

Министерство образования и науки Российской Федерации  
Федеральное агентство по образованию  
Ярославский государственный университет им. П. Г. Демидова

А. А. Короткин, В. Г. Фокин

# Модели и алгоритмы исследования операций

*Учебное пособие*

*Рекомендовано*

*Научно-методическим советом университета  
для студентов специальностей Прикладная математика и информатика,  
Математическое обеспечение и администрирование информационных систем*



Ярославль 2006

272 119



УДК 519.8

ББК В183.5я73

К68

*Рекомендовано*

*Редакционно-издательским советом университета  
в качестве учебного издания. План 2006 года*

Рецензенты:

доктор технических наук, профессор, зав. кафедрой прикладной математики  
Ярославского государственного технического университета Д. О. Бытев;  
кафедра теории и методики обучения информатике Ярославского  
государственного педагогического университета им. К. Д. Ушинского

К68 **Короткин, А. А.** Модели и алгоритмы исследования операций: учеб. пособие /  
А. А. Короткин, В. Г. Фокин; Яросл. гос. ун-т. – Ярославль: ЯрГУ, 2006. – 76 с.  
ISBN 5-8397-0459-8

Учебное пособие содержит изложение базовых вопросов, связанных с построением и анализом математических моделей в задачах принятия оптимальных решений. Особое внимание уделяется сетевым моделям, оценке трудоемкости алгоритмов, теории бескоалиционных игр и моделям массового обслуживания.

Предназначено для студентов, обучающихся по специальности 010501 Прикладная математика и информатика, 010503 Математическое обеспечение и администрирование информационных систем и направлению 010500 Прикладная математика и информатика (дисциплина «Теория игр и исследование операций», блоки ЕН и ОПД), очной формы обучения.

Табл. 5. Ил. 23. Библиогр.: 5 назв.

УДК 519.8

ББК В183.5я73

© Ярославский  
государственный  
университет, 2006

ISBN 5-8397-0459-8

© А. А. Короткин, В. Г. Фокин, 2006

## Оглавление

<b>Глава 1 Основные понятия исследования операций</b>	<b>4</b>
1.1. Модели исследования операций . . . . .	4
1.2. Выбор решения в условиях неопределенности . . . . .	7
1.3. Многокритериальные задачи . . . . .	9
<b>Глава 2 Сетевые модели и алгоритмы</b>	<b>12</b>
2.1. Основные понятия теории графов . . . . .	12
2.2. Задача о минимальном остове графа . . . . .	15
2.3. Кратчайшие пути в графе . . . . .	19
2.4. Сетевые графики . . . . .	23
2.5. Потоки в сетях . . . . .	27
<b>Глава 3 Оптимальное распределение ограниченного ресурса</b>	<b>34</b>
3.1. Метод динамического программирования . . . . .	34
3.2. Задача о рюкзаке . . . . .	38
<b>Глава 4 Элементы теории</b>	
<b>массового обслуживания</b>	<b>41</b>
4.1. Задачи теории массового обслуживания . . . . .	41
4.2. Формула Литтла . . . . .	43
4.3. Потоки событий в СМО . . . . .	45
4.4. Расчет марковских СМО . . . . .	48
<b>Глава 5 Теория игр</b>	<b>55</b>
5.1. Принципы рационального поведения . . . . .	55
5.2. Антагонистические игры . . . . .	60
5.3. Матричные игры . . . . .	63
5.4. Позиционные игры . . . . .	71
<b>Литература</b> . . . . .	<b>75</b>

# Глава 1

## Основные понятия исследования операций

*В технике, медицине, науке, финансах, бизнесе и даже в сфере государственного управления решения, затрагивающие жизнь каждого из нас, теперь принимаются в соответствии с упорядоченными процедурами, которые значительно эффективнее приблизительных и произвольных методов прошлого. Благодаря этому удается избежать или по крайней мере смягчить последствия многих катастрофических ошибок.*

*Л. Бернштейн. Против богов: Укрощение риска.*

*Хотели как лучше, а получилось как всегда.*

*В.С. Черномырдин*

### 1.1. Модели исследования операций

Под операциями обычно понимают целенаправленные управляемые процессы. Природа их может быть различной – это могут быть военные действия, производственные процессы, коммерческие мероприятия, административные решения и т.д., и т.п. Принципиальным здесь является то, что операции эти (совершенно несхожие по своей природе) могут быть описаны одними и теми же математическими моделями. Более того, анализ этих моделей позволяет лучше понять суть того или иного явления и даже предсказать его дальнейшее развитие. В теории систем это называется термином «изоморфизм моделей». Если бы не изоморфизм моделей, для каждой конкретной ситуации пришлось бы отыскивать собственный, уникальный метод решения и исследование операций как научное направление не сформировалось бы. К счастью, дело обстоит иначе: благодаря наличию общих закономерностей в развитии самых разных систем возможно исследование их математическими методами. Исследование операций как математический инструментарий, поддерживающий процесс принятия решений в самых разных областях человеческой деятельности, как совокупность средств, позволяющих обеспечить лицо, принимающее решение (ЛПР), необходимой количественной информацией, полученной



научными методами, сформировалось на стыке математики и разнообразных социально-экономических дисциплин; свой вклад в его становление внесли представители самых различных областей науки.

При всем многообразии содержания конкретных работ в области исследования операций каждое операционное исследование проходит последовательно через несколько этапов, основными из которых являются:

- постановка задачи и разработка концептуальной модели;
- разработка математической модели;
- выбор (разработка) метода и алгоритма решения;
- проверка адекватности и корректировка модели;
- поиск решения на модели;
- реализация найденного решения на практике.

**Постановка задачи и разработка концептуальной модели.** Это чрезвычайно важный и ответственный этап операционного исследования. Недаром говорят, что правильно поставить задачу – это наполовину ее решить. Первоначально цель и задачу операционного исследования формулирует заказчик (руководство фирмы, концерна, организации и т.п.). Как правило, эта цель и постановка задачи имеют довольно общий характер, например: исследовать организацию системы снабжения или основного производства, поставить диагноз и разработать конкретные рекомендации относительно ее улучшения. На этом этапе создается операционная группа из системных аналитиков, специалистов в области организации производства, социологов и психологов и т.п.

Операционная группа детально обследует соответствующую систему (объект), изучает информационные и материальные потоки как внутри самой системы, так и ее связи с внешней средой. Одновременно изучаются организация подсистемы управления данной системой (объектом), а также функционирование системы (показатели качества или критерии эффективности) и внешние факторы, которые влияют на эти характеристики. После сбора результатов обследования проводится их подробный анализ, в результате которого обнаруживаются существенные факторы и переменные, обосновывается выбор тех или иных показателей качества функционирования системы, а также существенных внешних факторов, структура самой системы, состав ее элементов, их взаимосвязи, внутренние переменные. Проводятся неоднократные консультации с заказчиками, в ходе которых уточняется постановка задачи. В случае необходимости проводится дополнительное обследование организационной системы с целью выявления неучтенных факторов и их взаимосвязей. Результатом этого этапа является концептуальная модель исследуемой системы (задачи), в которой в содержательной форме описывается состав системы, ее компоненты и их взаимосвязи, перечень основных показателей качества, переменных, как контролируемых, так и неконтролируемых внешних факторов, а также

их взаимосвязей с показателями качества системы, перечень стратегий управления (или решений), которые надо определить в результате решения поставленной задачи.

**Разработка математической модели.** После получения концептуальной модели системы (содержательной постановки задачи) нужно построить ее математическую модель. Этот процесс называется формализацией задачи. Любая задача принятия решений характеризуется следующими элементами:

1) совокупность *управляющих переменных*  $x = \{x_i\}$ , значения которых выбирает ЛПР; совокупность  $x$  иногда называют планом, стратегией, вариантом или альтернативой;

2) совокупность переменных, значения которых не регулируются ЛПР. Эти переменные могут играть роль внутренних параметров системы  $a = \{a_k\}$ , значения которых заранее фиксированы (например, запасы ресурсов у организации). В других случаях эти переменные могут быть внешними, которые изменяются независимо от ЛПР, и тогда их называют *неконтролируемыми факторами* операции  $\omega = \{\omega_l\}$ ;

3) множество  $D$  допустимых значений управляющих переменных. Во многих задачах множество  $D$  определяется естественными ограничениями вида  $x_j^- \leq x_j \leq x_j^+$ , а также ограничениями вида  $g_i(x, a) \leq b_i$ ,  $i = 1, \dots, m$ ;

4) *целевая функция* (критерий или показатель эффективности)  $f(x, a, \omega)$ , которая зависит от принятых стратегий, параметров системы и неконтролируемых факторов и которая характеризует «качество» управляющих переменных. Этот критерий может быть как скалярным, так и векторным. В последнем случае мы имеем так называемую *многокритериальную* задачу принятия решений. В самом общем случае математическая модель задачи принятия решений имеет вид:

$$F = f(x, a, \omega) \longrightarrow \max_{x \in D} (\min_{x \in D}). \quad (1.1)$$

**Выбор метода и алгоритма решения.** Для нахождения решения задачи (1.1) используют в зависимости от вида и структуры целевой функции и ограничений те или другие методы теории оптимальных решений (методы математического программирования):

- *линейное программирование*, если функции  $f(x, a, \omega)$ ,  $g_i(x, a, \omega)$  линейны относительно переменных  $x$ ;
- *нелинейное программирование*, если функции  $f(x, a, \omega)$  и (или)  $g_i(x, a, \omega)$  нелинейны относительно переменных  $x_i$ ;
- *дискретное программирование*, если на управляющие переменные  $x$  наложено условие дискретности, например, целочисленности ( $x_j \in \mathbb{Z}^+$ );
- *динамическое программирование*, если функции  $f(x, a, \omega)$  и  $g_i(x, a, \omega)$  имеют специальную структуру: например, являются аддитивными или мультипликативными от

переменных  $x$ ;

– *эвристическое программирование* – его применяют для решения тех задач, в которых точный оптимум найти невозможно из-за сложного комбинаторного характера задачи (так называемые  $\text{NP}$ -полные задачи) и связанного с ним огромного количества вариантов в множестве  $D$ . В таком случае отказываются от поиска оптимальных решений и ограничиваются поиском удовлетворительного решения с точки зрения ЛПР. При этом удачно пользуются специальными приемами – так называемыми «эвристиками», позволяющими существенно сократить число просматриваемых вариантов.

**Проверка адекватности и корректировка модели.** В сложных системах, к которым относятся и системы организационного типа, модель лишь частично отражает реальный объект (или процесс). Поэтому необходимо проводить проверку степени соответствия (или адекватности) модели и реального процесса.

Проверку можно проводить путем сравнения рассчитанного значения выходных характеристик  $y$  системы с фактическими значениями  $y_0$  при изменении значений внешних факторов в широком диапазоне. Выбор таких характеристик зависит от рассматриваемой системы. В качестве меры адекватности модели выбирают одно из соотношений:

а) абсолютное отклонение  $\varepsilon_a = |y - y_0| \leq \varepsilon_{\text{дон}}$ ;

б) относительное отклонение  $\varepsilon_b = |(y - y_0)/f_0| \cdot 100\% \leq \varepsilon_{\text{дон}}^b$ ;

в) вероятностная оценка  $\mathcal{P}\{\varepsilon_b \leq \varepsilon_{\text{дон}}^b\} \geq \mathcal{P}_{\text{дон}}$ ,

где  $\varepsilon_{\text{дон}}$  и  $\varepsilon_{\text{дон}}^b$  – допустимое отклонение, которое задается экспертом и определяет заданную степень адекватности модели,  $\mathcal{P}_{\text{дон}}$  – заданная величина вероятности неравенства  $\varepsilon_b \leq \varepsilon_{\text{дон}}^b$ . Если приведенные выше неравенства не выполнены, то это свидетельствует о том, что упущены некоторые важные факторы и модель требует корректировки. Возможны корректировки концептуальной модели, математической модели и соответственно метода решения.

**Реализация найденного решения на практике.** Даже если в результате исследования удастся найти наилучшее решение, окончательный выбор должен делать человек (ЛПР), который может использовать дополнительную информацию, не отраженную в модели.

## 1.2. Выбор решения в условиях неопределенности

При решении оптимизационной задачи (1.1) следует особенно выделить вопрос о полноте имеющейся информации относительно неконтролируемых факторов  $\omega = \{\omega_l\}$ . Если неконтролируемые факторы известны точно  $\omega = \omega_0$ , то задача имеет детерминированный характер и требует для своего решения один из перечисленных методов. Однако на практике исследователь операции может столкнуться с ситуацией, когда нужно нахо-

дить оптимальное решение в условиях неопределенности, т. е. неполноты информации относительно значений неконтролируемого фактора  $\omega$ . Неопределенность может быть следующего характера:

- 1) значение  $\omega$  являются случайной величиной (в общем случае векторной) с известным законом распределения вероятностей;
- 2) неконтролируемый фактор  $\omega$  – детерминированная величина, относительно которой известно лишь, что  $\omega \in \Omega$ , где  $\Omega$  – некоторая известная исследователю операции область возможных значений неконтролируемого фактора.

В первом случае целевая функция  $f(x, a, \omega)$  является случайной величиной, и один из возможных подходов к нахождению оптимальных значений управляющих переменных  $x^*$  заключается в получении экстремума математического ожидания  $f(x, a, \omega)$  («оптимизация в среднем»):

$$\bar{f}(x, a) = M[f(x, a, \omega)] = \int f(x, a, z) p_{\omega}(z) dz \longrightarrow \max_{x \in D} (\min_{x \in D}), \quad (1.2)$$

где  $p_{\omega}(z)$  – плотность распределения вероятностей случайной величины  $\omega$ . В случае, когда  $\omega$  принимает дискретные значения  $\omega^{(i)}$  с вероятностями  $p_i$ , интеграл заменяется суммой  $\sum_i f(x, a, \omega^{(i)}) p_i$ .

