathbf{x}^*$, а $z(\mathbf{x})$ — новым рекордом z^*. В противном случае решение-претендент и нижняя граница остаются без изменения. В обоих случаях ЗК исключается из списка.

Таким образом, задача-кандидат, удовлетворяющая любому из этих критериев, полностью исследована и удаляется из списка без дальнейшего ветвления. В списке остаются только те ЗК, для которых при решении оценочных ЗКР получены нецелочисленные векторы, достав-

ляющие целевой функции ЗКР значение, большее существующего рекорда z^*.

Анализ задач текущего списка, выполняемый по результатам решения их релаксаций, будет приводить или к удалению из списка бесперспективных ЗК, или к дальнейшему ветвлению некоторых из них на подзадачи *(задачи-потомки)*. Процесс окончится, когда в списке не останется задач для ветвления. Конечность процесса гарантирована, так как по определению допустимое множество задачи I конечно.

Прежде чем перейти к детализации общей схемы, отметим, что метод ветвей и границ обладает достаточной

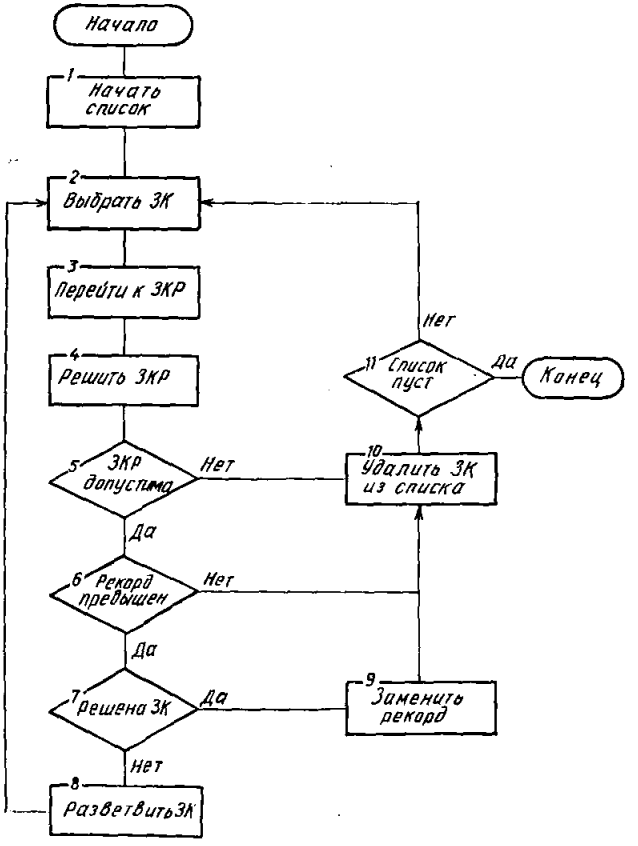

Рис. 3.4

универсальностью, позволяющей применять его не только в целочисленном программировании, но и в решении различных комбинаторных задач дискретного программирования. В связи с решением одной из них — задачи коммивояжера — он и был впервые использован. Достоинством алгоритмов типа ветвей и границ является то, что в ходе вычислений они дают последовательность целочисленных планов-претендентов, лучший из которых может быть взят в качестве приближенного решения, если счет прекращается до получения оптимального решения.

Вычислительная схема. Опишем основные операции алгоритма, укрупненная блок-схема которого представлена на рис. 3.4.

1. Включаем в список исходную задачу I, устанавливаем начальное значение рекорда равным $-\infty$.

2. Выбираем одну из задач текущего списка в качестве задачи-кандидата.

3. Заменяем ЗК ее релаксацией ЗКР. Релаксацией целочисленной задачи (3.22) — (3.24) будет непрерывная задача (3.22), (3.23).

4. Находим решение ЗКР и переходим к его анализу с использованием критериев зондирования.

5. Если ЗКР не имеет допустимого решения (критерий 1), переходим к шагу 10, где исключаем ЗК из списка.

6. Если максимум целевой функции ЗКР не больше рекорда (критерий 2), переходим к шагу 10, где исключаем ЗК из списка.

7. Если оптимальное решение ЗКР целочисленно (критерий 3), оно является допустимым для ЗК. Переходим к шагу 9. Если решение ЗКР нецелочисленно, переходим к шагу 8.

8. Разветвляем ЗК, заменяем ее в списке задачами-потомками и переходим к шагу 2.

9. Запоминаем решение ЗК, полученное в результате решения ЗКР, в качестве нового решения-претендента, заменяем рекорд и переходим к шагу 10, где исключаем из списка эту ЗК и все ЗК, для которых верхняя граница не больше нового рекорда.

10. По результатам зондирования ЗК удаляется из текущего списка.

11. Проверяем, пуст ли список задач. Если пуст, вычисления окончены. Если при этом рекорд равен $-\infty$, задача I не имеет допустимого решения. Если рекорд ко-

нечен, то имеющиеся решение-претендент и соответствующий ему рекорд являются оптимальным решением задачи I.

При описании схемы не определены способ ветвления ЗК (шаг 8) для получения задач-потомков и правила выбора ЗК из текущего списка (шаг 2). В связи с различными способами выполнения этих шагов существуют различные варианты метода.

Обычным приемом, используемым при ветвлении некоторой задачи-кандидата I_j на подзадачи, является *дихотомия* (деление на две части) допустимого множества P_j на два подмножества P_{j1} и P_{j2}. Для этого используется одна из нецелочисленных переменных, допустим, $x_k = [b_k] + f_k$, с помощью которой строятся две новые задачи I_{j1} и I_{j2}, где I_{j1} — задача I_j, дополненная условием $x_k \leqslant [b_k]$; I_{j2} — задача I_j, дополненная условием $x_k \geqslant [b_k] + 1$. Введением этих условий область допустимых решений задачи I_j разбивается на два непересекающихся подмножества P_{j1} и P_{j2}, и, кроме того, из нее исключается область $[b_k] < x_k < [b_k] + 1$, которая не может содержать целочисленное решение.

Вопрос о том, какая из нецелочисленных переменных x_k используется для построения подзадач, решается просто, если при постановке задачи все переменные, ограниченные условием целочисленности, ранжированы по приоритетам с учетом конкретных особенностей решаемой задачи. Тогда при каждом выполнении шага 8 для ветвления выбирается нецелочисленная переменная с наибольшим приоритетом. Применяется также случайный выбор и выбор с помощью штрафных добавок, учитывающих изменение целевой функции при переходе от дробного к целочисленному значению переменной.

Среди основных правил выбора ЗК для ветвления на шаге 2 назовем *одностороннее ветвление* и *ветвление по наибольшей верхней границе*. В первом случае ветвление дерева осуществляется в заранее определенном порядке. Это правило имеет два основных преимущества: упрощается ведение текущего списка задач и облегчается использование при решении задач-потомков результатов непосредственно предшествующих им задач. Поскольку рассматриваемая ветвь развивается до тех пор, пока не будет выполнен один из критериев зондирования, дерево ветвлений обычно имеет относительно небольшое количество сравнительно длинных ветвей.

Для второго правила, наоборот, характерны деревья, у которых много коротких ветвей. В основе этого правила лежит предположение, что оптимальная точка задачи I с большей вероятностью находится в том подмножестве, которому соответствует большее значение верхней границы. Таким образом, при ветвлении всякий раз выбирается наиболее перспективная в указанном смысле вершина дерева. При использовании правил этого типа объем требуемой памяти ЭВМ увеличивается, так как становится больше задач, одновременно находящихся в списке. Усложняется ведение текущего списка ЗК, возрастает объем вычислений на одну итерацию метода. Однако эти недостатки обычно компенсируются заметным сокращением общего количества итераций. В связи с этим правила, предусматривающие ветвление задач с большим значением верхней границы, широко применяются при построении пакетов прикладных программ.

Пример 3.5. Решить методом ветвей и границ целочисленную задачу I:

$$\left.\begin{array}{r} \max z = x_1 - 3x_2 + 3x_3; \\ 4x_1 - 3x_2 \leqslant 2; \\ -3x_1 + 2x_2 + x_3 \leqslant 3; \\ 2x_1 + x_2 - x_3 \leqslant 4; \\ x_1,\ x_2,\ x_3 \geqslant 0; \end{array}\right\} \qquad (3.26)$$

$$x_1,\ x_2,\ x_3 \text{ — целые.} \qquad (3.27)$$

Включаем задачу I в список, полагаем $z^* = -\infty$ (шаг 1), выбираем в качестве задачи-кандидата единственную находящуюся в нем задачу I (шаг 2) и переходим (шаг 3) к ее релаксации — задаче линейного программирования L_0, которую получаем, отбрасывая в задаче I (3.26) — (3.27) требование целочисленности (3.27). Приведя с помощью дополнительных переменных x_4, x_5, $x_6 \geqslant 0$ задачу (3.26) к каноническому виду, после двух итераций симплекс-метода находим (шаг 4) оптимальное решение задачи L_0: $x_1^0 = 1/2$, $x_2^0 = 0$, $x_3^0 =$

Таблица 3.7

c_B	x_B	b	1	−3	3	0	0	0
			a^1	a^2	a^3	a^4	a^5	a^6
1	x_1	1/2	1	−3/4	0	1/4	0	0
3	x_3	9/2	0	−1/4	1	3/4	1	0
0	x_6	15/2	0	9/4	0	1/4	1	1
	Δ	14	0	6/4	0	10/4	3	0

$= 4\frac{1}{2}$, $z^0 = 14$ (табл. 3.7). Значение $z^0 = 14$ становится верхней

границей (оценкой) $\varphi(P)$ целевой функции задачи I, $z(x) \leqslant \varphi(P) = 14$

Проверка критериев зондирования (шаги 5, 6, 7) приводит к шагу 8, где допустимое множество P задачи I разбивается на два подмножества P_1 и P_2 для получения двух задач-потомков I_1 и I_2. Для этого выбираем одну из нецелочисленных переменных, напри-

мер $x_3 = 4\frac{1}{2}$, и определяем задачу I_1 на точках $x \in P$, удовлетво-

ряющих требованию $x_3 \leqslant 4$, а задачу I_2 — на точках, удовлетворяю-

щих требованию $x_3 \geqslant 5$.

Теперь список содержит две задачи. Их релаксации L_1 и L_2 отличаются от L_0 одним дополнительным ограничением: $x_3 \leqslant 4$ в задаче L_1; $x_3 \geqslant 5$ в задаче L_2. Они соответственно имеют вид:

$$\max\ z = x_1 - 3x_2 + 3x_3; \quad 4x_1 - 3x_2 \leqslant 2;$$
$$-3x_1 + 2x_2 + x_3 \leqslant 3; \quad 2x_1 + x_2 - x_3 \leqslant 4;$$
$$x_1,\ x_2,\ x_3 \geqslant 0; \quad x_3 \leqslant 4;$$

$$\max\ z = x_1 - 3x_2 + 3x_3; \quad 4x_1 - 3x_2 \leqslant 2;$$
$$-3x_1 + 2x_2 + x_3 \leqslant 3; \quad 2x_1 + x_2 - x_3 \leqslant 4;$$
$$x_1,\ x_2,\ x_3 \geqslant 0; \quad x_3 \geqslant 5.$$

Решение задач L_1 и L_2 удобно выполнить, используя табл. 3.7 с результатами решения задачи L_0. Выразим из табл. 3.7 перемен-ную x_3: $x_3 = 9/2 + 1/4x_2 - 3/4x_4 - x_5$. Тогда неравенство $x_3 \leqslant 4$, преобразованное с помощью дополнительной переменной x_7 в урав-

Таблица 3.8

c_B	x_B	b	1	−3	3	0	0	0	0
			a^1	a^2	a^3	a^4	a^5	a^6	a^7
1	2	3	4	5	6	7	8	9	10
	Δ	14	0	6/4	0	10/4	3	0	0
1	x_1	1/2	1	−3/4	0	1/4	0	0	0
3	x_3	9/2	0	−1/4	1	3/4	1	0	0
0	x_6	15/2	0	9/4	0	1/4	1	1	0
0	x_7	−1/2	0	1/4	0	−3/4	**−1**	0	1
	Δ	25/2	0	9/4	0	1/4	0	0	3
1	x_1	1/2	1	−3/4	0	1/4	0	0	0
3	x_3	4	0	0	1	0	0	0	1
0	x_6	7	0	10/4	0	−2/4	0	1	1
0	x_5	1/2	0	−1/4	0	3/4	1	0	−1

нение $x_3 + x_7 = 4$, примет вид $1/4x_2 - 3/4x_4 - x_5 + x_7 = -1/2$, а условие $x_3 \geqslant 5$ или уравнение $-x_3 + x_7 = -5$ запишется в виде $-1/4x_2 + 3/4x_4 + x_5 + x_7 = -1/2$. Условия задачи L_1 и ее решение двойственным симплекс-методом приведены в табл. 3.8, а задачи L_2 — в табл. 3.9.

