Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра информатики

О.И. Костюкова

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Учебное пособие
для студентов специальности 31 03 04 «Информатика»
всех форм обучения

Минск 2003

УДК 519.854.3(519.852.35, 519.854.2, 519.832)
ББК 22.18 я 73
К 72

Р е ц е н з е н т:
зав. кафедрой математического обеспечения автоматизированных систем управления Белгосуниверситета, д-р техн. наук, проф. И.В. Совпель

Костюкова О.И.

К 72 Исследование операций: Учеб. пособие для студ. спец. 31 03 04 «Информатика» всех форм обучения / О.И. Костюкова. – Мн.: БГУИР, 2003, 94 с.: ил.

ISBN 985-444-548-8.

Учебное пособие составлено в соответствии с рабочей программой курса «Исследование операций». В него включены сведения об основных результатах и алгоритмах теории исследования операций. Дается представление о математическом аппарате исследования операций, рассматриваются и анализируются математические модели основных типов задач, встречающихся в приложениях.

Пособие может быть рекомендовано для курсового и дипломного проектирования.

УДК 519.854.3(519.852.35, 519.854.2, 519.832)
ББК 22.18 я 73

ISBN 985-444-548-8

СОДЕРЖАНИЕ

Введение 4

Глава 1. Целочисленное линейное программирование 5
§ 1. Примеры прикладных задач, содержащих условия целочисленности. Постановка задачи целочисленного программирования 6
§ 2. Метод ветвей и границ 11
§ 3. Метод Гомори (метод отсечений) для полностью целочисленных задач 20
Глава 2. Динамическое программирование 27
§ 1. Основные принципы динамического программирования 27
§ 2. Задача распределения ресурсов 28
§ 3. Задача сетевого планирования 32
Глава 3. Кратчайшие пути 37
§ 1. Задача о кратчайшем пути 38
§ 2. Кратчайшие пути между всеми парами вершин (задача о многополюсной кратчайшей цепи) 47
Глава 4. Потоки в сетях 54
§ 1. Примеры прикладных задач, имеющих сетевую форму 54
§ 2. Задача о максимальном потоке 59
§ 3. Задача о назначениях 67
§ 4. Задача коммивояжера 74
Глава 5. Линейное программирование и теория игр 82
§ 1. Постановка задачи 82
§ 2. Матричные игры. Смешанные стратегии 83
§ 3. Эквивалентность матричной игры и задачи линейного программирования 90
Литература 93

Введение

Среди многочисленных проблем, возникновение которых обусловлено возросшими запросами производства и бурным развитием информационных систем и технологий, одной из наиболее важных является проблема совершенствования управления во всех областях человеческой деятельности. При решении широкого круга задач оптимизации управляющих решений неоценимую услугу оказывает теория исследования операций.

Операцией называется совокупность взаимосогласованных действий, направленных на достижение некоторой цели. До тех пор, пока цель не определена, нет смысла говорить об операции. Если же цель определена и существуют разные пути ее достижения, то желательно найти оптимальный из этих путей.

Первоначально исследование операций было определено как научный метод, дающий в распоряжение руководителя количественные основания для принятия решения. С течением времени круг интересов исследователей операций значительно расширился и охватывает сегодня такие области, как экономика, военное дело, энергетика, освоение природных ресурсов, автоматизация проектирования.

Исследование операций – достаточно молодое научное направление. Становление исследования операций как научной дисциплины относится к 1952 г. и связывается с организацией Американского общества по исследованию операций. Интересно отметить, что к моменту официального рождения научного направления были опубликованы всего лишь две работы на эту тему.

Следующее десятилетие – 1952–1962 гг. было достаточно «тихим» в развитии исследования операций. Появляется ряд публикаций, которые представляют собой в основном либо труды конференций по исследованию операций, либо специальные выпуски различных институтов и организаций.

После 1962 г. положение резко изменяется – наступает «бурный» период в развитии исследования операций. Появляется множество научных работ, создаются научные общества и федерации по исследованию операций, публикуются специальные журналы.

Чем объяснить этот всплеск интереса к исследованию операций?

1. Начиная со второй мировой войны в производственно-коммерческой сфере резко возросла роль конкуренции. Руководители крупных корпораций стали понимать, что, с одной стороны, совершенствование традиционных способов накопления и обработки информации приводит к существенным выгодам, а с другой стороны – резко усложнился характер управленческих задач, например, план выпуска продукции того или иного предприятия должен учитывать спрос покупателей, потребности в сырье, необходимые оборотные фонды, мощности оборудования, вероятность возникновения технических неполадок, а также ограничения производственно-технологического характера. Для решения этих задач уже не хватало таких аргументов, как «здравый смысл» и «накопленный опыт», хотя и они очень важны.

2. Огромные успехи в развитии вычислительной техники.

Относительно термина «исследование операций». Это было начальное название данного научного направления. Может быть, этот термин не совсем удачен: а) термин «операция» является сильно перегруженным; б) слово «исследование» тоже не совсем удачное, так как порождает ложное впечатление «созерцательности» самого предмета. Фактически же наблюдается обратное – за последние десятилетия применение методов исследования операций неоднократно подтверждало большие возможности и высокую эффективность этих методов при решении практических задач.

Несмотря на то, что термин «исследование операций» не совсем точно отражает суть предмета и часто вводит в заблуждение, он сохранился до настоящего времени. Это дань традиции.

Термин «исследование операций» – operations research. Есть другие термины: operational research – операционные исследования (английская школа), американцы часто используют термин «наука об управлении» – management science.

Цели данного курса:

1) дать представление о математическом аппарате исследования операций;

2) научиться строить, исследовать и анализировать математические модели конкретных задач.

В исследовании операций моделирование играет важную роль. От выбора модели во многом зависит эффективность (а часто и просто возможность или невозможность) решения конкретной задачи.

С основными классами задач и основными приемами моделирования мы познакомимся в данном курсе.

Теоретическую основу курса составляют такие дисциплины, как линейное и нелинейное программирование, методы анализа сетей, теория игр.

В курсе будут рассмотрены следующие вопросы:

1. Целочисленное линейное программирование.
2. Динамическое программирование.
3. Кратчайшие пути.
4. Потоки в сетях.
5. Линейное программирование и теория игр.

Глава 1. Целочисленное линейное программирование

Целочисленное линейное программирование (ЦЛП) ориентировано на решение задач линейного программирования (ЛП), в которых на все или часть переменных наложены условия целочисленности. Различают полностью целочисленные задачи и частично целочисленные задачи.

К необходимости рассматривать целочисленные задачи мы приходим по двум причинам:

1) существует много практических задач, которые изначально по своей природе являются целочисленными;

2) существуют задачи, которые явно не содержат требований целочисленности, но содержат условия, которые трудно или невозможно (в исходной постановке) записать в виде формул (т.е. математически формализовать). Это так называемые «некорректные» задачи. Анализ таких задач, введение новых целочисленных переменных позволяют преодолеть эту трудность и свести задачу к стандартному виду.

Рассмотрим ряд примеров задач второго типа.

§1. Примеры прикладных задач, содержащих условия целочисленности. Постановка задачи целочисленного программирования

1. Задача с постоянными элементами затрат. Во многих типах задач планирования производства рассматривается следующая ситуация. Предприятие производит N видов продукции. Затраты на производство продукции j-го вида складываются из постоянных затрат в объеме K_j, не зависящем от количества произведенной продукции, и текущих издержек на производство единицы продукции.

Таким образом, если x_j – объем выпуска продукции j-го вида, то функцию суммарных затрат можно представить в виде

$$C_j(x_j) = \begin{cases} K_j + c_j x_j, & \text{если } x_j > 0; \\ 0, & \text{если } x_j = 0. \end{cases}$$

Естественно стремление минимизировать общие затраты по всем типам продукции:

$$Z = \sum_{j=1}^{N} C_j(x_j) \to min. \qquad (1)$$

Критерий (1) не является линейным вследствие разрыва в начале координат. Поэтому он практически не пригоден для дальнейшего исследования.

Введем новые булевы переменные, которые позволят нам записать критерий (1) в лучшем (с аналитической точки зрения) виде.

Пусть

$$y_j = \begin{cases} 0, \text{ если } x_j = 0, \\ 1, \text{ если } x_j > 0. \end{cases} \qquad (2)$$

Последнее условие можно переписать в виде

$$x_j \le M y_j, \qquad (3)$$

где M – достаточно большое число. Оно должно быть таким, чтобы $x_j \le M$ при любом допустимом значении x_j. Строго говоря, пока условие (3) не эквивалентно условию (2)!

Теперь рассмотрим критерий

$$Z = \sum_{j=1}^{N} \left(c_j x_j + K_j y_j \right) \to \min \tag{4}$$

при дополнительных ограничениях

$$0 \le x_j \le M y_j \ , \ y_j = 0 \vee 1, \ j = \overline{1, N} . \tag{5}$$

Покажем, что (4), (5) эквивалентны (1). Действительно, при $x_j > 0$ имеем $y_j = 1$ и

$$c_j x_j + K_j y_j = c_j x_j + K_j .$$

Пусть $x_j = 0$. Из (5) следует, что y_j может быть равным 0 или 1. Однако, согласно (4), наша цель – минимизировать критерий (4), следовательно, с учетом (4) мы обязательно должны выбрать $y_j = 0$.

С аналитической точки зрения, функция (4) является хорошей, так как она является линейной и непрерывной.

Следует подчеркнуть, что исходная задача с постоянными элементами затрат не имела никакого отношения к целочисленному программированию. Однако преобразованная задача является частично целочисленной с булевыми переменными. Необходимость такого преобразования была вызвана чисто аналитическими причинами.

2. Задача планирования производственной линии. Предположим, что имеется один станок, на котором надо произвести n различных операций. На порядок выполнения некоторых операций наложены ограничения технологического характера и, кроме того, есть ограничения, согласно которым каждая операция должна быть выполнена к заданному сроку. Таким образом, в задаче имеются ограничения трех типов:

1) ограничения на последовательность выполнения операций;

2) условие, что все операции выполняются на одном станке, т.е. операции нельзя распараллелить;

3) наличие временных сроков на осуществление операций.

Рассмотрим первый тип ограничений. Пусть x_j – момент начала выполнения j-й операции, a_j – время, требуемое на выполнение j-й операции. Если операция i должна предшествовать операции j, то должно выполняться неравенство

$$x_i + a_i \le x_j . \tag{6}$$

Рассмотрим ограничение второго типа. Для операций i и j, не выполняемых на станке одновременно, должно выполняться одно из соотношений:

$x_i - x_j \geq a_j$, если в оптимальном решении j предшествует i,

или

$x_j - x_i \geq a_i$, если в оптимальном решении i предшествует j.

Логическое ограничение вида «или–или» не укладывается в рамки обычной задачи ЛП, что вызывает существенные трудности при построении математической модели задачи. Эти трудности можно обойти путем введения вспомогательных булевых переменных

$$y_{ij} = \begin{cases} 0, & \text{если } j \text{ предшествует } i, \\ 1, & \text{если } i \text{ предшествует } j. \end{cases}$$

Пусть $M > 0$ достаточно велико, тогда ограничение вида «или–или» эквивалентно следующей системе неравенств:

$$My_{ij} + (x_i - x_j) \geq a_j, \tag{7}$$

$$M(1 - y_{ij}) + (x_j - x_i) \geq a_i. \tag{8}$$

Действительно, если на оптимальном решении $y_{ij} = 0$, то ограничение (8) становится избыточным, а ограничение (7) – активным:

$$(x_i - x_j) \geq a_j, \; M + (x_j - x_i) \geq a_i.$$

Если на оптимальном решении $y_{ij} = 1$, то ограничение (7) – избыточное, а ограничение (8) – активное:

$$M + (x_i - x_j) \geq a_j, \quad (x_j - x_i) \geq a_i.$$

Таким образом, введение булевых переменных позволило избежать логического «или–или» с помощью **линейных** неравенств.

Ограничения третьего типа учитываются следующим образом. Пусть операция j должна быть завершена к моменту времени d_j, тогда

$$x_j + a_j \leq d_j. \tag{9}$$

Если обозначить через t суммарное время, требуемое для выполнения всех n операций, то рассматриваемая задача принимает вид

$$Z = t \to min$$

при ограничениях (6) – (9) и

$$x_j + a_j \leq t, \quad j = \overline{1,n}; \; y_{ij} = 0 \vee 1, \; i = \overline{1,n}; \; j = \overline{1,n}.$$

3. Задачи с дихотомией. Предположим, что в некоторой ситуации требуется выполнение k ограничений из общего числа m ограничений. При этом заранее не известно, какие именно ограничения должны выполняться. Запишем ограничения в виде

$$g_i(x_1,...,x_n) \leq b_i, \quad i = 1,2,...,m.$$

Определим новую переменную

$$y_i = \begin{cases} 0, \text{если } i\text{-е ограничение должно обязательно выполняться;} \\ 1, \text{если } i\text{-е ограничение может нарушаться.} \end{cases}$$

Если $M > 0$ достаточно велико, то выполнение по крайней мере k ограничений из m обеспечивается соблюдением соотношений:

$$g_i(x_1, ..., x_n) \le b_i + My_i, \quad y_i = 0 \vee 1, \quad i = \overline{1,m},$$

$$\sum_{i=1}^{m} y_i = m - k.$$

Аналогичная ситуация возникает, когда правая часть некоторого ограничения может принимать **одно** из нескольких значений:

$$g(x_1, x_2, ..., x_n) \le b, \quad b \in \{b_1, b_2, ..., b_k\}.$$

Данное ограничение можно записать в виде

$$g(x_1, ..., x_n) \le \sum_{s=1}^{k} b_s y_s, \quad \sum_{s=1}^{k} y_s = 1, \quad y_s = 0 \vee 1, \quad s = \overline{1,k},$$

где

$$y_s = \begin{cases} 1, \text{ если } b = b_s, \\ 0 - \text{в противном случае.} \end{cases}$$

Есть еще много типов задач, которые по постановке не относятся к целочисленным, но построение оптимального решения в этих задачах требует перехода к моделям целочисленного программирования: задача коммивояжера, задача составления расписания обработки деталей на станках, задача баланса сборочной линии и др. Некоторые из этих задач мы еще будем рассматривать далее подробно. Построение моделей целочисленного программирования для таких ситуаций в некоторой степени является искусством.

4. Постановка задачи линейного целочисленного программирования и ее связь с задачами линейного программирования. В общей постановке задача ЦЛП имеет вид

$$c'x \to max, \tag{10}$$

$$Ax = b, \; d_{*j} \le x_j \le d_j^*, \; j \in J, \tag{11}$$

$$x_j - \text{целое}, \; j \in I, \tag{12}$$

здесь $x = (x_j, j \in J), J = \{1,2,...,n\}, I \subset J; \; A \in R^{m \times n}$, rank A $= m$, $b \in R^m$, $c \in R^n$.

Встает вопрос: что сложнее – задача ЦЛП (10) – (12) или задача ЛП (10) – (11)?

Многие ответят, что задача ЦЛП проще, так как в этой задаче меньше допустимых планов, например, если $-\infty < d_* \le d^* < \infty$, то количество допустимых планов конечно.

Однако справедливо обратное. Проиллюстрируем это на примере.

Пример. Рассмотрим задачу

$$21x_1 + 11x_2 \to \max, \tag{13}$$

$$7x_1 + 4x_2 \le 13,\ x_1 \ge 0, x_2 \ge 0, \tag{14}$$

$$x_1, x_2 - \text{целые}. \tag{15}$$

Эта задача имеет единственный оптимальный план $x^0 = (x_1 = 0, x_2 = 3)$ (см. рис. 1.1).

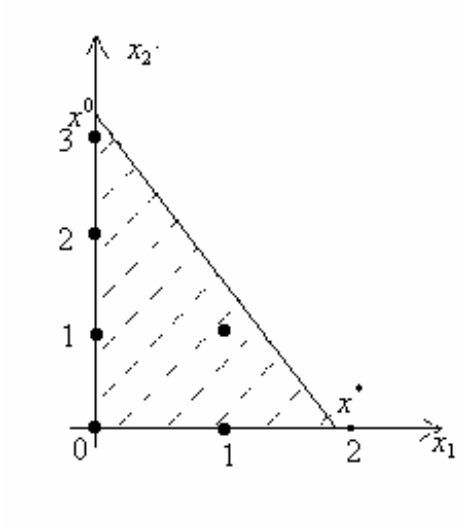


Рис. 1.1

Существует такое мнение, что задачи ЦЛП можно решать следующим образом: надо решить задачу без предположения целочисленности, а потом «округлить» полученное нецелочисленное решение до целочисленного.

Посмотрим, что мы получим, если воспользуемся этим приемом в нашей задаче. Оптимальное решение нецелочисленной задачи (13), (14) имеет вид $x^* = (x_1 = \frac{13}{7}, x_2 = 0)$. Очевидное округление приводит к вектору (x_1 = 2, x_2 = = 0), который не является даже планом.

Округление в «меньшую сторону» приводит к плану $(x_1 = 1, x_2 = 0)$. Этот план допустимый, но очень далек от оптимального.

Таким образом, мы видим, что «метод округления» нельзя считать универсальным.

Еще более неприятная особенность, свойственная задачам целочисленного программирования, состоит в том, что нет простого способа, позволяющего определить, является ли данный допустимый план оптимальным. В этом одно из главных отличий задач ЦЛП от задач ЛП. Чтобы проиллюстрировать это, предположим, что в задаче (13) – (15) надо проверить, является ли план $(x_1 = 1,\ x_2 = 1)$ оптимальным. По аналогии с ЛП приходит мысль, что для этого надо проверить, соответствует ли точка $(x_1 = 1,\ x_2 = 1)$ какому-либо локальному

оптимуму, в том смысле, что значение целевой функции не улучшится при переходе в любую соседнюю допустимую целочисленную точку:

$$x_1 = 1 + d, x_2 = 1 + l, \text{ где } d,\ l = -1 \vee 0 \vee 1.$$

В нашем примере допустимые соседние точки имеют вид

$$(x_1 = 0, x_2 = 0), (x_1 = 0, x_2 = 1), (x_1 = 0, x_2 = 2), (x_1 = 1, x_2 = 0). \qquad (16)$$

Легко проверить, что значение критерия качества в точке $(x_1 = 1, x_2 = 1)$ лучше, чем в точках (16). Однако вектор $(x_1 = 1, x_2 = 1)$ не является оптимальным планом!

Отметим, что для задач с булевыми переменными $x_j = 0 \vee 1$ проверка всех соседних дискретных точек сводится к полному перебору всех возможных планов. Возникает вопрос: существуют ли методы решения задач ЦЛП, отличные от полного перебора, либо полный перебор – это единственный способ решения?

Если задача имеет небольшую размерность, то метод полного перебора можно использовать. Однако, если размеры большие, то полный перебор по времени выливается приблизительно «в целую жизнь». Например, при рассмотрении задачи ЦЛП, в которой 100 переменных и $x_j = 0 \vee 1$, для полного перебора надо перебрать 2^{100} вариантов. Следовательно, нужны другие методы, отличные от полного перебора.

К настоящему времени наиболее популярными методами целочисленного программирования являются методы двух типов:

1) возврата;

2) отсекающих плоскостей.

Ниже рассматриваются метод ветвей и границ, относящийся к первой группе, и метод Гомори, относящийся ко второй группе методов.

§2. Метод ветвей и границ

1. Общая схема метода. Этот метод применим как к полностью целочисленным задачам, так и частично целочисленным задачам.

Рассмотрим задачу

$$c'x \to \max, \qquad (17)$$

$$Ax \le b, \qquad (18)$$

$$d_{*j} \le x_j \le d_j^*, \ j \in J, \qquad (19)$$

$$x_j - \text{целое}, \ j \in I, \qquad (20)$$

$$x = (x_j, j \in J), J = \{1, 2, ..., n\}, I \subset J.$$

Заметим, что без ограничения общности для $j \in I$ числа d_{*j}, d_j^* можно считать целыми.

Идея метода ветвей и границ основана на следующем элементарном факторе. Рассмотрим любую переменную x_j, $j \in I$. Пусть l – некоторое целое число, такое что

$$d_{*j} \le l \le d_j^* - 1.$$

Тогда оптимальное значение x_j^0 будет удовлетворять либо неравенству

$$l+1 \le x_j \le d_j^*,$$

либо неравенству

$$d_{*j} \le x_j \le l.$$

Для иллюстрации возможности использования этого факта предположим, что в задаче (17) – (19) условие целочисленности не учитывается. Пусть в результате решения непрерывной задачи мы получим оптимальный план, у которого $x_1 = 1\frac{1}{3}$. Далее рассмотрим две задачи:

задачу

$$c'x \to \max, \tag{21}$$

$$Ax \le b,\ d_{*j} \le x_j \le d_j^*,\ j \in J \setminus 1, \tag{22}$$

$$d_{*1} \le x_1 \le 1, \tag{23}$$

и задачу (21), (22) с условием

$$2 \le x_1 \le d_1^*. \tag{24}$$

Предположим теперь, что каждая из полученных задач имеет оптимальные целочисленные решения соответственно x^{01} и x^{02}. Тогда очевидно, что решением исходной задачи (17) – (20) является тот из векторов x^{01} или x^{02}, на котором значение $c'x$ больше.

2. Алгоритм. Рассмотрим общую итерацию метода. Пусть мы осуществляем *t*-ю итерацию.

В начале *t*-й итерации имеем:

1) число r_0^t, которое является оценкой снизу значения целевой функции исходной задачи на оптимальном плане;

2) список задач ЛП, которые подлежат решению. Эти задачи отличаются от (17) – (19) и друг от друга только условиями (19);

3) *n*-вектор μ, число μ_0.

Замечание. На первой итерации, т.е. при t=1, список задач ЛП состоит только из одной задачи (17) – (19). Для определения r_0^1, μ, μ_0 можно поступить следующим образом. Если известен какой-либо *целочисленный* план $\bar{x}$ задачи (17) – (20), то полагаем $r_0^1 = c'\bar{x}$, $\mu = \bar{x}$, $\mu_0 = 1$. Если такого плана нет, то пола-

гаем $\mu_0 = 0$, в качестве μ берем любой n-вектор, а для построения оценки r_0^1, удовлетворяющей неравенству $r_0^1 \le c'x^0$, где x^0 – решение задачи (17) – (20), можно привлечь любую дополнительную информацию. Например, если $c \ge 0$ и $d_* \ge 0$, то очевидно, что

$$c'x^0 \ge 0,$$

следовательно, можно положить $r_0^1 = 0$. В самом худшем случае, когда нет никакой дополнительной информации, мы полагаем $r_0^1 = -\infty$.

На произвольной итерации t выполняем следующие шаги.

Шаг 1. Если основной список пуст, идем на шаг 5. В противном случае выбираем любую задачу ЛП из списка и идем на шаг 2.

Шаг 2. Решаем выбранную задачу ЛП. Если эта задача не имеет решения либо она имеет решение x^*, для которого

$$c'x^* \le r_0^t,$$

то полагаем $r_0^{t+1} = r_0^t$, вычеркиваем данную задачу ЛП из списка и возвращаемся к началу новой (t + 1)-й итерации (идем на шаг 1, заменив t на t + 1).

Если выбранная задача ЛП имеет решение x^* и

$$c'x^* > r_0^t,$$

то идем на шаг 3.

Шаг 3. Если на решении x^* задачи ЛП выполняется условие целочисленности (20), то фиксируем это решение, т.е. полагаем $\mu = x^*$, $\mu_0 = 1$. Изменяем оценку

$$r_0^{t+1} = c'x^*,$$

вычеркиваем рассмотренную задачу ЛП из списка и переходим к новой (t + 1)-й итерации (идем на шаг 1, заменив t на t + 1).

Если на решении x^* задачи ЛП условие целочисленности (20) не выполняется, то идем на шаг 4.

Шаг 4. Выберем любую переменную $x_{j_0}^*$, $j_0 \in I$, которая не удовлетворяет условию целочисленности. Пусть $x_{j_0}^* = l_{j_0}$, l_{j_0} – нецелое число. Обозначим через $[l_{j_0}]$ целую часть числа l_{j_0}, т.е. $[l_{j_0}]$ – это наибольшее целое, удовлетворяющее неравенству

$$[l_{j_0}] \le l_{j_0}.$$

Например, [3] = 3, [3, 5] = 3, [–3, 5] = –4.

Удалим старую рассматриваемую задачу ЛП из списка, а вместо нее добавим две новые задачи ЛП. Эти задачи отличаются от задачи ЛП, выбранной на шаге 1, и друг от друга только прямыми ограничениями на переменную x_{j_0}.

В первой новой задаче ЛП эти ограничения имеют вид

$$\overline{d}_{*j_0} \le x_{j_0} \le [l_{j_0}],$$

во второй задаче эти ограничения имеют вид

$$[l_{j_0}]+1 \le x_{j_0} \le \overline{d}^{*}_{j_0}.$$

Здесь $\overline{d}_{*j_0}, \overline{d}^{*}_{j_0}$ – нижняя и верхняя границы на переменную x_{j_0}, которые были в задаче ЛП, выбранной на шаге 1.

Полагаем $r_0^{t+1} = r_0^t$, переходим к новой (t + 1)-й итерации (идем на шаг 1, заменив t на t + 1).

Шаг 5. Останавливаем алгоритм. При $\mu_0 = 1$ вектор μ принимаем за решение задачи (17) – (20). При $\mu_0 = 0$ в задаче (17) – (20) нет допустимых планов.

3. Пример. Рассмотрим следующую задачу ЦЛП:

$$3x_1 + 3x_2 + 13x_3 \to \max, \tag{25}$$

$$-3x_1 + 6x_2 + 7x_3 \le 8, \tag{26}$$

$$6x_1 - 3x_2 + 7x_3 \le 8, \tag{27}$$

$$0 \le x_j \le 5, j = \overline{1,3}, \tag{28}$$

$$x_j \text{ – целое}, j = \overline{1,3}. \tag{29}$$

Перед первой итерацией список задач состоит из одной задачи № 1, которая совпадает с задачей (25) – (28).