Другой подход к решению задачи заключается в следующей замене критерия операции (для определенности рассмотрим случай минимизации). Пусть  $f_0$  – некоторое возможное значение целевой функции. Тогда оптимальным будем считать значение управляющей переменной  $x^*$ , полученное как решение задачи

$$f_0 \longrightarrow \min_{x \in D}, \quad \mathcal{P}(\{f(x, a, \omega) \leq f_0\}) \geq \beta, \quad (1.3)$$

где «уровень достоверности»  $\beta$  выбирается достаточно большим: например,  $\beta = 0.9$ .

Первый подход («оптимизация в среднем») обычно применяется, когда операция повторяется многократно (либо во времени, либо в пространстве). Не рассчитывая на оптимум в каждом конкретном повторении операции, мы получим оптимум в среднем за всю серию повторений. Второй подход целесообразно использовать при однократной реализации решения.

Рассмотрим теперь неопределенность второго типа. Здесь наиболее часто используется подход, основанный на *принципе гарантированного результата*, суть которого заключается в следующем. Пусть для определенности рассматривается задача максимизации. Считают, что любое значение неконтролируемых факторов  $\omega$  может реализоваться, и поэтому при выборе вектора управляющих переменных  $x$  учитывают и наименее благоприятное значение  $\omega$ . Иными словами, принимая решение  $x$ , исследователь может рассчитывать лишь на то, что полученное значение целевой функции будет не меньше, чем  $\tilde{f}(x, a) = \min_{\omega \in \Omega} f(x, a, \omega)$ . Эта величина и есть гарантированный результат (в смысле

«хуже не будет»), соответствующий выбору  $x$ . Тогда необходимо выбирать вариант  $x$  так, чтобы достичь *максимально возможного* гарантированного результата:

$$\tilde{f}(x, a) = \min_{\omega \in \Omega} f(x, a, \omega) \longrightarrow \max_{x \in D}. \quad (1.4)$$

Очевидно, что привлечение дополнительной информации для описания области  $\Omega$  приводит к ее сужению и, следовательно, позволяет получить больший гарантированный результат. Выбор решения согласно (1.4) часто называют выбором по *максиминному критерию Вальда*.

Поясним этот критерий на следующем простом примере. Пусть множество допустимых вариантов решения состоит из трех элементов  $D = \{x^{(1)}, x^{(2)}, x^{(3)}\}$ , а множество значений неконтролируемого фактора из четырех —  $\Omega = \{\omega^{(1)}, \omega^{(2)}, \omega^{(3)}, \omega^{(4)}\}$ . Значения целевой функции  $f(x, \omega)$  приведены в табл. 1.1 (рассматривается задача на максимум).

Таблица 1.1

	$\omega^{(1)}$	$\omega^{(2)}$	$\omega^{(3)}$	$\omega^{(4)}$
$x^{(1)}$	4	-1	5	9
$x^{(2)}$	-5	10	6	5
$x^{(3)}$	2	1	3	1

Гарантированный результат при выборе  $x = x^{(1)}$  есть  $\tilde{f}(x^{(1)}) = \min\{4, -1, 5, 9\} = -1$ . При  $x = x^{(2)}$   $\tilde{f}(x^{(2)}) = \min\{-5, 10, 6, 5\} = -5$  и при  $x = x^{(3)}$   $\tilde{f}(x^{(3)}) = \min\{2, 1, 3, 1\} = 1$ . Тогда  $\max_{x^{(i)}} \{\tilde{f}(x^{(i)})\} = 1 = \tilde{f}(x^{(3)})$ . Значение  $x^{(3)}$  дает максимальный гарантированный результат, который и рекомендуется принять в качестве решения (хотя, надо заметить, он кажется чересчур «перестраховочным»).

### 1.3. Многокритериальные задачи

Особое место в исследовании операций занимает случай многокритериальной оптимизации, когда эффективность принимаемого решения  $x \in D$  оценивается по нескольким показателям  $f_1, f_2, \dots, f_k$ , одни из которых желательно обратить в максимум, другие — в минимум. Такая ситуация, как правило, возникает при проведении крупномасштабных, сложных операций, затрагивающих разнообразные интересы их организаторов. Критерии обычно противоречивы, т. е. решение, лучшее по одному показателю, может оказаться худшим по другому показателю. Например, минимизация стоимости и максимизация качества всегда противоречивы.

Набор целевых функций  $f_i(x)$  удобно рассматривать как векторный показатель  $F(x) = [f_1(x), f_2(x), \dots, f_k(x)]$ . Без ограничения общности далее будем предполагать,



что целью операции является максимизация всех показателей  $f_i(x)$ :

$$F(x) = [f_1(x), f_2(x), \dots, f_k(x)] \longrightarrow \max_{x \in D}$$

Еще раз подчеркнем, что эта задача математически некорректна, так как в общем случае

$$\text{Argmax} \{f_1(x)\} \cap \text{Argmax} \{f_2(x)\} \cap \dots \cap \text{Argmax} \{f_k(x)\} = \emptyset$$

и поэтому необходимо принять какое-то компромиссное решение. Базой для компромиссного решения может служить *оптимум по Парето*.

**Определение.** Элемент  $x^* \in D$  называется оптимальным по Парето (или эффективной точкой), если **не существует** элемента  $x \in D$  такого, что

$$F(x) = [f_1(x), \dots, f_k(x)] \geq F(x^*) = [f_1(x^*), \dots, f_k(x^*)], \text{ причем } F(x) \neq F(x^*). \quad (1.5)$$

Будем обозначать множество всех эффективных точек как  $P$ .

Соотношение (1.5) означает, что изменение оптимального по Парето элемента  $x^*$  с целью увеличения какой-либо компоненты  $f_i$  векторного показателя  $F$  обязательно уменьшит значение по крайней мере одной компоненты  $f_j$ ,  $j \neq i$ . В силу такой чувствительности это понятие иногда теряет смысл «оптимального» решения. Например, если имеется фиксированное количество  $A$  некоторого ресурса, который целиком распределяется между  $k$  потребителями, причем эффект  $f_i(x_i)$  для каждого  $i$ -го потребителя пропорционален количеству  $x_i$ , то всякое распределение  $x = (x_1, \dots, x_k)$ ,  $\sum_{i=1}^k x_i = A$ , оптимально по Парето, т. е.  $P = D$ . Действительно, улучшение положения любого потребителя может быть достигнуто только за счет передачи ему дополнительного количества ресурса, изымаемого у другого потребителя, положение которого ухудшается.

Тем не менее в ряде задач  $P \subset D$ , что позволяет отбросить варианты  $x \notin P$  как заведомо непригодные в качестве решения. Следовательно,  $P$  представляет собой множество компромиссных решений. Дело ЛПР выбрать из них тот вариант, который для него предпочтителен и приемлем по всем критериям.

Другой подход к выбору компромиссного решения заключается в построении обобщенного критерия задачи. Вначале все функции  $f_i(x)$  приводят к сопоставимому безразмерному виду («нормализуют»):  $\tilde{f}_i(x) = (f_i(x) - a_i^-)/(a_i^+ - a_i^-)$ , где  $a_i^+$  — некоторые «эталонные» (плановые, максимально возможные и т.п.) значения, а  $a_i^-$  — минимально допустимые (возможные) значения критериев  $f_i$ . Далее все критерии «сворачивают» в одну функцию — обобщенный критерий  $\mathcal{F}(x, \alpha_1, \dots, \alpha_k)$ , учитывая их относительную важность при помощи специальных положительных чисел  $\alpha_i$ ,  $\sum_{i=1}^k \alpha_i = 1$ , называемых *коэффициентами важности*:

$$\mathcal{F}(x, \alpha_1, \dots, \alpha_k) = \alpha_1 \tilde{f}_1(x) + \dots + \alpha_k \tilde{f}_k(x).$$

После такой операции задача становится однокритериальной. Основная трудность в таком подходе состоит в определении величин  $\alpha_i$ . Обычно для их определения приходится использовать специальную информацию, запрашиваемую у ЛПР или получаемую у экспертов<sup>1</sup>.

Изложим один подход к решению многокритериальных задач, основанный на следующем соображении. Будем считать, что в аддитивной свертке

$$\mathcal{F}(x, \alpha) = \alpha_1 \tilde{f}_1(x) + \dots + \alpha_k \tilde{f}_k(x)$$

вектор коэффициентов относительной важности  $\alpha = (\alpha_1, \dots, \alpha_k)$  является неконтролируемым фактором с областью возможных значений

$$A = \left\{ (\alpha_1, \dots, \alpha_k) : \alpha_i \geq 0, \sum_{i=1}^k \alpha_i = 1 \right\}.$$

Если нет никакой дополнительной информации относительно  $\alpha \in A$ , то, применяя принцип наилучшего гарантированного результата, придем к задаче

$$\tilde{\mathcal{F}}(x) = \min_{\alpha \in A} \mathcal{F}(x, \alpha) \longrightarrow \max_{x \in D}.$$

Нетрудно показать, что

$$\min_{\alpha \in A} \mathcal{F}(x, \alpha) = \min_{\alpha_i \geq 0, \sum_{i=1}^k \alpha_i = 1} \left\{ \alpha_1 \tilde{f}_1(x) + \dots + \alpha_k \tilde{f}_k(x) \right\} = \min_{i=1, \dots, k} \left\{ \tilde{f}_i(x) \right\}.$$

В результате мы приходим к максиминной задаче

$$\tilde{\mathcal{F}}(x) = \min_{i=1, \dots, k} \left\{ \tilde{f}_i(x) \right\} \longrightarrow \max_{x \in D}.$$

Любая дополнительная информация относительно важности критериев, например, «первый критерий важнее второго», приводит к сужению области  $A$  за счет появления в описании  $A$  дополнительного неравенства типа  $\alpha_1 \geq \alpha_2$  и, следовательно, к увеличению гарантированного результата  $\tilde{\mathcal{F}}(x)$ .

<sup>1</sup>Одну из возможных методик определения коэффициентов относительной важности  $\alpha_i$  см.: Глов В. А., Павельев В. В. Экспертные методы определения весовых коэффициентов // Автоматика и телемеханика. 1976. №12.

## Глава 2

# Сетевые модели и алгоритмы

Во многих задачах исследования операций в основе математической модели лежит понятие графа. Важнейшими задачами (по их распространенности на практике) являются задача построения минимального остова графа, задача нахождения кратчайших путей между вершинами графа, задача построения максимального потока в сети. В этой главе будут изложены алгоритмы решения этих ставших уже классическими задач.

Большое значение для любого алгоритма  $A$  имеет его *трудоемкость* (или *быстродействие*), которая определяется как зависимость числа  $t_A$  арифметических и логических операций алгоритма от размера решаемой задачи. Обычно под размером задачи понимают объем исходных данных. Для графовых моделей эта величина связана очевидным образом с количеством вершин графа. Поэтому трудоемкость графовых алгоритмов оценивают через эту величину, т. е.  $t_A = t_A(n)$ . Алгоритм принято называть *эффективным* или *полиномиальным*, если  $t_A(n) \leq Cn^k$ , где  $C, k$  – константы. Если же  $t_A(n) \geq Ca^n$ ,  $a > 1$ ,  $A$  считается алгоритмом с *экспоненциальной* трудоемкостью, и его использование возможно лишь при небольших значениях  $n$ . Для оценки трудоемкости часто используется обозначение  $O(g(n))$ , где  $g(n)$  – некоторая возрастающая функция натурального аргумента. По определению  $t(n) = O(g(n))$ , если  $t(n) \leq C \cdot g(n)$  для всех  $n$ , начиная с некоторого  $n_0$ .

### 2.1. Основные понятия теории графов

Напомним основные определения и понятия теории графов, которые будут использованы в этой главе для изложения сетевых алгоритмов.

*Неориентированным графом*, или просто графом  $G$ , называется пара конечных множеств  $(V, E)$ , где  $V = \{v_1, \dots, v_n\}$  – некоторое множество помеченных объектов, называемых *вершинами*, а  $E \subseteq \{e = (v_i, v_j) : v_i, v_j \in V, i \neq j\}$  – множество неупорядоченных пар вершин, называемых *ребрами*. *Ориентированным графом* (орграфом) называется пара  $(V, U)$ , где  $V$  – множество вершин и  $U = \{u = (v_i, v_j)\} \subseteq V \times V$  – множество упорядоченных пар вершин, называемых *дугами*, при этом вершина  $v_i$  называется начальной вершиной, а  $v_j$  – конечной вершиной дуги. В дальнейшем для удобства будем помечать вершины числами  $1, \dots, i, \dots, n$ . Будем также использовать обозначения  $|V| = n$ ,  $|E| = m$ . Граф с  $n$  вершинами и  $m$  ребрами называется  $(n, m)$ -графом. Граф,



в котором каждая пара вершин соединена ребром, называется *полным*. Для полного графа  $m = n(n - 1)/2$ .

Наглядным способом задания графа является представление его в виде рисунка, как показано на рис. 2.1. Вершины  $i$  изображаются помеченными узлами (кружками, точками и т.п.), а ребра — линиями, соединяющими узлы. В случае ориентированного графа дуга  $(i, j)$  изображается направленной линией, идущей от начала дуги (вершина  $i$ ) к концу (вершина  $j$ ). Направление линии обозначается стрелкой.