Таблица 3.9

c_B	x_B	b	1	−3	3	0	0	0	0
			a^1	a^2	a^3	a^4	a^5	a^6	a^7
	Δ	14	0	6/4	0	10/4	3	0	0
1	x_1	1/2	1	−3/4	0	1/4	0	0	0
3	x_3	9/2	0	−1/4	1	3/4	1	0	0
0	x_6	15/2	0	9/4	0	1/4	1	1	0
0	x_7	−1/2	0	**−1/4**	0	3/4	1	0	1
	Δ	11	0	0	0	7	9	0	6
1	x_1	2	1	0	0	−2	−3	0	−3
3	x_3	5	0	0	1	0	0	0	−1
0	x_6	3	0	0	0	7	10	1	9
−3	x_2	2	0	1	0	−3	−4	0	−4

Из табл. 3.8 и 3.9 получаем оптимальное решение задачи L_1 в виде $x_1^1 = 1/2$, $x_2^1 = 0$, $x_3^1 = 4$, $z^1 = 12\frac{1}{2}$, задачи L_2 — в виде $x_1^2 = 2$, $x_2^2 = 2$, $x_3^2 = 5$, $z^2 = 11$. Значения z^1, z^2 становятся верхними границами целевых функций задач I_1, I_2: $\varphi(P_1) \leqslant 12\frac{1}{2}$, $\varphi(P_2) \leqslant 11$.

Вычисления удобно сопровождать построением дерева решений (рис. 3.5, *а*), на котором каждая подзадача I_j отображается вершиной с указанием оптимального решения ее ЗКР.

Поскольку оптимальное решение $\mathbf{x}^2 = (x_1^2, x_2^2, x_3^2) = (2, 2, 5)$ задачи L_2 целочисленно (шаг 7), оно принадлежит P. Следовательно, критерий зондирования 3 для задачи I_2 выполнен, значение $z^2 = 11$ становится рекордом $z^* = z^2 = 11$, а вектор $\mathbf{x}^2 = (2, 2, 5)$ — решением-претендентом $\mathbf{x}^*$ (шаг 9). Задача I_2 из списка удаляется.

Для задачи I_1 ни один критерий зондирования не выполняется. Так как $\varphi(P_1) = 12\frac{1}{2} > z^* = 11$, не исключено, что оптимальный план задачи I принадлежит P_1. Продолжаем анализ множества P_1. Переходя к шагу 8, разветвим I_1 на две подзадачи I_3 и I_4, используя для этого единственную нецелочисленную переменную $x_1 = 1/2$ вектора $\mathbf{x}^1 = (1/2, 0, 4)$. Релаксации L_3 и L_4 отличаются от L_1 дополнительными условиями $x_1 \leqslant 0$ в задаче L_3 и $x_1 \geqslant 1$ в задаче L_4.

Оптимальное решение задачи L_3 получаем в виде $x_1^3 = 0$, $x_2^3 = 0$, $x_3^3 = 3$, $z^3 = 9$, задачи L_4 — в виде $x_1^4 = 1$, $x_2^4 = 2/3$, $x_3^4 = 4$, $z^4 = 11$. В обоих случаях выполняется критерий зондирования 2

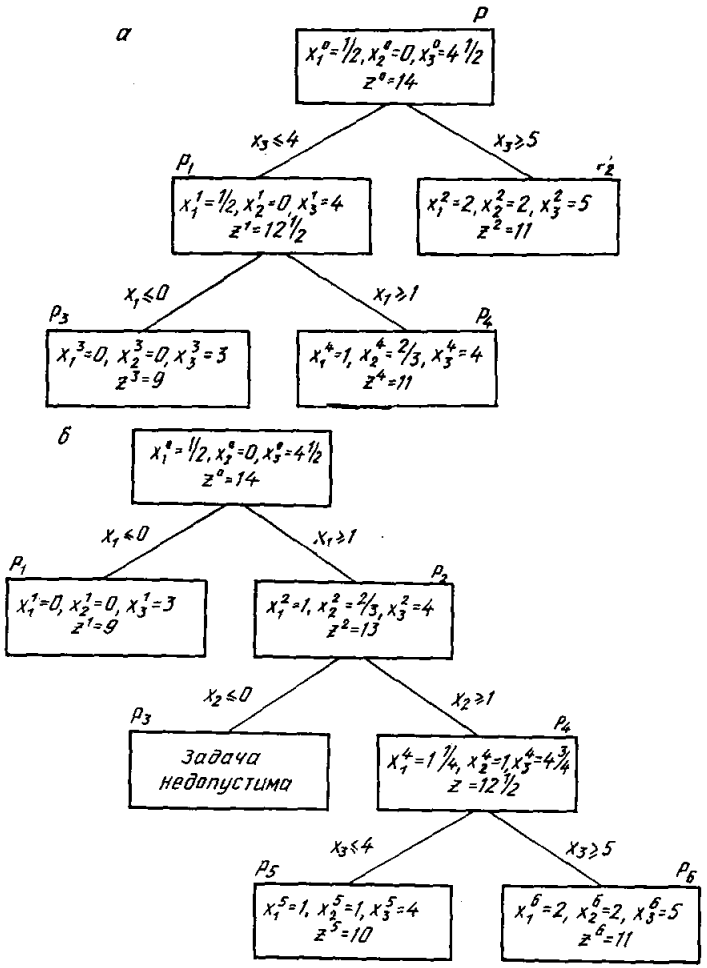

Рис. 3.5.

(шаг 6), I_3 и I_4 исключаются из списка и в нем нет больше задач. Таким образом, решение-претендент $\mathbf{x}^* = \mathbf{x}^2 = (2, 2, 5)$ является оптимальным решением задачи I со значением целевой функции $z(\mathbf{x}) = z^* = 11$. Действительно, рассмотрим висячие вершины I_2, I_3, I_4 дерева решений. Любое допустимое решение задачи I должно при-

надлежать одному из непересекающихся подмножеств P_2, P_3 или P_4, для которых определены верхние границы $\varphi(P_2)=z^2=11$, $\varphi(P_3)=z^3=9$, $\varphi(P_4)=z^4=11$ целевой функции $z(x)$. Поскольку верхняя граница $\varphi(P_2)$ найдена на целочисленной точке x^2 множества P_2, причем она не меньше, чем верхние границы на остальных подмножествах, x^2 является решением I.

На рис. 3.5, *б* представлено дерево решений той же задачи (3,26), (3.27) при условии, что для ветвления задачи I используется переменная x_1, а не x_3. Как видно из сопоставления рис. 3.5, *а* и 3.5, *б*, размеры дерева могут существенно зависеть от порядка выбора переменных. В пакетах прикладных программ обычно предусматривается возможность выбора переменных для ветвления с учетом их приоритетов, заданных при постановке задачи.

3.4. ЗАДАЧА КОММИВОЯЖЕРА

Постановка задачи. Задача коммивояжера служит математической моделью многих задач календарного планирования (теории расписаний), маршрутизации перевозок и т. д. Ее содержательная постановка такова.

Имеется n городов, расстояния между которыми заданы матрицей $(c_{ij})_{n\times n}$. Коммивояжер должен побывать в каждом городе один раз и вернуться в начальный пункт маршрута, совершив путь минимальной длины.

В понятиях теории графов задача состоит в том, чтобы найти кратчайший простой цикл из n ребер в графе с n вершинами при заданных длинах c_{ij} ребер (i, j). Существование решения задачи очевидно: в ней имеется $(n-1)!$ возможных циклов; один или несколько из них являются кратчайшими. Примером задачи календарного планирования, в математическом отношении эквивалентной задаче коммивояжера, будет следующая *задача о переналадке автоматической линии.*

Пусть некоторый набор деталей проходит обработку на автоматической линии. Время наладки линии для обработки очередной детали зависит от того, какая деталь обрабатывалась до нее. Продолжительности переналадок заданы матрицей $(t_{ij})_{n\times n}$, где элемент t_{ij} означает время переналадки при переходе от детали i к детали j. Требуется установить порядок обработки деталей, при котором затраты времени на наладку были бы минимальными.

Обратим внимание на то, что собственно время обработки деталей нас не интересует, поскольку оно входит в общее время выполнения заказа и является величиной постоянной при любом порядке обработки деталей. Если рассматривать вместо деталей города, а вместо матрицы переналадок (t_{ij}) матрицу расстояний (c_{ij}) и считать, что по завершении всех обработок линия вновь возвра-

щается в исходное состояние, то задача о переналадке линии совпадает с задачей коммивояжера.

Задача коммивояжера может быть сформулирована как целочисленная введением булевых переменных $x_{ij}=1$, если маршрут коммивояжера включает переезд из города i непосредственно в город j, и $x_{ij}=0$ в противном случае. Тогда можем записать целевую функцию и систему ограничений аналогично задаче о назначениях:

$$\min z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}; \tag{3.28}$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,\ \ldots,\ n; \tag{3.29}$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,\ \ldots,\ n; \tag{3.30}$$

$$x_{ij} \geqslant 0,\ \text{целые},\ i,\ j = 1,\ \ldots,\ n. \tag{3.31}$$

Условия (3.29) — (3.31) в совокупности обеспечивают, что каждая переменная x_{ij} равна или 0 или 1. При этом ограничения (3.29), (3.30) выражают условие, что коммивояжер побывает в каждом городе, один раз в него прибыв (в соответствии с ограничением (3.29)) и один раз выехав (ограничение (3.30)). Однако этих ограничений еще недостаточно для постановки задачи коммивояжера, так как они не исключают решения (рис. 3.6), где вместо простого цикла, проходящего через n вершин, отыскиваются два или более отдельных цикла (подцикла), проходящих через меньшее число вершин. Поэтому задача (3.28) — (3.31) должна быть дополнена ограничениями, обеспечивающими связность искомого цикла.

Способ формирования таких ограничений поясним с помощью рис. 3.6. Если бы требовалось исключить возникновение только подцикла (1,2,4,1) или (3,5,6,7,3), то достаточно было бы дополнить систему ограничений (3.29) — (3.31) условиями $x_{12} + x_{24} + x_{41} \leqslant 2$ или $x_{35} + x_{56} + x_{67} + x_{73} \leqslant 3$. Для того чтобы при постановке задачи заранее исключить все возможные подциклы, поступим следующим образом. Считая началом цикла вершину 1, введем переменные u_i, $i=2,\ \ldots,\ n$, и образуем $(n-1)(n-2)$ ограничений

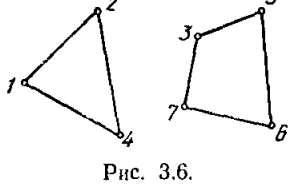

Рис. 3.6.

$$u_i - u_j + nx_{ij} \leqslant n-1,\ i=2,\ \ldots,\ n,\ j=2,\ \ldots,\ n,\ i \neq j. \tag{3.32}$$

Покажем, например, что ограничения (3.32) действительно исключают подцикл (3, 5, 6, 7, 3), которому отвечают переменные $x_{35}=x_{56}=x_{67}=x_{73}=1$ (рис. 3.6). Подставим их в ограничения (3.32):

$$u_3-u_5+nx_{35}\leqslant n-1;\ u_5-u_6+nx_{56}\leqslant n-1;$$
$$u_6-u_7+nx_{67}\leqslant n-1;\ u_7-u_3+nx_{73}\leqslant n-1.$$

Сложив эти ограничения, получим неравенство $n(x_{35}+x_{56}+x_{67}+x_{73})\leqslant(n-1)4$, которое не удовлетворяется, если все переменные x_{35}, x_{56}, x_{67}, x_{73} одновременно равны 1 (получаем $n\cdot 4\leqslant(n-1)4$), и, следовательно, исключает возможность образования подцикла (3, 5, 6, 7, 3).