Так как $x_j \ge 0, \quad j = \overline{1,3}$, то очевидно, что

$$3x_1 + 3x_2 + 13x_3 \ge 0$$

при любом допустимом плане. Значит, в качестве оценки снизу целевой функции исходной задачи можем взять число $r_0^1 = 0.$

Опишем алгоритм решения задачи (25) – (28) по итерациям.

Итерация 1. Список состоит из одной задачи, у которой оценка снизу равна $r_0^1 = 0.$ Решим задачу № 1 из списка задач. Эта задача имеет решение:

$$x_1^* = x_2^* = 2\frac{2}{3}, \quad x_3^* = 0, \quad c'x^* = 16. \qquad \text{(задача № 1).}$$

Решение задачи № 1 нецелочисленное, $c'x^* = 16 > 0 = r_0^1$. Переходим к шагу 4.

Выберем переменную x_1 для ветвления. Задачу № 1 вычеркнем из списка, вместо неё включаем две новые задачи № 2, № 3.

Задача № 2: условия (25) – (27) + прямые ограничения

$$3 \le x_1 \le 5, \quad 0 \le x_2 \le 5, \quad 0 \le x_3 \le 5.$$

Задача № 3: условия (25) – (27) + прямые ограничения

$$0 \le x_1 \le 2, \quad 0 \le x_2 \le 5, \quad 0 \le x_3 \le 5.$$

Полагаем $r_0^2 = r_0^1 = 0$ и переходим к шагу 1, начиная новую итерацию.

Итерация 2. Список состоит из задач № 2, 3.

Из списка задач рассмотрим задачу № 2. Легко увидеть, что ограничения этой задачи несовместны. Значит, мы вычёркиваем эту задачу из списка, полагаем $r_0^3 = r_0^2 = 0$ и переходим к новой итерации.

Итерация 3. Теперь список состоит только из задачи № 3. Рассмотрим эту задачу. Она имеет решение

$$x_1^* = x_2^* = 2, \quad x_3^* = \frac{2}{7}, \quad c'x^* = 15\frac{5}{7}. \qquad \text{(задача № 3).}$$

Переменная x_3^* – нецелая. Выбираем её для ветвления и идём на шаг 4.

Удаляем задачу № 3 из списка, вместо неё включаем новые две задачи № 4, № 5, порождённые задачей № 3.

Задача № 4: условия (25) – (27) + прямые ограничения

$$0 \le x_1 \le 2, \quad 0 \le x_2 \le 5, \quad 1 \le x_3 \le 5.$$

Задача № 5: условия (25) – (27) + прямые ограничения

$$0 \le x_1 \le 2, \quad 0 \le x_2 \le 5, \quad x_3 = 0.$$

Полагаем $r_0^4 = r_0^3 = 0$ и переходим к следующей итерации.

Итерация 4. Список состоит из задач № 4, № 5. Рассмотрим задачу № 4. Она имеет решение

$$x_1^* = x_2^* = \frac{1}{3}, \; x_3^* = 1, \quad c'x^* = 15. \qquad \text{(задача № 4).}$$

Переходим к шагу 4, выбрав переменную x_2 для ветвления. Задачу № 4 вычёркиваем из списка, вместо неё включаем в список две новые задачи № 6, № 7, порождённые задачей № 4.

Задача № 6: условия (25) – (7) + прямые ограничения

$$0 \le x_1 \le 2, \quad 1 \le x_2 \le 5, \quad 1 \le x_3 \le 5.$$

Задача № 7: условия (25) – (27) + прямые ограничения

$$0 \le x_1 \le 2, \quad x_2 = 0, \quad 1 \le x_3 \le 5.$$

Полагаем $r_0^5 = r_0^4 = 0$ и переходим к новой итерации.

Итерация 5. На данной итерации из списка задач № 5, № 6, № 7 выберем задачу № 6. Анализируя условия задачи № 6, легко заметить, что она не имеет допустимых планов, поэтому вычёркиваем её из списка. Полагаем $r_0^6 = r_0^5 = 0$ и переходим к новой итерации, возвращаясь к шагу 1.

Итерация 6. Теперь список состоит из задач № 5, № 7. Выберем задачу № 7. Она имеет решение

$$x_1^* = x_2^* = 0, \quad x_3^* = 1\frac{1}{7}, \quad c'x^* = 14\frac{6}{7}. \quad \text{(задача № 7).}$$

Переходим к шагу 4, используя x_3 для ветвления. Задачу № 7 исключаем из списка, вместо неё включаем две новые задачи № 8, № 9, порождённые задачей № 7.

Задача № 8: условия (25) – (27) + прямые ограничения

$$0 \le x_1 \le 2, \quad x_2 = 0, \quad 2 \le x_3 \le 5.$$

Задача № 9: условия (25) – (27) + прямые ограничения

$$0 \le x_1 \le 2, \; x_2 = 0, \; x_3 = 1.$$

Полагаем $r_0^7 = r_0^6 = 0$ и переходим к следующей итерации.

Итерация 7. Список состоит из задач № 5, 8, 9. Выберем задачу № 8. Она не имеет решения, так как её ограничения несовместны. Вычёркиваем её из списка. Полагаем $r_0^8 = r_0^7 = 0$ и переходим к новой итерации.

Итерация 8. Список состоит из задач № 5, 9. Выберем задачу № 9. Эта задача имеет решение

$$x_1^* = x_2^* = 0, \; x_3^* = 1, \; c'x^* = 13. \qquad \text{(задача № 9).}$$

Решение задачи № 9 – целое, поэтому полагаем $\mu_0 = 1$, $\mu = x^*$. Задачу № 9 вычёркиваем из списка, полагаем $r_0^9 = r_0^8 = 13$ и переходим к новой итерации.

Итерация 9. Список состоит только из одной задачи № 5. Задача № 5 имеет решение

$$x_1^* = 2, \; x_2^* = 2\frac{1}{3}, \; x_3^* = 0, \; c'x^* = 13. \qquad \text{(задача № 5).}$$

Поскольку $c'x^* = 13 \le r_0^0 = 13$, то мы не «дробим» задачу № 5, а просто вычёркиваем её из списка. Оценку r_0^9 не меняем: $r_0^{10} = r_0^9 = 13$. Переходим к шагу 1 новой итерации.

Итерация 10. На новой итерации список пуст. Алгоритм заканчивает работу. Так как у нас $\mu_0 = 1$, то вектор μ принимаем за решение задачи:

$$\mu = x^0 = (0, 0, 1).$$

Задача решена. Ход решения задачи схематично можно представить в виде дерева (рис. 1.2).

Замечания:

1. В описанном выше алгоритме возможность произвольного выбора возникает в двух местах:

а) при выборе задачи из списка;

б) при выборе нецелой переменной, по которой производится ветвление.

Число итераций метода зависит от того, какой именно выбор мы осуществим в пунктах «а» и «б».

К настоящему времени разработаны специальные процедуры, помогающие оценить «качество» выбора. Это повышает эффективность алгоритма.
2. На любой итерации (кроме первой) мы имеем список задач, которые надо решить и которые отличаются от породившей их задачи (решение которой уже найдено) **только одним** прямым ограничением. Для эффективного решения задач из списка можно использовать двойственный симплекс-метод [2, 9], выбирая в качестве начального двойственного базисного плана оптимальный двойственный план порождающей задачи.

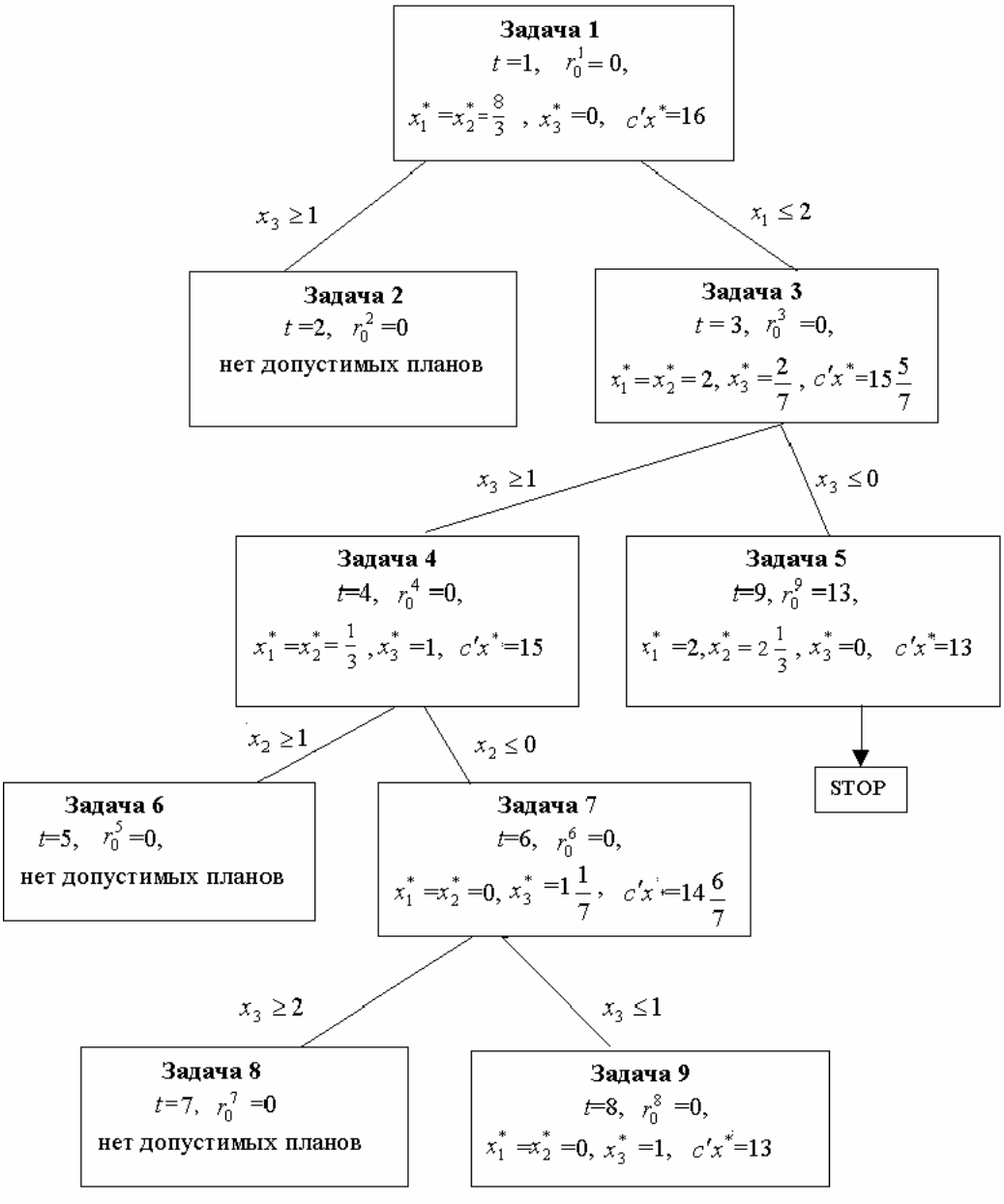


Рис. 1.2

4. Двойственный симплекс-метод для задачи линейного программирования с двусторонними ограничениями. Рассмотрим задачу линейного программирования следующего вида:

$$c'x \to max,$$

$$Ax = b, \tag{30}$$

$$d_* \le x \le d^*,$$

где $A \in R^{m\times n}$; $c, d_*, d^* \in R^n$; $b \in R^m$; $c = (c_j,\ j = \overline{1,n})$, $d_* = (d_{*j},\ j = \overline{1,n})$, $d^* = (d_j^*, j = \overline{1,n})$, $A = (A_j, j = \overline{1,n})$, $A_j \in R^m$ – заданные параметры задачи. Без ограничения общности будем считать, что $rank\ A = m$.

Подмножество индексов $J_Б = \{j_1, j_2, ..., j_m\} \subset J = \{1, 2, ..., n\}$ назовем базисом задачи (30), если $det\ A_Б \ne 0$, где $A_Б = (A_j, j \in J_Б)$ – базисная матрица.

Решение задачи (30) двойственным методом начинается с задания некоторого базиса $J_Б = \{j_1, j_2, ..., j_m\}$ и соответствующей ему базисной матрицы $A_Б = (A_j, j \in J_Б)$. Матрицу, обратную к базисной, обозначим через B: $B = A_Б^{-1}$.

Итерация метода состоит из следующих шагов.

1. Найдем m-вектор

$$y' := c'_Б B$$

и оценки

$$\Delta_j := y'A_j - c_j, \quad j \in J;$$

сформируем множества

$$J_H = J \setminus J_Б; \quad J_H^+ = \{j \in J_H : \Delta_j \ge 0\}; \quad J_H^- = J_H \setminus J_H^+.$$

2. Построим вектор $\aleph = (\aleph_j, j \in J)$ по следующему правилу:

$$\aleph_j = d_{*j}, \quad j \in J_H^+; \quad \aleph_j = d_j^*, \quad j \in J_H^-;$$

$$\aleph_Б = (\aleph_j, j \in J_Б) = B\Big(b - \sum_{j \in J_H^+ \cup J_H^-} A_j \aleph_j\Big).$$

3. Проверим критерий оптимальности: если выполняются соотношения

$$d_{*j} \le \aleph_j \le d_j^*,\ j \in J_Б, \tag{31}$$

то вектор $x^0 := \aleph = (\aleph_j, j \in J)$ – оптимальный план задачи (30). Решение задачи (30) прекращается.

В противном случае (т.е. если соотношения (31) не выполняются) идем на шаг 4.

4. Найдем такой индекс $j_k \in J_Б$, что $\aleph_{j_k} \notin [d_{*j_k}, d_{j_k}^*]$.

5. Положим

$$\mu_{j_k} = 1, \text{ если } \aleph_{j_k} < d_{*j_k},$$

$$\mu_{j_k} = -1, \text{ если } \aleph_{j_k} > d^*_{j_k}.$$

Подсчитаем m-вектор

$$\Delta y' = \mu_{j_k} e'_k B$$

и числа

$$\mu_j = \Delta y' A_j,\ j \in J_H^+ \cup J_H^-.$$

6. Найдем шаги $\sigma_j,\ j \in J_H = J_H^+ \cup J_H^-$ по правилу:

$$\sigma_j = \begin{cases} -\Delta_j / \mu_j, \text{ если } j \in J_H^+ \text{ и } \mu_j < 0 \text{ либо } j \in J_H^- \text{ и } \mu_j > 0; \\ \\ \infty \text{ в противном случае.} \end{cases}$$

Положим $\sigma_0 = \min\limits_{j \in J_H} \sigma_j = \sigma_{j*}$, здесь $j* \in J_H$ – индекс, на котором достигается минимум в последнем выражении. Если минимум достигается на нескольких индексах, то в качестве индекса $j*$ можно взять любой из них.

7. Если $\sigma_0 = \infty$, то прекращаем решение задачи (10), так как она не имеет допустимых планов. Если $\sigma_0 < \infty$, идем на шаг 8.

8. Построим новый коплан $\overline{\Delta} = (\overline{\Delta}_j, j \in J)$ по правилу:

$$\overline{\Delta}_j = \Delta_j + \sigma_0 \mu_j, \quad j \in J_H \cup j_k;$$

$$\overline{\Delta}_j = 0,\ j \in J_Б \setminus j_k.$$

9. Построим новый базис $\overline{J}_Б = (J_Б \setminus j_k) \cup j_*$, соответствующую ему базисную матрицу $\overline{A}_Б = (A_j, j \in \overline{J}_Б)$ и обратную матрицу $\overline{B} = (\overline{A}_Б)^{-1}$ по правилу $\overline{B} = M B$, где $m \times m-$матрица M получается из единичной $m \times m-$ матрицы заменой k-го столбца e_k на столбец d:

$$e_k = \begin{pmatrix} 0 \\ \dots \\ 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{pmatrix}, \quad d = -\begin{pmatrix} z_1 \\ \dots \\ z_{k-1} \\ -1 \\ z_{k+1} \\ \dots \\ z_m \end{pmatrix} \cdot \frac{1}{z_k}, \quad z = \begin{pmatrix} z_1 \\ \dots \\ z_{k-1} \\ z_k \\ z_{k+1} \\ \dots \\ z_m \end{pmatrix} = BA_{j*}.$$

10. Построим новые множества $\bar{J}_H$, $\bar{J}_H^-$ и $\bar{J}_H^+$:

$$\bar{J}_H = J \setminus \bar{J}_Б ;$$
$$\bar{J}_H^+ = (J_H^+ \setminus j_*) \cup j_k , \text{ если } \mu_{j_k} = 1,\ j_* \in J_H^+ ;$$
$$\bar{J}_H^+ = (J_H^+ \setminus j_*), \text{ если } \mu_{j_k} = -1,\ j_* \in J_H^+ ;$$
$$\bar{J}_H^+ = (J_H^+ \cup j_k), \text{ если } \mu_{j_k} = 1,\ j_* \notin J_H^+ ;$$
$$\bar{J}_H^+ = J_H^+ , \text{ если } \mu_{j_k} = -1,\ j_* \notin J_H^+ ;$$
$$\bar{J}_H^- = \bar{J}_H \setminus \bar{J}_H^+ .$$

Идем на шаг 2, используя новые базис $\bar{J}_Б$, коплан $\bar{\Delta}$, базисную матрицу $\bar{A}_Б$ и обратную к ней матрицу $\bar{B}$.

Совокупность шагов 2 – 10 назовем итерацией $J_Б \to \bar{J}_Б$. Данная итерация называется *невырожденной*, если $\sigma_0 > 0$.

Можно показать, что описанный алгоритм решает задачу за конечное число итераций, если в процессе его реализации встречается конечное число вырожденных итераций.

§3. Метод Гомори (метод отсечений) для полностью целочисленных задач

1. Интуитивное обоснование метода. Отметим ещё одну особенность задач ЦЛП, которая отличает их от соответствующих задач ЛП. Рассмотрим задачу:

$$c'x \to \max, \tag{32}$$
$$Ax = b,\ x \geq 0,$$
$$x_j - \text{целое},\ j \in J, \tag{33}$$

где $A \in R^{m \times n}$, rank $A = m < n$, $J = \{1, 2, ..., n\}$.

Если мы рассмотрим задачу ЛП (32), то увидим, что среди оптимальных планов задачи (32) всегда существует базисный план, т. е. план, у которого не более чем m компонент, отличных от нуля.

Рассмотрим теперь задачу ЦЛП (32), (33) с m основными ограничениями. В общем случае оптимальный план задачи (32), (33) будет содержать более чем m компонент, отличных от нуля. Это сразу же наводит на мысль о том, что если мы хотим получить решение целочисленной задачи (32), (33) как решение некоторой «непрерывной» задачи ЛП, то эта «непрерывная» задача должна со-

держать более чем *m* основных ограничений, т.е. мы должны к задаче (32) добавить еще некоторые основные ограничения.
Необходимость введения новых ограничений в задачу (32) для получения решения задачи (32), (33) можно объяснить и с другой позиции. Рассмотрим множество допустимых планов X задачи (32) (рис.1.3). Понятно, что множеством допустимых планов $X_ц$ задачи (32), (33) будут все «целые» точки, принадлежащие X. Рассмотрим ещё одно множество X_B, называемое *выпуклой оболочкой* точек $X_ц$. Согласно определению, X_B – это **наименьшее выпуклое** множество, содержащее все точки $X_ц$ (см. рис. 1.3). Справедливы включения

$$X_ц \subset X_B \subset X. \tag{34}$$

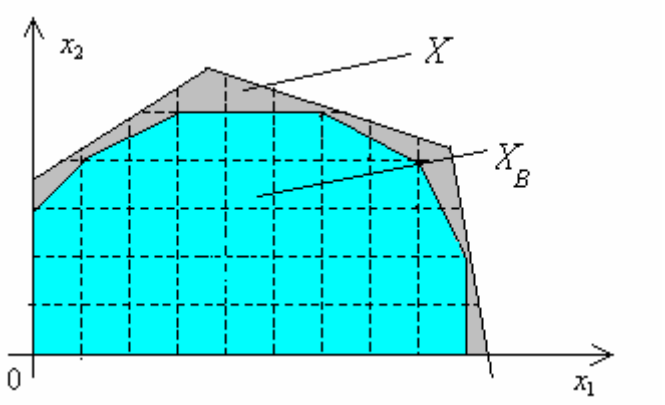


Рис. 1.3

Множество X_B, так же как и множество X, является «непрерывным». Легко видеть, что все угловые точки множества X_B – целые точки. Отсюда, учитывая свойства симплекс-метода и включения (34), заключаем, что среди решений задачи ЛП

$$c'x \to max, x \in X_B \tag{35}$$

обязательно есть целочисленное решение и это решение будет решением задачи (32), (33). Мы видим, что множество X_B получилось из X путём «отсечения» лишних кусков, не содержащих целых точек. При этом важно, что мы не отсекли ни одной целой точки! На рисунке для R^2 построить множество X_B просто. При большом *n* сделать это заранее перед решением задачи невозможно. Кроме того, нам не надо всё «чистое» множество X_B. Нам важно «высечь» только часть этого множества в окрестности оптимальной точки.

На этих идеях отсечений и основан метод Гомори. Суть метода состоит в том, что на каждом шаге к ограничениям соответствующей задачи ЛП добавляется новое ограничение, называемое отсекающей плоскостью. Эта плоскость должна обладать следующими свойствами:

1) отсекать имеющееся в наличии нецелочисленное решение задачи ЛП;

2) не отсекать ни одного целочисленного плана исходной задачи ЦЛП;

3) отсечения должны строиться таким образом, чтобы обеспечить конечность алгоритма, т.е. решение задачи ЦЛП должно строиться за конечное число шагов.

2. Построение отсекающей плоскости. Рассмотрим задачу ЦЛП вида (32), (33). Предположим, что:

1) ограничения задачи (32), (33) совместны;
2) целевая функция ограничена сверху.

Прежде чем описать весь алгоритм, покажем в общем случае, как построить дополнительное ограничение, которому удовлетворяют все **целые** планы задачи (32), (33).

Рассмотрим любое линейное уравнение, которое можно получить из уравнений $Ax = b$ путём алгебраических операций над этими уравнениями. В общем случае такое уравнение можно представить в виде

$$y' Ax = y' b, \tag{36}$$

где $y \in R^m$, $y \neq 0$, – произвольный вектор.

Очевидно, из условия $Ax = b$ следует справедливость (36).
Обозначим

$$a_j = y' A_j,\ j \in J, \beta = y' b.$$

Тогда равенство (36) примет вид

$$\sum_{j=1}^{n} a_j x_j = \beta . \tag{37}$$

Предположим, что хотя бы одно из чисел $a_j, j \in J$, β является дробным. Как и раньше, обозначим через $[d]$ целую часть числа d ($[d]$ – *наибольшее* целое, не превосходящее d).

По построению любой вектор x, удовлетворяющий $Ax = b$, должен удовлетворять и (37). Так как все $x_j \geq 0, j \in J$, и $[a_j] \leq a_j,\ j \in J$, то из (37) следует более слабое условие

$$\sum_{j=1}^{n} [a_j] x_j \leq \beta . \tag{38}$$

Теперь учтём, что по условию все $x_j, j \in J$ – целые. Отсюда следует, что все целые $x_j, j \in J$, удовлетворяющие (38), должны удовлетворять более сильному, чем (38), неравенству

$$\sum_{j=1}^{n} [a_j] x_j \leq [\beta]. \tag{39}$$

Заметим, что в общем случае, без учёта целочисленности, из (38) не следует (39)!

Условие (39) можно переписать в эквивалентной форме

$$\sum_{j=1}^{n}[a_j]x_j + x_* = [\beta],\ \ x_* \geq 0,\ \ x_* - \text{целое}. \qquad (40)$$

Таким образом, условие (40) есть новое условие, которому удовлетворяют все **целые** планы исходной задачи (32) – (33).

В алгоритме используется не условие (40), а разность между (40) и (37).

Определим значения f_j и f тождественными

$$[a_j] + f_j \equiv a_j,\ \ j \in J,\ \ [\beta] + f = \beta,$$

т.е. f_j – дробная часть числа a_j, $j \in J$, а f – дробная часть числа β. По построению,

$$0 \leq f_j < 1,\ \ j \in J,\ \ 0 \leq f < 1.$$

Вычтем из (40) равенство (37), в результате получим

$$\sum_{j=1}^{n}[-f_j]x_j + x_* = -f,\ \ x_* \geq 0 - \text{целое}. \qquad (41)$$

В алгоритме вместо (40) используется отсекающее ограничение (41).

3. Алгоритм Гомори (общая схема). Общую схему алгоритма Гомори можно представить в виде последовательности следующих шагов.

Шаг 1. Найти оптимальное решение задачи линейного программирования.

Шаг 2. Если в оптимальном базисе задачи ЛП есть искусственные индексы и размеры задачи ЛП велики, то уменьшаем размеры текущей задачи ЛП: удаляем из задачи ЛП искусственные переменные с базисными индексами и соответствующие им ограничения. В противном случае сразу переходим к шагу 3 без уменьшения размеров задачи ЛП.

Шаг 3. Прекращаем решение задачи ЦЛП, если все неискусственные переменные задачи ЛП целые. В противном случае переходим к шагу 4.

Шаг 4. Сформируем отсекающую плоскость (ограничение). Для этого выберем любую дробную неискусственную переменную (это обязательно базисная переменная!) $x^0_{i_0}$, $i_0 \in J^0_Б$. Здесь $J^0_Б$ – оптимальный базис текущей задачи ЛП. Положим $y' = e'_{i_0}(A^0_Б)^{-1}$. Сформируем отсекающее ограничение (41), исходя из ограничения (37), построенного по правилам предыдущего пункта.