Граф, диаграмму которого на плоскости можно нарисовать так, что ребра пересекаются лишь в вершинах, называется *планарным*.

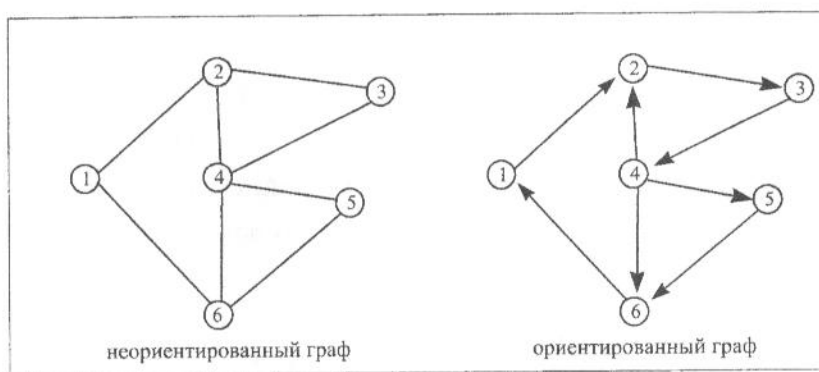


Рис. 2.1. Представление графов в виде диаграммы

Алгебраическим представлением графа является его *матрица смежности*  $A$ :

$$A = [a_{ij}], \quad i, j = \overline{1, n}, \quad a_{ij} = 1, \text{ если } (i, j) \in E \text{ и } a_{ij} = 0, \text{ если } (i, j) \notin E.$$

Матрица смежности графа симметрична и имеет нулевую диагональ. Для орграфа определение матрицы смежности такое же, как и для графа, только условие симметричности не сохраняется. Если граф (орграф) содержит относительно мало ребер (дуг), то в матрице смежности будет много нулей. Для компактного представления таких разреженных матриц используются списковые структуры данных. При создании эффективных алгоритмов часто полезно представление графов в виде обычного списка ребер: например, неориентированный граф на рис. 2.1 может быть задан списком  $L = \{(1, 2), (1, 6), (2, 3), (2, 4), (3, 4), (4, 5), (4, 6)\}$ .

*Взвешенным графом* называется граф, в котором каждому ребру  $e \in E$  приписано некоторое число (вес)  $a(e)$ , которое может в реальной задаче иметь смысл длины участка дороги между двумя городами, стоимости проведения линии связи между двумя пунктами или стоимости передачи единицы информации по этой линии и т.п. Реберно взвешенный граф можно описать аналогом матрицы смежности  $A = [a_{ij}]$ , только  $a_{ij}$  в этом случае равно весу ребра  $e = (i, j)$ . Если ребро  $e = (i, j)$  отсутствует, то полагают  $a(e) = a_{ij} = \infty$  (для задач на минимум). Аналогично определяется взвешенный оргграф. Взвешенные графы (орграфы) обычно называют *сетями*.

Пусть  $G = (V, E)$  – неориентированный граф. Вершины  $i$  и  $j$  называются *смежными*, если пара  $e = (i, j) \in E$ ; в этом случае также говорят, что ребро  $e$  *инцидентно* вершинам  $i$  и  $j$ . *Степенью* вершины  $i$  неориентированного графа называется число  $\rho(i)$  инцидентных ей ребер. Справедливо следующее очевидное равенство:

$$\sum_{i=1}^n \rho(i) = 2m. \quad (2.1)$$

Последовательность вершин  $\mu = (i_1, i_2, \dots, i_r)$  в неориентированном графе  $G = (V, E)$  такая, что  $(i_k, i_{k+1}) \in E$ ,  $k = \overline{1, r-1}$ , называется *маршрутом*, соединяющим вершины  $i_1$  и  $i_k$ . Маршрут  $\mu$  называется *цепью*, если все его ребра различны, и *простой цепью*, если в последовательности  $\mu$  все вершины, кроме, возможно, крайних, различны. Если  $i_1 = i_k$ , то маршрут называется *циклическим*. Циклическая цепь (простая циклическая цепь) называется *циклом* (простым циклом). Граф называется *связным*, если для любой пары его вершин существует маршрут, их соединяющий. Несвязный граф  $G$  естественным образом представим в виде объединения своих *компонент связности* (или просто *компонент*), т. е. максимальных по включению связных подграфов.

Для орграфа  $G = (V, U)$  рассматривают соответственно ориентированные маршруты вида  $\mu = (i_1, i_2, \dots, i_k, i_{k+1}, \dots, i_r)$ , где пары  $(i_k, i_{k+1}) \in U$ ,  $k = 1, \dots, r-1$ . Маршрут называется *цепью*, если все входящие в него дуги различны и *путем*, если все его вершины, кроме, возможно, крайних, различны. Циклический путь называется *контуром*.

Граф  $G' = (V', E')$  называется *остовным подграфом* графа  $G = (V, E)$ , если  $V' = V$  и  $E' \subset E$ .

Связный граф без циклов называется *деревом*. Существует несколько вариантов определения деревьев; некоторые из них отражены в следующей теореме.

**Теорема 2.1.** Для  $(n, m)$ -графа  $G$  следующие утверждения эквивалентны:

- 1)  $G$  – дерево;
- 2)  $G$  – связный граф и  $m = n - 1$ ;
- 3)  $G$  – граф без циклов и  $m = n - 1$ ;
- 4) любые две несовпадающие вершины соединяет единственная простая цепь;
- 5)  $G$  – граф без циклов, обладающий следующим свойством: добавление ребра между любыми двумя несмежными вершинами приводит к появлению ровно одного цикла.

Из утверждения 2) легко получается следующее

**Следствие.** В любом дереве с  $n \geq 2$  вершинами имеется не менее двух вершин со степенью 1.

Действительно, пусть  $\rho_1, \rho_2, \dots, \rho_n$  – последовательность степеней вершин  $1, 2, \dots, n$ . Все  $\rho_i > 0$  и  $\sum_{i=1}^n \rho_i = 2m = 2(n-1)$ . Следовательно, в последовательности степеней хотя бы два числа равны 1.

Представляет интерес число способов соединения  $n$  различных элементов ребрами так, чтобы получившийся граф был деревом. Ответ на этот вопрос дает следующая

**Теорема 2.2 (Кэли).** Число различных деревьев на  $n$  ( $n \geq 2$ ) помеченных вершинах равно  $n^{n-2}$ .

## 2.2. Задача о минимальном остове графа

Приведем две практические задачи, которые сводятся к построению оптимального в некотором смысле дерева.

**Задача 1.** Между населенными пунктами  $A, B, C, D, F$  и  $G$  существует сеть проселочных дорог, как показано на рис. 2.2. Длины дорог указаны непосредственно на самих дорогах. После многочисленных жалоб населения администрация области решила заасфальтировать некоторые дороги так, чтобы: 1) из любого пункта можно было бы проехать в любой пункт по асфальту, 2) суммарные затраты на асфальтирование были бы минимальны. Стоимость асфальтирования дороги пропорциональна ее длине. Найти план асфальтирования, устраивающий администрацию области.

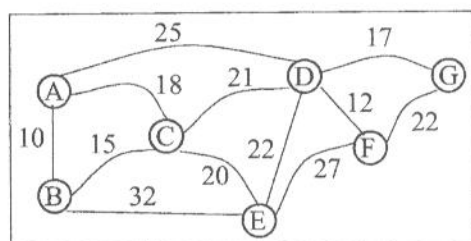


Рис. 2.2. Сеть дорог

**Задача 2.** Нужно соединить  $n$  узлов обработки информации сетью коммуникаций (например, оптоволоконным кабелем). Известна стоимость проведения коммуникационной линии между каждой парой узлов. Требования к сети: 1) связность, т. е. информацию можно передать от любого  $i$ -го узла любому  $j$ -му (возможно, транзитом через другие узлы), и 2) суммарная стоимость коммуникаций должна быть минимально возможной.

Проект асфальтирования дорог в первой задаче и проект сети коммуникаций во второй задаче задаются некоторым графом. Этот граф, очевидно, должен быть связным и не содержать циклов. Последнее следует из того, что разрыв цикла за счет удаления любого ребра, не нарушая свойства связности, только удешевляет проект. При этом в первой задаче нужно определить остовный подграф графа, описывающего сеть проселочных дорог, а во второй – построить оптимальное дерево коммуникаций. Можно считать, что и во второй задаче нужно определить остовный подграф полного графа на  $n$  вершинах.

**Остовом** (или каркасом) связного графа  $G = (V, E)$  называется остовный граф  $T$ , являющийся деревом. Задача о минимальном остове формулируется следующим образом. В множестве всех остовов  $\{T = (V, E_T), E_T \subset E\}$  данного реберно взвешенного

графа, задаваемого двумерным массивом стоимостей (весов) ребер  $C[i, j]$ ,  $i, j = \overline{1, n}$ , найти остов минимальной суммарной стоимости:

$$C(T) = \sum_{(i,j) \in E_T} C[i, j] \longrightarrow \min_T. \quad (2.2)$$

### 2.2.1. Алгоритм Прима

Идея алгоритма достаточно проста: строится последовательность «растущих» деревьев  $T_0 = (V_0, E_0), T_1 = (V_1, E_1), T_2 = (V_2, E_2), \dots, T_{n-1} = (V_{n-1}, E_{n-1})$ ,  $T_i \subset T_{i+1}$  по следующему правилу:

шаг 0.  $T_0 = (V_0, E_0)$ , где  $V_0 = \{1\}, E_0 = \emptyset$ ;

шаг 1.  $T_1 = (V_1, E_1)$ , где  $V_1 = \{1, j_1\}$ ,  $E_1 = \{(1, j_1)\}$ ,  $C[1, j_1] = \min_j C[1, j]$ ;

.....

шаг  $k+1$ .  $T_{k+1} = T_k \cup e_{k+1}$ , где ребро  $e_{k+1} = (i_{k+1}, j_{k+1})$  такое, что

$$\min_{i \in V_k, j \in V \setminus V_k} C[i, j] = C[i_{k+1}, j_{k+1}], \quad k = 1, \dots, n-2.$$

На рис. 2.3 изображен процесс построения минимального остова для исходных данных задачи 1.

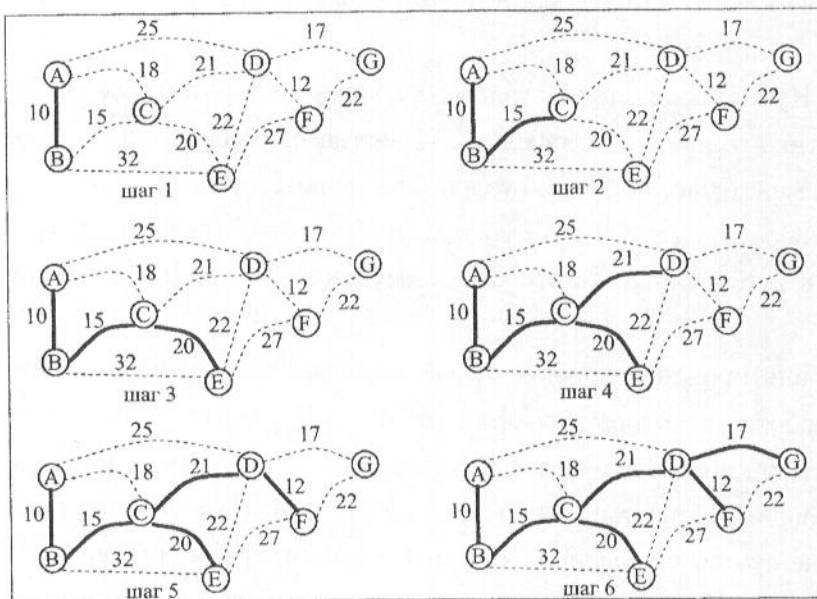


Рис. 2.3. Построение минимального остова для сети дорог

**Обоснование алгоритма.** Сначала покажем, что алгоритм строит действительно остов. На первом шаге дерево  $T_1$  содержит две вершины, соединенные ребром. Каждый последующий шаг подключает к построенному дереву одну вершину из оставшихся, при

этом она достижима из любой ранее подключенной вершины. За  $n - 2$  шагов будут подключены все  $n$  вершин. Очевидно, что при такой процедуре циклы не могут возникнуть.

Покажем, что алгоритм строит минимальный остов. Для простоты изложения предположим, что решение задачи – минимальный остов  $T^* = (V, E_{T^*})$  – единственно. Покажем, что выбранное на первом шаге ребро  $e_1 = (i_1, j_1)$  входит в  $T^*$ . Предположим противное:  $e_1 \notin E_{T^*}$ . Добавим в  $T^*$  ребро  $e_1$ . В силу утверждения 5 теоремы 2.1 возникнет один цикл. Разорвем этот цикл, удалив из него ребро  $e' \neq e_1$ , инцидентное вершине  $i_1$ . Стоимость полученного нового остова  $T'$  выражается через стоимость  $T^*$  следующим образом:

$$C(T') = C(T^*) + C(e_1) - C(e') \leq C(T^*), \text{ т.к. } C(e_1) \leq C(e'),$$

т. е. новый остов не хуже минимального, что противоречит предположению о единственности минимального остова  $T^*$ . Совершенно аналогично можно показать, что выбранное на втором шаге ребро  $e_2$  входит в  $T^*$ , выбранное на третьем шаге ребро  $e_3$  также входит в  $T^*$  и т.д. до последнего  $(n - 1)$ -го шага.