Отметим, что введение условия (3.32) резко увеличивает размеры модели. Так, при $n=50$ получаем порядка 2500 ограничений. Решение целочисленных задач таких размеров представляет серьезную вычислительную проблему. Между тем во многих практических ситуациях задача коммивояжера является стандартной подзадачей, решение которой должно выполняться многократно. Поэтому практического значения модель (3.28) — (3.32) не имеет. При разработке пакетов прикладных программ используются различные хорошо зарекомендовавшие себя эвристические методы и точные методы, в основе которых лежит комбинаторная структура задачи (например, методы ветвей и границ, динамического программирования). Рассмотрим первый из них.

Общая схема метода ветвей и границ для задачи коммивояжера. Обозначим через P множество всех простых циклов из n ребер, среди которых отыскивается кратчайший. Пусть имеется некоторая вспомогательная (оценочная) задача, из решения которой может быть получена оценка (нижняя граница) φ длины кратчайшего цикла на множестве P или на любом его подмножестве. Допустим, имеется также способ разбиения множества P на подмножества и правило выбора подмножеств для ветвления.

Тогда схема метода (см. рис. 3.3) такова. Решение вспомогательной задачи на множестве P дает оценку $\varphi(P)$. Разветвляем P на подмножества P_1 и P_2, решаем

на них оценочные задачи и выбираем для дальнейшего ветвления одно из подмножеств, например то, на котором нижняя граница меньше. Процесс построения и оценивания подмножеств продолжаем, пока нижняя граница для одного из подмножеств не совпадет с длиной какого-либо из его циклов. В результате будет определен цикл-претендент и соответствующий ему рекорд. После этого просматриваем висячие вершины дерева, и если среди них найдутся вершины с нижними границами, меньшими, чем имеющийся рекорд, то их развиваем по тем же правилам. Процесс продолжается, пока нижние границы новых подмножеств остаются меньше длины выделенного цикла. В ходе ветвления либо будет построен новый цикл меньшей длины, либо обнаружится, что на новых подмножествах он не существует (оценки для каждого из них не меньше рекорда).

Для полноты описания следует указать способ вычисления нижней границы и правило ветвления подмножеств.

Оценивание подмножеств. Один из наиболее простых способов расчета нижних границ основан на приведении матрицы расстояний. Он использует следующее соображение. Поскольку в решении всегда находится по одному элементу от каждых строки и столбца, то изменение всех элементов столбца или строки на одну и ту же величину не повлияет на выбор цикла. Поэтому, если решить задачу коммивояжера с некоторой матрицей расстояний, а затем из всех элементов какой-либо строки или столбца этой матрицы вычесть произвольное число *(число приведения)* и найти кратчайший цикл с новой матрицей, то он останется тем же, а его длина изменится на величину этого числа. Если же эту операцию проделать и для других строк и столбцов, то длина цикла L_1 на приведенной матрице будет меньше длины цикла L на исходной матрице на сумму h чисел приведения по всем строкам и столбцам, а именно: $L=L_1+h$. Если каждый ряд (строку или столбец) приводить, вычитая наименьшее число в нем, то, как и в задаче об оптимальных назначениях, все элементы приведенной матрицы будут неотрицательны и в каждом ряду будет, по крайней мере, один нуль. Тогда $L_1 \geqslant 0$, $L \geqslant h$ и сумма чисел приведения $h>0$ может служить нижней границей длины искомого простого цикла.

Другим способом получения оценок для вершин дерева ветвлений может служить решение задач о назна-

чениях, получаемых из задачи коммивояжера релаксацией условия связности маршрута. Применение этого способа обычно дает более точные оценки подмножеств и, как следствие, требует рассмотрения деревьев с меньшим количеством ветвей.

Ветвление. Для разбиения любого множества циклов P_j на подмножества P_{j1} и P_{j2} удобно использовать дихотомию, при которой в одно из них, допустим P_{j1}, входят все циклы множества P_j, включающие определенную дугу, а в другое, например P_{j2},— все циклы, не включающие эту дугу.

Если для оценки подмножеств используется решение задачи о назначениях, то разбиение осуществляется следующим образом. После решения задачи о назначениях на всем множестве P получаем либо связный цикл, либо (что более вероятно) совокупность подциклов. В первом случае задача коммивояжера решена, во втором переходим к ее ветвлению. Для этого выбираем один из подциклов, содержащий минимальное число вершин. Одну из дуг этого подцикла, допустим (k, l), используем для получения двух новых задач назначения. В одной из них элемент $c_{kl}=0$, а все остальные элементы строки k и столбца l заменяются на ∞. Это означает, что из города k обязателен переезд в город l. Для запрещения подцикла (k, l, k) полагаем также $c_{lk}=\infty$. В другой задаче, наоборот, элемент c_{kl} заменяется на ∞, а все остальные остаются без изменения. Это означает, что переезд из города k непосредственно в город l запрещается. Из решения данных задач получаются оценки соответствующих им подмножеств, на основании которых аналогичным образом продолжается ветвление и оценивание до получения цикла с длиной, не большей, чем нижние границы на всех висячих вершинах дерева.

Пример 3.7. Решить задачу коммивояжера, заданную матрицей, представленной в табл. 3.10, используя для оценок подмножеств задачу о назначениях.

Дерево ветвлений показано на рис. 3.7, где обозначение вида (k, l) указывает, что дуга (k, l) включается в цикл, вида $\overline{(k, l)}$ — не включается. Вначале в списке задач-кандидатов находится одна задача коммивояжера, заданная матрицей табл. 3.10 на всем множестве циклов P. Решаем ее релаксацию — задачу о назначениях и получаем (табл. 3.11) распределение исполнителей по работам (1,5), (2,1), (3,4), (4,3), (5,6), (6,2), отмеченное звездочками. Суммирование соответствующих элементов табл. 3.10 дает величину затрат, связанных с данным распределением: $z=2+1+5+4+2+1=15$.

Таблица 3.10

i \ j	1	2	3	4	5	6
1	∞	9	10	8	2	4
2	1	∞	6	8	3	4
3	6	4	∞	5	4	7
4	2	3	4	∞	5	8
5	5	3	2	6	∞	2
6	6	1	5	7	4	∞

Таким образом, исследовано все множество циклов P задачи коммивояжера и для него получена нижняя граница длины кратчайшего цикла $\varphi(P)=15$, которая достигается при двух подциклах (1,5), (5,6), (6,2), (2,1) и (3,4), (4,3) или короче (1,5,6,2,1) и (3,4, 3). Выбирая подцикл (3,4,3) с меньшим числом вершин, вносим в список задач-кандидатов две задачи коммивояжера: одну на подмножестве P_1 всех циклов, включающих дугу (3,4), другую — на подмножестве P_2 всех циклов, не содержащих дугу (3,4). Переходим к их релаксациям — двум задачам назначения. Для получения первой из них используем табл. 3.11, в которой все элементы строки 3 и столбца 4, кроме элемента c_{34}, заменяем на ∞, а также полагаем $c_{43}=\infty$ (табл. 3.12). Решение этой задачи представлено в табл. 3.13.

Распределение исполнителей по работам имеет вид (1,5), (2,6), (3,4), (4,1), (5,3), (6,2), а сумма соответствующих ему элементов

Таблица 3.11

i \ j	1	2	3	4	5	6
1	∞	8	8	3	0*	2
2	0*	∞	3	2	0	1
3	6	3	∞	0*	2	5
4	0	0	0*	∞	1	4
5	5	2	0	1	∞	0*
6	6	0*	3	2	2	∞

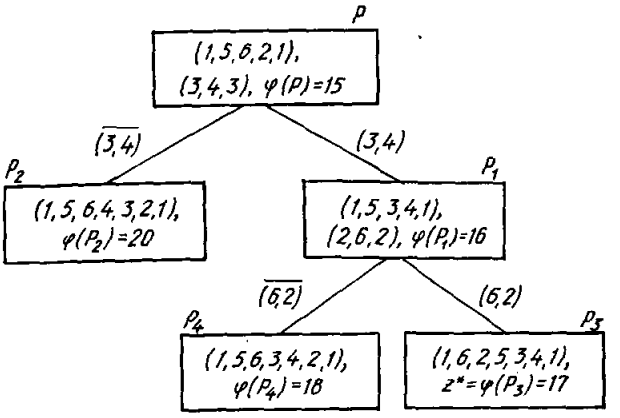

Рис. 3.7.

табл. 3.10 дает величину затрат $z=16$. Для задачи коммивояжера полученное решение означает образование двух подциклов (1,5,3,4,1) и (2,6,2) и указывает *нижнюю границу* $\varphi(P_1)=16$ *длины* кратчайшего из всех циклов, включающих дугу (3,4).

Задачу о назначениях на подмножестве P_2 получаем, заменяя в табл. 3.11 только элемент c_{34}: для исключения дуги (3,4) ему придается значение $c_{34}=\infty$ (табл. 3.14). В результате ее решения имеем простой цикл (1,5,6,4,3,2,1) *длиной* $z=20$ (табл. 3.15).

Исключаем эту задачу из списка, запомнив цикл и его длину соответственно как решение-претендент и рекорд, а задачу P_1 разветвляем на две подзадачи с множествами P_3 и P_4. Множество P_3 включает все циклы, содержащие дуги (3,4) и (6,2), а множество P_4 — все циклы, в которые входит дуга (3,4), но не входит дуга (6,2). Формулируем две задачи назначения.

Таблица 3.12

i \ j	1	2	3	4	5	6
1	∞	8	8	∞	0	2
2	0	∞	3	∞	0	1
3	∞	∞	∞	0	∞	∞
4	0	0	∞	∞	1	4
5	5	2	0	∞	∞	0
6	6	0	3	∞	2	∞

Таблица 3.13

i \ j	1	2	3	4	5	6
1	∞	8	7	∞	0*	1
2	0	∞	2	∞	0	0*
3	∞	∞	∞	0*	∞	∞
4	0*	0	∞	∞	1	3
5	6	3	0*	∞	∞	0
6	6	0*	2	∞	2	∞

Таблица 3.14

i \ j	1	2	3	4	5	6
1	∞	8	8	3	0	2
2	0	∞	3	2	0	1
3	6	3	∞	∞	2	5
4	0	0	0	∞	1	4
5	5	2	0	1	∞	0
6	6	0	3	2	2	∞

Таблица 3.15

i \ j	1	2	3	4	5	6
1	∞	7	6	0	0*	0
2	0*	∞	3	1	2	1
3	2	0*	∞	∞	0	1

Окончание табл. 3.15

i \ j	1	2	3	4	5	6
4	0	1	0*	∞	3	4
5	5	3	0	0	∞	0*
6	5	0	2	0*	3	∞

Таблица 3.16

i \ j	1	2	3	4	5	6
1	∞	∞	7	∞	0	1
2	0	∞	2	∞	0	∞
3	∞	∞	∞	0	∞	∞
4	0	∞	∞	∞	1	3
5	6	∞	0	∞	∞	0
6	∞	0	∞	∞	∞	∞

Таблица 3.17

i \ j	1	2	3	4	5	6
1	∞	∞	6	∞	0	0*
2	0	∞	1	∞	0*	∞
3	∞	∞	∞	0*	∞	∞
4	0*	∞	∞	∞	1	2
5	7	∞	0*	∞	∞	0
6	∞	0*	∞	∞	∞	∞

Таблица 3.18

i \ j	1	2	3	4	5	6
1	∞	8	7	∞	0	1
2	0	∞	2	∞	0	0
3	∞	∞	∞	0	∞	∞
4	0	0	∞	∞	1	3
5	6	3	0	∞	∞	0
6	6	∞	2	∞	2	∞

Таблица 3.19

i \ j	1	2	3	4	5	6
1	∞	8	7	∞	0*	1
2	0*	∞	2	∞	0	0
3	∞	∞	∞	0*	∞	∞
4	0	0*	∞	∞	1	3
5	6	3	0	∞	∞	0*
6	4	∞	0*	∞	0	∞

Задачу на множестве P_3 получаем, заменив в табл. 3.13 все элементы строки 6 и столбца 2, кроме c_{62}, на ∞ и положив для запрещения цикла (6,2,6) элемент $c_{26}=\infty$ (табл. 3.16). Ее решение (табл. 3.17) дает цикл (1,6,2,5,3,4,1) длиной $z=17$, который становится новым решением-претендентом для задачи коммивояжера, поскольку его длина меньше имеющегося рекорда $z^*=20$.