Шаг 5. Добавляем отсекающее ограничение (41) и новую (целую!) переменную x_* к задаче ЛП и получаем расширенную (новую) задачу ЛП. Переходим к шагу 1.

Опишем перечисленные шаги алгоритма подробнее.

Рассмотрим шаг 1. Пусть в текущей задаче ЛП есть переменные

$$x_j \geq 0,\ j \in J \cup J_{иск},$$

и есть $\overline{m} \geq m$ основных ограничений. По построению $\overline{m} = m + |J_{иск}|$ и каждой искусственной переменной x_{j*} поставлено в соответствие некоторое основное ограничение текущей задачи ЛП. Отметим, что это ограничение не входит в число основных ограничений исходной задачи ЛП или ЦЛП, т.е. является дополнительным отсекающим ограничением. Решим текущую задачу ЛП и обозначим через

$$x_j^0,\ j \in J \cup J_{иск},\ \ J_Б^0 \subset J \cup J_{иск},\ \ |J_Б^0| = \overline{m}$$

ее оптимальный базисный план.

Опустим пока подробное описание шага 2, поскольку на первой итерации этот шаг «пропускается» и поэтому его удобнее описывать после описания всех других шагов.

Шаг 3 описан подробно ранее.

Рассмотрим шаг 4. Предположим, что выбран некоторый индекс i_0, такой, что

$$i_0 \in J_Б^0 \bigcap J,\ \ x_{i_0}^0 > 0 - \text{нецелое}.$$

Положим

$$y' = e'_{i_0} (A_Б^0)^{-1},\ \ a_j = y' A_j,\ j \in J \cup J_{иск},\ \ \beta = y' b,$$

где A_j – j-й столбец матрицы условий и b – вектор правых частей основных ограничений текущей задачи ЛП. Рассмотрим ограничение

$$\sum_{j \in J \cup J_{иск}} a_j x_j = \beta. \tag{42}$$

Так как по построению, в силу специального выбора вектора y, имеют место равенства

$$a_{i_0} = 1,\ \ a_j = 0,\ j \in J_Б \setminus i_0,$$

то равенство (42) можно представить в виде

$$x_{i_0} + \sum_{j \in J_Н} a_j x_j = \beta, \tag{43}$$

где $J_Н \subset (J \cup J_{иск}) \setminus J_Б{}^0$. Так как все $x_j{}^0 = 0,\ j \in J_Н$, и вектор $x^0 = (x_j^0,\ j \in \in J \cup J_{иск})$ удовлетворяет (43), то имеем $\beta = x_{i_0}^0 > 0$ – нецелое. Отсекающее ограничение (41), построенное по (43), имеет вид

$$\sum_{j \in J_Н} [-f_j] x_j + x_{j*} = -f,\ \ x_{j*} \geq 0 - \text{целое}, \tag{44}$$

где x_{j*} – новая переменная, соответствующая отсекающему ограничению (44), j_* – наименьший целый индекс, не принадлежащий множеству $J \cup J_{иск}$.

Рассмотрим шаг 5. Ограничение (44) добавляем к основным ограничениям имеющейся задачи ЛП, индекс j_* и соответствующую ему переменную x_{j*}

добавляем к переменным текущей задачи ЛП. При этом произойдут следующие замены:

$$\overline{m} \to \overline{m}+1;\ J \cup J_{иск} \to J \cup \bar{J}_{иск},\ \ \bar{J}_{иск} = J_{иск} \cup j_*.$$

Покажем, что добавляемое новое ограничение (44) является «отсекающим», т.е. оно отсекает нецелочисленное оптимальное решение «старой» задачи ЛП. Действительно, на оптимальном плане x^0 «старой» задачи ЛП имеют место соотношения: $x_j{}^0 = 0,\ j \in J_{\text{Н}}$, и по построению, $0 < f < 1$. Следовательно, для выполнения ограничения (44) надо положить $x^0_{j*} = -f < 0$. Однако это противоречит условию $x^0_{j*} \geq 0$. Таким образом, мы видим, что оптимальный план «старой» задачи ЛП не удовлетворяет ограничению (44) и, следовательно, не является планом новой задачи ЛП. В этой ситуации новую задачу ЛП разумно решать двойственным симплекс-методом [2, 8], начиная процесс вычисления с оптимального двойственного базисного плана старой задачи ЛП.

Рассмотрим шаг 2. Для того чтобы размеры задачи ЛП не росли неограниченно, поступаем следующим образом. Если $J^0_{Б} \cap J_{иск} \neq 0$, то перед тем, как построить новое отсекающее ограничение, выбросим отсекающее ограничение, породившее искусственную переменную x_{j*}, $j_* \in J^0_{Б} \cap J_{иск}$. Исключим из задачи ЛП искусственную переменную x_{j*}, подставив вместо неё в другие отсекающие ограничения, содержащие переменную x_{j*}, её выражение через другие переменные, найденное из «выбрасываемого» отсекающего ограничения.

Дополнительное ограничение, соответствующее x_{j*}, можно выбросить, так как вхождение искусственной переменной x_{j*} в оптимальный базис означает, что это ограничение пассивно. Используя эту процедуру выбрасывания лишних ограничений, всегда можно добиться того, что в решаемой задаче ЛП будет не более чем *n* основных ограничений.

Теоретически доказано, что описанный алгоритм сходится к оптимуму за конечное число итераций.

Пример. Рассмотрим задачу

$$21\,x_1 + 11\,x_2 \to \max, \tag{45}$$

$$7\,x_1 + 4\,x_2 + x_3 = 13, \quad x_j \geq 0, \quad j = \overline{1,3}, \tag{46}$$

$$x_j \geq 0 - \text{целое}, \quad j = \overline{1,3}. \tag{47}$$

Решим данную задачу методом Гомори. Алгоритм опишем по итерациям.

Итерация 1. Решим задачу ЛП (45), (46) симплекс-методом [2, 9]. В результате получим оптимальный базисный план

$$x^0_1 = \frac{13}{7},\ \ x^0_2 = x^0_3 = 0,\ \ J^0_{Б} = \{1\}.$$

Базисная матрица $A_{\text{Б}} = (A_j, j \in J^0_{\text{Б}})$ и обратная к ней матрица $A^{-1}_{\text{Б}}$ имеют вид

$$A_{\text{Б}} = 7, \quad A_{\text{Б}}^{-1} = \frac{1}{7}.$$

Выберем $i_0 = 1$, подсчитаем вектор $y' = e'_{i_0} (A_{\text{Б}})^{-1} = \frac{1}{7}$ и получим ограничение вида (37)

$$x_1 + \frac{4}{7} x_2 + \frac{1}{7} x_3 = 1\frac{6}{7}.$$

По последнему ограничению построим отсекающее ограничение вида (41)

$$-\frac{4}{7} x_2 - \frac{1}{7} x_3 + x_4 = -\frac{6}{7}, \quad x_4 \geq 0. \qquad (48)$$

Добавим ограничение (48) к задаче (45), (46) и перейдем к следующей итерации.

Итерация 2. Решим задачу ЛП (45), (46), (48). Она имеет решение:

$$x_1^0 = 1, \quad x_2^0 = 1\frac{1}{2}, \quad x_3^0 = 0, \quad x_4{}^0 = 0, \quad J_{\text{Б}}^0 = \{1, 2\}.$$

Базисная матрица $A_{\text{Б}} = (A_j, j \in J_{\text{Б}}^0)$ и обратная к ней матрица $A_{\text{Б}}^{-1}$ имеют вид

$$A_{\text{Б}} = \begin{pmatrix} 7 & 4 \\ 0 & -\frac{4}{7} \end{pmatrix}, \; A_{\text{Б}}{}^{-1} = \begin{pmatrix} \frac{1}{7} & 1 \\ 0 & -\frac{7}{4} \end{pmatrix}.$$

Выберем $i_0 = 2$, положим $y' = (0, 1)$ $A_{\text{Б}}^{-1} = (0, -\frac{7}{4})$. Используя найденный вектор y, построим ограничение вида

$$\sum_{j \in J_{\text{Н}}} a_j x_j = \beta, \; (49)$$

где $a_j = y' A_j$, $\beta = y' b$. На данной итерации матрица условий A и вектор правых частей b текущей задачи ЛП имеют вид

$$A = \begin{pmatrix} 7 & 4 & 1 & 0 \\ 0 & -\frac{4}{7} & -\frac{1}{7} & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 13 \\ -\frac{6}{7} \end{pmatrix}.$$

Следовательно, $a_1 = 0$, $a_2 = 1$, $a_3 = 1/4$, $a_4 = -7/4$, $\beta = 3/2$ и ограничение (49) принимает вид $x_2 + \frac{1}{4} x_3 - \frac{7}{4} x_4 = 3/2$.

По этому ограничению сформируем отсекающее ограничение вида (41)

$$-\frac{1}{4} x_3 - \frac{1}{4} x_4 + x_5 = -\frac{1}{2}. \qquad (50)$$

Добавим ограничение (50) к задаче (45), (46), (48) и перейдем к следующей итерации.

Итерация 3. Решим задачи ЛП (45), (46), (48), (50) симплекс-методом. В результате будет найдено ее решение

$$x_1^0 = 0, \quad x_2^0 = 3, \quad x_3^0 = 1, \quad x_4^0 = 1, \quad x_5^0 = 0, \quad J_{\text{Б}}^0 = \{2, 3, 4\}.$$

Это решение является целочисленным. Следовательно, оно является решением исходной задачи ЦЛП (45) – (47).

Отметим, что если бы последнее решение было нецелочисленным, то на следующей итерации перед началом формирования нового отсекающего ограничения мы удалили бы из последней задачи ЛП (т.е. из задачи (45), (46), (48), (50)) ограничение (48), «породившее» искусственную переменную x_4, которая вошла в оптимальный базис, а в остальные ограничения (например, в (50)) подставили бы вместо переменной x_4 ее выражение через другие переменные, найденное из (48), т. е. в (50) вместо x_4 подставили бы

$$x_4 = \frac{4}{7} x_2 + \frac{1}{7} x_3 - \frac{6}{7}.$$

В результате ограничение (50) приняло бы вид

$$-\frac{1}{7} x_2 \ -\frac{2}{7} x_3 = -\frac{5}{7} \text{ или } x_2 + 2x_3 = 5.$$

Глава 2. Динамическое программирование

Динамическим программированием называется вычислительный метод решения специальных задач математического программирования и оптимального управления, математические модели которых имеют характер многоэтапных и динамических процессов.

§ 1. Основные принципы динамического программирования

В динамических методах приближения к решению (оптимальному плану) строятся по решениям последовательности аналогичных задач меньшей размерности (состоящих из меньшего числа этапов). Процесс решения как бы развертывается во времени. В статических методах количество этапов фиксировано и итерации представляют собой переходы от одного элемента (плана) к другому в пространстве фиксированной размерности.

Для того чтобы применить методы динамического программирования, необходимо, чтобы математическая модель решаемой задачи имела «динамический характер». Не всегда в исходной постановке задачи удается разглядеть ее «динамический характер». Поэтому очень важно научиться представлять модели в таком виде, в котором выявляются их динамические свойства.

Однако не существует простых правил, механическое применение которых в любой задаче позволяет выявить ее динамические свойства. Это в некоторой степени является искусством. И здесь лучшим учителем является опыт и интуиция.

Многие конкретные оптимизационные задачи удается сформулировать несколькими внешне различными способами, причем в каждой из формулировок в центре внимания находится та или иная структурная зависимость.

Метод динамического программирования основан на трех главных этапах.

1. *Инвариантное погружение* исходной задачи в семейство аналогичных задач (исходная задача должна принадлежать этому семейству). Реализация этого этапа может быть неоднозначной и в каждом конкретном случае зависит от опыта, изобретательности и интуиции исследователя. От вида инвариантного погружения зависят все последующие построения и эффективность полученного метода в целом. На первом этапе, когда настроено инвариантное погружение, вводится *функция Беллмана – оптимальное значение целевой функции произвольной задачи* из рассматриваемого семейства.

2. Второй этап решения задачи методом динамического программирования состоит в получении *уравнения для функции Беллмана.* На этом этапе используется *принцип оптимальности Беллмана*: оптимальная стратегия обладает тем свойством, что, каков бы ни был путь достижения некоторого состояния, последующие решения должны принадлежать оптимальной стратегии для оставшейся части процесса, начинающегося с этого состояния.

3. Поиск решения уравнения Беллмана и построение по нему решения исходной задачи.

Изложим основные приемы метода динамического программирования на ряде конкретных задач линейного программирования.

§ 2. Задача распределения ресурсов

Имеется сырье в объеме *c* и *n* технологических процессов. Если количество *x* сырья используется в *i*-м технологическом процессе, то получается прибыль $f_i(x)$. Как распределить сырье между процессами, чтобы получить максимальную прибыль?

Пусть x_i – количество сырья, выделяемое на *i*-й процесс. Тогда математическая модель сформулированной задачи имеет вид

$$\sum_{i=1}^{n} f_i(x_i) \to \max,\ \sum_{i=1}^{n} x_i = c,\ x_i \ge 0,\ i = \overline{1,n}. \qquad (1)$$

Специфика задачи нелинейного программирования (1) состоит в том, что его целевая функция и функция ограничений *сепарабельны*, т.е. представимы в виде суммы функций одной переменной.

Решим задачу (1) методом динамического программирования.

Осуществим первый этап – инвариантное погружение в семейство задач. Для задачи (1) этот этап состоит в рассмотрении совокупности задач распределения ресурсов в объеме *y* между *k* технологическими процессами:

$$P(k,y): \qquad \sum_{i=1}^{k} f_i(x_i) \to \max,\ \sum_{i=1}^{k} x_i = y,\ x_i \ge 0,\ i = \overline{1,k}, \qquad (2)$$

где $0 \le y \le c$, $0 \le k \le n$ – параметры семейства. При $y = c$ и $k = n$ получим исходную задачу.

Оптимальное значение целевой функции задачи (2) назовем *функцией Беллмана* $B_k(y)$:

$$B_k(y)=\max\sum_{i=1}^{k} f_i(x_i),\ \sum_{i=1}^{k} x_i=y,\ x_i\ge 0,\ i=\overline{1,k}. \qquad (3)$$

Перейдем ко второму этапу – **составлению уравнения Беллмана на основе принципа оптимальности. Сущность этого принципа для задачи (1) выражается приводимыми ниже рассуждениями.**

Отметим, что при составлении уравнения Беллмана проверяется правильность инвариантного погружения. С другой стороны, способ погружения сказывается на виде уравнения.

В задаче (2) с k процессами и запасом сырья y выделим k-му процессу сырье в количестве z, $0\le z\le y$. При этом размер прибыли от k-го процесса будет равен $f_k(z)$.

На оставшиеся процессы с номерами 1, 2, …, $k-1$ остается сырья в количестве $y-z$. Из принципа оптимальности следует, что это сырье $y-z$ между процессами 1, 2, …, $k-1$ нужно распределять оптимальным образом, ибо в противном случае при заданном количестве сырья z для k-го процесса можно получить большую прибыль, если сырье в объеме $y-z$ разделить между процессами 1, 2, …, $k-1$ оптимальным образом.

Согласно определению (3), размер максимальной прибыли от распределения $y-z$ единиц ресурса между процессами 1, 2, …, $k-1$ равен $B_{k-1}(y-z)$.

Таким образом, если запас сырья равен y, то при выделении k-му процессу z единиц ресурса от всех k-х процессов получаем прибыль

$$f_k(z)+B_{k-1}(y-z). \qquad (4)$$

Изменяя количество z в пределах $0\le z\le y$, находим значение $x_k^0(y)$ – оптимальное количество сырья на k-й процесс, при котором общая прибыль (4) максимальна:

$$f_k(x_k^0(y))+B_{k-1}(y-x_k^0(y))=\max[f_k(z)+B_{k-1}(y-z)],\ 0\le z\le y. \qquad (5)$$

С другой стороны, согласно (3), максимальная прибыль от k-х процессов при количестве сырья y равна $B_k(y)$. Учитывая это, получаем

$$B_k(y)=\max_{0\le z\le y}[f_k(z)+B_{k-1}(y-z)],\ k=\overline{1,n},\ 0\le y\le c. \qquad (6)$$

Параллельно с функцией $B_k(y)$ можно строить функцию $x_k^0(y)$, $0\le y\le c$, где $x_k^0(y)$ – значение параметра $z\in[0,y]$, на котором достигается максимум в правой части выражения (6).

Уравнение (6) называется *уравнением Беллмана*.

Поскольку уравнение (6) рекуррентно относительно аргумента k функции $B_k(y)$, то для его решения необходимо задать начальное условие. Это условие можно получить из (3), если положить $k=1$:

$$B_1(y)=\max f_1(x_1),\ x_1=y,\ x_1\ge 0.$$

Таким образом, начальное условие для уравнения Беллмана (6) имеет вид

$$B_1(y)=f_1(y). \qquad (7)$$

Рассмотрим третий этап – **поиск решения уравнения Беллмана (6), (7) и построение по нему решения исходной задачи.**

Начальное условие у нас задано – это условие (7). В уравнении (6) положим $k = 2$:

$$B_2(y)=\max_{0\le z\le y}[f_2(z)+B_1(y-z)]=\max_{0\le z\le y}[f_2(z)+f_1(y-z)]. \qquad (8)$$

В этом выражении под знаком максимума стоят известные функции. Поэтому формула (8) позволяет вычислить $B_2(y)$ максимизацией известной функции одной переменной. Положим далее в (6) $k = 3$:

$$B_3(y)=\max_{0\le z\le y}[f_3(z)+B_2(y-z)].$$

Функция $f_3(y)$ задана, функция $B_2(y)$ определена выше, следовательно, под знаком максимума стоит известная функция и мы можем теперь определить функцию $B_3(y)$ максимизацией известной функции одной переменной z. И так далее. В результате будут построены функции $B_1(y)$, …, $B_n(y)$, $0\le y\le c$. Согласно (3), число $B_n(c)$ – максимальная прибыль для исходной задачи (1).

Чтобы найти оптимальное распределение сырья по технологическим процессам, обратимся к выражению (5) и совершим *обратный ход решения уравнения Беллмана*.

Положим в (5) $k = n$, $y = c$ и, согласно (5), найдем число $x_n^0(c)$, которое, по определению, равно оптимальному количеству сырья, выделяемому на процесс n, если объем сырья на все n процессов равен c. Таким образом, компонента x_n^0 оптимального плана $x^0=(x_1^0, x_2^0, ..., x_n^0)$ исходной задачи (1) определена: $x_n^0=x_n^0(c)$.

Если n-му процессу выделили x_n^0 единиц сырья, то на остальные n-1 процессов осталось $c-x_n^0$ единиц.

Положим в (5) $k=n-1$, $y=c-x_n^0$ и найдем $x_{n-1}^0(c-x_n^0)$. По определению $x_{n-1}^0(c-x_n^0)$ равно оптимальному количеству сырья, которое дается $(n-1)$-му процессу при условии, что $c-x_n^0$ единиц сырья надо разделить оптимальным образом между первыми $n-1$ процессами. Таким образом, получаем $x_{n-1}^0=x_{n-1}^0(c-x_n^0)$.

Продолжив процесс решения, найдем компоненты $x_{n-2}^0, ..., x_1^0$ решения исходной задачи (1). Проанализируем результат.

Достоинства метода:

1. Исходная задача (1) максимизации по n переменным свелась к (n–1) задачам (6) максимизации по одной переменной, причем результат – глобально оптимальный план.

2. В процессе решения не использовались аналитические свойства элементов задачи, исходные функции могли быть заданы таблично, графически, алгоритмически и т.д.

3. По результатам вычислений $B_k(y)$ легко построить решение задачи (1) при варьированных значениях параметров c и n, что позволяет провести анализ чувствительности решений задачи (1) к изменениям указанных параметров.

Недостатки метода

Основным недостатком метода является *«проклятие размерности»*. Суть этого недостатка состоит в том, что при решении уравнения Беллмана (6) приходится запоминать функции $B_k(y)$. В рассмотренной выше задаче с распределением сырья одного вида ими оказались функции одного аргумента. В общем случае количество аргументов равно количеству видов сырья. Табулирование функций многих переменных ($n > 2$) требует очень много места в оперативной памяти, что затрудняет реализацию метода.

Существуют способы борьбы с «проклятием размерности», но эти способы годятся не для всех задач.

Пример. **Рассмотрим пример с данными из табл. 2.1.**

Таблица 2.1

$c = 5, n = 3$

x	0	1	2	3	4	5
$f_1(x)$	0	1	2	3	4	5
$f_2(x)$	0	0	1	2	4	7
$f_3(x)$	0	2	2	3	3	5

Определим функции Беллмана по правилу:

$$B_2(y)=\max_{0\le z\le y}[f_2(z)+B_1(y-z)],\ B_3(y)=\max_{0\le z\le y}[f_3(z)+B_2(y-z)].$$

Например

$$B_2(4)=\max_{0\le z\le 4}(f_2(z)+B_1(4-z))=$$

$$=\{f_2(0)+B_1(4),f_2(1)+B_1(3),f_2(2)+B_1(2),$$

$$f_2(3)+B_1(1),f_2(4)+B_1(0)\}=4.$$

Значения функций Беллмана представим в табл. 2.2, где в каждой клетке наряду со значением функции Беллмана $B_k(y)$ в скобках укажем значение $x_k^0(y)$, на котором достигает максимума правая часть уравнения (6).

Таблица 2.2

y	1	2	3	4	5
$B_1(y)$	1	2	3	4	5
$B_2(y)$	1(0)	2(0)	3(0)	4(0,4)	7(5)
$B_3(y)$	2(1)	3(1)	4(1)	5(1)	7(0)

Из табл. 2.2 видно, что максимальная прибыль в рассматриваемой задаче равна $B_3(5)=7$. Найдем оптимальное распределение ресурсов. Поскольку $x_3^0(5)=$ $= 0$, то третьему технологическому процессу назначаются ресурсы в объеме $x_3^0=0$. На остальные процессы 1 и 2 остается ресурсов $5-0=5$. Прибыль от реализации процессов 1, 2 при объеме ресурсов 5 равна $B_2(5)=7$ и $x_2^0(5)=5$. Значит, второму процессу назначается ресурс в объеме 5: $x_2^0=5$. На первый процесс остается ресурса в объеме 5 – 5 = 0. Следовательно, $x_1^0=0$. Получили оптимальный план ($x_1^0=0$, $x_2^0=5$, $x_3^0=0$).

Изменим теперь в задаче одно условие: положим теперь $c = 4$. Согласно таблице, имеем: $B_3(4)=5$ – это максимальная прибыль, $x_3^0(4)=1$. Следовательно, на первый и второй процессы остается ресурса в объеме $4-1=3$. Далее по таблице находим $B_2(3)=3$ и $x_2^0(3)=0$. На первый процесс остается ресурса в объеме 3 – 0 = 3. Оптимальный план распределения ресурсов ($x_1^0=3$, $x_2^0=0$, $x_3^0=1$).

§ 3. Задача сетевого планирования

В *сетевом планировании* исследуются проблемы реализации сложных проектов (комплексов работ), состоящих из большого количества отдельных работ, которые должны выполняться в определенной технологической последовательности. Одна из основных задач сетевого планирования – *расчет времени выполнения проекта*.

Составим сетевую модель задачи. Факт (явление) начала (или окончания) какого-либо множества работ из заданной совокупности работ проекта назовем *событием* и поставим ему в соответствие узел $i \in I$. Работу, которая может начаться после события $i \in I$ и которая предшествует событию j, обозначим дугой $(i, j) \in U$. Ни одна работа $(i, j) \in U$ не может начаться, если не завершены все

работы $(k, i) \in U$, $k \in I_i^- = \{k \in I: \exists (k, i) \in U\}$, т.е. ни одна работа $(i, j) \in U$ не может начаться до наступления события i; момент наступления события i определяется моментом завершения **всех** работ (k, i), $k \in I_i^-$.

На сети[1] $S = \{I, U\}$ выделим два узла: s – *начальное событие* (начало выполнения проекта) и t – *конечное событие* (завершение проекта). По построению верны соотношения $I_s^- = \varnothing$, $I_t^+ = \varnothing$, где $I_i^+ = \{j \in I: \exists (i,j) \in U\}$.

Каждой дуге $(i, j) \in U$ припишем одну характеристику $c_{ij} > 0$ – время выполнения работы (i, j). Обозначим через x_i, $i \in I$, момент наступления события i; $x_s = 0$. Из технологических требований, имеющихся в проекте и отраженных в структуре сети S, следуют неравенства

$$x_i + c_{ij} \le x_j, \; i \in I_j^-, \; j \in I, \tag{9}$$

которые означают, что событие j наступает не раньше, чем будут завершены все работы (i, j), $i \in I_j^-$, которые предшествуют этому событию j.

Из (9) следует, что в сети S нет контуров. Действительно, предположим, что в S существует контур и обозначим через $U_{\text{конт}} \subset U$ дуги этого контура. Просуммируем неравенства (9) по дугам $(i, j) \in U_{\text{конт}}$ и получим

$$\sum_{(i,j) \in U_{\text{конт}}} c_{ij} \le 0.$$

Однако это противоречит предположению $c_{ij} > 0$, $(i, j) \in U$.

Далее будем предполагать, что

$$I_i^+ \ne \varnothing, \; i \in I \setminus t; \; I_i^- \ne \varnothing, \; i \in I \setminus s,$$

ибо этого всегда можно добиться, модифицируя сеть S и не нарушая технологической последовательности выполнения работ.

Минимальное время выполнения проекта определяется наименьшим числом x_t^0, которое в совокупности с числами

$$x_i^0 \ge 0, i \in I \setminus t; \; x_s^0 = 0 \tag{10}$$

удовлетворяет неравенствам (9).