**Трудоемкость алгоритма Прима.** Основной операцией в алгоритме Прима является нахождение вершины, подключение которой дает минимальный прирост стоимости растущего дерева. Для быстрого поиска такой вершины организуем динамически обновляемый справочник в виде пары одномерных массивов  $COST$  и  $NOM$ , где после  $k$ -го шага  $COST[j]$  – минимальная стоимость подключения  $j$ -й вершины к частичному дереву  $T_k = (V_k, E_k)$ ,  $NOM[j]$  – номер вершины в  $V_k$ , подключение к которой дает эту минимальную стоимость. Вначале (шаг 0)  $COST[j] = C[j, 1]$ ,  $j \neq 1$ , и  $COST[1] = \infty$ ,  $NOM[j] = 1$  для всех  $j > 1$ . Определение подключаемой на  $k$ -м шаге вершины  $j_k$  сводится к нахождению  $\min_j COST[j] = COST[j_k]$ . После этого массивы  $COST$  и  $NOM$  пересчитываются по правилу:

$$COST[j] = \min \{ COST[j], C[j_k, j] \}, \text{ если } j \neq j_k, \quad COST[j_k] = \infty,$$

$$NOM[j] = \begin{cases} NOM[j], & \text{если } COST[j] < C[j_k, j], \\ j_k, & \text{если } COST[j] \geq C[j_k, j]. \end{cases}$$

На первом шаге нахождение минимума из  $n - 1$  чисел требует  $n - 2$  операций сравнений. Пересчет элементов массивов  $COST$  и  $NOM$  потребует еще столько же операций. Итого для выполнения первого шага необходимо  $2(n - 2)$  операций. На  $k$ -м шаге будет выполнено  $2(n - k - 1)$  операций. Тогда трудоемкость алгоритма Прима

$$t_{\text{Прим}}(n) = 2 \sum_{k=1}^{n-1} (n - k - 1) = n(n - 1) = O(n^2).$$

Следовательно, алгоритм Прима имеет квадратичную трудоемкость.



Если граф  $G$  задан весовой матрицей размера  $n \times n$ , то всякий алгоритм построения минимального остова в таком графе будет иметь трудоемкость не менее чем  $O(n^2)$ , поскольку он должен просмотреть все элементы матрицы. Следовательно, в этой ситуации алгоритм Прима имеет неуменияемую по порядку трудоемкость. Если же граф задан списком ребер и соответствующих весов  $L = \{(e_1, C(e_1)) \dots, (e_m, C(e_m))\}$  и  $m$  существенно меньше, чем  $n^2$ , то быстроедействие алгоритма Прима далеко от оптимального. Другими словами, алгоритм недостаточно эффективен в применении к графам, слабо насыщенным ребрами. В этой ситуации лучше использовать другой алгоритм построения минимального остова – алгоритм Краскала.

### 2.2.2. Алгоритм Краскала

Идея этого алгоритма заключается в последовательном пополнении списка ребер, образующих минимальный остов, начиная с самого «дешевого» ребра. Упорядочим ребра  $e_i$  по возрастанию их веса:

$$e_1, e_2, \dots, e_m : C(e_1) \leq C(e_2) \leq \dots \leq C(e_m). \quad (2.3)$$

Множество ребер  $E_{T^*}$  минимального остова  $T^*$  наращивается следующей пошаговой процедурой:

шаг 0.  $E_{T^*} = \emptyset$ ;

шаг 1.  $E_{T^*} = e_1$ ;

.....

шаг  $k$ . Если очередное ребро  $e_j$  из последовательности (2.3) не образует цикла с уже включенными в  $E_{T^*}$  ребрами, то  $E_{T^*} = E_{T^*} \cup e_j$ , иначе – перейти к рассмотрению следующего ребра в (2.3).

После  $n - 1$  шагов множество  $E_{T^*}$  будет содержать  $n - 1$  ребро, не образующих цикла, т. е. в силу утверждения 3 теоремы 2.1 остов графа. Доказательство корректности алгоритма практически такое же, как и для алгоритма Прима. Основная операция, выполняемая на каждом шаге (кроме первых двух) в алгоритме Краскала, – проверка на отсутствие цикла в наращиваемом множестве ребер при добавлении нового ребра. Эту операцию выполняют с помощью справочника номеров компонент связности  $NK$ ;  $NK[i]$  – номер компоненты, в которую входит вершина  $i$ . На шаге 0  $NK[i] = i$  (каждая  $i$ -я компонента состоит из одной вершины  $i$ ). Перед тем как включить очередное ребро  $(i, j)$  в  $E_{T^*}$  сначала проверяется условие:  $NK[i] = NK[j]$ . Если это условие выполнено, то при включении ребра  $(i, j)$  возникнет цикл. В противном случае ребро включается, но при этом две компоненты с номерами  $NK[i]$  и  $NK[j]$  надо объединить в одну. Объединение происходит следующим образом. Пусть, например, число вершин в компоненте с

номером  $NK[i]$  меньше, чем в компоненте  $NK[j]$ . Тогда для всех вершин  $l$  из компоненты  $NK[i]$  положим  $NK[l] = NK[j]$ . За время работы алгоритма каждая вершина изменит номер своей компоненты не более  $\log_2 n$  раз (доказать!). Следовательно, число всех коррекций справочника  $NK$  не превосходит  $n \log_2 n$ .

Трудоёмкость первого этапа алгоритма (упорядочивание списка ребер) есть  $O(m \log_2 m)$ . Таким образом, полное число операций в алгоритме Краскала

$$t_{\text{Краскал}}(n, m) = O(m \log_2 m + n \log_2 n).$$

В заключение этого раздела отметим, что наличие каких-то специфических особенностей графа позволяет строить более эффективные алгоритмы построения минимального остова. Так, например, для планарных графов известен алгоритм Тарьяна-Черитона (1976 г.) с оценкой трудоёмкости  $O(n)$ .

## 2.3. Кратчайшие пути в графе

### 2.3.1. Кратчайший путь в графе с неотрицательными весами

Пусть дан связный граф  $G(V, E)$ , каждой дуге  $(i, j)$  которого приписан неотрицательный вес  $l_{ij} \geq 0$ . Если  $(i, j) \notin E$ , будем считать, что  $l_{ij} = +\infty$ . Выделены две вершины  $s, t \in V$ , первую из которых будем называть стартовой, а вторую – финишной. Будем рассматривать простые цепи  $\mu[s, t]$ , соединяющие эти две вершины. Всюду далее для простых цепей будем использовать термин «путь» как более удобный.

Определим длину  $L(\mu[s, t])$  пути  $\mu[s, t]$ :  $L(\mu[s, t]) = \sum_{(i,j) \in \mu[s,t]} l_{ij}$ . Требуется найти кратчайший путь между данными вершинами, т. е. решить задачу

$$\sum_{(i,j) \in \mu[s,t]} l_{ij} \longrightarrow \min_{\mu[s,t]}.$$

Далее длину кратчайшего пути от  $s$  до  $t$  будем обозначать через  $L_{\min}(t)$ .

Рассмотрим итерационный алгоритм решения этой задачи, принадлежащий Е. Дийкстре. Разобьем множество вершин  $V$  на два непересекающихся подмножества: подмножество «красных» вершин  $R$  и подмножество «белых» вершин  $W$ . «Красными» будем называть те вершины  $j$ , для которых уже найдена длина кратчайшего пути от  $s$  до  $j$   $L_{\min}(j)$ . Перед первой итерацией  $R = \{s\}$ ,  $W = V \setminus R$ .

Далее припишем каждой «белой» вершине  $j$  комбинированную метку  $(\lambda(j), p(j))$ . Инициализируем метки следующим образом:  $\forall j \in W \lambda(j) = l_{sj}$ ,  $p(j) = s$ . Каждая итерация состоит из трех шагов:

а) среди «белых» вершин определяется вершина  $j^*$  с минимальной величиной  $\lambda$ :

$$\lambda(j^*) = \min_{j \in W} \lambda(j);$$

б) вершина  $j^*$  переводится из «белых» в «красные», т. е.  $R = R \cup \{j^*\}$ ,  $W = W \setminus \{j^*\}$  и ее метка больше не меняется;

в) метки оставшихся «белых» вершин пересчитаем по правилу

$$\forall j \in W \quad \lambda(j) = \min\{\lambda(j), \lambda(j^*) + l_{jj^*}\}, \quad (2.4)$$

$$p(j) = \begin{cases} p(j), & \text{если значение } \lambda(j) \text{ осталось без изменения} \\ j^*, & \text{в противном случае} \end{cases} \quad (2.5)$$

Итерации продолжаются до тех пор, пока финишная вершина  $t$  не будет «покрашена», т. е.  $\lambda(t) = \min_{j \in W} \lambda(j)$ .

**Обоснование алгоритма.** Корректность алгоритма следует из утверждения:

*А. Если вершина  $j^*$  становится «красной» ( $\lambda(j^*) = \min_{j \in W} \lambda(j)$ ), то длина  $L_{\min}(j^*)$  кратчайшего пути от  $s$  до  $j^*$  равна  $\lambda(j^*)$ .*

Для доказательства нам потребуются следующие свойства меток:

- 1) значение  $\lambda(j)$  есть длина некоторого пути от  $s$  до  $j$ ;
- 2) если  $r \in R$ , а  $j \in W$ , то  $\lambda(j) \leq \lambda(r) + l_{rj}$ .

Свойство 1 вытекает непосредственно из построения алгоритма. Свойство 2 можно пояснить следующим образом. Пусть на некоторой итерации вершина  $r$  стала «красной». Тогда для каждой вершины  $j \in W$  величина  $\lambda(j)$  пересчитывалась по правилу (2.5). Поэтому, начиная с этой итерации  $\lambda(j) \leq \lambda(r) + l_{rj}$ , а на последующих итерациях  $\lambda(j)$ , очевидно, не возрастает.

Далее нам еще потребуется следующее очевидное свойство кратчайшего пути: *любой участок кратчайшего пути есть кратчайший путь между соответствующими концевыми вершинами этого участка.*

Доказательство утверждения А проводится индукцией по номеру итерации. Если на первой итерации  $\lambda(j^*) = l_{sj^*} = \min_j \lambda_j = \min_j l_{sj}$ , то, очевидно, кратчайший путь от  $s$  до  $j^*$  состоит из одного ребра  $(s, j^*)$  и  $L_{\min}(j^*) = \lambda(j^*)$ . Предположим, что утверждение справедливо для всех итераций с номерами  $2, 3, \dots, k-1$ . Покажем его справедливость для итерации  $k$ . Пусть  $\lambda(j^*) = \min_{j \in W} \lambda_j$  и  $\mu^*$  – кратчайший путь от  $s$  до  $j^*$ . Двигаясь по этому пути от  $j^*$  к  $s$ , найдем на нем первую «красную» вершину  $r$  (см. рис. 2.4). Подсчитаем длину пути  $\mu^*$ :

$$\begin{aligned} L[\mu^*] &= \text{длина участка от } s \text{ до } r + l_{rj} + \Delta = L_{\min}(r) \text{ (по свойству кратч. пути)} + l_{rj} + \Delta = \\ &= \lambda(r) \text{ (по предположению индукции)} + l_{rj} + \Delta \geq \lambda(j) + \Delta \geq \lambda(j) \geq \lambda(j^*). \end{aligned} \quad (2.6)$$

По свойству 1 меток  $\lambda(j^*)$  есть длина некоторого пути от  $s$  до  $j^*$ . Но этот путь не может быть меньше кратчайшего пути, поэтому в (2.6) может быть только строгое равенство  $L[\mu^*] = \lambda(j^*)$ .  $\square$



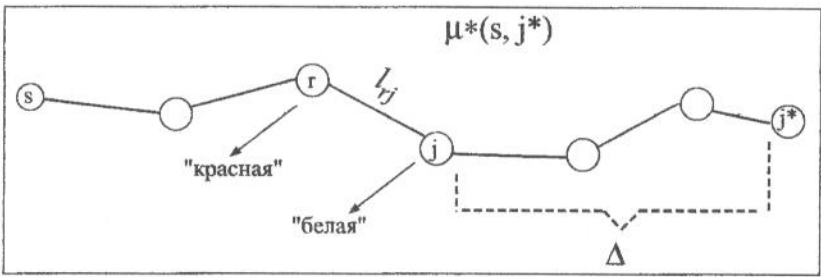


Рис. 2.4. К доказательству корректности алгоритма Дийкстры

**Восстановление кратчайшего пути.** В задаче представляет интерес нахождение не столько длины кратчайшего пути между данными вершинами  $s$  и  $t$ , сколько самого пути. Для этой цели воспользуемся второй компонентой метки – значением  $p(j)$ , которое, как нетрудно заметить, есть не что иное, как номер предпоследней вершины на пути от  $s$  до  $j$ . Таким образом, предпоследняя вершина на кратчайшем пути от  $s$  до  $t$  есть  $j_1 = p(t)$ , предпоследняя вершина на кратчайшем пути от  $s$  до  $j_1$  есть  $j_2 = p(j_1)$ , и т.д. Двигаясь таким образом от финишной вершины к стартовой, можно восстановить весь кратчайший путь.

**Трудоёмкость алгоритма Дийкстры.** Максимальное число итераций в алгоритме равно  $n - 1$ . На  $k$ -й итерации необходимо найти минимальную метку  $\lambda(j)$  среди  $n - k$  меток; это требует  $n - k - 1$  операций сравнения. Пересчет меток у оставшихся  $n - k - 1$  «белых» вершин требует одну операцию сложения плюс одну операцию сравнения на одну вершину, т. е. всего  $2 \cdot (n - k - 1)$  операций. Таким образом,

$$t_{\text{Дийкстра}}(n) = \sum_{k=1}^{n-1} [(n - k - 1) + 2(n - k - 1)] = 3 \frac{(n - 1)(n - 2)}{2} = O(n^2).$$

Алгоритм Дийкстры работает корректно при положительных весах ребер. Если в графе есть ребра отрицательного веса, то алгоритм может выдать неверный ответ.