Матрицу для задачи о назначениях на множестве P_4 записываем в виде табл. 3.18, заменив на ∞ элемент c_{62} матрицы 3.13. Решением задачи является цикл (1,5,6,3,4,2,1) длиной $z=18$ (табл. 3.19).

После решения этих задач имеем три висячие вершины с оценками $\varphi(P_2)$, $\varphi(P_3)$, $\varphi(P_4)$, из которых оценка $\varphi(P_3)=17$ наименьшая и получена на цикле (1,6,2,5,3,4,1), проходящем через все вершины графа. Следовательно, этот цикл оптимален.

Контрольные вопросы и упражнения

1. Дайте постановку задач целочисленного и дискретного программирования. В чем их отличие?

2. Назовите основные типы прикладных задач целочисленного программирования.

3. Какие переменные называются булевыми?

4. Какая задача называется частично целочисленной; полностью целочисленной?

5. Дайте краткую характеристику методов решения задач целочисленного программирования. В чем преимущества и недостатки точных методов; приближенных?

6. Опишите основную идею методов отсечений; ветвей и границ.

7. Постройте модель целочисленного программирования следующей задачи с неделимостями. Для изготовления s наименований заготовок применяют листы одного типоразмера, которые можно раскраивать n различными способами. Обозначим через r_{ij} количество заготовок i-го наименования, которое можно получить при раскрое одного листа j-м способом ($i=1, ..., s$, $j=1, ..., n$), а через f_i — количество деталей i-го наименования, которое надо изготовить по плану. Требуется выпустить заданное количество заготовок каждого типа при минимальном расходе листов.

8. Постройте целочисленную модель, если условия предыдущей задачи видоизменены следующим образом. Для изготовления s наименований заготовок имеется m различных типов листов. Каждый лист можно раскраивать под заготовки n различными способами. Обозначим через r_{ikj} количество заготовок i-го наименования, которое можно получить при раскрое одного листа k-го типа j-м способом ($i=1, ..., s$, $k=1, ..., m$, $j=1, ..., n$). Пусть c_{kj} — величина отходов, получаемых при j-м способе раскроя одного листа k-го типа; f_i — количество заготовок i-го наименования, которое должно быть изготовлено; a_k — имеющееся количество листов k-го типа. Требуется обеспечить потребность в заготовках при минимальном суммарном отходе материала.

9. Запишите с помощью булевых переменных условие, согласно которому в месячной программе данная деталь либо выпускается в пределах от 30 до 80 штук, либо совсем не выпускается.

10. Запишите с помощью булевых переменных условие, что партия изделий запускается в производство в одном из четырех вариантов: 20, 30, 60, 80 штук, или не запускается вовсе. Как изменится запись условия, если потребовать, чтобы какой-либо из вариантов был обязательно принят?

11. Решите следующие задачи:

а) $\max z = 2x_1 + x_2;$
$12x_1 + 7x_2 \leqslant 55;$
$5x_1 + 2x_2 \leqslant 18;$
$x_1, x_2 \geqslant 0$, целые;

б) $\max z = 2{,}5x_1 + 4x_2 + 3x_3;$
$4{,}5x_1 + 3x_2 + 5x_3 \leqslant 14;$
$2x_1 + 6x_2 + x_3 \leqslant 11;$
$x_1, x_2, x_3 \geqslant 0$, целые;

в) $\max z = 16x_1 + 3x_2;$
$4x_1 - 8x_2 \leqslant 30;$
$-2x_1 + 11x_2 \leqslant 20;$
$x_1, x_2 \geqslant 0$, целые;

г) $\max z = 3x_1 + 4x_2;$
$3x_1 + 2x_2 \leqslant 8;$
$x_1 + 4x_2 \leqslant 10;$
$x_1, x_2 \geqslant 0$, целые.

4. НЕЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

4.1. ЗАДАЧА НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Постановка и особенности. *Задачей нелинейного программирования* называется задача математического программирования, в которой нелинейны или целевая функция или функции, задающие ограничения. Мы будем рассматривать задачи следующего вида:

$$\min z = F(\mathbf{x}); \; f_i(\mathbf{x}) \leqslant 0, \; i = 1, \ldots, m; \; \mathbf{x} \geqslant 0, \qquad (4.1)$$

где или $F(\mathbf{x})$, или хотя бы одна из функций $f_i(\mathbf{x})$ — нелинейные функции n переменных $x_1, \ldots, x_n$. Если возникает задача *максимизации* некоторой функции $F(\mathbf{x})$, то она приводится к модели (4.1) переходом к задаче о *минимуме* функции $w = -F(\mathbf{x})$.

Нелинейность функций $F(\mathbf{x})$ и $f_i(\mathbf{x})$ вызывает существенные отличия задачи (4.1) от задачи линейного программирования. Так, если в задаче линейного программирования экстремальная точка является крайней точкой многогранника допустимых решений, то в задаче нелинейного программирования она может находиться как на границе области допустимых решений (рис. 4.1, *а*), так и внутри нее (рис. 4.1, *б*). Далее, нелинейная целевая функция может иметь несколько экстремальных точек. На рис. 4.2 показана функция одной переменной, которая имеет три минимума. Один из них находится во вну-

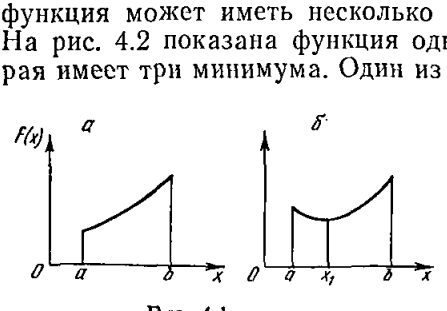

Рис. 4.1.

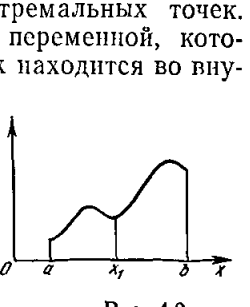

Рис. 4.2.

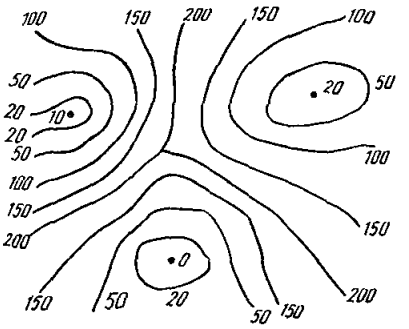

Рис. 4.3.

тренней точке x_1 и два — на концах отрезка. Глобальный минимум расположен в граничной точке a. На рис. 4.3 изображена многоэкстремальная функция двух переменных, для задания которой на плоскости использованы линии равных уровней $F(x_1, x_2) = c$. Она имеет два локальных минимума и один глобальный. Многоэкстремальность целевой функции создает значительные вычислительные трудности, так как в практических задачах обычно требуется найти среди всех минимумов точку глобального минимума.

Нелинейность ограничений также налагает свои особенности. Если в задачах линейного программирования оптимальное решение отыскивается на конечном множестве крайних точек выпуклой области, то для задачи нелинейного программирования эти условия в общем случае не сохраняются. Рассмотрим, например, рис. 4.4. Напомним, что выпуклой называется область, которая вместе с любыми двумя своими точками содержит и соединяющий их отрезок. Нелинейность ограничений может

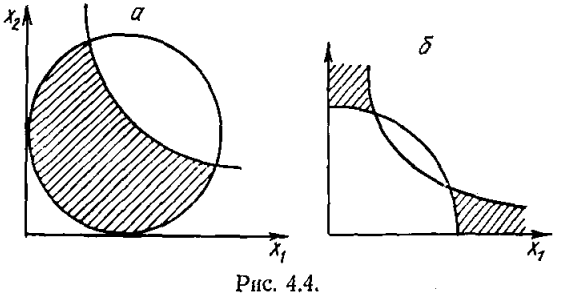

Рис. 4.4.

привести к *невыпуклости* (рис. 4.4, *а*) или *несвязности* (рис. 4.4, *б*) области допустимых решений, которая к тому же может иметь бесконечное множество крайних точек.

Эти особенности, отличающие нелинейные задачи от линейных, существенно усложняют их решение. В частности, если для задачи линейного программирования имеется признак оптимальности, то для общей задачи нелинейного программирования, где функции $F(\mathbf{x})$ и $f_i(\mathbf{x})$ — произвольного вида, такого признака нет. В задаче линейного программирования всегда может быть установлено, является ли произвольное допустимое решение оптимальным. Если некоторый план не оптимален, то можно перейти к новому плану, на котором значение целевой функции ближе к оптимальному, чем на предыдущем. В общей же задаче нелинейного программирования такой возможности нет. Если даже на каком-то этапе решения получен план, являющийся оптимальным, установить это можно только путем вычисления целевой функции во всех остальных подозреваемых на экстремум точках и сравнения их между собой. Трудности решения задач нелинейного программирования общего вида заставляют подразделять их на отдельные классы со своими методами решения. При их рассмотрении мы будем постоянно встречаться с понятием выпуклости функций.

Выпуклые функции. Функция $F(\mathbf{x})$, заданная на выпуклом множестве X, называется *выпуклой*, если для любых двух точек $\mathbf{x}^1$, $\mathbf{x}^2 \in X$ и любого λ, $0 \leqslant \lambda \leqslant 1$, справедливо неравенство $F(\mathbf{x}) \leqslant \lambda F(\mathbf{x}^1) + (1-\lambda) F(\mathbf{x}^2)$. Здесь $\mathbf{x} = \lambda \mathbf{x}^1 + (1-\lambda)\mathbf{x}^2$.

Функцию $F(\mathbf{x})$ назовем *строго выпуклой*, если в данном условии знак $\leqslant$ можно заменить знаком $<$. Функции одной переменной, изображенные на рис. 4.1, строго выпуклы.

Аналогично определяются понятия вогнутой и строго вогнутой функции. Так, функция $F(\mathbf{x})$, заданная на выпуклом множестве X, называется *вогнутой*, если для любых двух точек $\mathbf{x}^1$, $\mathbf{x}^2 \in X$ и любого λ, $0 \leqslant \lambda \leqslant 1$, имеет место неравенство $F(\mathbf{x}) \geqslant \lambda F(\mathbf{x}^1) + (1-\lambda) F(\mathbf{x}^2)$, $\mathbf{x} = \lambda \mathbf{x}^1 + (1-\lambda)\mathbf{x}^2$.

Очевидно, если $F(\mathbf{x})$ — выпуклая функция, то $-F(\mathbf{x})$ — вогнутая, и наоборот. Линейная функция $z = \mathbf{cx}$ является одновременно выпуклой и вогнутой, так

как для любых двух точек $\mathbf{x}^1$ и $\mathbf{x}^2$ и любого λ выполняется равенство $c[\lambda\mathbf{x}^1+(1-\lambda)\mathbf{x}^2]=\lambda c\mathbf{x}^1+(1-\lambda)c\mathbf{x}^2$.

Приведенные определения выпуклых и вогнутых функций дают одновременно и правила для установления их выпуклости. Однако практически использовать эти правила трудно. В ряде частных случаев определение выпуклости или вогнутости функции упрощается.

Пусть $F(\mathbf{x})$ непрерывная дважды дифференцируемая по всем переменным функция. Запишем для нее матрицу

$$H_F=\begin{bmatrix} f_{11}(\mathbf{x}^0) & f_{12}(\mathbf{x}^0) & \dots & f_{1n}(\mathbf{x}^0) \\ f_{21}(\mathbf{x}^0) & f_{22}(\mathbf{x}^0) & \dots & f_{2n}(\mathbf{x}^0) \\ \dots & \dots & \dots & \dots \\ f_{n1}(\mathbf{x}^0) & f_{n2}(\mathbf{x}^0) & \dots & f_{nn}(\mathbf{x}^0) \end{bmatrix},$$

где $f_{ij}(\mathbf{x}^0)$ — вторая частная производная по x_i и x_j, вычисленная в точке $\mathbf{x}^0$.

Матрица H_F называется *матрицей Гессе* или *гессианом функции* F. Пусть X — выпуклое множество и $\mathbf{x}^0\in X$. Тогда для того чтобы функция $F(\mathbf{x})$ была выпукла в некоторой окрестности точки $\mathbf{x}^0$, необходимо и достаточно, чтобы в этой окрестности были неотрицательны все главные миноры ее гессиана H_F. Если все нечетные главные миноры H_F отрицательны, а четные положительны, то функция строго вогнута в окрестности точки $\mathbf{x}^0$.