Поскольку для окончания проекта необходимо, чтобы все работы были завершены, то длина каждого пути из s в t, равная сумме Σc_{ij}, вычисленной вдоль дуг пути, не меньше чем x_t^0. Чтобы убедиться в этом, достаточно сложить неравенства (9) вдоль этого пути. С другой стороны, очевидно, что найдется такая последовательность работ, составляющая путь из s в t, общая продолжительность которых равна x_t^0.

[1] Определения основных понятий на сети S приведены в главе 3.

Таким образом, задача вычисления x_t^0 сводится к поиску пути из s в t с максимальной длиной $\sum c_{ij}$. Такой путь принято называть *критическим*.

Заметим, что сформулированная задача:

$$\text{найти } x_t - x_s \to min \text{ при условиях (9), (10)} \qquad (11)$$

является двойственной к задаче о потоке минимальной стоимости на сети S [2, 7] с дуговыми стоимостями $-c_{ij} < 0$ и интенсивностями узлов $a_s = 1$, $a_t = -1$, $a_t = -1$, $a_i = 0$, $i \in I \setminus \{s, t\}$. При этом числа x_i^0, $i \in I$ – оптимальные потенциалы в задаче о потоке минимальной стоимости. Следовательно, задачу (11) можно решать *методом потенциалов*, который мы рассмотрим в курсе «Методы оптимизации» [9].

Используя специфику задачи (11), решим её методом динамического программирования. Итак, мы решаем задачу о построении критического (максимального) пути из s в t.

Согласно общей схеме динамического программирования, осуществим **первый этап** – инвариантное погружение в семейство задач.

Общая задача семейства состоит в построении максимального пути из s в любой узел $j \in I$. Длина B_j этого пути называется *функцией Беллмана*.

Осуществим **второй этап** – составим уравнение Беллмана, которому удовлетворяет функция Беллмана B_j. Для этого поступим следующим образом. Будем исследовать пути из s в j. Вначале предположим, что последней дугой пути из s в j является дуга $(i, j) \in U$, где $i \in I_j^-$, что в узел i из s мы попали вдоль пути максимальной длины, т. е. вдоль пути длины B_i. Тогда длина рассматриваемого пути из s в j через i равна

$$B_i + c_{ij}. \qquad (12)$$

Переберем все узлы $i \in I_j^-$ и найдем максимальное среди чисел (4):

$$\max_{i \in I_j^-} (B_i + c_{ij}). \qquad (13)$$

Рассуждая от противного, легко показать, что число (13) равно длине максимального пути из s в j. По определению длина максимального пути из s в j равна B_j, следовательно,

$$B_j = \max_{i \in I_j^-} (B_i + c_{ij}). \qquad (14)$$

Уравнение (14) – это **уравнение Беллмана.**

Значение функции Беллмана для узла s известно:

$$B_s = 0. \qquad (15)$$

Равенство (15) – краевое (начальное) условие для уравнения (14). В отличие от рассмотренного ранее примера уравнение Беллмана (14) не является явно рекуррентным.

Для решения уравнения (14) с начальным условием (15) разработаем специальный метод, называемый методом пометок.

Алгоритм метода пометок. Обозначим через I_* множество узлов $i \in I$, для которых известна функция Беллмана B_i. Множество I_* не пусто, так как, согласно (7), $s \in I_*$. Если $t \in I_*$, то задача решена, B_t – длина максимального пути из s в t. Для восстановления этого пути надо осуществить «*обратный ход*» решения уравнения Беллмана. Эту процедуру рассмотрим ниже. Для того чтобы легче осуществить «обратный ход», будем для узлов $i \in I_*$ кроме функции Беллмана B_i задавать еще функцию $f(i) \in I$. На первом шаге имеем

$$I_* = \{s\},\ B_s = 0,\ f(s) = 0.$$

Пусть $t \notin I_*$. В сети S построим множество узлов $w(I_*) = \{j \in I: (i, j) \in U, j \notin I_*, i \in I_*\}$, соседних с I_*. В силу того, что в сети S нет контуров, среди узлов $j \in w(I_*)$ обязательно найдется узел j_*, для которого

$$I^-_{j*} \subset I_*. \qquad (16)$$

Поскольку для узлов $i \in I_*$ значения B_i функции Беллмана уже известны, то из уравнения (14) можно найти

$$B_{j*} = \max_{i \in I^-_{j*}} (c_{ij*} + B_i) = c_{i*j*} + B_{i*}. \qquad (17)$$

Полагаем $I_* = I_* \cup j_*$, $f(j_*) = i_*$ и переходим к следующей итерации.

Замечание. Узлов типа j_*, для которых верно (16), может оказаться несколько. Для всех них по правилам (17) находим функцию Беллмана и функцию $f(j)$ и все эти узлы добавляем к множеству I_*.

Очевидно, что через конечное число (меньшее либо равное $|I|$) шагов мы придем к ситуации, когда $t \in I_*$. Это означает, что задача решена: B_t – длина максимального пути из s в t.

Осуществим «обратный ход» для нахождения максимального пути: $\{t, i_1, i_2, \ldots, i_k, s\}$ по правилу:

$$i_1 = f(t),\ i_2 = f(i_1),\ \ldots,\ i_k = f(i_{k-1}),\ s = f(i_k). \qquad (18)$$

Пример. Рассмотрим сеть, изображенную на рис. 2.1, где $s = 1$, $t = 4$.

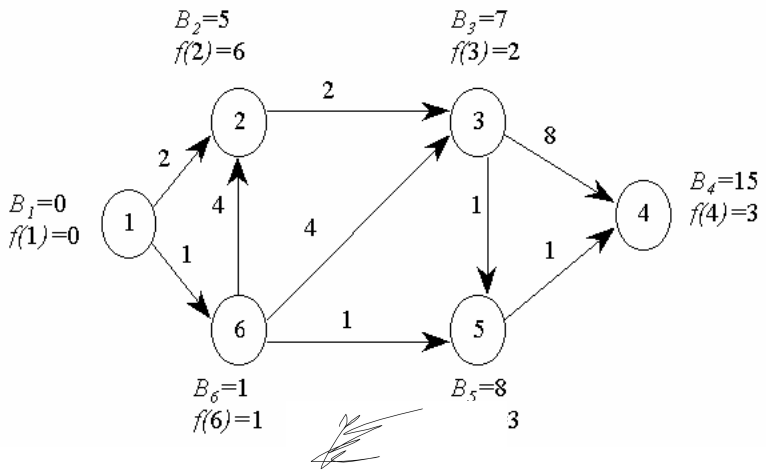


Рис. 2.1

Числа, указанные на дугах, равны длине соответствующей дуги. На этом же рисунке приведены значения функции Беллмана B_j, $j=\overline{1,6}$, определенные по методу пометок. Вторые метки $f(j)$, $j=\overline{1,6}$, позволяют восстановить критический путь из *s* в *t* по правилу (18):

$$i_1 := f(t)=3;\ \ i_2 := f(3)=2;\ \ i_3 := f(2)=6;\ \ i_4 := f(1).$$

Следовательно, искомый путь имеет вид

$$s = 1 \to 6 \to 2 \to 3 \to 4 = t.$$

Замечание. Работы, входящие в критический путь, должны начинаться в строго фиксированные моменты времени: например, работа $(i, j) \in U_{кр.путь}$ не может (чисто физически) начаться раньше момента времени x_i^0; если же она начнется позже момента x_i^0, то это приведет к увеличению времени выполнения всего проекта.

Для работ, не принадлежащих критическому пути, есть возможность несколько варьировать момент их начала без увеличения общей продолжительности выполнения всего проекта. Пусть $(i, j) \notin U_{кр.путь}$, тогда работа (i, j) чисто физически не может начаться раньше момента x_i^0. Но мы можем начать работу (i, j) в любой момент вида

$$x_i^0 + \Delta t_{ij}\ ,\ \text{где}\ \Delta t_{ij} \in [\,0, x_j^0 - c_{ij} - x_i^0\,],$$

без ущерба для общего времени выполнения проекта.

Эту возможность «сдвига» работ некритического пути используют при сетевом планировании распределения ресурсов (например трудовых ресурсов), при решении следующей задачи: определить такие Δt_{ij}, $(i, j) \in U$, чтобы в каждый момент времени для выполнения текущих работ требовалось минимальное количество рабочих. Моменты времени x_i^0, $i \in I$, считаются уже найденными и фиксированными.

§ 2. Кратчайшие пути между всеми парами вершин (задача о многополюсной кратчайшей цепи)

1. Обоснование метода решения с помощью динамического программирования. Рассмотрим задачу нахождения кратчайших путей между всеми парами узлов сети $S = \{I, U\}$. Пусть $I = \{1,2,...,n\}$ – множество узлов сети *S*, $c_{ij} \geq 0$ – длина дуги $(i, j) \in U$. Считаем, что $c_{ij} = \infty$, если $(i, j) \notin U$, и дуги из *U* являются ориентированными. Если одна из дуг (или несколько дуг) является неориентированной, то считаем, что в сети есть ориентированные дуги (i, j) и (j, i) с одинаковой длиной $c_{ij} = c_{ji}$.

Для решения поставленной задачи будем использовать алгоритм, разработанный Флойдом. Для обоснования алгоритма воспользуемся методом динамического программирования.

1-й этап. Осуществим инвариантное погружение исходной задачи в семейство (состоящее из *n* задач) аналогичных задач. Каждая *j*-я задача, где $j = 1,2,...,n$, данного семейства формулируется следующим образом: для каждой пары узлов $i, k \in I$ найти путь минимальной длины, промежуточные узлы которого могут принадлежать только множеству узлов $\{1,2,..., j\}$.

Обозначим через d_{ik}^{j} длину минимального пути из i в k при условии, что промежуточными могут быть только узлы из $\{1,2,..., j\}$. По построению d_{ik}^{j} – это функция Беллмана.

2-й этап. Составим уравнение Беллмана для функции Беллмана. С этой целью для пары узлов $i, k \in I$ рассмотрим все пути, которые могут проходить через узлы $\{1,2,..., j+1\}$. Все такие пути можно разбить на две группы:

а) пути, не содержащие узел $j+1$;

б) пути, содержащие узел $j+1$.

Найдём длину минимального пути в группе «а». Согласно определению функции Беллмана, длина такого пути равна d_{ik}^{j}.

Теперь найдём длину минимального пути в группе «б». Ясно, что длина такого пути равна $d_{ij+1}^{j} + d_{j+1k}^{j}$. Значит, длина минимального пути из i в k с промежуточными узлами, принадлежащими множеству $\{1,2,..., j+1\}$, равна $min\{d_{ik}^{j}, d_{ij+1}^{j} + d_{j+1k}^{j}\}$.

Следовательно, согласно определению функции Беллмана, имеем

$$d_{ik}^{j+1} = min\{d_{ik}^{j}, d_{i\,j+1}^{j} + d_{j+1\,k}^{j}\}, \forall i, \ k \in I. \qquad (18)$$

Уравнение (18) – это уравнение Беллмана. Из него, в частности, следует, что

$$d_{j+1\,k}^{j+1} = d_{j+1\,k}^{j}, \ \ d_{i\,j+1}^{j+1} = d_{i\,j+1}^{j}, \ \forall i, k \in I.$$

Кроме того, очевидно, что

$$d_{ii}^{j} = 0 \text{ при } \forall i \in I, \forall j = \overline{1,n}.$$

Зададим начальные условия для уравнения Беллмана:

$$d_{ii}^0 = 0,\ i = \overline{1,n};\ \ d_{ik}^0 = c_{ik},\ i = \overline{1,n};\ k = \overline{1,n};\ i \neq k. \qquad (19)$$

Напомним, что равенство $c_{ik} = \infty$ означает, что в сети $S = \{I, U\}$ нет дуги (i, k).

3-й этап. Решим уравнение Беллмана (прямой ход) и по нему восстановим решение исходной задачи (обратный ход). Уравнение (18) имеет явно выраженный динамический характер (динамика идёт по $j = 1,2,...,n$). Для построения и запоминания функции Беллмана удобно использовать матричный метод, который позволит нам запоминать длины кратчайших путей и восстановить дуги, входящие в эти пути.

2. Алгоритм Флойда. На каждой итерации алгоритма строятся две $(n \times n)$-матрицы: *D* и *R*. Матрица *D* называется *матрицей длин кратчайших путей* и содержит текущие оценки длин кратчайших путей, т.е. на *j*-й итерации имеем

$$D^j = (d_{ik}^j,\ k = \overline{1,n},\ i = \overline{1,n}).$$

Алгоритм начинает работу при $D^0 = (d_{ik}^0,\ k = \overline{1,n},\ i = \overline{1,n})$, где числа d_{ik}^0 определены согласно (19). Для построения D^1 используется *трёхместная операция* (18) при $j = 1$, и т.д.

Матрица *R* называется *матрицей маршрутов* и служит для нахождения промежуточных узлов (если такие имеются) кратчайших путей. На *j*-й итерации она определяется как $R^j = (r_{ik}^j,\ k = \overline{1,n},\ i = \overline{1,n})$, где r_{ik}^j – первый промежуточный узел кратчайшего пути из *i* в *k*, получаемого на *j*-й итерации. Алгоритм начинает работу при $R^0 = (r_{ik}^0,\ k = \overline{1,n},\ i = \overline{1,n})$, где $r_{ik}^0 = k$. На *j*-й итерации элемент r_{ik}^j получается по следующим правилам:

$$r_{ik}^j = \begin{cases} r_{ij}^{j-1}, & \text{если } d_{ij}^{j-1} > d_{ij}^{j-1} + d_{jk}^{j-1}, \\ r_{ik}^{j-1} - & \text{в противном случае}. \end{cases} \qquad (20)$$

После построения D^0 и R^0 нужно последовательно построить матрицы $D^j, R^j,\ j = 1,2,...,n$, используя правила пересчёта (18), (20).

Опишем подробнее реализацию трёхместной операции (18) на *j*-й итерации, используя матрицы D^{j-1} и R^{j-1}, полученные на (*j*-1)-й итерации.

Шаг 1. Вычеркнем элементы *j*-й строки и *j*-го столбца матрицы D^{j-1}. Назовём элементы *j*-й строки и *j*-го столбца базисными элементами (базисной строкой и столбцом соответственно).

Шаг 2. Для каждого элемента (i,k), $i \neq j,\ k \neq j$, матрицы расстояний (начиная с первого и не принадлежащего ни базисной строке, ни базисному

столбцу) сравним числа d_{ik}^{j-1} и сумму соответствующих элементов базового столбца и базовой строки $d_{ij}^{j-1}+d_{jk}^{j-1}$. Если $d_{ij}^{j-1}+d_{jk}^{j-1}\geq d_{ik}^{j-1}$, то полагаем $d_{ik}^{j}=d_{ik}^{j-1}$ и выбираем новые значения для переменных i и k. Если $d_{ij}^{j-1}+d_{jk}^{j-1}<d_{ik}^{j-1}$, то полагаем

$$d_{ik}^{j}=d_{ij}^{j-1}+d_{jk}^{j-1}$$

и переходим к новым переменным i и k. При этом параллельно изменяем элементы матрицы R^{j-1} на элементы матрицы R^{j} согласно правилам (20). Для элементов базисной строки и базисного столбца полагаем

$$d_{ij}^{j}=d_{ij}^{j-1},\ i=\overline{1,n};\ d_{jk}^{j}=d_{jk}^{j-1},\ k=\overline{1,n},$$

т.е. они не меняются.

Анализируя соотношения (18), можно разработать правила, упрощающие пересчёт матриц $D^{j-1}\to D^{j}$.

Ясно, что если $d_{ij}^{j-1}=\infty$ (т.е. *i*-й элемент базисного столбца равен ∞), то все элементы *i*-й строки остаются прежними и не пересчитываются:

$$d_{ik}^{j}=d_{ik}^{j-1},\ k=\overline{1,n}.$$

Аналогично, если $d_{kj}^{j-1}=\infty$ (т.е. k-й элемент базисной строки равен ∞), то все элементы *k*-го столба остаются прежними и не пересчитываются:

$$d_{ik}^{j}=d_{ik}^{j-1},\ i=\overline{1,n}.$$

При реализации пересчёта $D^{j-1}\to D^{j}$ удобно из таблицы вычёркивать строки и столбцы, не подлежащие пересчёту.

Пусть матрицы D^{j} и R^{j} найдены. При $j<n$ переходим к новой (*j*+1)-й итерации. При *j* = *n* STOP. Матрица D^{n} состоит из длин минимальных путей. Матрица R^{n} используется для восстановления минимального пути.

Опишем процедуру восстановления кратчайшего пути.

Предположим, нас интересует минимальный путь из i_0 в j_0. Длина этого пути равна $d_{i_0 j_0}^{n}$. Для восстановления пути из i_0 в j_0 рассмотрим элементы матрицы R^{n}:

1. Найдём элементы $r_{i_0 j_0}^{n}$. Пусть $i_1:=r_{i_0 j_0}^{n}$, значит, i_1 – первый промежуточный узел кратчайшего пути из i_0 в j_0.

2. Найдём элемент $r_{i_1 j_0}^{n}$. Пусть $r_{i_1 j_0}^{n}=i_2$, следовательно, i_2– первый промежуточный узел кратчайшего пути из i_1 в j_0; i_2 – второй промежуточный

узел кратчайшего пути из i_0 в j_0 и т.д., пока для некоторого i_s ни получим $r^n_{i_s j_0} = j_0$. Таким образом, минимальный путь из i_0 в j_0 последовательно проходит через узлы $i_0, i_1, i_2, ..., i_s, j_0$.

Пример. Крупное учреждение планирует разработать систему внутренней доставки почты, основанную на использовании линии пневматической связи, для распределения корреспонденции между 8 отделами. Некоторые отделы будут только отсылать корреспонденцию в другие отделы, не имея при этом возможности получать почту. Все остальные отделы могут получать и отправлять почту. Расположение линий пневматической связи изображено на рис. 3.6.

Каждому отделу соответствует узел, каждая дуга – это линия связи. Числа на дугах – расстояния между отделами.

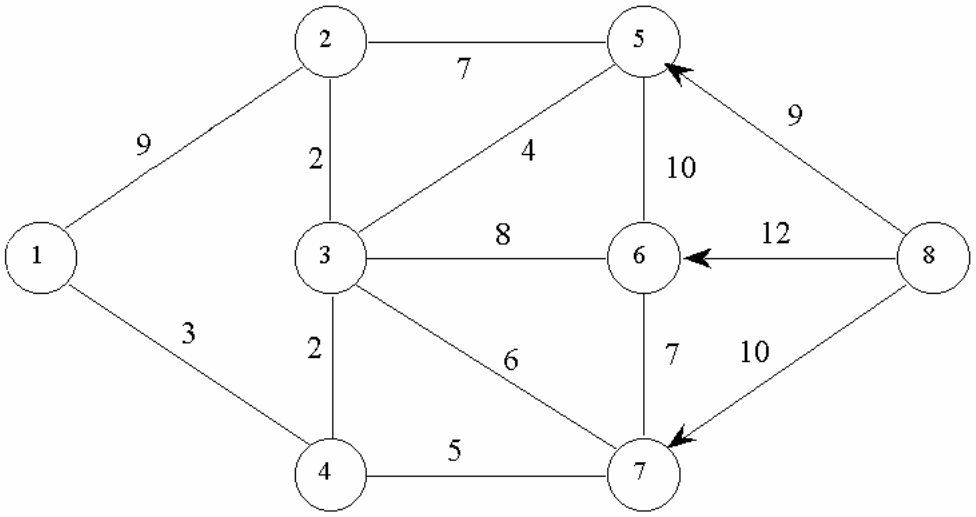


Рис. 3.6

Для того чтобы каждый отдел при пересылке почты в другой отдел смог бы определить оптимальный путь, необходимо заготовить таблицу, указывающую кратчайший путь между каждой парой отделов. Для построения такой таблицы воспользуемся алгоритмом Флойда.

Согласно алгоритму, начальные матрицы кратчайших путей и маршрутов имеют вид

$$D^0 = \begin{bmatrix} 0 & 9 & \infty & 3 & \infty & \infty & \infty & \infty \\ 9 & 0 & 2 & \infty & 7 & \infty & \infty & \infty \\ \infty & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & \infty & 2 & 0 & \infty & \infty & 5 & \infty \\ \infty & 7 & 4 & \infty & 0 & 10 & \infty & \infty \\ \infty & \infty & 8 & \infty & 10 & 0 & 7 & \infty \\ \infty & \infty & 6 & 5 & \infty & 7 & 0 & \infty \\ \infty & \infty & \infty & \infty & 9 & 12 & 10 & 0 \end{bmatrix}, \; R^0 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}.$$

Итерация 1. $j = 1$ – базовый элемент. Из матрицы D^0 вычёркиваем 1-й столбец и 1-ю строку. Кроме того, столбцы 3, 5, 6, 7, 8 и строки 3, 5, 6, 7, 8 также можно вычеркнуть, так как в базовой строке и в базовом столбце на соответствующих местах стоят ∞. Следовательно, рабочая матрица имеет вид

	1	2	4	
1	0	9	3	← Базовая строка
2	9	0	∞	
4	3	∞	0	

↑ Базовый столбец

Диагональные элементы матрицы D^0 можно не рассматривать. Значит, необходимо исследовать две оценки d_{24}^0 и d_{42}^0. Применение трёхместной операции даёт следующие результаты:

$$d_{24}^1 = min\ \{d_{24}^0, d_{21}^0 + d_{11}^0\} = min\ \{\infty, 9+3\} = 12,$$

$$d_{42}^1 = min\ \{d_{42}^0, d_{41}^0 + d_{12}^0\} = min\ \{\infty, 3+9\} = 12.$$

Новые оценки лучше старых: $d_{24}^1 < d_{24}^0$, $d_{42}^1 < d_{42}^0$, поэтому полагаем $r_{24}^1 = 1$, $r_{42}^1 = 1$. Все остальные элементы матриц D^1 и R^1 остаются прежними. Выпишем матрицы D^1 и R^1

$$D^1=\begin{bmatrix} 0 & 9 & \infty & 3 & \infty & \infty & \infty & \infty \\ 9 & 0 & 2 & 12 & 7 & \infty & \infty & \infty \\ \infty & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & 12 & 2 & 0 & \infty & \infty & 5 & \infty \\ \infty & 7 & 4 & \infty & 0 & 10 & \infty & \infty \\ \infty & \infty & 8 & \infty & 10 & 0 & 7 & \infty \\ \infty & \infty & 6 & 5 & \infty & 7 & 0 & \infty \\ \infty & \infty & \infty & \infty & 9 & 12 & 10 & 0 \end{bmatrix},\ R^1=\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 1 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 1 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}.$$

Итерация 2. Определим узел $j=2$ как базовый и выделим в D^1 вторую строку и второй столбец – это базовые строка и столбец. Кроме того, можно вычеркнуть столбцы 6, 7, 8 и строки 6, 7, 8, так как в базовых строке и столбце на соответствующих местах стоят ∞. «Рабочая» матрица D_p^1 имеет вид

$$D_p^1=\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 9 & \infty & 3 & \infty \\ 2 & 9 & 0 & 2 & 12 & 7 \\ 3 & \infty & 2 & 0 & 2 & 4 \\ 4 & 3 & 12 & 2 & 0 & \infty \\ 5 & \infty & 7 & 4 & \infty & 0 \end{array}.$$

Применяем трёхместную операцию к элементам матрицы D_p^1 (диагональные элементы не пересчитываем):

$$d_{13}^2=min\left\{d_{13}^1,\ d_{12}^1+d_{23}^1\right\}=min\{\infty,\ 9+2\}=11<d_{13}^1,$$

$$d_{14}^2=min\left\{d_{14}^1,\ d_{12}^1+d_{24}^1\right\}=min\{3,\ 9+12\}=3<d_{14}^1,$$

$$d_{15}^2=min\left\{d_{15}^1,\ d_{12}^1+d_{25}^1\right\}=min\{\infty,\ 9+7\}=16<d_{15}^1,$$

$$d_{31}^2=min\left\{d_{31}^1,\ d_{32}^1+d_{23}^1\right\}=min\{\infty,\ 9+2\}=11<d_{31}^1,$$

$$d_{34}^2=min\{2,\ 12+2\}=2=d_{34}^1,$$

$$d_{35}^2=min\{4,\ 2+7\}=4=d_{35}^1,$$

$$d_{41}^2=min\{3,\ \infty+12\}=3=d_{41}^1,$$

$$d_{43}^2=min\{2,\ 12+0\}=2=d_{43}^1,$$

$$d_{45}^2=min\{\infty,\ 12+7\}=19<d_{45}^1,$$

$$d_{51}^2=min\{\infty,\ 9+7\}=16<d_{51}^1,$$

$$d_{53}^2 = min\{4,\ 7+2\} = 4 = d_{53}^1,$$

$$d_{54}^2 = min\{\infty,\ 7+12\} = 2 < d_{54}^1.$$

Остальные $d_{ik}^2 = d_{ik}^1$! Следовательно, полагаем $r_{13}^2 = r_{15}^2 = r_{31}^2 = r_{45}^2 = r_{51}^2 = r_{54}^2 = 2$ и все остальные r_{ik}^2 не меняем: $r_{ik}^2 = r_{ik}^1$. Запишем матрицы D^2 и R^2:

$$D^2 = \begin{bmatrix} 0 & 9 & 11 & 3 & 16 & \infty & \infty & \infty \\ 9 & 0 & 2 & 12 & 7 & \infty & \infty & \infty \\ 11 & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & 12 & 2 & 0 & 19 & \infty & 5 & \infty \\ 16 & 7 & 4 & 19 & 0 & 10 & \infty & \infty \\ \infty & \infty & 8 & \infty & 10 & 0 & 7 & \infty \\ \infty & \infty & 6 & 5 & \infty & 7 & 0 & \infty \\ \infty & \infty & \infty & \infty & 9 & 12 & 10 & 0 \end{bmatrix}, \quad R^2 = \begin{bmatrix} 1 & 2 & 2 & 4 & 2 & 6 & 7 & 8 \\ 1 & 2 & 3 & 1 & 5 & 6 & 7 & 8 \\ 2 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 1 & 3 & 4 & 2 & 6 & 7 & 8 \\ 2 & 2 & 3 & 2 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}.$$

Выполняя аналогичные операции на итерациях 3, 4, 5, 6, 7, 8, мы получим матрицы D^i, R^i, $i = \overline{3,8}$. Матрицы D^8 и R^8 приведены ниже:

$$D^8 = \begin{bmatrix} 0 & 7 & 5 & 3 & 9 & 13 & 8 & \infty \\ 7 & 0 & 2 & 4 & 6 & 10 & 8 & \infty \\ 5 & 2 & 0 & 2 & 4 & 8 & 6 & \infty \\ 3 & 4 & 2 & 0 & 6 & 10 & 5 & \infty \\ 9 & 6 & 4 & 6 & 0 & 10 & 10 & \infty \\ 13 & 10 & 8 & 10 & 10 & 0 & 7 & \infty \\ 8 & 8 & 6 & 5 & 10 & 7 & 0 & \infty \\ 18 & 15 & 13 & 15 & 9 & 12 & 10 & 0 \end{bmatrix}, \quad R^8 = \begin{bmatrix} 1 & 4 & 4 & 4 & 4 & 4 & 4 & 8 \\ 3 & 2 & 3 & 3 & 3 & 3 & 3 & 8 \\ 4 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 3 & 3 & 4 & 3 & 3 & 7 & 8 \\ 3 & 3 & 3 & 3 & 5 & 6 & 3 & 8 \\ 3 & 3 & 3 & 3 & 5 & 6 & 7 & 8 \\ 4 & 3 & 3 & 4 & 3 & 6 & 7 & 8 \\ 5 & 5 & 5 & 5 & 5 & 6 & 7 & 8 \end{bmatrix}.$$

Для иллюстрации результатов, содержащихся в матрицах D^8, R^8, рассмотрим кратчайший путь из узла 1 в узел 5. Длина этого пути равна $d_{15}^8 = 9$. Для того чтобы найти сам путь из 1 в 5, обратимся к матрице R^8. Поскольку r_{15}^8 равно 4, то узел 4 является первым промежуточным узлом пути из 1 в 5. Затем, для того чтобы найти узел, следующий за узлом 4 в пути, ведущем в 5, определяем значение r_{45}^8. Данное значение равно 3. Значит, за узлом 4 следует узел 3. Далее находим $r_{35}^8 = 5$. Следовательно, кратчайший путь из 1 в 5 проходит через узлы $1 \to 4 \to 3 \to 5$.