Это можно показать на простом примере графа (рис. 2.5). Действительно,  $\lambda(a) = 6$ ,  $\lambda(t) = 5$ ,  $\lambda(t)$  – минимальная метка. Алгоритм будет утверждать, что длина кратчайшего пути от  $s$  до  $t$  равна  $\lambda(t) = 5$ , что неверно.

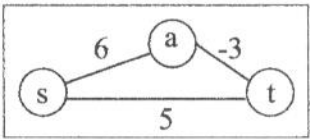


Рис. 2.5.

2.3.2. Матрица кратчайших путей (алгоритм Флойда)

Во многих задачах исследования операций требуется нахождение кратчайших путей между всеми парами вершин. Многократное ( $C_n^2 = n(n - 1)/2$  раз) использование для этой цели алгоритма Дийкстры не очень эффективно: трудоёмкость растёт как  $O(n^4)$ . Излагаемый ниже алгоритм Флойда имеет трудоёмкость  $O(n^3)$ . Кроме того, этот алгоритм применим как к неотрицательным, так и к произвольным весам ребер.

Пусть граф  $G$  задан матрицей весов  $W = [w_{ij}]$ . Полагаем  $w_{ii} = 0$  для всех  $i = 1, \dots, n$  и  $w_{ij} = +\infty$ , если в графе отсутствует дуга  $(i, j)$ . Пусть матрица  $W^k = [w_{ij}^k]$ ,  $k = 1, \dots, n$ , обладает следующим свойством: элемент  $w_{ij}^k$  — это длина пути от  $i$  до  $j$ , кратчайшего среди всех путей  $\mu[i, j]$ , внутренние вершины которых принадлежат множеству  $\{1, 2, \dots, k\}$ . Обозначим этот условно кратчайший путь как  $\mu_{кр}^k(i, j)$ . Из данного определения следует, что элемент  $w_{ij}^n$  матрицы  $W^n$  — это длина *безусловно* кратчайшего пути между вершинами  $i$  и  $j$ .

Для матрицы  $W^{k+1} = [w_{ij}^{k+1}]$  справедливо соотношение:

$$w_{ij}^{k+1} = \min\{w_{ij}^k, w_{i,k+1}^k + w_{k+1,j}^k\}. \quad (2.7)$$

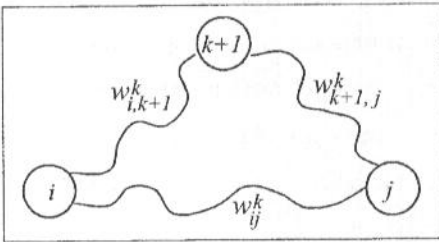


Рис. 2.6. К обоснованию (2.7)

Действительно, для любого пути  $\mu^{k+1}(i, j)$  от  $i$  до  $j$ , внутренние вершины которого принадлежат множеству  $\{1, 2, \dots, k, k+1\}$ , есть только две альтернативы: а) путь не содержит в качестве внутренней вершины вершину  $k+1$  и б) путь содержит вершину  $k+1$ . В случае а) кратчайший путь  $\mu_{кр}^{k+1}(i, j)$  совпадает с  $\mu_{кр}^k(i, j)$ , а в случае б)

он складывается из двух частей: кратчайший путь  $\mu_{кр}^k(i, k+1)$  от вершины  $i$  до вершины  $k+1$  плюс кратчайший путь  $\mu_{кр}^k(k+1, j)$  от вершины  $k+1$  до вершины  $j$ . Эти альтернативы поясняются на рис. 2.6. Выбор из альтернатив а) и б) наилучшей и приводит к (2.7). Формула (2.7) представляет собой рекуррентное соотношение, связывающее матрицу  $W^{k+1}$  с матрицей  $W^k$ . В качестве начального условия берется матрица  $W^0 = W$ . Последовательно вычисляя матрицы  $W^k$  по формуле (2.7) для  $k = 1, 2, \dots$ , мы за  $n$  итераций получим матрицу  $W^n$ .

Описанный алгоритм будет корректно работать только при одном условии: в графе  $G$  не должно быть циклов (для орграфа — контуров) отрицательной длины. Проверку на отсутствие отрицательных циклов можно осуществить в самом алгоритме, основываясь на следующем очевидном соображении: если на  $k$ -й итерации для некоторой вершины  $i$  выполнено  $w_{ii}^k < 0$ , то в графе есть цикл отрицательной длины. В этом случае работу алгоритма надо прерывать.

Для того чтобы после окончания работы алгоритма иметь возможность быстро найти кратчайший путь между любой парой вершин, будем на  $k$ -й итерации вместе с матрицей  $W^k$  строить матрицу  $P^k = [p_{ij}^k]$ . Вначале полагаем  $p_{ij}^0 = i$  для всех  $i, j = 1, \dots, n$ . Далее, на  $(k+1)$ -й итерации полагаем  $p_{ij}^{k+1} = p_{ij}^k$ , если  $w_{ij}^{k+1} = w_{ij}^k$ , и  $p_{ij}^{k+1} = p_{k+1,j}^k$ , если  $w_{ij}^{k+1} = w_{i,k+1}^k + w_{k+1,j}^k$ . Таким образом,  $p_{ij}^k$  — это номер предпоследней вершины на кратчайшем пути  $\mu_{кр}^k(i, j)$ . Матрица  $P^n$  играет ту же роль, что и метки  $p(j)$  в алгоритме Дейкстры. С помощью этой матрицы кратчайший путь  $\mu_{кр}^k(i, j)$  определяется

следующим образом:  $\mu_{np}^k(i, j) = (i, \dots, j_3, j_2, j_1, j)$ , где  $j_1 = p_{ij}^n$ ,  $j_2 = p_{ij_1}^n$ ,  $j_3 = p_{ij_2}^n, \dots$

Для реализации алгоритма Флойда не надо, разумеется, хранить в памяти компьютера всю последовательность матриц  $W^0, W^1, \dots, W^n$ : все итерации можно проводить «внутри» одной матрицы  $W = W^0$  (или ее копии, если нежелательно «портить» исходную матрицу весов). Это же соображение справедливо и для матриц  $P^k$ . Программная реализация алгоритма будет содержать три вложенных цикла:

```
{Программа FLOYD}
for i:=1 to n do for j:=1 to n do P[i,j]:=i; {инициализация матрицы P}
for k:=1 to n do
  for i:=1 to n do
    begin
      for j:=1 to n do
        begin
          W[i,j] := min(W[i,j], W[i,k]+W[k,j]);
          if W[i,j] > W[i,k]+W[k,j] then P[i,j]:=P[k,j];
        end;
      if W[i,i]<0 then EXIT;
    end.
```

При такой реализации трудоемкость алгоритма оценивается как  $t_{\text{Флойд}}(n) = O(n^3)$ .

## 2.4. Сетевые графики

Многие крупные проекты, такие, как строительство дома, разработка автоматизированной системы бухгалтерского учета, обучение в университете, можно разбить на большое число различных операций (работ). Некоторые из этих работ могут выполняться одновременно, другие – только последовательно: одна работа после окончания другой.

Задача управления объектом состоит в том, чтобы обеспечить его своевременное завершение с учетом времени, необходимого для выполнения каждой работы, и определенных взаимосвязей, характеризующих последовательность их выполнения. Если будет задержана одна работа, то это может привести к задержке выполнения всего проекта.

Каждый проект можно представить в виде некоторого орграфа, называемого *сетевым графиком*. Для этого зададим:

- 1) перечень всех работ проекта;
- 2) время, необходимое для выполнения каждой работы;
- 3) перечни работ, непосредственно предшествующих каждой работе.

Представим теперь каждую работу в виде дуги орграфа, построенного по следующему правилу: если некоторая работа представляется дугой  $(i, j)$ , то в вершину  $i$  входят только дуги, представляющие работы, непосредственно предшествующие данной работе. При необходимости введем фиктивные дуги, не представляющие никакой работы. В качестве примера рассмотрим следующий абстрактный проект (табл. 2.1):

Таблица 2.1

Работа	Предшествующие работы	Время выполнения ( $t$ )
A	—	10
B	A	9
C	A	6
D	A	18
E	B, C	10
F	B, C, D	7
G	E, F	8

Чтобы обеспечить предшествование работ  $B$  и  $C$  работе  $F$ , необходимо ввести фиктивную работу и соответствующую ей фиктивную дугу  $(2, 3)$ , как показано на рис. 2.7. Время выполнения фиктивных работ всегда полагается равным нулю.

Граф, изображающий отношения предшествования между работами проекта, называется *сетевым графиком*. Вершины сетевого графика называются *событиями*. Говорят, что событие *произошло*, если все работы, которые отображаются дугами, входящими в соответствующую вершину, полностью завершены. Сетевой график не может содержать контуров. В противном случае проект никогда не может быть закончен. Отсутствие контуров в сетевом графике позволяет пронумеровать события  $0, 1, \dots$  таким образом, чтобы для каждой дуги  $(i, j)$  соблюдать условие  $i < j$ . Обычно сетевой график изображают так, чтобы в нем было ровно одно событие, в которое не входит ни одной дуги, и ровно одно событие, из которого не выходит ни одной дуги. Эти события называются *начальным* и *конечным* событиями соответственно.

Обозначим через  $t(i, j)$  время выполнения работы  $(i, j)$ . Для каждого события  $i$  пусть  $E(j)$  обозначает *наиболее ранний* из возможных сроков его наступления, а  $L(j)$  — *наиболее поздний* срок появления события  $j$ , еще допускающий своевременное окончание всего проекта. Идею расчета величин  $E(j)$  поясним на примере фрагмента сетевого графика, приведенного на рис. 2.8(а).

В этом примере продолжительности работ, отображенных дугами  $(p, i)$ ,  $(q, i)$ ,  $(r, i)$ , равны соответственно  $t(p, j) = 8$ ,  $t(q, j) = 12$ ,  $t(r, j) = 10$ , а самые ранние сроки наступления событий  $E(p) = 10$ ,  $E(q) = 8$ ,  $E(r) = 12$ . Наиболее ранний срок

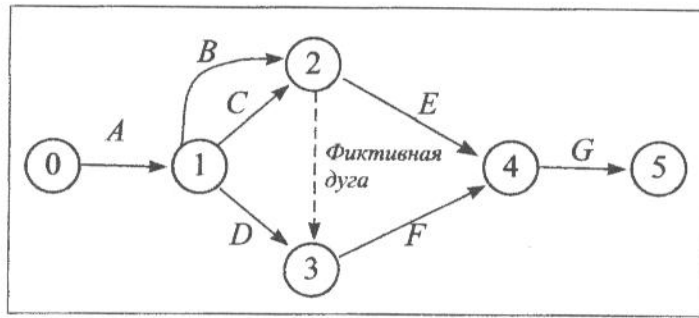


Рис. 2.7. Сетевой график с фиктивной дугой

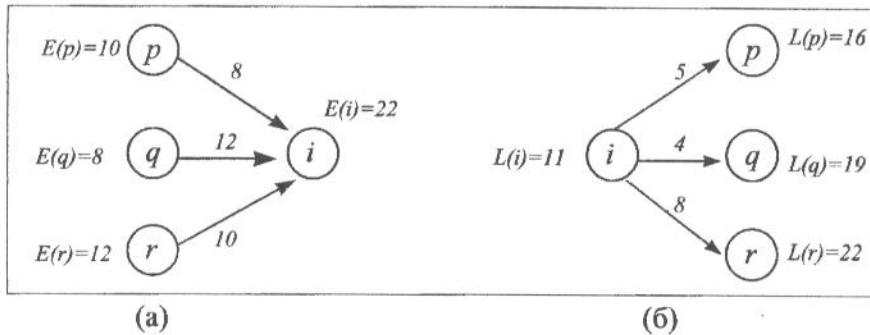


Рис. 2.8. Расчет раннего и позднего сроков наступления события

завершения работы  $(p, j)$  очевидно равен  $E(p) + t(p, j) = 10 + 8 = 18$ . Аналогично для работ  $(q, j)$  и  $(r, j)$  эти сроки равны  $E(q) + t(q, j) = 20$ ,  $E(r) + t(r, j) = 22$ . Поскольку событие  $i$  может наступить только при завершении *всех* этих работ, то  $E(j) = \max \{E(p) + t(p, j), E(q) + t(q, j), E(r) + t(r, j)\} = 22$ . В общем случае формула расчета  $E(j)$  имеет вид:

$$E(j) = \max_{(i,j) \in U} \{E(i) + t(i, j)\}, \quad (2.8)$$

где  $U$  – множество дуг сетевого графика.

Рекуррентное соотношение (2.8) лежит в основе алгоритма расчета наиболее ранних сроков наступления событий. Пусть дан сетевой график  $G = (V, U)$ , где  $V = \{0, 1, 2, \dots, n\}$  – множество событий,  $U = \{(i, j)\}$  – множество дуг (работ). Будем считать, что нумерация событий такая, что для каждой работы  $(i, j)$  выполняется условие  $i < j$ .

#### Алгоритм расчета наиболее ранних сроков наступления событий

Шаг 0. Положить  $E(0) = 0$ .