Если приведенные условия выпуклости выполняются для всех $\mathbf{x}\in X$, а не только в одной точке, то дифференцируемая функция $F(\mathbf{x})$ является строго выпуклой всюду на множестве точек X. Аналогичное утверждение имеет место для строго вогнутых функций.

Пример 4.1. Определить поведение функции $F(\mathbf{x})=3x_1^2-2x_1x_2+x_2^2$ в окрестности точки $\mathbf{x}^0=(1,\ 1)$.

Записав матрицу Гессе

$$\begin{bmatrix} f_{11}(\mathbf{x}^0) & f_{12}(\mathbf{x}^0) \\ f_{21}(\mathbf{x}^0) & f_{22}(\mathbf{x}_0) \end{bmatrix}=\begin{bmatrix} 6 & -2 \\ -2 & 2 \end{bmatrix},$$

видим, что она состоит лишь из числовых коэффициентов. Следовательно, вывод о характере функции будет справедлив для всех точек $\mathbf{x}$. Вычислим определители

$$|f_{11}|=6>0;\ \begin{vmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{vmatrix}=\begin{vmatrix} 6 & -2 \\ -2 & 2 \end{vmatrix}=8>0.$$

Функция является строго выпуклой при всех значениях переменных $(x_1,\ x_2)$.

Во многих случаях проверка функций на выпуклость облегчается благодаря следующему утверждению: *сумма выпуклых на множестве X функций является выпуклой функцией. Точно так же сумма вогнутых функций есть вогнутая функция.*

Типы задач нелинейного программирования. В нелинейном программировании выделяют два основных типа задач — выпуклого и невыпуклого программирования. В задачах *выпуклого программирования* минимизируемая целевая функция выпукла (в задачах на максимум — вогнута), а ограничения задают выпуклое множество допустимых решений. *Невыпуклое программирование* охватывает все остальные задачи, не обладающие особенностями задач выпуклого программирования.

Важным свойством задач выпуклого программирования является совпадение локального и глобального минимумов. Покажем это. Пусть функция $F(\mathbf{x})$ имеет локальный минимум в точке $\mathbf{x}^0 \in X$. Это значит, что для некоторой окрестности $U(\mathbf{x}^0)$ точки $\mathbf{x}^0$ выполняется отношение $F(\mathbf{x}^0) \leqslant F(\mathbf{x})$ для всех $\mathbf{x} \in U(\mathbf{x}^0)$. Пусть в то же время в некоторой точке $\mathbf{x}^* \in X$ имеет место $F(\mathbf{x}^*) < F(\mathbf{x}^0)$.

Рассмотрим отрезок, соединяющий точки $\mathbf{x}^0$ и $\mathbf{x}^*$. В силу выпуклости области X этот отрезок принадлежит X. Возьмем на нем произвольную точку $\lambda\mathbf{x}^0 + (1-\lambda)\mathbf{x}^*$, $0<\lambda<1$. В силу того, что функция $F(\mathbf{x})$ выпукла, имеем $F(\lambda\mathbf{x}^0 + (1-\lambda)\mathbf{x}^*) \leqslant \lambda F(\mathbf{x}^0) + (1-\lambda)F(\mathbf{x}^*)$. Данное неравенство можно усилить, если вместо $F(\mathbf{x}^*)$ подставить справа $F(\mathbf{x}^0) > F(\mathbf{x}^*)$. Это дает $F(\lambda\mathbf{x}^0 + (1-\lambda)\mathbf{x}^*) < F(\mathbf{x}^0)$. Однако λ можно выбрать так, чтобы точка $\lambda\mathbf{x}^0 + (1-\lambda)\mathbf{x}^*$ была расположена сколь угодно близко к $\mathbf{x}^0$. Тогда в окрестности $\mathbf{x}^0$ найдется точка $\mathbf{x}$, в которой $F(\mathbf{x}) < F(\mathbf{x}^0)$, а это означает, что точка $\mathbf{x}^0$ не является точкой локального минимума. Полученное противоречие доказывает утверждение, а именно: в задачах выпуклого программирования локальный минимум является одновременно и глобальным. Эта особенность упрощает решение задач выпуклого программирования.

В выпуклом программировании в свою очередь специальный класс образуют задачи *квадратичного программирования*. В них целевая функция представляет полином второй степени, а область допустимых решений задается, как и в линейном программировании, системой линейных ограничений.

Если обозначить выпуклую квадратичную функцию (форму) через $Q(\mathbf{x})$, то задача квадратичного программирования запишется в виде $\min Q(\mathbf{x})$; $A\mathbf{x} \geqslant \mathbf{b}$; $\mathbf{x} \geqslant \mathbf{0}$. Здесь функция $Q(\mathbf{x})$ в соответствии с ее определением имеет вид $Q(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_i x_j$. Квадратичную форму Q можно записать в виде

$$
\begin{aligned}
Q(\mathbf{x}) = {} & (c_{11} x_1 + c_{12} x_2 + \ldots + c_{1n} x_n) x_1 + \\
& + (c_{21} x_1 + c_{22} x_2 + \ldots + c_{2n} x_n) x_2 + \\
& \ldots\ldots\ldots\ldots\ldots\ldots \\
& + (c_{n1} x_1 + c_{n2} x_2 + \ldots + c_{nn} x_n) x_n.
\end{aligned}
$$

Заметим, что если $i \neq j$, то коэффициент при $x_i x_j$ равен $c_{ij} + c_{ji}$. Отсюда следует, что матрицу (c_{ij}) коэффициентов квадратичной формы всегда можно считать симметричной. Действительно, определяя новые коэффициенты равенством $d_{ij} = d_{ji} = (c_{ij} + c_{ji})/2$ для всех i, j, получаем, что $d_{ij} + d_{ji} = c_{ij} + c_{ji}$ и матрица (d_{ij}) — симметричная. Очевидно, что при таком переходе к новым коэффициентам значение Q для любых $\mathbf{x}$ остается прежним. Поэтому будем предполагать, что матрица коэффициентов квадратичной формы всегда симметрична. Нетрудно убедиться, что при этом предположении она с точностью до коэффициента 2 совпадает с матрицей Гессе. Отсюда приведенные ранее критерии выпуклости для случая квадратичных функций становятся очень простыми, а именно: квадратичная функция $Q(\mathbf{x})$ строго выпукла, если все главные миноры матрицы ее коэффициентов положительны; если все нечетные главные миноры матрицы коэффициентов квадратичной формы отрицательны, а четные положительны, то она строго вогнута.

Пример 4.2. Пусть $Q(\mathbf{x}) = 3x_1^2 - 2x_1x_2 + x_2^2$. Проверим ее выпуклость.

Построив симметричную матрицу коэффициентов данной квадратичной функции

$$
C = \begin{bmatrix} 3 & -1 \\ -1 & 1 \end{bmatrix},
$$

находим значения ее главных определителей

$$
|C_1| = |3| > 0; \quad |C_2| = \begin{vmatrix} 3 & -1 \\ -1 & 1 \end{vmatrix} = 2 > 0.
$$

Квадратичная форма строго выпукла. Отметим, что этот вывод мы получили также в примере 4.1, пользуясь критерием выпуклости в общем виде.

Как видно из постановки, задача квадратичного программирования близка к задаче линейного программирования и отличается от нее лишь тем, что целевая функция является не линейной, а квадратичной формой. Следовательно, если аппроксимировать в окрестности некоторой точки квадратичную форму линейной, то от задачи квадратичного программирования можно перейти к решению задачи линейного программирования. На основании этого разработаны методы, строящие последовательность решений задач линейного программирования, сходящуюся к точке оптимума квадратичной задачи.

Очевидно, к задачам квадратичного программирования применимы также методы, позволяющие решать общую задачу выпуклого программирования. Прежде чем перейти к их рассмотрению, обратим внимание на некоторые детали решения известной из математического анализа задачи безусловной минимизации заданной функции $F(\mathbf{x})$ в неограниченной области изменения $\mathbf{x}$.

Классическая задача. При аналитическом решении задачи безусловной минимизации функции $F(\mathbf{x})$ обычно используют необходимые и достаточные условия экстремума функции. Напомним, что необходимое условие экстремума функции в некоторой точке состоит в том, что все ее частные производные в этой точке равны нулю или, по крайней мере, одна из частных производных не существует или бесконечна. Достаточным условием экстремума функции в точке, где все ее частные производные равны нулю, является постоянство знака второго дифференциала функции в некоторой окрестности рассматриваемой точки. При этом отрицательный знак второго дифференциала является признаком максимума функции, а положительный — признаком минимума. Таким образом, поиск экстремумов дважды дифференцируемой функции $F(\mathbf{x})$ в неограниченной области осуществляется в два этапа: на первом в соответствии с необходимым условием определяется множество стационарных точек, где возможен экстремум, а на втором в соответствии с достаточным условием выявляются экстремальные точки и устанавливается вид экстремума.

Выполнение первого этапа требует решения системы n уравнений с n неизвестными

$$\frac{\partial F(\mathbf{x})}{\partial x_1} = 0, \quad \frac{\partial F(\mathbf{x})}{\partial x_2} = 0, \quad \ldots, \quad \frac{\partial F(\mathbf{x})}{\partial x_n} = 0. \tag{4.2}$$

Следует иметь в виду, что система (4.2) обычно является системой нелинейных уравнений, решение которой может представлять задачу того же порядка сложности, что и решение исходной задачи отыскания экстремума функции $F(\mathbf{x})$.

Если минимизируемая функция стеснена ограничениями типа равенств, экстремальные точки могут быть найдены известным из математического анализа методом множителей Лагранжа. Этот метод сводит задачу условной минимизации вида

$$\min z = F(\mathbf{x}); \; f_i(\mathbf{x}) = 0, \; i = 1, \ldots, m; \; \mathbf{x} \geqslant \mathbf{0},$$

где ограничения заданы равенствами, к задаче безусловной минимизации *функции Лагранжа* $L(\mathbf{x}, \mathbf{y})$. Функция Лагранжа имеет вид

$$L(\mathbf{x}, \mathbf{y}) = F(\mathbf{x}) + \sum_{i=1}^{m} y_i f_i(\mathbf{x}). \tag{4.3}$$

Для этой функции необходимое условие экстремума представляется в виде системы $n + m$ уравнений относительно переменных x_j и неопределенных множителей y_i:

$$\left.\begin{aligned} &\frac{\partial L}{\partial x_j} = \frac{\partial F}{\partial x_j} + \sum_{i=1}^{m} y_i \frac{\partial f_i}{\partial x_j} = 0, \; j = 1, \ldots, n; \\ &\frac{\partial L}{\partial y_i} = f_i(\mathbf{x}) = 0, \; i = 1, \ldots, m. \end{aligned}\right\} \tag{4.4}$$

Из решения системы (4.4) находится точка $(\mathbf{x}, \mathbf{y})$, которая доставляет минимум функции Лагранжа (4.3) в неограниченной области и одновременно является точкой границы, заданной условиями $f_i(\mathbf{x}) = 0$, $i = 1, \ldots, m$, где достигает минимума целевая функция $F(\mathbf{x})$ исходной задачи на условный минимум. Однако в общем случае система (4.4), как и (4.2), является нелинейной, и ее решение может быть задачей не менее сложной, чем непосредственное отыскание минимума функции Лагранжа (4.3). Отсюда возникает необходимость в численных методах, отыскивающих точку оптимума из решения прямой задачи безусловной минимизации функции (4.3), а не из системы уравнений вида (4.4).

4.2. ОДНОМЕРНАЯ МИНИМИЗАЦИЯ

Классификация методов одномерной минимизации. Будем рассматривать задачу в следующей постановке: найти на интервале (a, b) точку x^*, в которой заданная функция $F(x)$ принимает минимальное значение. Решение этой задачи является существенным элементом методов многомерной минимизации.

Способ решения задачи во многом определяется априорной информацией о свойствах функции на рассматриваемом интервале. Когда о поведении функции в заданном интервале ничего заранее не известно, минимум можно найти только с помощью перебора ее значений. Если ε — заданная точность определения искомой точки x^* на интервале (a, b), то количество точек n, в которых потребуется вычислить значения $F(x)$, чтобы выделить подынтервал, содержащий точку x^*, будет $n=[(b-a)/\varepsilon]+1$, где скобки означают целую часть заключенного в них числа.