Глава 4. Потоки в сетях

Популярность сетевых оптимизационных моделей, которые обычно являются частными случаями моделей ЛП, можно объяснить прежде всего следующими причинами.

Часто они относятся к задачам распределения продукции. Следовательно, модели этого класса имеют экономическую интерпретацию и могут найти применение во многих промышленных фирмах и предприятиях. Кроме того, математическая структура сетей идентична структуре других оптимизационных моделей, на первый взгляд не имеющих с ними ничего общего. Однако указанные две причины не могут служить основанием для выделения сетевых моделей в качестве предмета специального изучения.

Важнейшей причиной, обусловливающей целесообразность такого выделения, являются особенности математических характеристик сетевых моделей. Используя эти особенности, можно существенно повысить эффективность процесса отыскания оптимальных решений задач, которые удаётся описать на «сетевом языке». В реальных примерах сетевые модели часто содержат тысячи переменных и сотни ограничений. В связи с этим применение эффективных методов становится не только выгодным, но просто необходимым.

Наконец, исследуя сети, можно убедиться, что разнообразные, на первый взгляд, совершенно непохожие оптимизационные модели допускают применение общего метода, что, несомненно, обеспечивает существенные преимущества.

§ 1. Примеры прикладных задач, имеющих сетевую форму

1. Классическая транспортная задача (в матричной форме). Имеется n пунктов производства некоторого (одного) продукта. Обозначим через a_i объём производства продукта в i-м пункте производства, $i = \overline{1,n}$. Имеется m пунктов потребления этого продукта. Обозначим через b_j объём потребления продукта в j-м пункте потребления, $j = \overline{1,m}$.

Перевозка единицы продукции из i-го пункта производства в j-й пункт потребления стоит $c_{ij}, i = \overline{1,n}, \quad j = \overline{1,m}$. Требуется составить такой план перевозок продукта от пунктов производства в пункты потребления, чтобы:

1) вывести весь продукт из каждого пункта производства;
2) удовлетворить спрос каждого пункта потребления;
3) минимизировать общую стоимость перевозок.

Обозначим через x_{ij} объём продукта, перевозимый из i-го пункта производства в j-й пункт потребления. Математическая модель задачи состоит в следующем: найти такой план перевозок ($x = x_{ij}$, $i = \overline{1,n}$, $j = \overline{1,m}$), чтобы

$$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \to min, \tag{1}$$

$$\sum_{j=1}^{m} x_{ij} = a_i, \quad i = \overline{1,n}; \quad \sum_{i=1}^{n} x_{ij} = b_j, \quad j = \overline{1,m}, \tag{2}$$

$$x_{ij} \geq 0, \quad i = \overline{1,n}; \quad j = \overline{1,m}. \tag{3}$$

Иногда в условия задачи добавляют ещё одно требование: объём перевозки продукта из i-го пункта производства в j-й пункт потребления не должен превышать заданного числа d_{ij}. В этом случае ограничение (3) заменяется ограничением

$$0 \leq x_{ij} \leq d_{ij}, \quad i = \overline{1,n}; \quad j = \overline{1,m}. \tag{3'}$$

Задача (1) – (3) и метод её решения (метод потенциалов) были рассмотрены в курсе «Методы оптимизации» [9]. Метод решения задачи (1), (2), (3') легко получить из метода решения задачи (1) – (3), зная правила симплекс-метода для задачи ЛП с двухсторонними ограничениями на переменные [3].

2. Сетевая модель транспортной задачи (задача о потоке минимальной стоимости). Большое значение имеет обобщение классической транспортной задачи путём включения в неё случаев, когда некоторые пункты являются транзитными.

Имеется n пунктов $i \in I = \{1,2,\ldots,n\}$, которые связаны между собой системой дорог, которую удобно представлять в виде набора пар

$$(i,j) \in U \subset \{(i,j),\ i = \overline{1,n},\ j = \overline{1,n}\}.$$

Если пара $(i, j) \in U$, то это означает, что есть дорога из i в j. (Но это ещё не означает, что есть дорога из j в i. Для существования такой дороги надо, чтобы существовала дуга $(j, i) \in U$!)

Все пункты $i \in I$ делятся на непересекающиеся три группы:

$$I = I_{пр} \cup I_{потр} \cup I_{тр},$$

где $i \in I_{пр}$ – пункты производства; обозначим через $a_i > 0$ объёмы производства в этих пунктах;

$i \in I_{потр}$ – пункты потребления; обозначим через $a_i < 0$ объёмы потребления в этих пунктах;

$i \in I_{тр}$ – транзитные пункты, для них $a_i = 0$.

Числа a_i, $i \in I$, называются интенсивностями узлов $i \in I$.

Обозначим через c_{ij}, $(i, j) \in U$, стоимость перевозки единицы продукции по дороге (i, j) из i в j. Требуется так организовать перевозки от пунктов производства к пунктам потребления, чтобы:

1) для каждого пункта соблюдалось условие баланса:

– для пункта производства – сумма произведённого продукта плюс сумма введённого продукта равны сумме вывезенного продукта;

– для пункта потребления – сумма ввезенного продукта равна потреблённому продукту плюс сумма вывезенного продукта;

– для транзитного пункта – сумма ввезённого продукта равна сумме вывезенного продукта;

2) суммарная стоимость всех перевозок была минимальна.

Обозначим через x_{ij} объём перевозки по дуге (i, j). Тогда математическая модель задачи имеет вид

$$\sum_{(i,j)\in U} c_{ij} x_{ij} \to min, \tag{4}$$

$$\sum_{j\in I_i^+} x_{ij} - \sum_{j\in I_i^-} x_{ji} = a_i \ , \ i \in I, \tag{5}$$

$$x_{ij} \geq 0, \ (i, j) \in U. \tag{6}$$

К сформулированной выше задаче можно добавить ещё одно условие: объём перевозки по дороге (i, j) не должен превышать числа d_{ij}. Тогда ограничение (6) заменяется на ограничение

$$0 \leq x_{ij} \leq d_{ij} \ , \ (i, j) \in U. \tag{6'}$$

Задача (4), (5), (6') называется задачей о потоке минимальной стоимости с ограничениями на пропускные способности дуг. Методы решения задачи (4) – (6) рассмотрены в [2, 3, 7].

3. Многопродуктовая транспортная задача. В задачах, описанных в пп. 1, 2, речь шла о перевозках одного вида продукта из пунктов производства в пункты потребления. Можно рассматривать аналогичные задачи, в которых речь идёт о нескольких видах продуктов. Рассмотрим двухпродуктовую задачу.

Пусть есть пункты $i \in I$, связанные сетью дорог $(i, j) \in U$. Каждая дорога $(i, j) \in U$ имеет ограниченную пропускную способность d_{ij} , $(i, j) \in U$.

Для каждого пункта $i \in I$ заданы два числа a_i, b_i:

a_i – интенсивность узла (пункта) i по первому продукту (если $a_i > 0$, то первый продукт производится в i; если $a_i < 0$, то первый продукт потребляется в i; если $a_i = 0$, то пункт i – транзитный по первому продукту);

b_i – интенсивность узла (пункта) i по второму продукту (если $b_i > 0$, то второй продукт производится в i; если $b_i < 0$, то второй продукт потребляется в i, если $b_i = 0$, то i – транзитный пункт по второму продукту).

Заданы числа c_{ij}, f_{ij} – стоимость перевозки единицы продукции первого и второго видов по дуге $(i, j) \in U$. Требуется найти план перевозок продукции первого и второго вида, такой, что:

1) для каждого узла выполняются условия баланса по продукции первого и второго видов;

2) суммарный объём перевозок продукции первого и второго видов по дуге (i, j) не превосходит её пропускной способности d_{ij};

3) суммарная стоимость перевозок продукции двух видов минимальная.

Обозначим через x_{ij} и y_{ij} объёмы перевозок первого и второго видов продукции по дуге (i, j). Математическая модель двухпродуктовой транспортной задачи имеет вид

$$\sum_{(i,j)\in U} c_{ij} x_{ij} + \sum_{(i,j)\in U} f_{ij} y_{ij} \to \min,$$

$$\sum_{j\in I_i^+} x_{ij} - \sum_{j\in I_i^-} x_{ji} = a_i \ , \ i\in I \ ,$$

$$\sum_{j\in I_i^+} y_{ij} - \sum_{j\in I_i^-} y_{ji} = b_i \ , \ i\in I \ ;$$

$$x_{ij} \ge 0 \ , \ y_{ij} \ge 0 \ , \ x_{ij} + y_{ij} \le d_{ij} \ , \ (i,j)\in U \ .$$

Последнее ограничение кажется малосущественным, однако именно оно значительно усложняет решение последней задачи по сравнению с аналогичной однопродуктовой задачей (4), (5), (6'). Методы решения двухпродуктовой транспортной задачи описаны в [3].

4. Задача о построении дерева кратчайших путей из заданного узла *s*. Эта задача подробно описана в главе 3. Там же отмечалось, что она – частный случай задачи (4) – (6) с интенсивностями узлов, заданными следующим образом:

$$a_s = n-1, \quad a_i = -1, \quad i \in I \setminus \{s\}.$$

Один из методов решения задачи о построении дерева кратчайших путей описан также в главе 3.

5. Задача о расчёте минимального времени выполнения комплекса работ (задача о критическом пути из *s* в *t*). Математическая модель этой задачи описана в главе 2. Там же показано, что эта задача – частный случай задачи (4) – (6), когда min заменяем на max и

$$a_s = 1, \quad a_t = -1, \quad a_i = 0, \quad i \in I \setminus \{s,t\}.$$

Методы решения данной задачи изложены также в главе 2.

6. Задача о назначениях. Задачу о назначениях можно кратко сформулировать следующим образом. Задано *n* работ, каждую из которых может выполнить любой из *n* исполнителей. Стоимость выполнения работы *i* исполнителем *j* равна c_{ij}. Нужно распределить исполнителей по работам, т.е. назначить по одному исполнителю на каждую работу таким образом, чтобы минимизировать общие затраты.

Построим математическую модель данной задачи. Определим переменную

$$x_{ij} = \begin{cases} 1, & \text{если на работу } i \text{ назначается исполнитель } j; \\ 0 - & \text{в противном случае.} \end{cases}$$

Тогда математическая модель рассматриваемой задачи имеет вид

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \to min \, , \qquad (7)$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = \overline{1,n}; \qquad \sum_{i=1}^{n} x_{ij} = 1,\ j = \overline{1,n}; \tag{8}$$

$$0 \le x_{ij} \le 1, \quad i = \overline{1,n}; \quad j = \overline{1,n}; \tag{9}$$

$$x_{ij} - \text{целое}, \quad i = \overline{1,n}; \quad j = \overline{1,n}. \tag{10}$$

Очевидно, что задача (7) – (9) – частный случай транспортной задачи (1), (2), (3'), когда

$$a_i = 1, \quad i = \overline{1,n}; \quad b_j = 1, \quad j = \overline{1,m}, \quad n = m \quad \text{и} \quad d_{ij} = 1, \quad i = \overline{1,n}; \quad j = \overline{1,m}.$$

Нетрудно показать, что задача (1), (2), (3') обладает следующим свойством: она имеет целочисленное решение, если $a_i, i = \overline{1,n}$; $b_j, j = \overline{1,m}$ – целые числа. Следовательно, и задача (7) – (10) – частный случай задачи (1), (2), (3') – имеет целочисленное решение. Значит, для решения задачи (7) – (10) можно использовать метод потенциалов, отбросив условия целочисленности. Отметим, что задача (7) – (10) – специальная транспортная задача и для её решения можно разработать другие методы, отличные от метода потенциалов, учитывающие специфику этой задачи. Один из таких методов будет рассмотрен в §3.

7. Задача коммивояжёра. Эта задача относится к следующей ситуации: коммивояжёр собирается посетить каждый из *n* городов по одному разу, выехав из первого города и вернувшись в него же. Ни один город коммивояжёр не должен посещать дважды. Расстояние между городами *i* и *j* равно c_{ij} (если между городами *i* и *j* нет дороги, то полагаем $c_{ij} = \infty$). Надо найти кратчайший маршрут коммивояжёра.

Математическая модель этой задачи отображает также ситуацию совершенно иного характера. Имеется *n* сортов мороженого, которое изготавливается на одном и том же оборудовании. Пусть c_{ij} означает затраты времени на очистку и подготовку оборудования, когда сорт *j* изготавливается после сорта *i*. Предполагается, что заданная последовательность производства повторяется каждый день, т.е. оборудование после последнего сорта мороженого опять настраивается на производство первого сорта.

Требуется найти такую последовательность производства, при которой затраты на переналадку были бы минимальными.

Обозначим

$$x_{ij} = \begin{cases} 1, & \text{если из города } i \text{ идём в город } j, \\ 0 & - \text{ в противном случае.} \end{cases}$$

Математическая модель задачи коммивояжёра совпадает с задачей (7) – (10) плюс ещё одно дополнительное требование[2]:

[2] В (11) слово «цикл» можно было бы заменить словом «контур», поскольку с учетом ограничений (8) – (10) легко показать, что все циклы, образованные совокупностью дуг $U_*(x)$, являются контурами.

совокупность дуг $U_*(x)=\{(i,j)\in U, x_{ij}=1\}$ образует один цикл. (11)

Дополнительное ограничение (11) является существенным. Решив задачи (7) – (9) (без дополнительного условия (11)), мы можем получить такой оптимальный план x^0, для которого множество $U_*(x^0)$ состоит из двух и более циклов, что недопустимо в исходной задаче о коммивояжёре. Отметим, что ограничение (11) существенно усложняет решение задачи о коммивояжёре. К настоящему времени разработано много методов решения задачи о коммивояжере. Некоторые из них будут описаны в § 4.

8. Задача о максимальном потоке. Пусть задана некоторая ориентированная сеть $S = \{I, U\}$. На каждой дуге $(i,j)\in U$ задано число $d_{ij}\geq 0$ – пропускная способность дуги $(i,j)\in U$. В сети S выделены два узла $s\in I$ и $t\in I$, $s\neq t$; s – источник, t – сток. Требуется найти максимальный поток из узла s в узел t по дугам сети S при условии, что величина x_{ij} дугового потока по дуге $(i,j)\in U$ положительна и не превышает числа d_{ij} – пропускной способности дуги (i,j).

Математическая модель данной задачи имеет вид

$$v\to \max_{v,x},$$

$$\sum_{j\in I_i^+} x_{ij} - \sum_{j\in I_i^-} x_{ji} = \begin{cases} v & \text{при} \quad i=s, \\ 0 & \text{при} \quad i\in I\setminus\{s,t\}, \\ -v & \text{при} \quad i=t, \end{cases} \qquad (12)$$

$$0\leq x_{ij}\leq d_{ij}, \quad (i,j)\in U.$$

Свойства задачи (12) и методы ее решения описаны в следующем параграфе.

§ 2. Задача о максимальном потоке

1. Связь задачи о максимальном потоке с задачей о потоке минимальной стоимости. Как отмечалось в предыдущем параграфе, математическая модель задачи о максимальном потоке имеет вид (12). Покажем, что задача (12) является частным случаем задачи о потоке минимальной стоимости. Рассмотрим расширенную сеть $\overline{S}=\{I,\overline{U}\}$, которая получается из исходной сети $S=\{I, U\}$ добавлением дуги (t,s): $\overline{U}=U\cup(t,s)$.

Положим $a_i = 0$, $i\in I$, $c_{ij}=0$, $(i,j)\in U$, $c_{ts}=-1$, $d_{ts}=\infty$ и рассмотрим задачу о потоке минимальной стоимости на сети $\overline{S}=\{I,\overline{U}\}$:

$$\sum_{(i,j)\in\overline{U}} c_{ij}\, x_{ij} \to \min,$$
$$\sum_{j\in I_i^+} x_{ij} - \sum_{j\in I_i^-} x_{ji} = a_i\ ,\ i\in I\ ; \qquad (13)$$
$$0\le x_{ij} \le d_{ij}\ ,\ (i,j)\in\overline{U}\ .$$

Очевидно, что задача (13) эквивалентна задаче (14). Следовательно, для построения максимального потока можно воспользоваться методом потенциалов [2, 3], применив его для решения задачи (13), которая эквивалентна задаче о максимальном потоке.

Метод потенциалов рассчитан на решение произвольной задачи вида (13), а у нас задача (13) имеет ряд специальных особенностей. Учёт этих особенностей позволяет разработать для её решения и другие методы.

Замечание. Если $\nu^0 = x_{ts}^0 > 0$, то дуга $(t,s) \in U_{\text{Б}}^0$, т.е. принадлежит оптимальному базису. Следовательно, оптимальное дерево имеет следующую структуру

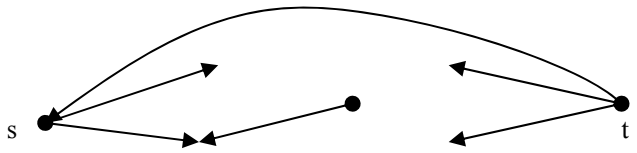


Рис. 4.1

Рассмотрим задачу, двойственную к (12). Задача, двойственная к (12), имеет вид

$$\sum_{(i,j)\in U} d_{ij}\,\delta_{ij} \to \min_{u,\delta},$$
$$u_i - u_j + \delta_{ij} \ge 0\,,\ (i,j)\in U\,, \qquad (14)$$
$$u_t - u_s \ge 1,\ u_s = 0,\ \delta_{ij} \ge 0\,,\ (i,j)\in U\,.$$

Теоретически возможно рассматривать переменные δ_{ij} как некие «переменные-индексаторы». Расшифруем смысл этого названия. Если в оптимальном двойственном решении $\delta_{ij} > 0$, то дуга (i,j) является элементом множества дуг, образующих «узкое место» в сети, т.е. именно эти дуги ограничивают значение максимального потока. Следует заметить, что таких узких мест может быть несколько.

2. Максимальный поток и минимальный разрез. С понятием максимального потока тесно связано понятие разреза.

Рассмотрим любое множество узлов I_*, обладающее свойством: $I_* \subset I$, $s \in I_*$, $t \notin I_*$. По нему построим множество дуг

$$\omega = \omega(I_*) = \{(i,j) \in U: i \in I_*, j \notin I_*\}.$$

Совокупность дуг $\omega(I_*)$ называется *разрезом* сети S, порождённым множеством узлов I_*. Действительно, удаление дуг $\omega(I_*)$ из сети S разрывает все пути, ведущие из s в t. Сеть S становится как бы разрезанной на две части: часть с узлом s и часть с узлом t и нет ни одного пути из s в t.

Число

$$\rho(\omega) = \sum_{(i,j)\in\omega(I_*)} d_{ij} \qquad \textbf{(15)}$$

называется *пропускной способностью разреза* $\omega(I_*)$. Разрез с минимальной пропускной способностью называется *минимальным разрезом*.

Покажем, что каждому разрезу $\omega(I_*)$ соответствует двойственный план, на котором значение целевой функции двойственной задачи (3) равно пропускной способности данного разреза.

Действительно, положим

$$u_i = 0,\ i \in I_*;\ u_i = 1,\ i \in I \setminus I_*. \qquad (16)$$

Числа δ_{ij}, $(i,j) \in U$, определим таким образом, чтобы при заданных числах (16) выполнялись ограничения задачи (14) и значение ее целевой функции было минимальным:

$$\delta_{ij} = 0, \text{ если } u_i - u_j \geq 0;\ \delta_{ij} = -(u_i - u_j), \text{ если } u_i - u_j < 0;\ (i,j) \in U. \qquad (17)$$

Из соотношений (16) и (17) получаем

$$\delta_{ij} = 1, (i,j) \in \omega(I_*);\ \delta_{ij} = 0, (i,j) \in U \setminus \omega(I_*).$$

Следовательно,

$$\sum_{(i,j)\in U} \delta_{ij} d_{ij} = \sum_{(i,j)\in\omega(I_*)} d_{ij} = \rho(\omega(I_*)). \qquad (18)$$

Из теории двойственности мы знаем, что значение прямой целевой функции на любом допустимом плане меньше значения двойственной целевой функции на любом двойственном допустимом плане. Отсюда с учётом (18) делаем вывод, что

$$v \leq \rho(\omega(I_*))$$

для любого допустимого потока v и любого разреза $\omega(I_*)$. Покажем теперь, что существуют такие v^0 и I_*^0, для которых $v^0 = \rho(\omega(I_*^0))$.

Действительно, выше мы отмечали, что задача о максимальном потоке эквивалентна специальной задаче о потоке минимальной стоимости (13). Предположим, что мы решим эту задачу методом потенциалов. В результате получим некоторый оптимальный поток

$$v^0 = x_{ts}^0,\ x_{ij}^0,\ (i,j) \in U,$$

и соответствующий ему оптимальный базис-дерево

$$U_{\text{Б}} \subset \overline{U} = U \cup (t, s).$$

Выше мы также отмечали, что $(t, s) \in U_{\text{Б}}$ и множество $U_{\text{Б}}$ имеет структуру, приведенную на рис. 4.1. Убрав дугу (t, s) из $U_{\text{Б}}$, мы получаем две компоненты связности: содержащую узел

$$s \to \{ I_*^0, U_*^0 \}$$

и содержащую узел

$$t \to \{ I \setminus I_*^0, (U_{\text{Б}} \setminus (t, s)) U_*^0 \}.$$

По дугам $U_{\text{Б}}$ подсчитаем потенциалы, согласно правилам

$$u_i^0 - u_j^0 = c_{ij}, \ (i, j) \in U_{\text{Б}}, \ u_s^0 = 0. \tag{19}$$

В силу специфики данной сети из (19) получаем

$$u_i^0 = 0, \ i \in I_*^0; \ u_i^0 = 1, \ i \in I \setminus I_*^0. \tag{20}$$

Положим

$$\delta_{ij}^0 = 0, \text{если} \, u_i^0 - u_j^0 \geq 0; \ \delta_{ij}^0 = -(u_i^0 - u_j^0), \text{если} \, u_i^0 - u_j^0 < 0, (i, j) \in U. \tag{21}$$

Легко проверить, что построенная совокупность (20), (21) – план задачи (14). Из теории двойственности мы знаем, что этот план – оптимальный двойственный план и имеет место равенство

$$v^0 = \sum_{(i,j) \in U} d_{ij} \delta_{ij}^0. \tag{22}$$

Из (22) с учетом (20), (21) получаем, что

$$v^0 = \sum_{(i,j) \in U} d_{ij} \delta_{ij}^0 = \sum_{\substack{(i,j) \in U, \\ i \in I_*^0, j \in I \setminus I_*^0}} d_{ij} = \rho(\omega(I_*^0)).$$

Таким образом, мы доказали теорему о максимальном потоке и минимальном разрезе, которая формулируется следующим образом.

Теорема. *Величина максимального потока в сети S равна пропускной способности минимального разреза на сети S.*

Значение теоремы о максимальном потоке и минимальном разрезе заключается в том, что максимальный поток в сети можно найти, вычисляя пропускные способности всех разрезов и выбирая среди них минимальный. Конечно, при решении задачи о максимальном потоке этот результат имеет небольшое практическое значение, так как мы не получаем никакой информации о самих величинах x_{ij} дуговых потоков. Однако данный результат важен с теоретической точки зрения и часто используется при разработке сложных потоковых алгоритмов и при проверке решения на оптимальность.