Шаг 1. Для  $j = 1, 2, \dots, n$  вычислить

$$E(j) = \max_{i: (i,j) \in U} \{E(i) + t(i, j)\}.$$



Поясним работу алгоритма на примере сетевого графика, изображенного на рис. 2.7:

$$\begin{aligned} E(0) &= 0; & E(1) &= 0 + 10 = 10; & E(2) &= \max\{E(1) + 6, E(1) + 9\} = 19; \\ E(3) &= \max\{E(1) + 8, E(2) + 0\} = 19; & E(4) &= \max\{E(2) + 10, E(3) + 7\} = 29; \\ E(5) &= 29 + 8 = 37. \end{aligned}$$

Таким образом, весь проект не может завершиться раньше, чем через 37 единиц времени.

Вычислим теперь поздние сроки наступления событий. Рассмотрим рис. 2.8(б). Событие  $i$  предшествует ровно трем событиям:  $p$ ,  $q$  и  $r$ . Время наступления события  $i$  не должно превышать 11; в противном случае событие  $p$  будет задержано и произойдет позднее  $L(j) = 16$ . Аналогично время события  $i$  не должно превышать 15, так как  $L(q) - t(i, q) = 19 - 4 = 15$ , и не должно превышать 14, так как  $L(r) - t(i, r) = 22 - 8 = 14$ .

В общем случае

$$L(i) = \min_{j: (i,j) \in U} \{L(j) - t(i, j)\}. \quad (2.9)$$

Опираясь на рекуррентное соотношение (2.9), можно описать алгоритм расчета наиболее поздних сроков событий.

#### Алгоритм расчета наиболее поздних сроков наступления событий

Шаг 0. Положить  $L(n)$  равным заданному времени завершения проекта (во всех реальных ситуациях  $L(n) \geq E(n)$ ).

Шаг 1. Для  $i = n - 1, n - 2, \dots, 0$  вычислить

$$L(i) = \min_{j: (i,j) \in U} \{L(j) - t(i, j)\}.$$

Наиболее ранний срок  $E(i)$  наступления события  $i$  можно интерпретировать как *длину пути наибольшей длины* от начального события к событию  $i$ , а разность  $L(i) - E(i)$  — как длину пути наибольшей длины от события  $i$  к конечному событию. Отметим еще, что если  $L(n) \geq E(n)$ , то  $L(i) \geq E(i)$  для всех событий  $i$ . Путь максимальной длины от начального события к конечному называется *критическим*. Длина критического пути равна минимальному времени, за которое может быть выполнен весь проект.

**Трудоёмкость расчета самых ранних и самых поздних сроков.** Алгоритм расчета  $E(i)$  требует одной операции сложения для каждой дуги сетевого графика и одной операции на максимум для каждого события, кроме начального. Алгоритм расчета величин  $L(i)$  требует одной операции вычитания и одной операции сравнения на минимум для каждого события, кроме конечного. Отсюда следует, что полная трудоёмкость обоих алгоритмов оценивается величиной  $O(m + n)$ , где  $m$  — число дуг (работ).

Рассмотрим некоторую работу  $(i, j)$ . Эта работа может начаться не ранее  $E(i)$  и должна закончиться не позднее  $L(j)$ . Таким образом, без задержки окончания проекта

на выполнение работы  $(i, j)$  можно выделить не более  $L(j) - E(i)$  единиц времени. Следовательно, при выполнении этой работы можно допустить максимальную задержку  $\Pi(i, j) = L(j) - E(i) - t(i, j) \geq 0$ . Величина  $\Pi(i, j)$  называется *полным резервом* работы  $(i, j)$ .

Величина  $C(i, j) = E(j) - E(i) - t(i, j)$  называется *свободным резервом времени* работы  $(i, j)$ . Свободный резерв времени равен максимальной задержке выполнения работы  $(i, j)$ , не влияющей на выполнение последующих работ.

*Независимый резерв времени* работы  $(i, j)$  определяется как максимальная задержка, которую можно допустить при выполнении работы  $(i, j)$  без введения дополнительных ограничений на любую другую работу проекта:  $H(i, j) = E(j) - E(i) - t(i, j)$ . Отрицательное значение независимого резерва означает, что любая задержка в выполнении работы приведет к дополнительным ограничениям на выполнение других работ.

Все три вида временных резервов находятся в следующем соотношении:

$$\Pi(i, j) \geq C(i, j) \geq H(i, j). \quad (2.10)$$

**Задача.** Построить сетевой график и провести полный расчет всех временных характеристик для проекта, представленного в табл. 2.2, при условии, что плановое время выполнения проекта равно минимально возможному времени выполнения всего проекта.

Таблица 2.2

Работа	Предшествующие работы	Время выполнения ( $t$ )
A	—	12
B	—	10
C	A	8
D	A	13
E	A, B	10
F	A, B	11
G	D, F	15
H	C, E,	14
I	E, F, H	16

## 2.5. Потоки в сетях

Поток определяет способ пересылки некоторых объектов из одного пункта в другой. Потоки, например, возникают при транспортировке готовой продукции от завода до распределительного склада, при движении людей от места проживания к местам работы, при передаче информационных массивов по коммуникационным сетям и т.п.

Несмотря на разнообразие ситуаций, связанных с потоками, в них возникает целый ряд достаточно общих задач. Примерами таких задач могли бы служить максимизация суммарного объема перевозок в некоторой системе из одной точки в другую; максимизация объема передачи информации в единицу времени, минимизация стоимости пересылки через некоторую систему определенного объема информации (или материальных ресурсов). Далее будем рассматривать следующую математическую модель, связанную с задачами подобного типа.

Сетью с пропускными способностями дуг будем называть объект  $N = \langle V, U, s, t, q \rangle$ , где  $V$  и  $U$  – множество вершин и дуг соответственно некоторого ориентированного графа,  $s \in V$  – вершина, для которой множество входящих дуг пусто,  $t$  – вершина, для которой множество выходящих дуг пусто,  $q : U \rightarrow \mathbb{R}^+$  – функция, которая каждой дуге  $u = (i, j)$  ставит в соответствие неотрицательное число  $q(i, j)$ , называемое *пропускной способностью*. Вершина  $s$  называется *источником*, а вершина  $t$  – *стоком* сети. Будем предполагать, что сеть содержит только один источник и только один сток. Пример сети приведен на рис. 2.9(а). *Потоком* величины  $F$  в сети  $N$  называется функция  $f : U \rightarrow \mathbb{R}$ ,

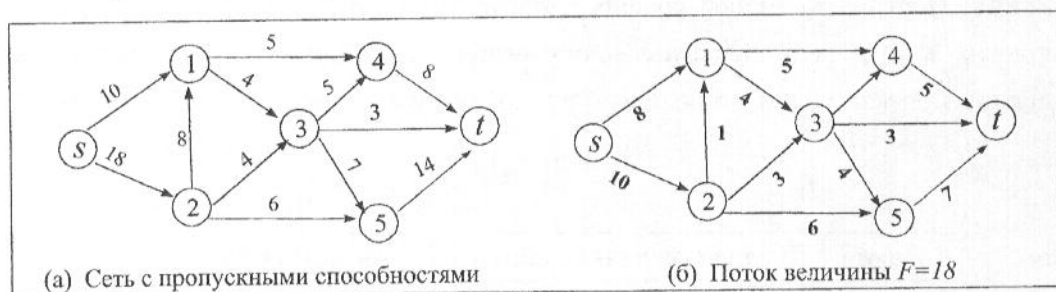


Рис. 2.9. Пример сети

удовлетворяющая условиям:

$$1) 0 \leq f(i, j) \leq q(i, j); \quad (2.11)$$

$$2) \forall j \in V : \sum_{(i,j) \in U} f(i, j) - \sum_{(j,k) \in U} f(j, k) = \begin{cases} -F, & \text{если } j = s, \\ F, & \text{если } j = t, \\ 0, & \text{в остальных случаях.} \end{cases} \quad (2.12)$$

Число  $f(i, j)$  будем называть величиной потока на дуге  $(i, j)$ . Условие (2.11) задает естественные ограничения на величины потоков по каждой дуге. Интерпретация условия (2.12) следующая: из источника  $s$  выходит суммарный поток величины  $F$  (первая сумма при  $j = s$  равна нулю, вторая равна  $-F$ ), далее он распространяется по дугам сети, при этом для каждой промежуточной вершины  $j \neq s, t$ , входящий в  $j$  суммарный поток, равен суммарному потоку, выходящему из  $j$ , и в конечном счете весь поток величины  $F$  входит в вершину-сток  $t$  (первая сумма при  $j = t$  равна  $F$ , вторая равна нулю). Условие



(2.12) называют *уравнением баланса*. На рис. 2.9(б) приведен один из возможных потоков величины  $F = 18$ . Далее величину потока  $f$  будем обозначать как  $\text{val}(f)$ . Очевидно  $\text{val}(f) = \sum_{(i,t) \in U} f(i,t) = \sum_{(s,j) \in U} f(s,j)$ .

Как уже говорилось выше, многие практические задачи сводятся к организации в сети потока максимальной величины, т. е. к оптимизационной задаче

$$\text{val}(f^*) = \max_f \text{val}(f). \quad (2.13)$$

Для решения задачи (2.13) введем понятие *разреза*. Разрезом сети  $N$ , отделяющим  $s$  от  $t$ , называется разбиение  $R = \{V_s, V_t\}$  множества вершин сети  $V$  на два подмножества  $V_s \cup V_t = V$ ,  $V_s \cap V_t = \emptyset$ , при котором  $s \in V_s$ ,  $t \in V_t$ . Множество дуг  $(i,j)$ , у которых начальная вершина  $i \in V_s$ , а конечная вершина  $j \in V_t$ , будем называть *прямыми дугами* разреза  $R$  и обозначать  $\vec{U}_R$ . Аналогично вводится понятие множества *обратных дуг* разреза:  $\overleftarrow{U}_R = \{(k,l) : k \in V_t, l \in V_s\}$ . Понятие разреза поясняется на рис. 2.10.

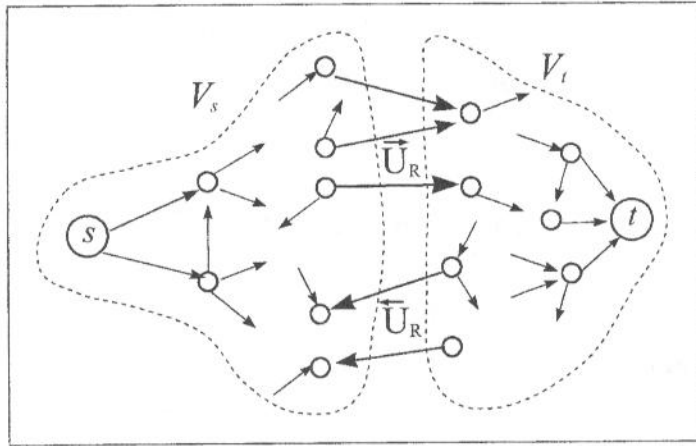


Рис. 2.10. Прямые и обратные дуги разреза  $R = (V_s, V_t)$

*Пропускной способностью* разреза  $R$  называется величина

$$Q(R) = \sum_{(i,j) \in \vec{U}_R} q(i,j).$$

Разрез с минимальной пропускной способностью будем называть *минимальным разрезом*.

**Теорема 2.3.** В сети для любого потока  $f$  и любого разреза  $R$  выполняется равенство

$$\text{val}(f) = \sum_{(i,j) \in \vec{U}_R} f(i,j) - \sum_{(i,j) \in \overleftarrow{U}_R} f(i,j). \quad (2.14)$$

*Доказательство.* Приведем «физическое» доказательство теоремы. Из источника выходит поток величины  $\text{val}(f)$ . Определенное количество, а именно  $\sum_{(i,j) \in \vec{U}_R} f(i,j)$

единиц потока, проходит по прямым дугам разреза из  $V_s$  в  $V_t$ . Другое количество,  $\sum_{(i,j) \in \vec{U}_R} f(i,j)$ , проходит по обратным дугам из  $V_t$  в  $V_s$ . Но в  $V_t$  должно остаться  $\text{val}(f)$  единиц потока, так как именно это количество «втекает» в сток  $t$  и остается там. Следовательно, разность этих количеств должна равняться величине потока.  $\square$

**Следствие 1.** Для любого потока  $f$  и любого разреза  $R$

$$\text{val}(f) \leq Q(R).$$

Действительно,

$$\text{val}(f) = \sum_{(i,j) \in \vec{U}_R} f(i,j) - \sum_{(i,j) \in \vec{U}_R} f(i,j) \leq \sum_{(i,j) \in \vec{U}_R} f(i,j) \leq \sum_{(i,j) \in \vec{U}_R} q(i,j) = Q(R). \quad \square$$

**Следствие 2.**  $\max_f \text{val}(f) \leq \min_R Q(R)$ .

**Следствие 3.** Если для некоторого потока  $f^*$  и некоторого разреза  $R^*$  выполняется равенство  $\text{val}(f^*) = Q(R^*)$ , то  $\text{val}(f^*) = \max_f \text{val}(f)$  и  $Q(R^*) = \min_R Q(R)$ .

Доказательство. Пусть  $\text{val}(f^*) = Q(R^*)$ . Из следствия 2 получаем

$$\text{val}(f^*) \leq \max_f \text{val}(f) \leq \min_R Q(R) \leq Q(R^*).$$

Поскольку крайние значения цепочки неравенств совпадают, то все выражения в ней равны между собой.  $\square$

**Метод Форда – Фалкерсона построения максимального потока.** Идея метода заключается в следующем. Пусть в сети  $\mathcal{N} = \langle V, U, s, t, q \rangle$  уже есть некоторый поток  $f : U \rightarrow \mathbb{R}^+$  величины  $\text{val}(f)$ .