Непосредственное сравнение полученных значений $F(x_i)$, $i = 1, ..., n$, позволит определить точку x_k и значение $F(x_k)=\min F(x_i)$, которые дают оценки положения x^* и величины $F(x^*)$ экстремума функции.

Как видно из выражения для n, трудоемкость метода, измеряемая количеством вычислений функции, необходимых для достижения заданной точности, обратно пропорциональна ε. Чтобы определить точку минимума с точностью в 10^{-3} от длины интервала (a, b), потребуется 1001 измерение; с точностью в 10^{-4} — 10 001 измерение и т. д. Однако в практических задачах подобная ситуация встречается редко. Обычно физический смысл задачи или предварительное исследование функции позволяют сделать некоторые заключения о ее свойствах. Например, можно установить, непрерывна функция или нет, монотонна она или имеет экстремум и т. д.

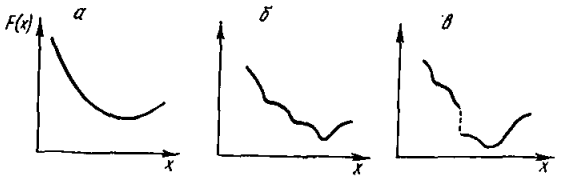

Рис. 4.5.

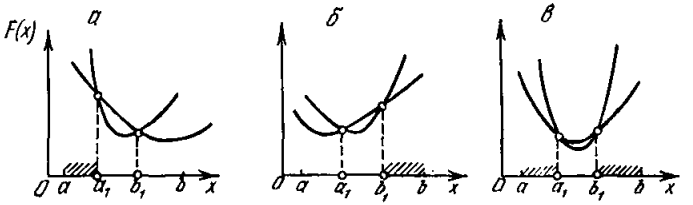

Рис. 4.6.

Одним из наиболее простых и чаще всего выполняющихся на практике предположений о функции является свойство ее *унимодальности*, означающее существование единственного экстремума на рассматриваемом интервале. Примеры унимодальных функций приведены на рис. 4.5. Как видно, этим понятием охватываются не только гладкие выпуклые (рис. 4.5, *а*), но и недифференцируемые (рис. 4.5, *б, в*) функции. Оно достаточно содержательно для построения методов минимизации, по эффективности превосходящих простой перебор.

Метод дихотомии. При рассмотрении вычислительной схемы метода будем считать, что минимум унимодальной функции ищется на отрезке (a, b) длины l_0. Выясним, как добиться наибольшей точности в определении положения точки минимума, производя только два вычисления функции. Обозначим точки эксперимента через a_1 и b_1, $a_1 < b_1$, и вычислим значения функции в них. При этом возможны три случая: $F(a_1) > F(b_1)$ (рис. 4.6, *а*), $F(a_1) < F(b_1)$ (рис. 4.6, *б*), $F(a_1) = F(b_1)$ (рис. 4.6, *в*). Соответственно в силу унимодальности функции искомая точка минимума может находиться в интервале или (a_1, b), или (a, b_1), или (a_1, b_1). Поскольку в результате эксперимента может встретиться любой из этих случаев, оценкой длины отрезка l, содержащего искомую точку, будет величина наибольшего из интервалов: $l = \max(b - a_1, b_1 - a, b_1 - a_1)$. Из анализа данного выражения видно, что для сокращения отрезка l необходимо увеличивать a_1 и уменьшать b_1 при соблюдении условия $a_1 < b_1$. Для этого следует брать точки a_1 и b_1 возможно ближе к середине c исходного интервала, оставляя между ними лишь расстояние ε, на котором еще практически можно отличить значения функции в соседних точках.

Итак, измерения функции следует выполнить в точках $a_1 = c - \varepsilon/2$ и $b_1 = c + \varepsilon/2$. Точка с меньшим зна-

чением функции указывает ту часть интервала, которую надо взять для дальнейшего рассмотрения. На рис. 4.6 заштрихованы части отрезка, не содержащие минимум.

Таким образом, в результате одного шага дихотомии исходный интервал сокращается вдвое. Поступая аналогично с новым интервалом, после второй пары измерений сократим исходный отрезок в четыре раза, после третьей — в восемь и т. д. После n шагов ($2n$ измерений функции) получим отрезок длиной $l_n = 2^{-n} l_0$. Процесс заканчивается, когда длина интервала станет не больше заданной точности ε определения точки минимума.

Метод золотого сечения. Метод дихотомии прост, надежен и гораздо эффективнее простого перебора. Для сравнения заметим, что при точности определения минимума в $\varepsilon = 10^{-3}$ от длины исходного интервала l_0 количество вычислений функции уменьшается от 1001 до 20, так как после 10 пар измерений методом дихотомии длина интервала $l_{10} = l_0/2^{10} = l_0/1024$ становится меньше заданной точности.

Однако в методе дихотомии при переходе к новому шагу не используется ни одно из ранее найденных значений функции. Оба измерения, необходимые для выделения части интервала, содержащей точку минимума, производятся заново. Метод, требующий вычисления функции только в одной точке, был бы более эффективен.

Это соображение реализовано в методах с использованием чисел Фибоначчи и золотого сечения. Первый из них по эффективности несколько превосходит второй, но метод золотого сечения проще.

Золотым сечением называется такое деление отрезка на две неравные части, при котором отношение длины всего отрезка к большей части равно отношению большей части к меньшей. Нетрудно проверить, что некоторая точка производит золотое сечение отрезка (a, b) длины l, если она расположена так, что ее расстояние от одного конца отрезка $l' = \left(3 - \sqrt{5}\right) l/2$, а от другого — $l'' = \left(\sqrt{5} - 1\right) l/2$, $l'' > l'$. Действительно, подставляя выражения для l' и l'' в отношения l/l'' и l''/l', получаем равенство $l/l'' = l''/l' = \left(\sqrt{5} + 1\right)/2 \approx 1{,}618$.

Возьмем на отрезке (a, b) две точки a_1, b_1, производящие его золотое сечение (рис. 4.7), и вычислим значения функции $F(a_1)$, $F(b_1)$. Точка с бо́льшим значением функции укажет отрезок (a, a_1) или (b_1, b) длины l', который

в силу унимодальности функции не может содержать точку ее минимума. Точка с меньшим значением функции укажет отрезок (a, b_1) или (a_1, b) длины l'', который содержит искомую точку минимума. Кроме того, для этого отрезка уже известна одна из двух точек, производящих его золотое сечение. Пусть для определенности на следующем шаге рассматривается отрезок (a, b_1), где $(a, a_1) > (a_1, b_1)$. Вычислим отношения $(a, b_1)/(a, a_1)$ и $(a, a_1)/(a_1, b_1)$. Подставляя значения $(a, b_1) = l'' = (\sqrt{5}-1)\,l/2$, $(a, a_1) = l' = (3-\sqrt{5})\,l/2$, $(a_1, b_1) = l'' - l' = (\sqrt{5}-2)\,l$, получаем $(a, b_1)/(a, a_1) = (a, a_1)/(a_1, b_1) = (\sqrt{5}+1)/2 \approx \approx 1{,}618$.

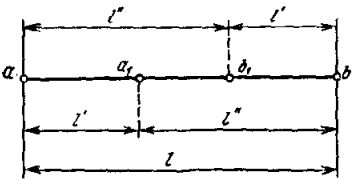

Рис. 4.7.

Таким образом, после двух вычислений функции интервал, содержащий точку оптимума, сократился меньше, чем в методе дихотомии: $l'' = (\sqrt{5}-1)\,l/2 > l/2$. Однако новый отрезок содержит точку (a_1 или b_1), которая производит его золотое сечение, и для продолжения процесса на следующем шаге понадобится одно вычисление функции, а не два, как в методе дихотомии. Второе значение функции на полученном отрезке измеряется в точке, отстоящей на расстоянии $(\sqrt{5}-2)\,l''$ от имеющейся точки золотого сечения.

Проведенные рассуждения позволяют сформулировать следующий алгоритм минимизации унимодальной функции $F(x)$ на интервале (a, b) с точностью ε (алгоритм золотого сечения).

1. Вычисляем координаты точек $a_1 = b - (\sqrt{5}-1) \times \times (b-a)/2$ и $b_1 = a + (\sqrt{5}-1)\,(b-a)/2 = a + b - a_1$.

2. Вычисляем $F(a_1)$, $F(b_1)$.

3. Если $F(a_1) > F(b_1)$, то переходим к 7.

4. Полагаем $b = b_1$, $b_1 = a_1$, $F(b_1) = F(a_1)$.

5. Если $(b-a) \leqslant \varepsilon$, то переходим к 10.

6. Вычисляем $a_1 = b_1 - (\sqrt{5}-2)\,(b-a)$, $F(a_1)$ и переходим к 3.

7. Полагаем $a = a_1$, $a_1 = b_1$, $F(a_1) = F(b_1)$.

8. Если $(b-a) \leqslant \varepsilon$, то переходим к 10.

9. Вычисляем $b_1 = a_1 + (\sqrt{5} - 2)(b - a)$, $F(b_1)$ и переходим к 3.

10. Выполняем заключительные операции по оценке положения минимума $x^* = (a + b)/2$ и значения функции $F(x^*)$.

4.3. МНОГОМЕРНАЯ МИНИМИЗАЦИЯ

Признак оптимальности. Для задачи выпуклого программирования

$$\min F(\mathbf{x});\ \mathbf{x} \geqslant 0;\ f_i(\mathbf{x}) \leqslant 0,\ i = 1, \ldots, m, \qquad (4.5)$$

в отличие от общей задачи нелинейного программирования существует критерий оптимальности решения. Это так называемые условия *Куна — Таккера*, распространяющие метод неопределенных множителей Лагранжа на случай, когда среди ограничений имеются не только равенства, но и неравенства. С их помощью устанавливается эквивалентность задачи выпуклого программирования и задачи отыскания седловой точки функции Лагранжа. Поясним понятие седловой точки.

Пусть для задачи (4.5) составлена функция Лагранжа (4.3). Точка $(\mathbf{x}^0, \mathbf{y}^0)$ называется *седловой точкой* функции $L(\mathbf{x}, \mathbf{y})$, если при любых $\mathbf{x} \geqslant 0$, $\mathbf{y} \geqslant 0$ имеют место неравенства

$$L(\mathbf{x}^0, \mathbf{y}) \leqslant L(\mathbf{x}^0, \mathbf{y}^0) \leqslant L(\mathbf{x}, \mathbf{y}^0). \qquad (4.6)$$

Из неравенств (4.6) видно, что седловой является такая точка $(\mathbf{x}^0, \mathbf{y}^0)$, в которой функция $L(\mathbf{x}, \mathbf{y})$ по направлениям (переменным) $\mathbf{x}$ имеет минимум, а по переменным $\mathbf{y}$ — максимум (рис. 4.8).

Запишем условия существования седловой точки $(\mathbf{x}^0, \mathbf{y}^0)$ функции Лагранжа в предположении дифференцируемости функций $F(\mathbf{x})$, $f_i(\mathbf{x})$, а следовательно, и функции $L(\mathbf{x}, \mathbf{y})$. При отсутствии ограничений на знак переменных x_j, y_i производные $\partial L/\partial x_j$, $\partial L/\partial y_i$ в этой точке должны обратиться в нуль. При наличии ограничений $x_j \geqslant 0$, $y_i \geqslant 0$ точка экстремума может быть на границе $x_j = 0$ или $y_i = 0$ (рис. 4.9). Здесь, как

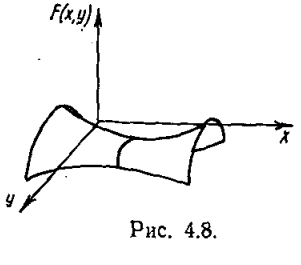

Рис. 4.8.

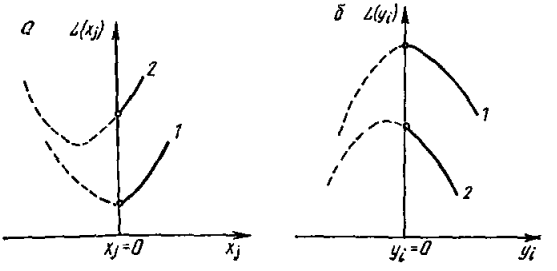

Рис. 4.9.