Основываясь на полученных результатах, можно дать следующую интерпретацию двойственной задачи (14): среди всех разрезов сети *S* найти разрез с минимальной пропускной способностью.

3. Метод Форда–Фалкерсона (построение максимального потока). Опишем алгоритм решения задачи о минимальном потоке, известный в литературе как метод Форда–Фалкерсона. Для описания алгоритма надо ввести два понятия: пометки и увеличивающего пути. Пометка узла используется для указания как величины потока, так и источника потока, вызывающего изменение текущей величины потока по дуге, т.е. указывается узел, с помощью которого помечается данный узел.

Увеличивающий путь потока из s в t определяется как связная последовательность прямых и обратных дуг, по которым из s в t можно послать несколько единиц потока. Поток по каждой прямой дуге при этом увеличивается, не превышая её пропускной способности, а поток по каждой обратной дуге уменьшается, оставаясь при этом неотрицательным.

Алгоритм составим из следующих этапов.

Этап 1. Определим начальный поток следующим образом:

$$v=0,\ x_{ij}=0,\ (i,j)\in U.$$

Этап 2. Найдем увеличивающий путь в сети S с заданным потоком. Если такой путь существует, то перейдем к этапу 3. В противном случае STOP: текущий поток является максимальным.

Этап 3. **Увеличим поток, изменив при этом дуговые потоки. Используя новый поток, перейдем опять к этапу 2.**

Опишем алгоритм построения увеличивающего пути. Считаем, что заданы дуговые потоки x_{ij} , $(i,j)\in U$.

Шаг 1. Полагаем $I_c = 1,\ I_t = 1,\ L = \{s\},\ g(s) = 0,\ i = s,\ p_s = 1$. Здесь

I_c – счётчик итераций,

I_t – счётчик меток,

L – множество помеченных узлов,

$g(j)$ – метка узла j,

p_j – вторая метка узла j.

Шаг 2. Рассмотрим непомеченный узел j, для которого существует дуга $(i,j)\in U$ с $x_{ij} < d_{ij}$. Помечаем узел j, полагая $g(j) = i,\ I_t := I_t + 1,\ p_j = I_t$. Так поступаем с каждым непомеченным узлом j, для которого существует дуга $(i,j)\in U,$ с $x_{ij} < d_{ij}$. Помеченные узлы добавляем ко множеству помеченных узлов L.

Шаг 3. Рассмотрим непомеченный узел j, для которого существует дуга $(j,i)\in U$ с $x_{ji} > 0$. Помечаем узел j, полагая $g(j) = -i,\ I_t := I_t + 1,\ p_j = I_t$. Так поступаем с каждым непомеченным узлом j, для которого существует дуга $(j,i)\in U,$ с $x_{ji} > 0$. Помеченные узлы добавляем ко множеству помеченных узлов.

Шаг 4. Если узел t помечен, то STOP: увеличивающий путь найден. Переходим к алгоритму восстановления пути и увеличения потока. Если узел t не помечен, то переходим к шагу 5.

Шаг 5. Положим $I_c := I_c + 1$. Найдём помеченный узел j_0 с меткой $p_{j_0} = I_c$.

Если такой узел найден, то полагаем $i := j_0$ и возвращаемся к шагу 2. Если такого узла найти не удалось, то не существует увеличивающего пути из s в t. STOP.

Опишем алгоритм восстановления пути и увеличения потока.

По условию узел t помечен. Пусть $q(t) = i_1$, следовательно, из узла i_1 попадаем в узел t по прямой дуге (i_1, t). Полагаем

$$\alpha_1 = d_{i_1 t} - x_{i_1 t}.$$

Рассмотрим метку узла i_1. Предположим, что $q(i_1) = i_2$. Полагаем

$$\alpha_2 := min\{\alpha_1, d_{i_2 i_1} - x_{i_2 i_1}\}.$$

Если $q(i_1) = -i_2$, то полагаем

$$\alpha_2 := min\{\alpha_1, x_{i_1 i_2}\}$$

(в последнем случае в увеличивающем пути дуга (i_1, i_2) является обратной). И так далее, пока ни получим на некотором шаге

$$q(i_m) = \pm s,\ \alpha_m := \begin{cases} \min\{\alpha_{m-1},\ d_{si_m} - x_{si_m}\} \text{ при } q(i_m) = s, \\ \min\{\alpha_{m-1},\ x_{i_m s}\} \text{ при } q(i_m) = -s. \end{cases}$$

Таким образом, увеличивающий путь проходит через узлы s, i_m, i_{m-1}, …, i_2, i_1, t.

Изменяем дуговые потоки на дугах этого пути: дуговые потоки на прямых дугах увеличиваем на α_m, дуговые потоки на обратных дугах уменьшаем на α_m, поток ν увеличиваем на α_m. STOP.

Пример. Рассмотрим сеть, приведенную на рис. 4.2. Пропускные способности дуг d_{ij} указаны на дугах «жирными» цифрами. Пусть на данной сети имеется допустимый поток $x = (x_{ij}, (i, j) \in U)$ величины $\nu = 9$. Дуговые потоки x_{ij} указаны на дугах «нежирными» цифрами. Проверим, является ли данный поток максимальным, и если нет, то построим новый поток, для которого $\bar{\nu} > \nu = 9$. Для этого осуществим итерации метода Форда–Фалкерсона.

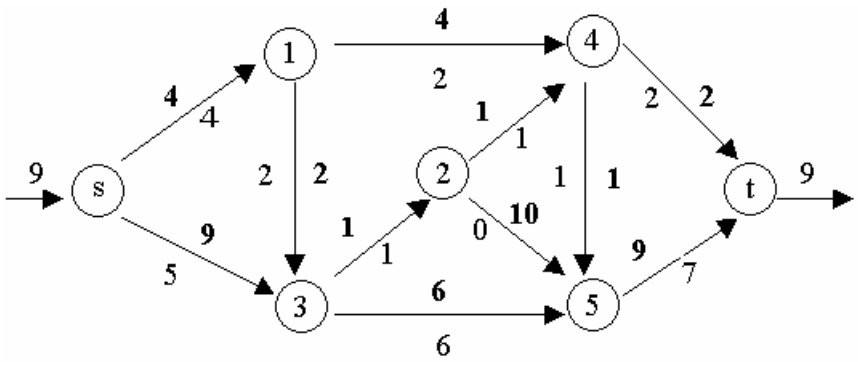


Рис. 4.2

Построение увеличивающего пути

Итерация 1

Шаг 1. Полагаем

$$I_c = 1, I_t = 1, L = \{s\}, \ g(s) = 0, \ i = s, \ p_s = 1.$$

Шаг 2. Из узла $i = s$ есть только одна дуга $(s,3) \in U$, для которой $x_{s3} = 5 < 9 = d_{s3}$. Помечаем узел 3, полагая $I_t = 2, L = \{s,3\}, \ g(3) = s, \ p_3 = 2.$

Шаг 3. Для узла $i = s$ нет ни одной дуги $(j,i) \in U$, для которой $x_{ji} > 0.$ Переходим к шагу 4.

Шаг 4. Поскольку $t \notin L$, то переходим к шагу 5.

Шаг 5. Полагаем $I_c := I_c + 1$, и, поскольку $p_3 = I_c$, то полагаем $j_0 = 3.$ Переходим к шагу 2 итерации 2, положив $i = j_0$.

Итерация 2

Шаг 2. Из узла $i = 3$ нет дуг $(i,j) \in U$, для которых $x_{ij} < d_{ij}$. Переходим к шагу 3.

Шаг 3. Для узла $i = 3$ есть одна дуга $(1,3) \in U$, для которой $x_{13} > 0.$ Помечаем узел 1, полагая $I_t = 3, \ L = \{s,3,1\}, \ g(1) = -3, \ p_1 = 3.$

Шаг 4. Поскольку $t \notin L$, то переходим к шагу 5.

Шаг 5. Полагаем $I_c := I_c + 1 = 3$ и находим узел $j_0 \in L,$ для которого $p_{j_0} = I_c$. На данной итерации $j_0 = 1.$ Переходим к шагу 2 итерации 3, полагая $i = j_0 = 1.$

Итерация 3

Шаги 2 - 3. С помощью узла $i = 1$ помечаем узел 4, полагая

$I_t = 4, L = \{s,3,1,4\}, g(4) = 1, p_4 = 4.$

Шаг 4. Узел $t \notin L$, переходим к шагу 5.

Шаг 5. Полагаем $I_c := I_c + 1 = 4$ и находим узел $j_0 \in L,$ для которого $p_{j_0} = I_c$. На данной итерации $j_0 = 4$. Переходим к шагу 2 итерации 4, полагая $i = j_0 = 4.$

Итерация 4

Шаги 2 - 3. С помощью узла $i = 4$ помечаем узел 2, полагая

$$I_t = 5, L = \{s,3,1,4,2\}, g(2) = -4, p_2 = 5.$$

Шаг 4. Узел $t \notin L$, переходим к шагу 5.

Шаг 5. Полагаем $I_c := I_c + 1 = 5, \ j_0 = 2$. Переходим к шагу 2 итерации 5, заменив i на $j_0 = 2$.

Итерация 5

Шаги 2 - 4. С помощью узла $i = 2$ помечаем узел 5, полагая

$$I_t = 6, L = \{ s, 3, 1, 4, 2, 5 \}, g(5) = 2, p_5 = 6.$$

Переходим к шагу 5.

Шаг 5. Полагаем $I_c := I_c + 1 = 6,\ j_0 = 5$. Переходим к шагу 2 итерации 6, заменив *i* на $j_0 = 5$.

Итерация 6

Шаги 2 - 4. С помощью узла $i = 5$ помечаем узел *t*, полагая $L = \{s, 3, 1, 4, 2, 5, t\}, g(t) = 5$. Узел $t \in L$. STOP – увеличивающий путь построен. Значит, имеющийся поток можно увеличить.

Переходим к алгоритму восстановления пути и увеличения потока. Применяя **алгоритм восстановления пути и увеличения потока**, получаем

$$q(t) = 5, \quad \alpha_1 = d_{5t} - x_{5t} = 2,$$
$$q(5) = 2, \quad \alpha_2 = \min\{ \alpha_1, 10 \} = 2,$$
$$q(2) = -4, \quad \alpha_3 = \min\{ \alpha_2, 1 \} = 1,$$
$$q(4) = 1, \quad \alpha_4 = \min\{ \alpha_3, 2 \} = 1,$$
$$q(1) = -3, \quad \alpha_5 = \min\{ \alpha_4, 2 \} = 1,$$
$$q(3) = s, \quad \alpha_6 = \min\{ \alpha_5, 1 \} = 1.$$

Увеличиваем поток вдоль построенного увеличивающего пути

$$U_n = \{ (s,3), (1,3), (1,4), (2,4), (2,5), (5,t) \},$$

изменяя дуговые потоки на дугах $(i, j) \in U$: по правилу

$$\bar{x}_{s3} = x_{s3} + \alpha_6 = 6,\ \bar{x}_{13} = x_{13} - \alpha_6 = 1,\ \bar{x}_{14} = x_{14} + \alpha_6 = 3,$$

$$\bar{x}_{24} = x_{24} - \alpha_6 = 0,\ \bar{x}_{25} = x_{25} + \alpha_6 = 1,\ \bar{x}_{5t} = x_{5t} + \alpha_6 = 8,$$

$$\bar{x}_{ij} = x_{ij},\ (i, j) \in U \therefore U_n,\ \bar{v} = v + \alpha_6.$$

Сеть S с новым потоком $\bar{x}$ приведена на рис. 4.3.

Поток $\bar{x}$ является максимальным. Действительно, применив к сети *S* с потоком $\bar{x}$ алгоритм построения увеличивающего пути, мы можем пометить только узлы $L = \{ s, 3, 1, 4, \}$. После чего на шаге 5 не удается найти узел j_0, для которого $p_{j_0} = I_c$. Легко проверить, что множество узлов *L* задают разрез

$$\omega(L) = \{ (3,2), (3,6), (4,5), (2,t) \},$$

пропускная способность которого равна $\rho(\omega(L)) = 1+6+1+2 = 10 = \bar{v}$.

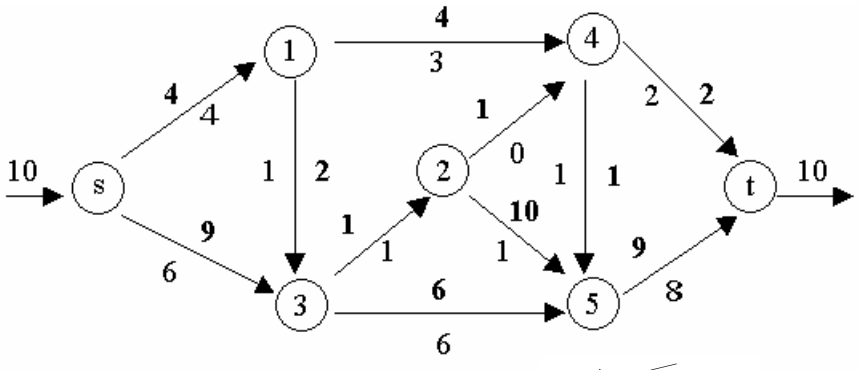


Рис. 4.3

Согласно теореме, $\bar{x}, \bar{v}$ – максимальный поток, $\omega(L)$ – минимальный разрез.

§ 3. Задача о назначениях

1. Постановка задачи. Имеется n видов работ и n исполнителей, каждый из которых может выполнять любую работу. При назначении j-го работника на i-ю работу затраты предприятия равны c_{ij}. Требуется на каждую работу назначить по исполнителю таким образом, чтобы общие расходы предприятия были минимальными. Математическая модель данной задачи имеет вид

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to min;$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i=\overline{1,n};\quad \sum_{i=1}^{n} x_{ij} = 1,\ j=\overline{1,n}; \tag{23}$$

$$x_{ij} = 0 \vee 1,\ i=\overline{1,n},\ j=\overline{1,n}.$$

Здесь $x_{ij} = \begin{cases} 0, \text{если на } i\text{-ю работу не назначается } j\text{-й работник}, \\ 1, \text{если на } i\text{-ю работу назначается } j\text{-й работник}. \end{cases}$

Задача (23) – частный случай транспортной задачи в матричной форме. Дополнительное требование целочисленности не является существенным (в данной задаче!), так как ранее мы отмечали, что если в транспортной задаче параметры a_i, b_i, d_{ij} – целые, то существует целочисленное решение задачи (23) и это решение можно построить с помощью классического метода потенциалов. Следовательно, для решения задачи (23) можно использовать классический метод потенциалов. Отметим, однако, что на каждой итерации метода потенциалов мы будем иметь вырожденный базисный план. Действительно, в задаче (23) каждая компонента плана может принимать только критические значения 0 или 1, и, следовательно, согласно определению, любой допустимый план будет «полностью» вырожденным.

Можно заменить условия

$$0 \le x_{ij} \le 1,\ i=\overline{1,n},\ j=\overline{1,n} \tag{24}$$

на условия

$$0 \le x_{ij},\ i=\overline{1,n},\ j=\overline{1,n}. \tag{25}$$

Но и в этом случае каждый базисный план будет вырожденным. Действительно, если мы начнём решение с целочисленного плана, то и на всех последующих итерациях у нас будет целочисленный план. С учётом этого заключаем, что на каждом плане только n компонент могут быть отличны от 0, а остальные равны 0. Базис состоит из $2n-1$ элементов. Следовательно, среди базисных компонент будет $n-1$ нулевых. Значит, и в случае использования ограничений (25) каждый базисный план будет вырожденным.

Хорошо известно, что вырожденность отрицательно сказывается на эффективности метода потенциалов (симплекс-метода) и при вырожденности велика вероятность зацикливания. В силу отмеченных причин нельзя ожидать, что метод потенциалов «в чистом виде» будет эффективен для решения задачи (23). Следовательно, нужны другие методы, учитывающие ее специфику. Рассмотрим один из таких методов, получивший название «венгерский метод», так как в нем используются результаты венгерского математика Эгервари.

2. Венгерский метод. По параметрам задачи (23) составим $(n \times n)$-матрицу стоимостей $C = (c_{ij},\ j = \overline{1,n},\ i = \overline{1,n})$. Предположим, что каждый элемент *i*-й строки складывается с действительным числом γ_i, а каждый элемент *j*-го столбца складывается с действительным числом δ_j. В результате такого преобразования матрицы C будет получена новая матрица стоимостей D с коэффициентами

$$d_{ij} = c_{ij} + \gamma_i + \delta_j,\ i = \overline{1,n},\ j = \overline{1,n}. \tag{26}$$

Из (23), (26) получаем

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_i x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n}\delta_j x_{ij} =$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} - \sum_{i=1}^{n}\gamma_i \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n}\delta_j \sum_{i=1}^{n} x_{ij} = \sum_i\sum_j d_{ij}x_{ij} - \underbrace{\sum_{i=1}^{n}\gamma_i - \sum_{j=1}^{n}\delta_j}_{\text{const}}.$$

Отсюда следует, что при ограничениях задачи о назначениях минимизация функции $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$ эквивалентна минимизации функции $\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij}$ (здесь γ_i, δ_i – любые действительные числа). Это свойство задачи (23) и составляет основу излагаемого ниже алгоритма.

Общая схема метода следующая:

1. Из элементов каждой строки и каждого столбца матрицы стоимостей вычитаются их наименьшие элементы.

2. Ведётся поиск допустимого плана задачи (23), единичным элементам которого соответствуют нулевые элементы модифицированной матрицы стоимостей, т.е. строится «нулевое» назначение.

3. Если такой допустимый план существует, то он является оптимальным планом назначений. Если такого плана не существует, то матрица стоимостей модифицируется ещё раз с целью получить в ней большее число нулевых элементов.

Прокомментируем кратко эти три шага алгоритма.

Шаг 1. *Редукция строк и столбцов.* Цель данного шага состоит в получении максимально возможного числа нулей в матрице стоимостей. Для этого можно последовательно из всех элементов каждой строки вычесть по минимальному элементу, затем в полученной матрице из каждого столбца вычесть по минимальному элементу, найденному среди элементов данного столбца. Заменить исходную матрицу стоимостей на новую.

Шаг 2. *Определение назначений.* Если после выполнения процедуры редукции в каждой строке и в каждом столбце матрицы стоимостей можно выбрать по одному нулевому элементу так, что соответствующее этим элементам решение будет допустимым планом, то данное назначение будет оптимальным. Действительно, стоимость построенного назначения равна нулю. Поскольку все элементы текущей модифицированной матрицы стоимостей неотрицательные, то стоимость любого другого допустимого назначения будет больше либо равной нулю. Отсюда заключаем, что построенное «ненулевое» назначение является оптимальным. Если назначений нулевой стоимости для редуцированной матрицы найти нельзя, то данная матрица стоимостей подлежит дальнейшей модификации (см. шаг 3).

Шаг 3. *Модификация редуцированной матрицы.* Эта процедура нацелена на получение новых нулей в матрице стоимостей. Из имеющейся матрицы стоимостей вычеркнем *минимально возможное* число строк и столбцов, содержащих нулевые элементы. Среди невычеркнутых элементов матрицы найдём минимальный элемент. Ясно, что он положительный. Пусть он равен $\alpha > 0$.

Если значение α вычесть из всех (вычеркнутых и невычеркнутых) элементов старой редуцированной матрицы, то среди вычеркнутых элементов могут появиться отрицательные (причём минимальные из них равны $-\alpha$ и стоят на месте старых нулей), а среди невычеркнутых элементов не будет отрицательных, но появится хотя бы один нулевой элемент.

Чтобы избавится от отрицательных элементов в вычеркнутых элементах, поступим следующим образом: к элементам каждой вычеркнутой строки и к элементам каждого вычеркнутого столбца добавим по числу α. Отметим, что в результате последней процедуры величина α будет прибавляться дважды к вычеркнутым элементам, стоящим на пересечении вычеркнутых строк и столбцов.

Кроме того, можно показать, что:

1) как и раньше, все отрицательные элементы будут преобразованы в нулевые или положительные элементы;

2) полученная матрица является редуцированной матрицей по отношению к исходной матрице стоимостей C, т.е. она может быть получена из C в

результате преобразования $c_{ij} \to d_{ij} = c_{ij} + \gamma_i + \delta_j$, где γ_i, δ_j – некоторые действительные числа;

3) в результате выполнения данной процедуры новая редуцированная матрица стала содержать больше нулей, расположенных вне строк и столбцов, соответствующих ненулевым элементам текущего неоптимального плана (построенного на шаге 2). Отметим, что в общем случае общее число нулей новой редуцированной матрицы может и уменьшиться!

Таким образом, при реализации венгерского метода надо осуществить две основные процедуры – шаг 2 и шаг 3.

Опишем эти процедуры более детально.

Процедура, используемая на шаге 2. На основе текущей матрицы стоимостей $C = (c_{ij},\ i = \overline{1,n},\ \ j = \overline{1,n})$ сформируем сеть $S = \{I, U\}$ со множеством узлов $I = \{s, t\} \cup N \cup N_*$, где $N = \{1, 2, ..., n\}$, $N_* = \{n+1, n+2, ..., 2n\}$ и множеством дуг $U = U_1 \cup U_0 \cup U_*$, где $U_1 = \{(s,i), i \in N\}$, $U_* = \{(i,t), i \in N_*\}$, $U_0 = \{(i,j), i \in N,\ \ j \in N_* : c_{i\,j-n} = 0\}$. На сети $S = \{I, U\}$ с пропускными способностями дуг

$$d_{ij} = 1,\ (i,j) \in U_1 \cup U_*; \quad d_{ij} = \infty,\ (i,j) \in U_0 \tag{27}$$

решим задачу о максимальном потоке из узла *s* в узел *t*. Для решения данной задачи можно использовать метод Форда–Фалкерсона (см. § 2).

Пусть

$$v^0, y_{ij}^0,\ (i,j) \in U, \tag{28}$$

– максимальный поток в сети S и I_*, $I_* \subset I$, $s \in I_*$, $t \notin I_*$ – множество узлов, помеченных на последней итерации метода Форда–Фалкерсона.

Если $v^0 = n$, то исходная задача о назначениях решена. Оптимальный план
назначений

$$x^0 = (x_{ij}^0,\ i = \overline{1,n},\ \ j = \overline{1,n})$$

строится по правилу

$$x_{ij}^0 = 1, \text{ если } (i, j+n) \in U \text{ и } y_{ij+n}^0 = 1; \quad x_{ij}^0 = 0 \text{ – в противном случае,} \tag{29}$$

$$i = \overline{1,n},\ \ j = \overline{1,n}.$$

Алгоритм прекращает свою работу.

Если $v^0 < n$, то в текущей матрице стоимостей нельзя осуществить полное “нулевое” назначение. Переходим к операциям шага 3.

Процедура, используемая на шаге 3. Пусть $C = (c_{ij},\ i = \overline{1,n},\ \ j = \overline{1,n})$ – текущая матрица стоимостей; $v^0, y^0_{ij},\ \ (i,j) \in U$ – максимальный поток и $I_* \subset I$ – множество помеченных узлов, построенных на шаге 2. Положим

$$N^{(1)} = \{ i \in N : i \in I_* \},\ \ N^{(2)} = \{ i \in N : i + n \in I_* \}.$$

Используя результаты § 2, нетрудно показать, что по построению выполняются следующие свойства:

а) если $(i, j+n) \in U_0$ и $i \in N^{(1)}$, то $j \in N^{(2)}$;

б) если $(i, j+n) \in U_0$ и $y^0_{i\,j+n} > 0$, то $i \in N^{(1)}$, $j \in N^{(2)}$ либо $i \notin N^{(1)}$, $j \notin N^{(2)}$;

в) если $v^0 < n$, то $N^{(1)} \neq \varnothing,\ \ N^{(2)} \neq N$.

Найдем число

$$\alpha = \min_{\substack{i \in N^{(1)}, \\ j \in N \setminus N^{(2)}}} c_{ij}. \tag{30}$$

Из свойств «а» – «в» и правил построения множества U_0 следует, что

$$\alpha > 0.$$

Изменим матрицу стоимостей следующим образом. От всех элементов строк с номерами $i \in N^{(1)}$ отнимем число α. Ко всем элементам столбцов с номерами $j \in N^{(2)}$ прибавим число α. В результате получим новую матрицу стоимостей $\overline{C} = (\overline{c}_{ij},\ i = \overline{1,n},\ j = \overline{1,n})$ с коэффициентами:

$$\begin{gathered} \overline{c}_{ij} = c_{ij}, \qquad \text{если} \quad i \in N^{(1)}, \quad j \in N^{(2)} \ \text{ либо } i \notin N^{(1)}, \ \ j \notin N^{(2)}; \\ \overline{c}_{ij} = c_{ij} - \alpha, \ \ \text{если} \quad i \in N^{(1)}, \ \ j \notin N^{(2)}; \\ \overline{c}_{ij} = c_{ij} + \alpha, \ \ \text{если} \quad i \notin N^{(1)}, \quad j \in N^{(2)}. \end{gathered} \tag{31}$$

Используя новую матрицу стоимостей $\overline{C}$, переходим к шагу 2.

3. Конечность алгоритма. Из соотношений (31) следует, что матрица $\overline{C}$, построенная на шаге 3, обладает свойствами:

1) $\overline{c}_{ij} \geq 0,\ i = \overline{1,n},\ j = \overline{1,n}$;

2) матрица $\overline{C}$ является редуцированной матрицей по отношению к матрице C;

3) $\overline{c}_{ij} = c_{ij} = 0,\ \ \text{если} \ \ y^0_{i\,j+n} > 0,\ \ (i, j+n) \in U_0$;

4) найдется такой элемент (i_*, j_*), что $\overline{c}_{i_* j_*} = 0,\ c_{i_* j_*} = \alpha > 0,\ i_* \in N^{(1)},$ $j_* \notin N^{(2)}$.