Шаг 1. Положить  $V_0 = \{s\}$ ,  $\tilde{i} = s$ .

Шаг 2. Если существует вершина  $j \notin V_0$ , для которой  $f(\tilde{i}, j) < q(\tilde{i}, j)$  или  $f(j, \tilde{i}) > 0$ , то включить вершину  $j$  в  $V_0$  и положить  $\tilde{i} := j$ . Повторять этот шаг до тех пор, пока множество  $V_0$  нельзя будет расширять дальше.

Теперь могут возникнуть два случая в зависимости от того, будет ли  $t \in V_0$  или  $t \notin V_0$ .

Случай 1.  $t \in V_0$ . В соответствии с шагом 2 включение  $t \in V_0$  означает следующее: существует такая «неориентированная» цепь  $\mu$ , идущая из  $s$  в  $t$ , что для каждой дуги  $(i, j) \in \mu$ , ориентированной в цепи в прямом направлении (прямой дуги),  $f(i, j) < q(i, j)$ , а для каждой дуги  $(k, i) \in \mu$ , идущей в обратном направлении (обратной дуги),  $f(k, i) > 0$ . Такую цепь  $\mu$  будем называть *аугментальной*<sup>1</sup> цепью потока.

<sup>1</sup>От *augmen* (лат.) – увеличение, приращение.

Пусть

$$\delta_f = \min_{(i,j) \in \mu} \{q(i,j) - f(i,j)\}, \quad (i,j) - \text{прямая дуга},$$

$$\delta_b = \min_{(k,i) \in \mu} \{f(k,i)\}, \quad (k,i) - \text{обратная дуга},$$

$$\delta = \min\{\delta_f, \delta_b\}.$$

Если теперь величину  $\delta$  добавить к потоку на всех прямых дугах цепи и вычесть на всех обратных дугах, то в результате получится новый допустимый поток  $f'$ , на  $\delta$  единиц больший, чем предыдущий:  $\text{val}(f') = \text{val}(f) + \delta$ . Это очевидно в силу того, что добавление  $\delta$  к потоку на прямых дугах не приводит к превышению ни одной из пропускных способностей этих дуг (так как  $\delta \leq \delta_f$ ), а вычитание  $\delta$  из потока на обратных дугах не может сделать поток на этих дугах отрицательным (так как  $\delta \leq \delta_b$ ).

Используя новый поток  $f'$ , можно затем повторить шаги 1 и 2, увеличивая каждый раз поток вдоль аугментальных цепей.

Случай 2.  $t \notin V_0$ . В соответствии с шагом 2 это означает, что не существует ни одной аугментальной цепи от  $s$  до  $t$ . Обозначим построенный к данному моменту поток как  $f^*$ . Построим отделяющий  $s$  от  $t$  разрез  $R^* = (V_s^*, V_t^*)$ , где  $V_s^* = V_0$ ,  $V_t^* = V \setminus V_0$ . На прямых дугах  $(i,j)$  этого разреза  $f^*(i,j) = q(i,j)$ , а на обратных дугах  $(k,l)$  разреза  $f^*(k,l) = 0$ . По теореме 2.3

$$\text{val}(f^*) = \sum_{(i,j) \in \vec{U}_R^*} f^*(i,j) - \sum_{(k,l) \in \overleftarrow{U}_R^*} f^*(k,l) = \sum_{(i,j) \in \vec{U}_R^*} q(i,j) - \sum_{(k,l) \in \overleftarrow{U}_R^*} 0 = Q(R^*).$$

Следовательно, в силу следствия 3 поток  $f^*$  максимальный, а разрез  $R^*$  – минимальный.

Фактически, изложенный метод представляет собой конструктивное доказательство следующего утверждения о максимальном потоке и минимальном разрезе:

**Теорема 2.4 (Форд – Фалкерсон).** *Для любой сети*

$$\max_f \text{val}(f) = \min_R Q(R).$$

Алгоритм начинает работу с произвольного допустимого потока (можно взять и нулевой поток  $f \equiv 0$ ), затем стремится увеличить величину потока с помощью систематического поиска всех возможных аугментальных цепей потока от  $s$  к  $t$ . Поиск аугментальной цепи осуществляется с помощью *расстановки пометок* на вершинах сети. Пометки указывают, вдоль каких дуг может быть увеличен поток и насколько. Как только будет найдена одна из таких цепей, поток вдоль нее увеличивают до максимально возможного значения, все пометки в вершинах стираются и вновь полученный поток используется

в качестве исходного при новой расстановке пометок. Алгоритм заканчивает работу и дает максимальный поток, если нельзя найти ни одну аугментальную цепь.

**Алгоритм расстановки пометок.** Вершина может находиться в одном из трех состояний: вершина помечена и просмотрена, вершина помечена и не просмотрена, вершина не помечена. Пометка произвольной вершины  $j$  состоит из двух частей и имеет вид  $[+i, \delta(i)]$  или  $[-i, \delta(i)]$ . Часть  $+i$  пометки первого типа означает, что поток допускает увеличение потока на дуге  $(i, j)$ . Часть  $-i$  пометки другого типа означает, что поток может быть уменьшен вдоль дуги  $(j, i)$ . Присвоение пометки вершине  $t$  соответствует нахождению аугментальной цепи от  $s$  до  $t$ .

Шаг 0. Положить  $f(i, j) = 0$  для всех дуг  $(i, j)$ .

Шаг 1. Присвоить вершине  $s$  пометку  $[+s, \infty]$ .

Шаг 2. Взять некоторую помеченную непросмотренную вершину  $i$ ; пусть ее пометка будет  $[\pm k, \delta(i)]$ .

(1) Каждой непомеченной вершине  $j$  такой, что существует дуга  $(i, j)$  и  $f(i, j) < q(i, j)$ , присвоить пометку  $[+i, \delta(j)]$ , где  $\delta(j) = \min\{\delta(i), q(i, j) - f(i, j)\}$ .

(2) Каждой непомеченной вершине  $j$  такой, что существует дуга  $(j, i)$  и  $f(j, i) > 0$ , присвоить пометку  $[-i, \delta(j)]$ , где  $\delta(j) = \min\{\delta(i), f(j, i)\}$ . Обозначить каким-либо способом, что вершина  $i$  просмотрена.

Шаг 3. Повторять шаг 2 до тех пор, пока либо вершина  $t$  будет помечена, и тогда перейти к шагу 4, либо  $t$  не будет помечена и никаких других пометок нельзя будет расставить; в этом случае алгоритм заканчивает работу с максимальным потоком.

Шаг 4. Положить  $x = t$ .

Шаг 5. Если пометка в вершине  $x$   $[+z, \delta(z)]$ , то  $f(z, x) := f(z, x) + \delta(z)$ ; если пометка в вершине  $x$   $[-z, \delta(z)]$ , то  $f(x, z) := f(x, z) - \delta(z)$ .

Шаг 6. Если  $z = s$ , то стереть все пометки и вернуться к шагу 1, чтобы вновь расставлять пометки, используя уже улучшенный поток, найденный на шаге 5. Если  $z \neq s$ , то  $x := z$  и вернуться к шагу 5.

Относительно трудоемкости алгоритма расстановки пометок можно отметить следующее. Если для всех дуг  $(i, j)$  пропускные способности  $q(i, j)$  — целые числа, то  $\delta(t)$  — целое, и поток вдоль аугментальной цепи увеличивается, по крайней мере, на единицу. Следовательно, максимальный поток будет получен за конечное число шагов<sup>1</sup>. Однако число итераций будет существенно зависеть от порядка выбора аугментальных цепей (точнее, от порядка, в котором помечаются вершины); при неудачном выборе это число может быть равным величине максимального потока. Например, для сети на рис. 2.11

<sup>1</sup>Это утверждение становится, вообще говоря, неверным, если  $q(i, j)$  иррациональны. Форд и Фалкерсон привели пример, когда их алгоритм строит поток, величина которого стремится к  $1/4$  величины максимального потока при бесконечном увеличении числа шагов.

величина максимального потока равна 2000. Этот поток можно получить из начального нулевого потока за две итерации: сначала взять аугментальную цепь  $(s, 1), (1, t)$  и пропустить по ней поток величины 1000, а затем по цепи  $((s, 2), (2, t))$  пропустить поток такой же величины. Это удачный выбор аугментальных цепей.

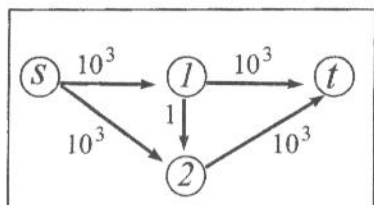


Рис. 2.11. Пример медленной работы алгоритма пометок

Неудачный выбор такой: при начальном нулевом потоке выбирается цепь  $\mu_1 = ((s, 1)(1, 2), (2, t))$  и увеличивается вдоль нее поток на единицу. Затем берется цепь  $\mu_2 = ((s, 2)(1, 2), (1, t))$  и вдоль нее увеличивается поток на единицу. Потом снова берется цепь  $\mu_1$ , увеличивается вдоль нее поток на единицу, затем берется  $\mu_2$  и т.д. При таком поочередном выборе цепей  $\mu_1$  и  $\mu_2$  максимальный поток будет получен за 2000 итераций. В общем случае алгоритм пометок не гарантирует удачного выбора аугментальных цепочек.

Удачную попытку разработать алгоритм решения задачи о максимальном потоке, трудоемкость которого не зависит от величин пропускных способностей дуг, впервые предприняли Эдмондс и Карп. Ими была доказана следующая

**Теорема 2.5 (Эдмондс и Карп).** *Если в алгоритме пометок Форда-Фалкерсона каждое увеличение потока выполняется вдоль кратчайшей по числу дуг аугментальной цепи, то максимальный поток можно получить не более чем после  $m(n-2)/2$  приращений величины потока, где  $m$  – число дуг, а  $n$  – число вершин в сети.*

Эдмондс и Карп предложили модификацию алгоритма пометок, в которой кратчайшая аугментальная цепь на каждой итерации отыскивается за  $O(m)$  операций. Таким образом, трудоемкость их алгоритма оценивается как  $O(m^2n)$ .

Впоследствии этот результат был улучшен. К настоящему времени известны алгоритмы (Е. А. Диниц, А. В. Карзанов, V. M. Malhotra) нахождения максимального потока с трудоемкостью  $O(n^3)$ .



# Глава 3

## Оптимальное распределение ограниченного ресурса

Задача распределения ограниченного ресурса в одной из самых простых постановок формулируется следующим образом. Пусть у некоторой организации имеется определенный ресурс (деньги, машины, комплекты оборудования, сырье и т.п.) в количестве  $A$  единиц. Имеется  $n$  потребителей этого ресурса, например филиалы данной организации. Если потребителю  $j$  выделить некоторое количество  $x \leq A$  ресурса, то дополнительный доход, который может принести этот филиал, оценивается величиной  $f_j = f_j(x)$ . Требуется так распределить имеющийся ресурс, чтобы получить максимальный суммарный доход. Нетрудно составить математическую модель этой задачи:

$$F(x_1, \dots, x_n) = \sum_{j=1}^n f_j(x_j) \longrightarrow \max_{x_1, \dots, x_n}, \tag{3.1}$$

$$\sum_{j=1}^n x_j \leq A, \quad x_1, \dots, x_n \geq 0, \tag{3.2}$$

где  $x_j$  – количество ресурса, выделенного  $j$ -му потребителю. Относительно функций  $f_j(x)$  могут быть сделаны разные предположения, касающиеся непрерывности, гладкости, монотонности и т.п. В практике функции  $f_j(x)$  часто задаются не в виде аналитических зависимостей, а в табулированном виде, т. е. таблицей следующего типа:

$x$	0	$\Delta$	$2\Delta$	...	$(k-1)\Delta$
$f_j(x)$	$f_{j,0}$	$f_{j,1}$	$f_{j,2}$	...	$f_{j,k-1}$

где  $f_{ji} = f_j(i \cdot \Delta)$  (впрочем, шаг табулирования по  $x$  может быть и непостоянным). На переменные  $x_j$  могут быть наложены условия дискретности. В такой ситуации применение классических методов решения задачи (3.1)–(3.2) как задачи на условный экстремум невозможно. Требуется использовать совсем другие идеи. Об одной такой идее – динамическом программировании – пойдет речь в следующем разделе.

### 3.1. Метод динамического программирования

Метод динамического программирования представляет собой вычислительную схему решения широкого класса оптимизационных (и не только!) задач. В рассматриваемом

методе слово «динамическое» означает, что процесс определения оптимального допустимого плана  $x^* = (x_1^*, \dots, x_n^*)$  рассматривается как  $n$ -шаговый процесс принятия решений: на первом шаге надо оптимально выбрать  $x_1$ , на втором –  $x_2$  и т.д. Трудность заключается в том, что при выборе на каждом шаге (кроме последнего) необходимо учитывать его будущие последствия для оставшихся шагов.

Сформулируем достаточно очевидный общий принцип, лежащий в основе динамического программирования и впервые четко сформулированный Р. Беллманом.

**Принцип оптимальности.** *Каково бы ни было состояние процесса перед очередным  $k$ -м шагом, надо выбирать решение  $x_k$  так, чтобы выигрыш на данном шаге плюс максимальный выигрыш на всех последующих шагах был максимальным.*

Обычно применение динамического программирования включает в себя разбиение исходной задачи на однотипные аналогичные подзадачи, в каждой из которых решение связано с решением предыдущей задачи, имеющей меньший размер. Эта связь между задачами выражается в виде некоторого *рекуррентного* соотношения, называемого уравнением Беллмана.