и прежде, возможен случай равенства нулю производных по x_j, y_i (кривые 1, рис. 4.9, *а*, *б*). Кроме того, поскольку точка $(\mathbf{x}^0, \mathbf{y}^0)$ есть точка минимума функции $L(\mathbf{x}, \mathbf{y})$ по переменным x и максимума по переменным y, на границе возможны значения производных $\partial L(\mathbf{x}^0, \mathbf{y}^0)/\partial x_j > 0$ и $\partial L(\mathbf{x}^0, \mathbf{y}^0)/\partial y_i < 0$ (кривые 2, рис. 4.9, *а*, *б*). Отсюда условия, состоящие в том, что в точке $(\mathbf{x}^0, \mathbf{y}^0)$ всегда либо производные, либо переменные обращаются в нуль, примут вид

$$\left.\begin{aligned} x_j^0 \frac{\partial L(\mathbf{x}^0, \mathbf{y}^0)}{\partial x_j} = 0, \; j = 1, \ldots, n; \\ y_i^0 \frac{\partial L(\mathbf{x}^0, \mathbf{y}^0)}{\partial y_i} = 0, \; i = 1, \ldots, m. \end{aligned}\right\} \tag{4.7}$$

Условия (4.7) являются необходимыми условиями того, что точка $(\mathbf{x}^0, \mathbf{y}^0)$ функции Лагранжа для задачи (4.3) является седловой точкой, и называются *условиями Куна — Таккера*. Понятие седловой точки используется в следующей теореме, которая устанавливает признак оптимальности для задачи выпуклого программирования.

Теорема 4.1 (теорема Куна — Таккера). *Для оптимальности вектора* $\mathbf{x}^0$ *в задаче (4.5) достаточно, чтобы при некотором векторе* $\mathbf{y}^0$ *точка* $(\mathbf{x}^0, \mathbf{y}^0)$ *была седловой для функции Лагранжа (4.3).*

◆ Пусть точка $(\mathbf{x}^0, \mathbf{y}^0)$ является седловой для функции (4.3). Тогда имеют место неравенства (4.6), которые запишем в следующем виде:

$$\sum_{i=1}^{m} y_i f_i(\mathbf{x}^0) \leqslant \sum_{i=1}^{m} y_i^0 f_i(\mathbf{x}^0); \tag{4.8}$$

$$F(\mathbf{x}^0) + \sum_{i=1}^{m} y_i^0 f_i(\mathbf{x}^0) \leqslant F(\mathbf{x}) + \sum_{i=1}^{m} y_i^0 f_i(\mathbf{x}). \tag{4.9}$$

По определению седловой точки неравенство (4.8) должно выполняться при всех $\mathbf{y} \geqslant 0$. Отсюда следует

$$f_i(\mathbf{x}^0) \leqslant 0, \quad i = 1, \ldots, m, \tag{4.10}$$

иначе можно было бы подобрать значения компонент y_i так, чтобы неравенство (4.8) нарушилось. Из (4.10) следует, что $\mathbf{x}^0$ принадлежит допустимой области задачи (4.5). Поскольку (4.8) справедливо при всех $\mathbf{y} \geqslant 0$, то оно имеет место и для $\mathbf{y} = 0$. Но (4.8) не нарушится при $\mathbf{y} = 0$ только тогда, когда $\sum_{i=1}^{m} y_i^0 f_i(\mathbf{x}^0) = 0$. Установив это, можем переписать неравенство (4.9) в виде $F(\mathbf{x}^0) \leqslant F(\mathbf{x}) + \sum_{i=1}^{m} y_i^0 f_i(\mathbf{x})$. При всех $y_i \geqslant 0$, $f_i(\mathbf{x}) \leqslant 0$ второе слагаемое в правой части неположительно. Тогда тем более $F(\mathbf{x}^0) \leqslant F(\mathbf{x})$, чем и устанавливается оптимальность вектора $\mathbf{x}^0$. ◆

Кун и Таккер в своей теореме дали доказательство того, что приведенный признак оптимальности является не только достаточным, но и необходимым.

Пример 4.3. Решить задачу

$$\min z = 5(x_1-3)^2 + 10(x_2-4)^2; \quad x_1 + x_2 \leqslant 8;$$
$$x_1 + 3x_2 \geqslant 15; \quad x_1, x_2 \geqslant 0.$$

Составляем функцию Лагранжа: $L(x_1, x_2, y_1, y_2) = 5(x_1-3)^2 + 10 \times (x_2-4)^2 + y_1(x_1+x_2-8) + y_2(-x_1-3x_2+15)$. Строим систему (4.7):

$$x_1[10(x_1-3) + y_1 - y_2] = 0; \quad y_1[x_1 + x_2 - 8] = 0;$$
$$x_2[20(x_2-4) + y_1 - 3y_2] = 0; \quad y_2[-x_1 - 3x_2 + 15] = 0.$$

При решении этой системы надо рассмотреть ряд вариантов значений неизвестных: $x_1 > 0$, $x_2 > 0$; $x_1 > 0$, $x_2 = 0$ и т. д. Возьмем случай $x_1 > 0$, $x_2 > 0$. Выразим из первого уравнения x_1, из второго x_2. Подставив их в третье и четвертое уравнения, получим

$$y_1\left[-\frac{3}{20}y_1 + \frac{5}{20}y_2 - 1\right] = 0; \; y_2\left[\frac{5}{20}y_1 - \frac{11}{20}y_2\right] = 0,$$

что дает два решения: $y_1 = 0$, $y_2 = 0$, $x_1 = 3$, $x_2 = 4$ и $y_1 = -27{,}5$, $y_2 = -12{,}5$, $x_1 = 4{,}5$, $x_2 = 3{,}5$. Поскольку вектор $\mathbf{y} = (y_1, y_2)$ должен быть неотрицательным, остается первое решение. Проверяем, является ли точка (3,4,0,0) седловой для функции $L(x_1, x_2, y_1, y_2)$. Полагая $y_1 = y_2 = 0$, получаем функцию $L(x_1, x_2, 0, 0) = 5(x_1-3)^2 + 10(x_2-4)^2$, которая при $x_1 = 3$, $x_2 = 4$ достигает минимума. Считая $x_1 = 3$, $x_2 = 4$, имеем функцию $L(3, 4, y_1, y_2) = -y_1$, которая при $y_1 = 0$ достигает максимума. Следовательно, точка (3,4,0,0) является седловой, а решение $x_1 = 3$, $x_2 = 4$ — оптимальным со значением $F(\mathbf{x}) = F(3,4) = 0$.

Как нетрудно убедиться, целевая функция $z=F(x_1, x_2)$ задачи выпукла, откуда следует, что найденная точка минимума единственна и нет необходимости рассматривать другие решения системы уравнений.

Таким образом, задача выпуклого программирования (4.5) эквивалентна задаче о седловой точке соответствующей функции Лагранжа и сводится к решению системы (4.7). Однако решение этой системы, как и системы (4.4) для классической задачи с ограничениями-равенствами, может быть связано с большими вычислительными трудностями, в связи с чем вновь возникает вопрос о построении прямых методов минимизации.

Методы минимизации, предназначенные для реализации на ЭВМ, являются обычно итерационными. Точные методы, позволяющие отыскивать решение за конечное число шагов, разработаны в настоящее время только для минимизации квадратичных функций.

Итерационные, или шаговые, методы сводят задачу минимизации заданной функции $F(\mathbf{x})$ к построению последовательности точек $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$, такой, что на каждом следующем члене последовательности значение функции убывает

$$F(\mathbf{x}^{k+1}) < F(\mathbf{x}^k), \quad k=1, 2, \ldots \tag{4.11}$$

Вычисление очередной точки $\mathbf{x}^{k+1}$ разбивается на два этапа: выбор направления s_k движения (спуска) из точки $\mathbf{x}^k$ в сторону уменьшения $F(\mathbf{x})$ и определение шага спуска λ_k. Различия между методами определяются способами выбора соответствующего направления и организации движения вдоль него. При классификации методов удобно различать их по величине наивысшего порядка производных минимизируемой функции, участвующих в расчетах. Так, в методах второго порядка используется матрица вторых частных производных, первого порядка — первые производные, а нулевого — только значения функции.

Градиентные методы. Рассмотрим задачу $\min F(\mathbf{x})$, $\mathbf{x} \in R^n$ минимизации нелинейной функции $F(\mathbf{x})$ без ограничений на вектор переменных $\mathbf{x}$. Будем предполагать, что $F(\mathbf{x})$ дифференцируема по всем переменным x_j, $j=1, \ldots, n$.

Дифференцируемая функция $F(\mathbf{x})$ может быть разложена в ряд Тейлора в точке $\mathbf{x}$. Если приращения Δx_1, Δx_2, ..., Δx_n малы, в разложении можно ограничиться главной частью приращения: $F(\mathbf{x}+\Delta\mathbf{x}) \approx F(\mathbf{x}) + F'(\mathbf{x})\,\Delta\mathbf{x}$.

Здесь $F'(\mathbf{x})$ — градиент функции $F(\mathbf{x})$, представляющий собой вектор $F'(\mathbf{x}) = (\partial F(\mathbf{x})/\partial x_1, \partial F(\mathbf{x})/\partial x_2, \ldots, \partial F(\mathbf{x})/\partial x_n)$, компонентами которого являются частные производные функции в точке $\mathbf{x}$. При $F'(\mathbf{x}) \neq \mathbf{0}$ направление наискорейшего возрастания функции $F(\mathbf{x})$ в точке $\mathbf{x}$ совпадает с направлением градиента $F'(\mathbf{x})$, а направление наискорейшего убывания — с направлением антиградиента $-F'(\mathbf{x})$.

Это свойство градиента и лежит в основе методов первого порядка, или градиентных. В каждом из них процесс начинается в некоторой *точке начального приближения* $\mathbf{x}^0$. Отметим, что удачный выбор точки $\mathbf{x}^0$ может существенно ускорить сходимость метода, однако общих правил выбора начального приближения нет.

Пусть некоторая начальная точка $\mathbf{x}^0$ выбрана. Тогда *метод градиента* минимизирует функцию $F(\mathbf{x})$ построением последовательности $\{\mathbf{x}^k\}$, элементы которой определяются следующим образом:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \lambda_k F'(\mathbf{x}^k), \ \lambda_k > 0, \ k = 0, 1, 2, \ldots \qquad (4.12)$$

Если в точке $\mathbf{x}^k$ градиент отличен от нуля, всегда можно выбрать такое число λ_k, называемое длиной шага, чтобы перейти в точку $\mathbf{x}^{k+1}$, где $F(\mathbf{x}^{k+1}) < F(\mathbf{x}^k)$. Если же $F'(\mathbf{x}^k) = \mathbf{0}$, то в случае выпуклости $F(\mathbf{x})$ точка $\mathbf{x}^k$ является точкой ее минимума; в других случаях необходимо проверить, является ли точка $\mathbf{x}$ экстремальной.

Градиентные методы различаются способами выбора градиентного шага λ_k. Наиболее простой из них состоит в том, что на каждой итерации берут одно и то же значение шага $\lambda_k = \lambda$, если только при этом выполняется условие монотонности (4.11). При нарушении монотонности число λ уменьшают, пока монотонность не восстановится. На практике применяется и более гибкая тактика «разгона и торможения», при которой величина шага увеличивается или уменьшается в зависимости от поведения функции в ходе поиска.

Для повышения быстродействия метода целесообразно каждую итерацию выполнять с максимально возможным шагом. Метод, реализующий эту идею, называется методом *скорейшего спуска*. Опишем его. Пусть в начальной точке $\mathbf{x}^0$ вычислен градиент $F'(\mathbf{x}^0)$. Будем двигаться в направлении антиградиента до тех пор, пока это приводит к уменьшению функции. Точка $\mathbf{x}^1$, в которой функция перестает уменьшаться, является точкой ее минимума в направлении $-F'(\mathbf{x}^0)$. В точке $\mathbf{x}^1$ вычисляем новый гра-

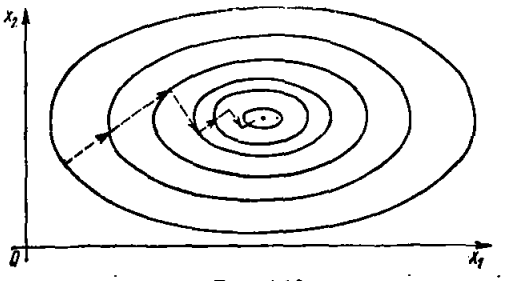

Рис. 4.10.