Из 3-го и 4-го свойств следует, что при каждом последующем использовании процедуры шага 2 либо увеличивается множество помеченных узлов I_*,

либо величина максимального потока по дугам с нулевой стоимостью увеличивается хотя бы на 1. Поскольку $I_* \subset I$, $|I| = 2n+2$ и $v^0 \le n$, то очевидно, что через конечное число повторений шага 2 мы придем к ситуации, когда будет найден максимальный поток (28) с $v^0 = n$. Согласно алгоритму, это означает, что исходная задача решена. Оптимальный план назначений строится по правилам (29).

Замечания: 1. Из свойств матрицы $\overline{C}$ следует, что при решении задачи о максимальном потоке на шаге 2 текущей итерации в качестве начального допустимого потока можно брать оптимальный поток, полученный на шаге 2 предыдущей итерации.

2. На шаге 2 алгоритма используется метод Форда–Фалкерсона для нахождения максимального потока в сети *S*, построенной по текущей матрице стоимостей. Сеть *S* имеет специальную структуру. Учет этой специфики позволяет разработать упрощенные табличные приемы реализации метода Форда–Фалкерсона, позволяющие сократить объем вычислений.

Пример. Начальная матрица стоимостей имеет вид

$$\begin{pmatrix} 2 & 10 & 9 & 7 \\ 15 & 4 & 14 & 8 \\ 13 & 14 & 16 & 11 \\ 4 & 15 & 13 & 19 \end{pmatrix}.$$

Шаг 1. После просмотра строк и соответствующих преобразований получаем

$$\begin{pmatrix} 0 & 8 & 7 & 5 \\ 11 & 0 & 10 & 4 \\ 2 & 3 & 5 & 0 \\ 0 & 11 & 9 & 15 \end{pmatrix}.$$

После просмотра столбцов и соответствующих преобразований получаем

$$\begin{pmatrix} 0 & 8 & 2 & 5 \\ 11 & 0 & 5 & 4 \\ 2 & 3 & 0 & 0 \\ 0 & 11 & 4 & 15 \end{pmatrix}.$$

Шаг 2. Используя последнюю матрицу, сформируем сеть *S,* приведенную на рис. 4.4.

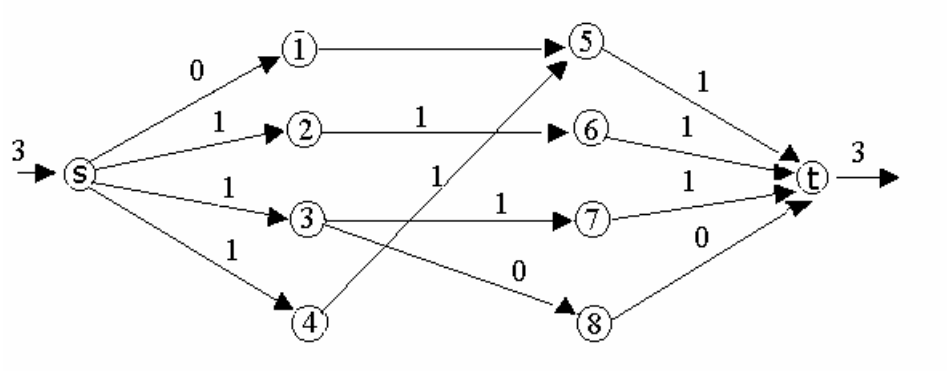


Рис. 4.4

На сети S с пропускными способностями (27) решим задачу о максимальном потоке. Максимальный поток (28) приведен на дугах S на рис. 4.4. Множество помеченных узлов I_* состоит из узлов $I_*=\{s,\ 1,\ 4,\ 5\}$. Поскольку $v^0 = 3 < n = 4$, то переходим к шагу 3.

Шаг 3. Построим множество $N^{(1)} = \{1, 4\}$ и $N^{(2)} = \{1\}$. Подсчитаем число α (30) для нашего примера

$$\alpha = min\{\ c_{1j},\ i = \overline{2,4},\ c_{nj},\ j = \overline{2,4}\} = min\{8,\ 2,\ 5,\ 11,\ 4,\ 15\} = 2.$$

Используя α, $N^{(1)}$ и $N^{(2)}$, построим новую матрицу стоимостей $\overline{C}$ по правилам (31). Матрица $\overline{C}$ имеет вид

$$\begin{pmatrix} 0 & 6 & 0 & 3 \\ 13 & 0 & 5 & 4 \\ 4 & 3 & 0 & 0 \\ 0 & 9 & 2 & 13 \end{pmatrix}.$$

Переходим к шагу 2 с новой матрицей $\overline{C}$.

Шаг 2. Сеть S, соответствующая $\overline{C}$, приведена на рис. 4.5.

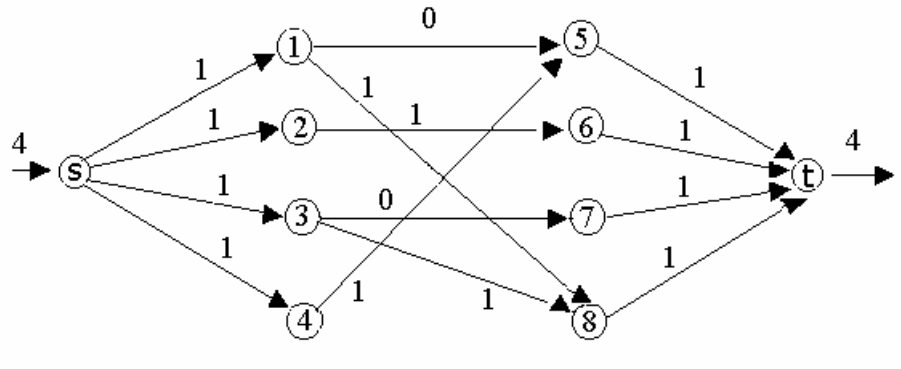


Рис. 4.5

На этом же рисунке приведен и максимальный поток. Величина этого потока равна $v^0 = 4 = n$. Используя правила (29) и найденный максимальный поток, определим оптимальное назначение:

$$x_{13}^0 = 1,\ x_{22}^0 = 1,\ \ x_{34}^0 = 1,\ \ x_{41}^0 = 1,$$
$$x_{ij}^0 = 0,\ i = \overline{1,n},\ j = \overline{1,n};\ (i,j) \in \{(1,3),\ (2,\ 2),(3,4),(4,1)\}.$$

Таким образом, 3-й работник назначается на 1-ю работу, 2-й работник – на 2-ю работу, 4-й работник – на 3-ю работу и 1-й работник назначается на 4-ю работу.

§ 4. Задача коммивояжера

1. Постановка задачи. Математическая модель задачи коммивояжера имеет вид

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to min\ (c_{ii} = \infty), \tag{32}$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = \overline{1,n}\ \ \text{(отъезд из города } i\text{)}, \tag{33}$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = \overline{1,n}\ \ \text{(прибытие в город } j\text{)}, \tag{34}$$

$$0 \le x_{ij} \le 1,\ i = \overline{1,n};\ j = \overline{1,n}, \tag{35}$$

множество $U_* = \{(i,j):\ i = \overline{1,n},\ j = \overline{1,n}; x_{ij} = 1\}$ есть единственный цикл.

(36)

Условие (36) отличает задачу коммивояжера от задачи о назначениях. Если отбросим (36), т.е. будем рассматривать задачу (32) – (34), то получим задачу о назначениях.

Равенство $x_{ij} = 1$ означает, что коммивояжер из города i идёт в город j, равенство $x_{ij} = 0$ означает, что дуга (i,j) не включается в маршрут коммивояжера.

Существует много методов решения задачи (32) – (36). Мы рассмотрим два метода, являющихся различными модификациями метода ветвей и границ.

2. Первая модификация (метод исключения подциклов). Эта модификация в наименьшей степени отличается от метода ветвей и границ, рассмотренного нами ранее. В начале итерации t известны верхняя граница (оценка) r_0^t оптимального значения целевой функции и соответствующий ей маршрут. Можно принять r_0^1 равным достаточно большому числу, скажем, сумме $(c_{12} + c_{23} + \ldots + c_{n1})$, соответствующей маршруту $1 \to 2 \to 3 \to \ldots \to n \to 1$. Кроме того, имеется основной список, содержащий ряд задач о назначениях. Все задачи о назначениях имеют вид (32) – (35), но отличаются друг от друга тем, что в них различные величины c_{ij} равны ∞. Равенство $c_{ij} = \infty$ означает, что дуга (i,j) исключается из маршрута.

На первой итерации основной список состоит из одной (исходной) задачи (32) – (35).

На итерации t выполняются следующие шаги.

Шаг 1. Прекратим вычисления, если основной список пуст: зафиксированный маршрут является оптимальным. В противном случае выберем одну задачу, вычеркнув её из основного списка.

Шаг 2. Решим выбранную задачу о назначениях. Если оптимальное значение целевой функции (которое может быть равно ∞, что означает, что её ограничения несовместны!) больше или равно r_0^t, то оценку не меняем: $r_0^{t+1} = r_0^t$ и возвращаемся к шагу 1. В противном случае, т.е. если значение целевой функции задачи о назначениях меньше r_0^t, перейдём к шагу 3.

Шаг 3. Если полученное оптимальное решение выбранной задачи о назначениях является одним циклом, то зафиксируем это решение и положим r_0^{t+1} равным оптимальному значению целевой функции рассматриваемой задачи о назначениях. Перейдем к шагу 1. В противном случае, т.е. если решение задачи о назначениях образует несколько подциклов, перейдем к шагу 4.

Шаг 4. Остановимся в полученном оптимальном решении задачи о назначениях на подцикле, содержащем минимальное количество дуг. Каждой дуге (*i, j*) из выбранного подцикла поставим в соответствие задачу о назначениях, внеся её в основной список и приняв соответствующее значение $c_{ij} = \infty$, а все остальные коэффициенты оставим теми же, что и в задаче, выбранной на шаге 1. Примем $r_0^{t+1} = r_0^t$ и вернемся к шагу 1.

Пример 1. Рассмотрим задачу коммивояжера со следующей матрицей маршрутов:

	1	**2**	**3**	**4**	**5**
1	∞	10	25	25	10
2	1	∞	10	15	2
3	8	9	∞	20	10
4	14	10	24	∞	15
5	10	8	25	27	∞

Дерево, соответствующее данному примеру, приведено на рис. 4.6.

Оценка $r_o^{t=1} = 65$ соответствует
$1 \to 2 \to 3 \to 4 \to 5 \to 1$

НАЧАЛО

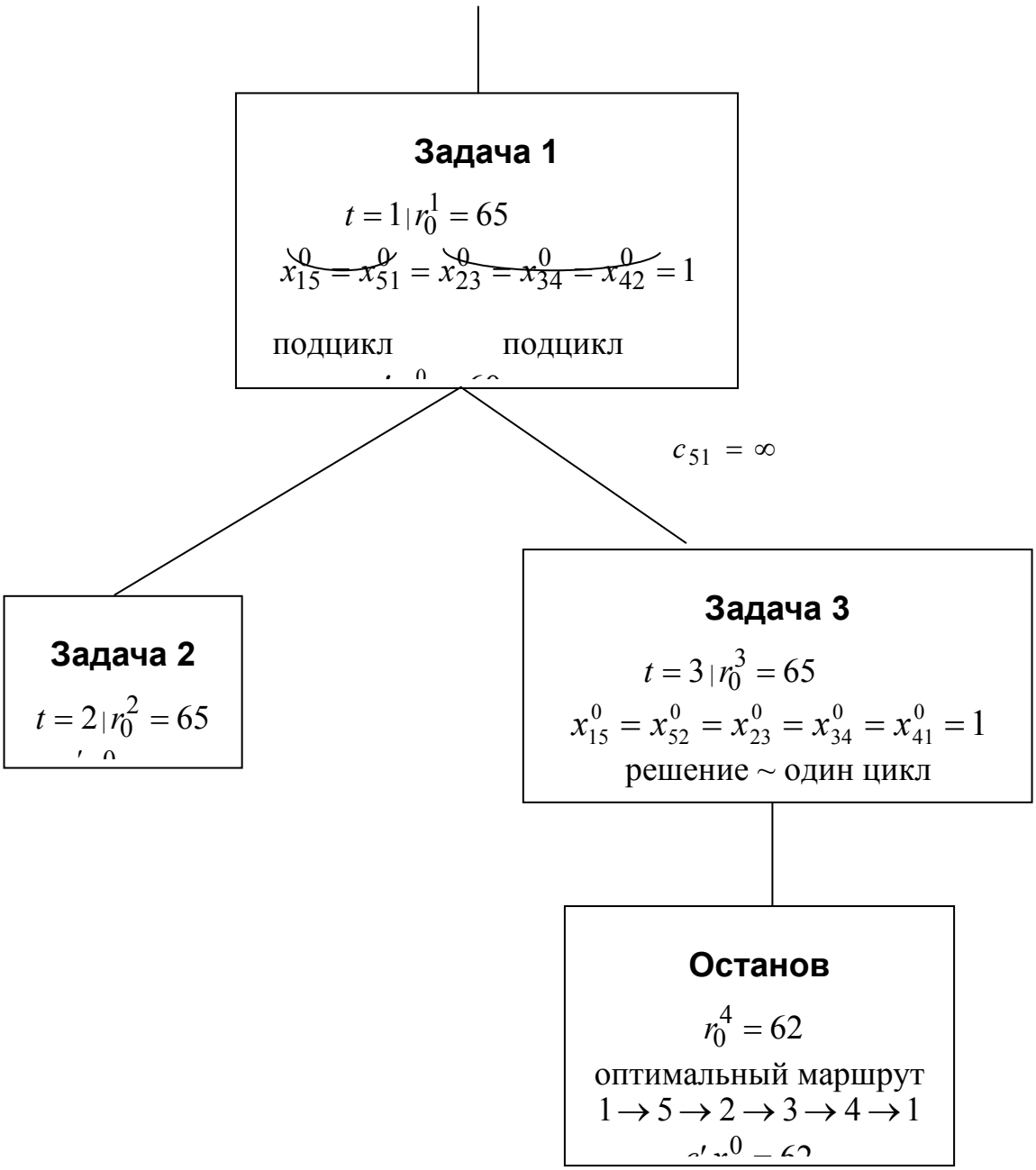


Рис. 4.6

3. Вторая модификация (метод задания маршрутов). Изложенный выше метод позволяет ограничить число просматриваемых вершин дерева в методе ветвей и границ, что достигается за счёт вычисления эффективной нижней оценки (границы) целевой функции для любого цикла, порождаемого каждой задачей. Но для получения этой оценки приходится находить оптимальное решение задачи о назначениях.

Покажем теперь, как можно вычислять оценки более простым способом. Однако цена, которую приходится платить за такое упрощение, определяется тем, что в новом алгоритме нужно исследовать большее число ветвей соответствующего дерева.

В начале каждой итерации t известна верхняя оценка r_0^t оптимального значения целевой функции. Значение r_0^t на первой итерации можно определить по тем же правилам, что и в предыдущем алгоритме. Кроме того, имеется основной список задач, в которых некоторое подмножество значений c_{ij} изменено и принято равным ∞, а некоторое подмножество x_{kp} принято равным 1. Среди значения $x_{kp} = 1$ отсутствуют наборы, образующие подциклы. Отметим, что равенство $c_{ij} = \infty$ означает, что дуга (i, j) исключается из маршрута, а равенство $x_{kp} = 1$ означает, что дуга (k, p) обязательно включается в маршрут.

На первой итерации основной список включает две задачи: в одной из них значение выбранного (выбираем произвольно) c_{ij} изменено на ∞ (это означает, что маршрут $i \to j$ запрещён), в другой – соответствующая переменная $x_{ij} = 1$ (это означает, что маршрут $i \to j$ задан), а $c_{ji} = \infty$ (полагая $c_{ji} = \infty$, запрещают маршрут $j \to i$, предотвращая образование подцикла $i \to j \to i$).

Рассмотрим любую задачу из основного списка и попытаемся вычислить для неё нижнюю оценку оптимального значения целевой функции для любого цикла, содержащего заданное подмножество дуг с $x_{ij} = 1$. Существует много способов вычисления таких оценок. В целом, чем больше нижняя оценка, тем меньшее число ветвей приходится исследовать.

Приведём один простой, но достаточно эффективный способ вычисления нижних границ. В основе этого способа лежат те же идеи, что использовались нами при обосновании венгерского метода решения задачи о назначениях. Прежде всего будем считать, что из матрицы $C = (c_{ij},\ i=\overline{1,n};\ j=\overline{1,n})$, соответствующей рассматриваемой задаче, вычеркнуты строки k и столбцы p, если задано, что $x_{kp} = 1$. Ясно, что указанная оценка должна быть, по крайней мере, равной сумме c_{ij} при заданных $x_{ij} = 1$, плюс сумма наименьших c_{ij} в каждой из невычеркнутых строк.

Эту оценку можно (и должно) ещё увеличить. Для этого вычитается минимальный коэффициент c_{ij} в каждой невычеркнутой строке из всех оставшихся c_{ij} этой строки. Далее к полученной выше оценке добавляется сумма минимальных чисел, найденных в каждом невычеркнутом столбце среди «уменьшенных» расстояний.

Пример 2, иллюстрирующий эту процедуру, приведен на рис. 4.7.

$x_{23} = 1,\ c_{23} = 10$

min по строкам

↓

	1	2	3	4	5	
1	∞	10		25	10	10
2						
3	8	∞		20	10	8
4	14	10		∞	15	10
5	10	8		27	∞	8

Матрица уменьшенных расстояний

	1	2	3	4	5	
1	∞	0		15	0	
2						
3	0	∞		12	2	
4	4	0		∞	5	
5	2	0		19	∞	
	0	0		12	0	← min по столбцам

Оценка равна c_{23}+(10+8+10+8)+(0+0+12+0) = 58.

Рис. 4.7

Ясно, что самую «хорошую» нижнюю оценку мы бы получили, если бы решили до конца задачу о назначениях, соответствующую невычеркнутой части таблицы (вычеркнутая часть соответствует закреплённой части маршрута). Однако такая оценка потребовала бы значительных вычислительных затрат.

На итерации t выполняются следующие шаги.

Шаг 1. Прекратить вычисления, если основной список пуст: зафиксированный цикл является оптимальным маршрутом. В противном случае выбрать одну задачу и вычеркнуть её из основного списка. Перейти к шагу 2.

Шаг 2. Определить нижнюю оценку целевой функции для любого цикла, порождённого выбранной задачей. Если нижняя оценка больше или равна r_0^t, то положить $r_0^{t+1} = r_0^t$ и перейти к шагу 1. В противном случае перейти к шагу 3.

Шаг 3. Если зафиксированные переменные $x_{ij} = 1$ в выбранной задаче образуют один цикл, то зафиксируем его; положим r_0^{t+1} равным длине получен-

ного цикла; вернёмся к шагу 1. В противном случае (т.е. если дуги с $x_{ij}=1$ не образуют цикла) перейдём к шагу 4.

Шаг 4. Попытаемся найти такую дугу (i_0, j_0):

1) которая до текущего момента не принадлежала множеству дуг с фиксированными значениями $x_{ij}=1$;

2) для которой текущее $c_{i_0 j_0} < \infty$;

3) во множестве дуг с фиксированными значениями нет дуг вида (i, j_0), (i_0, j);

4) добавление дуги (i_0, j_0) ко множеству дуг (i, j) с $x_{ij}=1$ не образует подцикла, т.е. цикла с количеством дуг меньше, чем *n*.

Если удаётся найти такую дугу (i_0, j_0), то в основной список вносим две новые задачи. Каждая из этих задач идентична задаче, выбранной на шаге 1, за исключением лишь того, что в одну из них надо внести изменение, положив $c_{i_0 j_0}=\infty$, в другую – условие $x_{i_0 j_0}=1$ и изменение $c_{j_0 i_0}=\infty$. Положим $r_0^{t+1}=r_0^t$ и вернёмся к шагу 1.

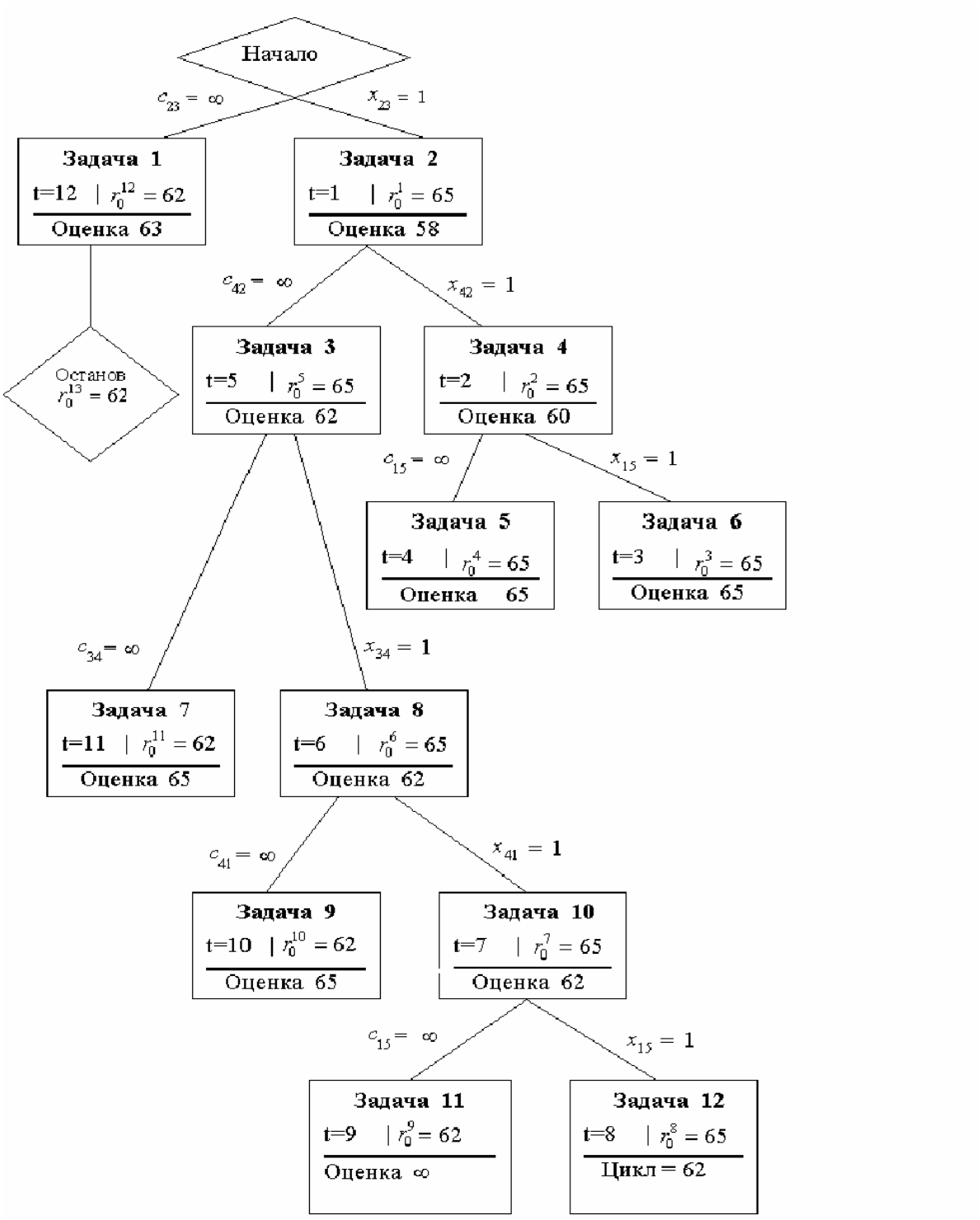
Если дугу (i_0, j_0) с указанными свойствами найти не удалось, то ничего не вносим в список и переходим к шагу 1.

Отметим два существенных отличия данного варианта алгоритма ветвей и границ от предыдущего. В данном варианте на шаге 2 вычисляется нижняя оценка для выбранной задачи, но *не отыскивается* её оптимальное решение. Кроме того, на шаге 4 в основной список могут вноситься *две задачи* или не добавляется *ни одной*, если не существует переменной $x_{i_0 j_0}$, удовлетворяющей указанным условиям 1 – 4. В предыдущей модификации на шаге 4 образуется от 2 до $n/2$ новых задач, вносимых в основной список.

Пример 2. Рассмотрим задачу коммивояжера с той же матрицей расстояний, что и в примере 1, т.е. с матрицей

	1	**2**	**3**	**4**	**5**
1	∞	10	25	25	10
2	1	∞	10	15	2
3	8	9	∞	20	10
4	14	10	24	∞	15
5	10	8	25	27	∞

Ход решения данной задачи с помощью второй модификации отражен на рис. 4.8.



Оптимальный цикл: r_0=62
(задача 12)
x_{15}=x_{52}=x_{23}=x_{34}=x_{41}= 1

Рис. 4.8

Глава 5. Линейное программирование и теория игр

Как и линейное программирование, теория игр является одной из областей современной математики. Если при исследовании общей задачи линейного программирования мы определяем способ эффективного использования или распределения ограниченных ресурсов для достижения желаемых целей, то в теории игр нас интересует стратегия, с помощью которой достигается выигрыш, максимально возможный в данной игре.

В то время, когда закладывались основы теории игр, замечательное соответствие между линейным программированием и теорией игр не было известно. Связь между ними впервые была установлена фон Нейманом и Данцигом. В данной главе мы установим эквивалентность общей задачи линейного программирования произвольной игре двух партнеров с нулевой суммой и конечным числом стратегий.