диент и в направлении, обратном градиенту, делаем максимально возможный шаг λ_1, приводящий в точку $\mathbf{x}^2$, и т. д. Таким образом, очередная точка $\mathbf{x}^{k+1}$ в (4.12) есть точка минимума функции $F(\mathbf{x})$ на луче $\mathbf{x}=\mathbf{x}^k-\lambda F'(\mathbf{x}^k)$, $\lambda\geqslant 0$, направленном из точки $\mathbf{x}^k$ по антиградиенту. Отсюда длина шага λ_k определится из условия минимума функции $F(\lambda)$ одной переменной — длины шага λ:

$$F_k(\lambda_k)=\min_{\lambda\geqslant 0} F(\mathbf{x}^k-\lambda F'(\mathbf{x}^k)).$$

Для поиска минимума функции одной переменной могут быть использованы методы одномерной минимизации, описанные в § 4.2. Итерационный процесс построения последовательности $\{\mathbf{x}^k\}$ прекращается при выполнении некоторого критерия окончания поиска, устанавливающего близость точки $\mathbf{x}^k$ к искомой точке минимума. В качестве критерия можно использовать оценку близости или двух соседних точек последовательности, или значений функции в соседних точках, или модуль градиента и прекратить счет при выполнении одного из следующих условий:

$$|\mathbf{x}^k-\mathbf{x}^{k+1}|\leqslant\delta_1, \text{или } |F(\mathbf{x}^k)-F(\mathbf{x}^{k+1})|\leqslant\delta_2, \text{или } |F'(\mathbf{x}^k)|\leqslant\delta_3,$$

где δ_1, δ_2, δ_3 — заданные числа.

Траектория поиска для функции двух переменных показана на рис. 4.10. Движение осуществляется по направлениям, ортогональным линиям равного уровня.

Итак, последовательность операций в методе скорейшего спуска такова. В начальной точке $\mathbf{x}^0$ вычисляется градиент $F'(\mathbf{x}^0)$ и определяется направление антиградиента $-F'(\mathbf{x}^0)$. Затем в этом направлении одним из методов одномерной минимизации выполняется одномерный поиск

минимума. Получаем длину шага λ, определяющего величину смещения в направлении $-F'(\mathbf{x}^0)$ к новой точке $\mathbf{x}^1$. В точке $\mathbf{x}^1$ операции повторяются, и так до тех пор, пока не будет выполнено одно из условий окончания поиска или какая-либо их комбинация.

Методы нулевого порядка. Методы, требующие для своей реализации вычисления первых или вторых производных, при определенных условиях обеспечивают хорошую сходимость. Однако они обычно чувствительны к выбору начальной точки поиска: для их сходимости она должна быть близка к окрестности минимума. Кроме того, не во всех практических задачах имеется возможность вычислить первые и тем более вторые производные. И хотя методы нулевого порядка сходятся относительно медленно, они получили достаточно широкое распространение.

Общим для большинства методов этой группы является то, что спуск из очередной точки производится по направлению одной из координатных осей. Последовательность выбора координатных осей устанавливается различными способами. В известном *методе Гаусса — Зайделя* все оси выбираются поочередно в фиксированном порядке. Существуют методы, в которых направление координатных осей остается неизменным в течение всего поиска, и методы, в которых направление осей меняется в зависимости от результатов предыдущих шагов. Методы нулевого порядка реализуются на ЭВМ наиболее просто, благодаря чему они широко применяются на практике.

К методам, использующим лишь значения функции и не требующим вычисления производных, относятся *методы случайного поиска*, у которых перемещение к точке экстремума осуществляется случайным образом. Один из простейших методов случайного поиска состоит в следующем. Выбирают величину шага λ и из начальной точки $\mathbf{x}^0$ совершают в случайном направлении перемещение на расстояние λ. Если значение $F(\mathbf{x})$ в новой точке не уменьшилось, то возвращаются в точку $\mathbf{x}^0$ и осуществляют в случайном направлении новый шаг на расстояние λ. В случае неудачи опять возвращаются в точку $\mathbf{x}^0$ и процесс повторяют до тех пор, пока очередное перемещение не приведет к уменьшению целевой функции. После этого в направлении последнего перемещения делают шаги величиной λ до тех пор, пока приращение функции

$F(\mathbf{x})$ не изменит знак. Затем процесс повторяют. При этом если из очередной точки $\mathbf{x}^k$ определенное число случайных перемещений на расстояние λ не привело к уменьшению функции, то точку $\mathbf{x}^k$ считают точкой минимума.

Главное преимущество методов случайного поиска заключается в их простоте; эти методы не налагают никаких ограничений на функцию $F(\mathbf{x})$. Основной их недостаток — медленная сходимость. Для ее ускорения включают процедуры, которые анализируют результаты выполненных операций и выделяют направления с большей вероятностью убывания минимизируемой функции. Методы такого типа называются методами *случайного поиска с обучением*.

Метод штрафных функций. Если минимум целевой функции находится внутри допустимой области изменения переменных, то наличие ограничений не влияет на процесс поиска решения. Поиск минимума осуществляется практически так же, как и в случае без ограничений. Если точка минимума находится за пределами допустимой области, то отыскание наименьшего значения целевой функции при ограничениях на переменные становится существенно сложнее.

Одним из наиболее эффективных методов решения задач с ограничениями является метод штрафных функций. Идея этого метода такова. Вместо того чтобы непосредственно решать задачу вида (4.1) с ограничениями, переходят к последовательности задач минимизации вспомогательной функции $R(\mathbf{x}) = F(\mathbf{x}) + H(\mathbf{x})$, в которой на $\mathbf{x}$ не накладывается каких-либо ограничений. Добавление слагаемого $H(\mathbf{x})$ к функции $F(\mathbf{x})$ можно рассматривать как введение специального штрафа за неточное выполнение ограничений $f_i(\mathbf{x}) \leqslant 0$, $i=1, \ldots, m$. Поэтому слагаемое $H(\mathbf{x})$ часто называется *штрафной функцией*, откуда и происходит название рассматриваемого метода. Метод штрафных функций позволяет, таким образом, сводить задачи на условный экстремум к задачам на безусловный экстремум. Штрафные функции должны быть равны нулю для тех $\mathbf{x}$, при которых ограничения выполняются, и бесконечно велики для тех $\mathbf{x}$, при которых ограничения нарушаются.

Исходя из этих соображений, следует выбирать штрафную функцию $H_k(\mathbf{x})$ так, чтобы она обладала свойствами

$$\lim_{k\to\infty} H_k(\mathbf{x}) = \begin{cases} 0 & \text{при } f_i(\mathbf{x}) \leqslant 0; \\ \infty & \text{при } f_i(\mathbf{x}) > 0. \end{cases}$$

Тогда точка $\mathbf{x}$, минимизирующая $R(\mathbf{x})$, минимизирует и $F(\mathbf{x})$. Однако из-за невозможности точно представить на ЭВМ величину ∞ используются различные приближенные способы задания штрафной функции, предусматривающие, в частности, увеличение влияния ограничений на минимизируемую функцию при приближении к точке минимума.

Можно образовать много функций, обладающих указанными свойствами. Например, возьмем функцию

$$H_k(\mathbf{x}) = A_k H(\mathbf{x}), \quad H(\mathbf{x}) = \sum_{i=1}^{m} (\max(f_i(\mathbf{x}), 0))^p,$$

где $A_k > 0$, $k = 1, 2, \ldots, \lim\limits_{k\to\infty} A_k = \infty$, $p \geqslant 1$ — фиксированное число. Таким образом, исходную задачу решают в результате последовательной минимизации функции $R(\mathbf{x})$ с возрастающим штрафом за нарушение ограничений. При этом точка минимума, полученная на предыдущем шаге k, служит начальной точкой поиска на следующем шаге $k + 1$.

Контрольные вопросы и упражнения

1. Какие задачи относятся к задачам нелинейного программирования?
2. Назовите типы задач нелинейного программирования. Чем они характеризуются?
3. Каковы особенности задач нелинейного программирования по сравнению с задачами линейного программирования?
4. В решении каких задач эффективно используется симплекс-метод линейного программирования?
5. Что лежит в основе градиентных методов? Опишите их преимущества и недостатки.
6. Что такое штрафная функция и каково ее назначение?
7. Опишите основную идею метода случайного поиска в простейшем случае. Каковы недостатки и преимущества методов случайного поиска?
8. Дана функция $F(x_1, x_2, x_3) = 2x_1 + 3x_2 + 4x_3 - x_1x_2 - x_2x_3 - x_1x_3 - 2x_1^2 + 2x_2^2 + 2x_3^2$. В какую точку приведет поиск оптимума с шагом λ из начальной точки $\mathbf{x}^0$ методами покоординатного спуска, градиента и скорейшего подъема (спуска) после пяти шагов? Значения параметров для различных вариантов приведены в табл. 4.1.

Таблица 4.1

Номер варианта	x^0	λ
1	(0,0,0)	0,2
2	(0,0,0)	0,1
3	(1,1,1)	0,2
4	(1,1,1)	0,1

Указание. Предварительно установить вид выпуклости функции и характер экстремума (максимум, минимум).

ЛИТЕРАТУРА

Основная

Абрамов Л. М., Капустин В. Ф. Математическое программирование.— Л.: Изд-во ЛГУ, 1981.— 328 с.

Ашманов С. А. Линейное программирование.— М.: Наука, 1981.— 384 с.

Карманов В. Г. Математическое программирование.— М.: Наука, 1975.— 272 с.

Кузнецов Ю. Н., Кузубов В. И., Волощенко А. Б. Математическое программирование.— М.: Высш. шк., 1980.— 300 с.

Линейное и нелинейное программирование/ Под ред. И. Н. Ляшенко.— Киев: Вища шк., 1975.— 372 с.

Дополнительная

Гейл Д. Теория линейных экономических моделей.— М.: Изд-во иностр. лит-ры, 1963.— 418 с.

Гольштейн Е. Г., Юдин Д. Б. Задачи линейного программирования транспортного типа.— М.: Наука, 1969.— 389 с.

Данциг Дж. Линейное программирование, его обобщения и применения.— М.: Прогресс, 1966.— 600 с.

Корбут А. А., Финкельштейн Ю. Ю. Дискретное программирование.— М.: Наука, 1968.— 368 с.

Финкельштейн Ю. Ю. Приближенные методы и прикладные задачи дискретного программирования.— М.: Наука, 1976.— 264 с.

Форд Л. Р., Фалкерсон Д. Р. Потоки в сетях.— М.: Мир, 1966.— 276 с.

Химмельблау Д. Прикладное нелинейное программирование.— М.: Мир, 1975.— 534 с.

Юдин Д. Б., Гольштейн Е. Г. Линейное программирование (теория, методы и приложения).— М.: Наука, 1969.—424 с.

ОГЛАВЛЕНИЕ

Владимир Антонович Балашевич

ОСНОВЫ МАТЕМАТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Зав. редакцией Л. Д. Духвалов
Редактор Н. М. Латышева
Мл. редакторы В. М. Кушнлевич, М. С. Молчанова
Обложка и художественное редактирование Ю. С. Сергачева
Техн. редактор Н. П. Тихонова
Корректор Л. А. Еркович

ИБ № 1921

Сдано в набор 06.06.84. Подписано в печать 13.03.85. АТ 18522. Формат 84×108¹/₃₂. Бумага тип. № 1. Гарнитура литературная. Высокая печать. Усл. печ. л. 9.24. Усл. кр.-отт. 9,45. Уч.-изд. л. 9,38. Тираж 5900 экз. Зак. 466. Цена 35 к.

Издательство «Вышэйшая школа» Государственного комитета БССР по делам издательств, полиграфии и книжной торговли. 220048, Минск, проспект Машерова, 11.

Минский ордена Трудового Красного Знамени полиграфкомбинат МППО им. Я. Коласа. 220005. Минск, ул. Красная, 23.

Балашевич В. А.

Б20 Основы математического программирования: [Учеб. пособие для инж.-экон. и экон. спец.]— Мн.: Выш. шк., 1985.— 173 с., ил.

35 к.

Излагаются основные разделы курса математического программирования: линейное, целочисленное, нелинейное программирование, задачи транспортного типа. Рассматриваются комбинаторные свойства задач транспортного типа и целочисленного программирования, вопросы реализации алгоритмов их решения на ЭВМ. Изложение ведется в векторно-матричных обозначениях.

Б $\frac{1702000000-043}{M304(05)-85}$14—85 ББК 22.143я73