§ 1. Постановка задачи

Основное содержание теории игр состоит в изучении следующей проблемы: если n партнеров $P_1, P_2, ..., P_n$ играют в данную игру Г, то как должен вести партию i-й игрок для достижения наиболее благоприятного для себя исхода?

Здесь под термином *игра* понимается совокупность предварительно оговоренных правил и условий игры, а термин *партия* связан с частной возможной реализацией этих правил. В дальнейшем предполагается, что в конце каждой партии игры каждый игрок P_i получает сумму денег v_i, называемую *выигрышем* этого игрока, и стремится максимизировать сумму получаемых им денег.

В большинстве салонных игр общая сумма денег, теряемых проигравшими игроками, равна сумме денег, получаемых выигравшими партнерами. В этом случае для каждой партии

$$v_1 + v_2 + ... + v_n = 0.$$

Число v_i может быть любого знака, при этом:

$v_i > 0$ соответствует выигрышу,

$v_i < 0$ соответствует проигрышу,

$v_i = 0$ соответствует нейтральному исходу.

Игры, в которых алгебраическая сумма выигрышей равна нулю, называются *играми с нулевой суммой*.

Игры также классифицируются по числу игроков и числу возможных ходов. Далее игры можно подразделять на *кооперативные* и *некооперативные*. В кооперативных играх партнеры могут образовывать коалиции и играть как команды, тогда как в некооперативных играх каждого игрока интересует лишь его собственный результат. Игры двух партнеров являются, очевидно, некооперативными.

Мы будем рассматривать только игры двух партнеров с нулевой суммой и конечным числом возможных ходов. Такие игры называются *прямоугольными*

или *матричными*, поскольку они задаются платежной матрицей (или матрицей выигрышей):

$$A = \left(a_{ij},\ j = \overline{1,n} \atop i = \overline{1,m} \right).$$

Первый игрок имеет возможность сделать m выборов (m чистых стратегий), а второй – n выборов (n чистых стратегий). Если первый игрок выбирает i-ю чистую стратегию, а второй – j-ю, то выигрыш первого (проигрыш второго) равен a_{ij}, сумма выигрышей обоих игроков равна нулю.

Задача теории игр заключается в выборе принципов поведения игроков в каждой конкретной ситуации.

Решение игры – это выбор линии поведения игроков, обеспечивающих состояние равновесия, т.е. состояние, к которому стремился бы каждый разумный игрок, сознавая, что отступление от этой линии может только уменьшить его выигрыш.

В большинстве конкретных ситуаций невозможно указать такие чистые стратегии игроков, которые обеспечивали бы ситуацию равновесия независимо от поведения противников. Однако теория игр позволяет выработать такую линию поведения, придерживаясь которой в каждой партии каждый игрок может обеспечить ситуацию равновесия в среднем (для многих партий) независимо от поведения противника.

Если один из противников не будет в процессе последовательного повторения партий придерживаться правил оптимального выбора стратегий, то средний выигрыш другого противника может увеличиться.

§ 2. Матричные игры. Смешанные стратегии

Вектор $x = (x_1, x_2, ..., x_m)$, каждая компонента которого указывает относительную частоту (вероятность), с которой соответствующая чистая стратегия используется в игре, называется *смешанной стратегией* первого игрока.

Вектор $y = (y_1, y_2, ..., y_n)$ – смешанная стратегия второго игрока. Ясно, что

$$x_i \geq 0,\ i = 1,m;\ y_j \geq 0,\ j = 1,n;\ \sum_{i=1}^{m} x_i = 1,\ \sum_{j=1}^{n} y_j = 1.$$

Чистая стратегия может быть определена как смешанная стратегия, в которой все компоненты, кроме одной, равны нулю.

В дальнейшем будем обозначать чистые стратегии обоих игроков в виде единичных векторов

$$e_i = \left(\underbrace{0,0,...,0,1}_{i},0...0 \right),\ e_j = \left(\underbrace{0,0,...,0,1}_{j},0...0 \right).$$

Оптимальная стратегия игрока – это стратегия, обеспечивающая ему максимально возможный *гарантированный средний* выигрыш. (При этом предупреждается, что игра ведется без обмана и подглядывания).

Рассмотрим матричную игру, определяемую матрицей выигрышей:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}.$$

Если первый партнер P_1 выбирает любую чистую стратегию i, то он уверен, что выиграет по крайней мере $\min\limits_j a_{ij}$.

Например, если

$$A = \begin{pmatrix} 1 & 5 & 0 & 4 \\ 2 & 1 & 3 & 3 \\ 4 & 2 & -1 & 0 \end{pmatrix},$$

то мы имеем: $\min\limits_j a_{1j} = 0$, $\min\limits_j a_{2j} = 1$, $\min\limits_j a_{3j} = -1$.

Поскольку игрок P_1 может выбирать любые *i*, то естественно предполагать, что он выберет ту стратегию *i*, при которой его гарантированный выигрыш максимальный. Следовательно, при использовании чистой стратегии игрок P_1 может выиграть не менее

$$\max\limits_i \min\limits_j a_{ij} =: g_1.$$

Число g_1 – гарантированный выигрыш игрока P_1 при использовании только чистых стратегий. В нашем примере

$$g_1 = 1 = a_{22} = 1.$$

Если второй игрок P_2 выбирает стратегию j, то наихудшее, что с ним может случиться, – это проигрыш в размере $\max\limits_i a_{ij}$. Игрок P_2 может выбирать ту чистую стратегию j, при которой его проигрыш минимален. При выборе этой стратегии игрок P_2 гарантирует, что игрок P_1 не сможет выиграть больше чем

$$g_2 := \min\limits_j \max\limits_i a_{ij}.$$

Для нашей матрицы $g_2 := \min\limits_j \max\limits_i a_{ij} = 3$.

Отметим, что всегда

$$\min\limits_{y \in Y} \max\limits_{x \in X} f(x,y) \geq \max\limits_{x \in X} \min\limits_{y \in Y} f(x,y), \text{ следовательно, } g_2 \geq g_1.$$

Таким образом, в нашем примере при выборе только чистых стратегий игрок P_1 может гарантировать, что он выиграет не менее 1 (а хотел бы выиграть с гарантией больше!); игрок P_2 может гарантировать, что он не проиграет более 3 (а хотел бы проиграть с гарантией меньше!).

Если бы имело место равенство

$$\max_i \min_j a_{ij} = \min_j \max_i a_{ij} = v, \tag{1}$$

то P_1 может быть уверен, что выиграет не менее v, а игрок P_2 может добиться того, что гарантированно не проиграет больше чем v.

Матричная игра, для которой имеет место равенство (1), наилучшим образом разыгрывается партнерами P_1 и P_2, избирающими соответствующие чистые стратегии. Поскольку при любом отклонении игрока P_1 от этой стратегии его гарантированный выигрыш не увеличивается (а возможно, и уменьшится) и поскольку при каждом отклонении партнера P_2 от своей чистой стратегии он не уменьшит своего проигрыша (а возможно, увеличит его), указанные чистые стратегии естественно назвать оптимальными чистыми стратегиями.

Следующая матричная игра является одной из тех игр, которые имеют оптимальные чистые стратегии:

$$A = \begin{pmatrix} 3 & 5 & 6 \\ 1 & 2 & 3 \\ 0 & 7 & 4 \end{pmatrix}, \ \max_i \min_j a_{ij} = \min_j \max_i a_{ij} = 3.$$

Поскольку не все матричные игры могут оптимально разыгрываться при помощи чистых стратегий, необходимо ввести понятие оптимальной смешанной стратегии.

Если игрок P_1 при длительном процессе игры выбирает смешанную стратегию $x = (x_1, x_2, ..., x_m)$, т.е. с вероятностью x_i выбирает i-ю чистую стратегию, а игрок P_2 при длительном процессе игры выбирает смешанную стратегию $y = (y_1, y_2, ..., y_n)$, т.е. с вероятностью y_j выбирает j-ю стратегию, то математическое ожидание (средний выигрыш) выигрыша игрока P_1 равно

$$M(x, y) = \sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij} x_i y_j = x'Ay. \tag{2}$$

Соответственно выигрыш игрока P_2 при этом равен $-M(x, y)$, т.е. его проигрыш равен $M(x, y)$.

Функция $M(x, y)$ называется *функцией платежей*.

Говорят, что ***игра имеет решение в смешанных стратегиях***, если существуют такие стратегии x^*, y^* и число v, что при любых смешанных стратегиях x, y выполняется соотношение

$$M(x, y^*) \le v \le M(x^*, y). \qquad (3)$$

Полагая в (3) $x = x^*, y = y^*$, получим

$$v = M(x^*, y^*). \qquad (4)$$

Число v называется ***ценой игры***.

Неравенство (3) означает, что если игрок P_1 будет использовать смешанную стратегию x^* (при длительном процессе игры), то его гарантированный выигрыш равен v, так как при любом y имеем

$$v \le M(x^*, y).$$

Для игрока P_2 соотношение (3) означает, что если игрок P_2 будет придерживаться стратегии y^*, то не проиграет больше чем v, так как при каждом выборе смешанной стратегии x игроком P_1 имеем

$$M(x, y^*) \le v.$$

Стратегии x^*, y^*, удовлетворяющие (3), называются *оптимальными стратегиями.*

Рассмотрим примеры.

Пример 1. Игрок P_1 выбирает одну из двух сторон монеты. Игрок P_2, не зная выбора первого игрока, также выбирает одну из сторон монеты. После того, как оба игрока произвели свой выбор, игрок P_2 платит 1 дол. игроку P_1, если выбранные стороны совпали, и –1 дол. – в противном случае, т.е. в противном случае игрок P_1 платит игроку P_2 1 дол. (т.е. игрок P_1 играет на максимум, а игрок P_2 – на минимум).

Запишем матрицу платежей

		Выбор P_2: «Орел»	*Выбор* P_2: «Решка»
Выбор P_1	«*орёл*»	1	–1
	«*решка*»	–1	1

$$\Rightarrow A = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

Если бы игроки использовали только чистые стратегии (т.е. играли бы только один раз или играли много раз, но использовали один и тот же выбранный заранее фиксированный ход), то игрок P_1 мог бы гарантировать себе выигрыш в размере

$$g_1 = \max_i \min_j a_{ij} = -1,$$

а игрок P_2 мог бы гарантировать, что не проиграет более чем

$$g_2 = \min_j \max_i a_{ij} = 1.$$

Но первый игрок хотел бы выиграть больше, а второй проиграть меньше.

Предположим теперь, что игрокам разрешается использовать смешанные стратегии, т.е. предполагается, что игра повторяется много раз и каждый из игроков должен определить, с какой частотой (вероятностью) он будет выбирать «орла» и с какой частотой «решку», так, чтобы «улучшить» свой гарантированный результат: для игрока P_1 это означает увеличить свой гарантированный выигрыш, а для P_2 это означает уменьшить свой гарантированный проигрыш.

Очевидно, что в данной простой игре оптимальными смешанными стратегиями будут

$$x^* = \left(x_1 = \frac{1}{2}, x_2 = \frac{1}{2} \right), \; y^* = \left(y_1 = \frac{1}{2}, y_2 = \frac{1}{2} \right).$$

Игрок P_1 с вероятностью $\frac{1}{2}$ выбирает «орла», с вероятностью $\frac{1}{2}$ – «решку». Игрок P_2 поступает аналогично. При этом, используя стратегии x^*, y^* в длинном процессе игры, игрок P_1 гарантирует себе выигрыш (средний) $v = 0$ (это больше чем $g_1 = -1$!), а игрок P_2 гарантирует себе, что не проиграет больше чем $v = 0$ (это меньше, чем было раньше $g_2 = 1$!).

Пример 2. Игра «в жулика». *Эта игра является примером игры, которая на первый взгляд кажется беспроигрышной для каждого игрока, т.е. имеет цену $v = 0$, но на самом деле это не так.*

Каждому из двух партнеров выдается по тузу бубен и треф. P_1 получает также бубновую двойку, а P_2 – трефовую. При первом ходе P_1 выбирает и откладывает одну из своих карт, и P_2, не знающий выбора карты партнером P_1, также откладывает одну свою карту.

Если были отложены карты одной масти, выигрывает игрок P_1, в противном случае выигравшим считается игрок P_2. Размер выигрыша определяется картой, отложенной победителем (тузу приписывается 1 очко, двойке – 2 очка). Если отложены две двойки, выигрыш равен нулю. Матрица выигрышей этой игры имеет вид

	♦	♣	2
			♣
♦	1	-1	-2
♣	-1	1	1
2	2	-1	0
♦			

Поскольку каждый элемент третьей строки не меньше соответствующего элемента первой строки, то игроку P_1 не имеет смысла использовать первую стратегию. Следовательно, можно считать, что $x_1 = 0$. В этом случае говорят, что третья стратегия доминирует над первой. После исключения первой строки матрица выигрышей имеет вид

$$\begin{pmatrix} -1 & 1 & 1 \\ 2 & -1 & 0 \end{pmatrix}.$$

Так как элементы второго столбца меньше либо равны элементам третьего столбца, то игроку P_2 нет необходимости использовать третью стратегию, следовательно, $y_3 = 0$. Из матрицы удалим третий столбец. Получим матрицу

$$\begin{pmatrix} -1 & 1 \\ 2 & -1 \end{pmatrix}.$$

Исследование данной матрицы показывает, что она не имеет седловой точки, следовательно, данная игра не имеет решения в чистых стратегиях.

Рассмотрим смешанные стратегии. Пусть P_1 с вероятностью s выбирает вторую стратегию, тогда с вероятностью $(1-s)$ он выбирает третью стратегию и его смешанная стратегия имеет вид

$$x = (x_1 = 0,\ x_2 = s,\ x_3 = (1-s)).$$

Аналогично для P_2: пусть он с вероятностью p выбирает первую стратегию, тогда он с вероятностью $(1-p)$ выбирает вторую стратегию. Его смешанная стратегия имеет вид

$$y = (y_1 = p, y_2 = (1-p), y_3 = 0).$$

При этом цена игры равна $M(x,y) = (-3s+2)p + (2s-1)(1-p) = \overline{M}(s,p)$. Надо найти такие s^*, p^*, $0 \le s^* \le 1$, $0 \le p^* \le 1$, что

$$\overline{M}(s,p^*) \le \overline{M}(s^*,p^*) \le \overline{M}(s^*,p).$$

Нетрудно проверить, что в нашем примере $s^* = \frac{3}{5}$, $p^* = \frac{2}{5}$, значит,

$$x^0 = \left(0, \frac{3}{5}, \frac{2}{5}\right),\ y^0 = \left(\frac{2}{5}, \frac{3}{5}, 0\right) \text{ и } M(x^0, y^0) = \frac{1}{5}.$$

Следовательно, эта игра выгодна для игрока P_1. Иными словами, если P_1 избирает свою оптимальную стратегию x^0, а серия игр достаточно длинна, то ему

гарантирован, независимо от стратегии y, применяемой игроком P_2, выигрыш не менее чем $M(x^0, y^0) = v^0 = \frac{1}{5}$.

Аналогично, если P_2 избирает свою оптимальную стратегию y^0, то при длительной игре он может быть застрахован от проигрыша, большего, чем v^0. (Если он отступит от своей стратегии y^0, то может проиграть больше). Если оба игрока изберут свои оптимальные стратегии, то можно предсказать вероятный исход игры.

Определение. Игра называется *симметричной*, если соответствующая ей матрица выигрышей является кососимметричной, т.е. $a_{ij} = -a_{ji}$.

Утверждение. Цена симметричной игры равна $v = 0$, и оптимальные стратегии обоих игроков совпадают.

Доказательство. Запишем функцию выигрыша для P_1 (проигрыша для P_2):

$$M(x, y) = x'Ay = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i a_{ij} y_j .$$

Легко проверить, что в случае кососимметричной матрицы A и $x = y$ имеем $M(x,x) = x'Ax = 0$. Обозначим через x^0, y^0 оптимальные стратегии игроков P_1 и P_2 соответственно. Имеем

$$\max_x \min_y x'Ay = \min_y x^{0'} Ay = v .$$

Если P_2 использует любую свою стратегию y, то $x^{0'} Ay \geq v$. Но при $y = x^0$ мы знаем, что $x^{0'} Ax^0 = 0$, следовательно, $0 \geq v$.

Аналогично для игрока P_1:

$$\min_y \max_x x'Ay = \max_x x' Ay^0 = v .$$

Если P_1 использует каждую свою стратегию x, то $x'Ay^0 \leq v$. Для $x = y^0$ получаем $y^{0'} Ay^0 = 0$. Следовательно, $0 \leq v$.

Из полученных неравенств следует, что $v = 0$, следовательно, цена игры равна нулю и игроки имеют одинаковые смешанные стратегии. Что и требовалось доказать.

Выше мы рассматривали два небольших примера и специальную (симметричную) игру и смогли найти решения или описать свойства решений. А как поступить в общем случае? Каждая ли матричная игра имеет решение? И если имеет, то как его найти? Ответы на эти вопросы дадим в следующем параграфе.

§3. Эквивалентность матричной игры и задачи линейного программирования

Сформулируем основную теорему теории игр.

Теорема 1. *Каждая матричная игра с нулевой суммой имеет решение в смешанных стратегиях, т.е. существуют такие смешанные стратегии* x^0 *и* y^0 *первого и второго игроков соответственно, что при любых смешанных стратегиях* x, y *имеют место неравенства и равенства:*

$$M(x, y^0) \le M(x^0, y^0) \le M(x^0, y),$$

$$\max_{x \in X} \min_{y \in Y} M(x, y) = \min_{y \in Y} \max_{x \in X} M(x, y) = M(x^0, y^0),$$

где

$$X = \left\{ x \in R^m : \sum_{i=1}^{m} x_i = 1, x_i \ge 0, i = \overline{1,m} \right\},\ Y = \left\{ y \in R^n : \sum_{j=1}^{n} y_j = 1, y_j \ge 0, j = \overline{1,n} \right\} -$$

множества смешанных стратегий игроков соответственно P_1 *и* P_2.

Пусть задана матрица A

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}.$$

Согласно определению, задача игрока P_1 заключается в том, чтобы найти такое максимальное число v и вектор $x^0 = \left(x_1^0, x_2^0, \dots, x_m^0\right) \in X$, что

$$M(x^0, y) \ge v \text{ для } \forall y \in Y. \tag{5}$$

Рассмотрим ограничения (5) подробнее:

$$x^{0'} Ay \ge v, \forall y \in Y. \tag{6}$$

Покажем, что неравенства (6) эквивалентны условиям

$$x^{0'} Ae_j \ge v,\ j = \overline{1,n}. \tag{7}$$

Отметим, что в (6) мы имеем континуум ограничений, так как неравенства должны выполняться для $\forall y \in Y$; в (7) мы имеем только n ограничений. Действительно, очевидно, что из (6) следует (7), так как $e_j \in Y, j = \overline{1,n}$. Покажем, что из (7) следует (6).

Пусть (7) имеет место. Рассмотрим $\forall y = (y_1, y_2, ..., y_n) \in Y$. Правую и левую части каждого j-го неравенства (7) умножим на $y_j \geq 0$ и просуммируем все неравенства

$$\sum_{j=1}^{n} x^{0'} A y_j e_j \geq \sum_{j=1}^{n} y_j v. \tag{8}$$

Учитывая, что $\sum_{j=1}^{n} y_j = 1$ и $y = \sum_{j=1}^{n} y_j e_j$, из (8) получаем (6). Эквивалентность (6) и (7) доказана.

Таким образом, мы пришли к тому, что задача игрока P_1 состоит в поиске таких чисел v и вектора x^0, которые являются решением следующей задачи:

$$\begin{gathered} v \to \max_{x,v} \\ \sum_{i=1}^{m} a_{ij} x_i \geq v,\ j = \overline{1,n}; \\ \sum_{j=1}^{n} x_i = 1,\ x_i \geq 0,\ i = \overline{1,m}. \end{gathered} \tag{9}$$

Задача (9) является задачей линейного программирования.

Аналогично, рассуждая за второго игрока P_2, мы приходим к тому, что его задача состоит в нахождении такого минимального числа v и вектора $y^0 \in Y$, при которых

$$M(x, y^0) \leq v, \forall x \in X. \tag{10}$$

Можно показать, что (10) эквивалентно условиям $e_j' A y^0 \leq 0,\ i = \overline{1,m}$. Следовательно, задача игрока P_2 состоит в нахождении таких числа v и вектора y^0, которые являются решением следующей задачи:

$$\begin{gathered} v \to \min_{v,y}; \\ \sum_{j=1}^{n} a_{ij} y_j \leq v,\ i = \overline{1,m}; \\ \sum_{j=1}^{n} y_j = 1,\ y_j \geq 0,\ j = \overline{1,n}. \end{gathered} \tag{11}$$

Задача (11) является задачей линейного программирования.

Легко проверить, что задачи (9) и (11) составляют пару двойственных задач! Следовательно, не нужно решать каждую из этих задач отдельно. Из теории двойственности следует, что достаточно решить, например,

симплекс-методом одну из этих задач и по оптимальному плану решенной задачи легко восстановить оптимальный план второй задачи.

Таким образом, мы показали, что верна следующая теорема.

Теорема 2. *Каждая матричная игра с платежной матрицей* $A = \begin{pmatrix} a_{ij},\ j=\overline{1,n} \\ i=\overline{1,m} \end{pmatrix}$ *эквивалентна паре двойственных задач (9) и (11).*

Рассмотрим теперь обратную задачу. Попытаемся представить данную задачу линейного программирования в форме матричной игры. Каждой паре двойственных задач линейного программирования можно поставить в соответствие матричную игру, цена и оптимальные стратегии которой позволяют вычислить оптимальные прямой и двойственный планы (если они существуют!).

Подчеркнем, что в то время как матричные игры всегда имеют оптимальные стратегии (т.е. имеют решение), задачи линейного программирования могут и не иметь решений.

Рассмотрим пару двойственных задач:

$$\begin{aligned} &c'x \to max, \\ &Ax \le b, x \ge 0, \end{aligned} \tag{12}$$

$$\begin{aligned} &b'y \to min, \\ &A'y \ge c, y \ge 0. \end{aligned} \tag{13}$$

Здесь $A \in R^{m\times n}$. Построим игру с платежной матрицей

$$\Pi = \begin{pmatrix} 0 & A & -b \\ -A' & 0 & c \\ b' & -c' & 0 \end{pmatrix} \in R^{(n+m+1)\times(n+m+1)}. \tag{14}$$

Отметим, что матрица Π является кососимметричной, следовательно, если рассматривать матричную игру с платежной матрицей Π, то стратегии (оптимальные) первого и второго игроков должны совпадать! Справедлива

Теорема 3. *Пара двойственных задач (12) и (13) имеет решение тогда и только тогда, когда игра с платежной матрицей (14) имеет такую оптимальную стратегию*

$$u^* = (u_1^*, u_2^*, ..., u_{n+m}^*, u_{n+m+1}^*),$$

что $u^*_{n+m+1} > 0$. *При этом*

$$y^0_i = \frac{u^*_i}{u^*_{n+m+1}},\ i = \overline{1,m}; \qquad x^0_j = \frac{u^*_{m+j}}{u^*_{n+m+1}},\ j = \overline{1,n}.$$

Из теоремы 3 следует, что решение пары двойственных задач линейного программирования может быть сведено к вычислению оптимальных стратегий (которые совпадают!) симметричной игры с платежной матрицей *П* (14).

Таким образом, в этом параграфе мы показали, что существует тесная связь между задачами линейного программирования и матричными играми. Наличие этой связи, с одной стороны, позволяет привлечь к исследованию матричных игр методы, применяемые в линейном программировании, с другой стороны, делает возможным использование идей и методов теории игр для разработки новых алгоритмов решения задач линейного программирования.

ЛИТЕРАТУРА

1. Вагнер Г. Основы исследования операций. Т. 1–3. – М.: Мир, 1973.
2. Габасов Р., Кириллова Ф.М. Методы оптимизации: Учеб. пособие. – Мн.: БГУ, 1981.
3. Габасов Р., Кириллова Ф.М. Методы линейного программирования: В 3 ч. Ч. 2. – Мн.: БГУ, 1978.
4. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. – М.: Мир, 1985.
5. Дегтярев Ю.И. Исследование операций. – М.: Высш. шк., 1986.
6. Исследование операций. Т. 1. Методологические основы и математические методы. Т. 2. Модели и применения. Под ред. Дж. Моудера, С. Элмалмаграби. – М.: Мир, 1981.
7. Йенсен П., Барнес Д. Потоковое программирование. – М.: Радио и связь, 1984.
8. Карманов В.Г. Математическое программирование: Учеб. пособие. – М.: Наука, 1980.
9. Костюкова О.И. Методы оптимизации: Учеб. пособие для студентов специальности Н.08.02.00 «Информатика»: В 2 ч. Ч. 1. Линейное и квадратичное программирование. – Мн.: БГУИР, 2000.
10. Таха Х. Введение в исследование операций. – М.: Изд. дом «Вильямс», 2001.

11. Химмельблау Д. Прикладное нелинейное программирование. – М.: Мир, 1975.
12. Wolsey L.A. Integer Programming. Wiley-Interseries in Discrete Mathematics and Optimization, John Wiley & Sons, New York, 1996.

Св. план 2003, поз. 38

Учебное издание

Костюкова Ольга Ивановна

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Учебное пособие
для студентов специальности 31 03 04 «Информатика»
всех форм обучения

Редактор Н.А. Бебель

Корректор Е.Н. Батурчик

Подписано в печать .10.2003. Формат 60x84 1/16. Бумага офсетная.
Печать ризографическая. Гарнитура «Таймс». Усл. печ. л.
Уч.-изд. л. 6,0. Тираж 100 экз. Заказ 257.

Издатель и полиграфическое исполнение:
Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники».

Лицензия ЛП № 156 от 30.12.2002.
Лицензия ЛВ № 509 от 05.08.2001.
220013, Минск, П. Бровки, 6