Рассмотрим теперь задачу распределения ресурса (3.1) – (3.2) при условии, что ресурс и варианты его распределения целочисленны, т. е.

$$F(x_1, \dots, x_n) = \sum_{j=1}^n f_j(x_j) \longrightarrow \max_{x_1, \dots, x_n}, \quad \sum_{j=1}^n x_j \leq A, \quad A, x_1, \dots, x_n \in \mathbb{Z}^+. \quad (3.3)$$

Будем рассматривать процедуру распределения ресурса как  $n$ -шаговый процесс: на 1-м шаге выделяется  $x_1$  единиц ресурса первому потребителю, на 2-м шаге –  $x_2$  единиц из оставшихся  $A - x_1$  единиц – второму потребителю и т.д. Определим состояние процесса распределения ресурса перед  $k$ -м шагом как величину  $y$ , равную количеству единиц ресурса, оставшихся после первых  $k - 1$  шагов:  $y = A - x_1 - \dots - x_{k-1}$ .

Обозначим через  $\hat{F}_k(y)$  максимальный суммарный доход, полученный от потребителей с номерами  $k, k + 1, \dots, n$  при условии, что для распределения между ними осталось  $y$  единиц ресурса:

$$\hat{F}_k(y) = \max_{x_k + \dots + x_n \leq y} \{f_k(x_k) + \dots + f_n(x_n)\}, \quad x_k, \dots, x_n \in \mathbb{Z}^+.$$

В силу принципа оптимальности решение  $x_k$  на  $k$ -м шаге надо выбрать таким, чтобы доход на этом шаге плюс максимальный доход от последующих шагов (он равен по определению  $\hat{F}_{k+1}(y - x)$ , так как после  $k$ -го шага состояние процесса распределения изменится с  $y$  на  $y - x_k$ ) был максимален, т. е. равен  $\hat{F}_k(y)$ . Отсюда получаем

$$\hat{F}_k(y) = \max_{x_k=0,1,\dots,y-1,y} \{f_k(x_k) + \hat{F}_k(y - x_k)\}. \quad (3.4)$$

Значение  $x_k$ , при котором достигается максимум в задаче (3.4), обозначим  $\hat{x}_k(y)$ .

Полученное равенство (3.4) называется *основным рекуррентным уравнением динамического программирования* для рассматриваемой задачи. При  $k = n$  (последний шаг, за которым уже ничего не распределяется) уравнение (3.4) выглядит очень просто:

$$\hat{F}_n(y) = \max_{x_n=0,1,\dots,y-1,y} \{f_n(x_n)\} = f_n(y), \tag{3.5}$$

а  $\hat{x}_n(y) = y$ . Иными словами, если перед последним шагом осталось  $y$  единиц ресурса, то весь этот ресурс надо отдать  $n$ -му потребителю для того, чтобы получить от него максимально возможный доход  $f_n(y)$  (напомним, что функция дохода предполагается неубывающей).

Покажем, как использовать соотношения (3.4) - (3.5) для решения исходной задачи (3.1) - (3.2) с ограничением целочисленности  $A \in \mathbb{Z}^+$ ,  $x_j \in \mathbb{Z}^+$ ,  $j = 1, \dots, n$ . Фактически нам надо получить  $\hat{F}_1(A)$  - максимум суммарного дохода от распределения  $A$  единиц ресурса между потребителями  $1, 2, \dots, n$ . Однако сразу получить этот максимум трудно. Поступим следующим образом. Сначала по формуле (3.5) определим  $\hat{F}_n(y)$  при всех значениях  $y$  от 0 до  $A$ . Затем, используя (3.4), вычислим  $\hat{F}_{n-1}(y)$  для всех  $y = 0, 1, \dots, A - 1, A$  и соответствующее значение  $\hat{x}_{n-1}(y)$ . Зная  $\hat{F}_{n-1}(y)$ , найдем  $\hat{F}_{n-2}(y)$  для всех  $y$  от 0 до  $A$  и т.д. до  $\hat{F}_1(A)$ . Таким образом, процесс решения задачи разворачивается от конца к началу: сначала планируется последний  $n$ -й шаг, затем  $(n - 1)$ -й и т.д.

Поясним эти вычисления на конкретном примере. Рассмотрим задачу с четырьмя потребителями, между которыми надо распределить  $A = 8$  единиц некоторого ресурса. Функции  $f_j$  заданы в табл. 3.1. Определим сначала  $\hat{F}_4(y)$  при всех  $y = 0, 1, \dots, 8$ . Используя (3.5), запишем  $\hat{F}_4(y)$  в предпоследний столбец табл. 3.2. В последний столбец запишем соответствующие  $\hat{x}_4(y)$ .

Таблица 3.1

$x$	0	1	2	3	4	5	6	7	8
$f_1(x)$	0	10	13	16	19	22	26	29	31
$f_2(x)$	0	9	12	18	21	24	27	27	27
$f_3(x)$	0	11	14	17	20	25	28	28	28
$f_4(x)$	0	12	16	18	22	27	30	32	35



Вычислим теперь  $\hat{F}_3(y)$ , используя уравнение (3.4):

$$\hat{F}_3(0) = \max_{x_3=0} \{f_3(x_3) + \hat{F}_4(0 - x_3)\} = 0, \quad \hat{x}_3(0) = 0,$$

$$\hat{F}_3(1) = \max_{x_3=0,1} \{f_3(x_3) + \hat{F}_4(1 - x_3)\} = \max \begin{cases} f_3(0) + \hat{F}_4(1 - 0) = 0 + 12 \\ f_3(1) + \hat{F}_4(1 - 1) = 11 + 0 \end{cases} = 12, \quad \hat{x}_3(1) = 0,$$

$$\hat{F}_3(2) = \max_{x_3=0,1,2} \{f_3(x_3) + \hat{F}_4(2 - x_3)\} = \max \begin{cases} f_3(0) + \hat{F}_4(2 - 0) = 0 + 16 \\ f_3(1) + \hat{F}_4(2 - 1) = 11 + 12 = 23 \\ f_3(2) + \hat{F}_4(2 - 2) = 14 + 0 \end{cases} = 23, \quad \hat{x}_3(2) = 1,$$

Аналогично вычисляются  $\hat{F}_3(3), \hat{F}_3(4), \dots, \hat{F}_3(8)$  и соответствующие  $\hat{x}_3(3), \dots, \hat{x}_3(8)$ . Продолжая вычисления, мы получим значения  $\hat{F}_2(y)$  при всех  $y = 0, 1, \dots, 8$ . При этом для вычисления  $\hat{F}_2(y)$  будет использоваться лишь столбец со значениями  $\hat{F}_3(y)$ . На последнем этапе вычислим только  $\hat{F}_1(8) = \max_{x_1=0,1,\dots,8} \{f_1(x_1) + \hat{F}_2(8 - x_1)\} = 58, \quad \hat{x}_1(8) = 1$ . Результаты всех вычислений записаны в табл. 3.2.

Таблица 3.2

$y$	$\hat{F}_1(y)$	$\hat{x}_1(y)$	$\hat{F}_2(y)$	$\hat{x}_2(y)$	$\hat{F}_3(y)$	$\hat{x}_3(y)$	$\hat{F}_4(y)$	$\hat{x}_4(y)$
0			0	0	0	0	0	0
1			12	0	12	0	12	1
2			23	0	23	1	16	<span style="border: 1px solid black;">2</span>
3			32	1	27	1	18	3
4			36	1	30	<span style="border: 1px solid black;">2</span>	22	4
5			41	3	33	1	27	5
6			45	3	38	1	30	6
7			48	<span style="border: 1px solid black;">3</span>	41	1	32	7
8	58	<span style="border: 1px solid black;">1</span>	51	3	44	2	35	8

Максимальное значение в задаче (3.3) равно 58 и достигается при  $x_1^* = \hat{x}_1(8) = 1$ . После первого шага для потребителей 2, 3 и 4 остается  $y = 8 - x_1^* = 7$  единиц ресурса. Эти 7 единиц должны быть оптимально распределены между ними. В столбце  $\hat{x}_2$  видно, что при  $y = 7$  надо взять оптимальное решение  $x_2^* = \hat{x}_2(7) = 3$ . Для потребителей 3 и 4 останется после этого  $y = 8 - x_1^* - x_2^* = 4$  единицы ресурса. Оптимальное решение на третьем шаге  $x_3^* = \hat{x}_3(4) = 2$ . Последний шаг приводит к  $x_4^* = \hat{x}_4(8 - 1 - 3 - 2) = 2$ . Оптимальные решения выделены в табл. 3.2 рамками. Полученное распределение  $x^* = (1, 3, 2, 2)$  является решением задачи (3.3).

В ряде случаев целесообразно считать целевой функцией распределения ограниченного ресурса не суммарный доход, а доход, получаемый от самого «слабого» потреби-

теля, т. е.  $F(x_1, \dots, x_n) = \min_{j=1, \dots, n} \{f_j(x_j)\}$ . В качестве примера рассмотрим следующую задачу.

У студента имеется  $A$  дней на подготовку к досрочной сдаче экзаменов по  $n$  предметам. Реально оценивая свои силы, студент полагает, что, потратив  $x$  дней на подготовку по  $j$ -му предмету, он получит на экзамене оценку  $f_j(x)$ . Поскольку в российских вузах результат сессии оценивается не по сумме набранных баллов, а по минимальной полученной оценке (с оценками 5, 5, 5, 2 студент считается в деканате «двоечником»), то задача оптимального распределения ресурса (дни на подготовку) между потребителями (сдаваемые предметы) будет формулироваться так:

$$F(x_1, \dots, x_n) = \min_{j=1, \dots, n} \{f_j(x_j)\} \longrightarrow \max_{x_1, \dots, x_n}, \quad \sum_{j=1}^n x_j \leq A, \quad x_1, \dots, x_n \in \mathbb{Z}^+. \quad (3.6)$$

Для задачи (3.6), так же как и для задачи с аддитивной целевой функцией, справедлив принцип оптимальности Беллмана. Поэтому, применяя описанные выше рассуждения, легко получить необходимое рекуррентное уравнение для  $\hat{F}_k(y)$ :

$$\hat{F}_k(y) = \max_{x_k \leq y, x_k \in \mathbb{Z}^+} \min \left\{ f_k(x_k), \hat{F}_{k+1}(y - x_k) \right\}.$$

Вся вычислительная схема решения этого уравнения остается по сути такой же, как и для уравнения (3.4).

**Трудоемкость алгоритма динамического программирования.** Оценим число операций, необходимых для решения задачи (3.1) - (3.2). Для того чтобы вычислить  $\hat{F}_k(y)$ , надо выполнить  $y + 1$  операций сложения и  $y$  операций сравнения двух чисел, так как максимум в (3.4) находится прямым перебором  $y + 1$  чисел. Тогда заполнение  $k$ -го столбца потребует  $\sum_{y=0}^A (2y + 1) = (A + 1)^2$  операций. Суммируя эту величину по всем столбцам, получим оценку общего числа  $T(n, A)$  операций

$$T(n, A) \leq n(A + 1)^2 = O(nA^2).$$

Можно считать этот алгоритм эффективным или алгоритмом с полиномиальной временной сложностью, если только величина  $A$  ограничена сверху константой. В общем же случае алгоритмы, в степенную оценку трудоемкости которых входит величина, характеризующая размер исходных данных (в данном случае количество единиц целочисленного ресурса), называются *псевдополиномиальными*.

## 3.2. Задача о рюкзаке

Рассмотрим теперь применение метода динамического программирования для решения следующей задачи. Пусть имеется  $n$  предметов  $P_1, \dots, P_n$ , которые надо загрузить

в некоторую емкость, условно называемую рюкзаком. Каждый предмет  $P_j$  характеризуется стоимостью  $c_j > 0$  руб. и весом  $q_j > 0$  кг. Грузоподъемность рюкзака равна  $Q$ , причем  $Q < \sum_{j=1}^n q_j$ , т. е. все предметы в рюкзак поместить невозможно. Желательно отобрать для загрузки те предметы, суммарная стоимость которых будет максимальна среди всех вариантов загрузки. Математическая модель оптимального выбора предметов для загрузки, очевидно, следующая:

$$\sum_{j=1}^n c_j x_j \longrightarrow \max_{x_1, \dots, x_n}, \quad (3.7)$$

$$\sum_{j=1}^n q_j x_j \leq Q, \quad x_j \in \{0, 1\}, \quad j = 1, \dots, n. \quad (3.8)$$

Здесь  $x_j = 1$ , если предмет  $P_j$  решено поместить в рюкзак, и  $x_j = 0$  в противном случае. Задача (3.7) - (3.8) называется задачей о рюкзаке с булевыми переменными.

Применим для решения задачи (3.7) - (3.8) метод динамического программирования. Пусть  $\hat{F}_k(y)$  - максимум стоимости, который можно получить от загрузки среди предметов  $P_k, P_{k+1}, \dots, P_n$  при условии, что в рюкзаке осталось свободного места на  $y$  кг ( $y$  - состояние процесса). После  $k$ -го шага принятия решения  $x_k$  (загружать  $P_k$  или нет) состояние процесса станет  $y - q_k x_k$ . Рекуррентное уравнение динамического программирования будет иметь следующий вид:

$$\hat{F}_k(y) = \max_{\substack{q_k x_k \leq y, \\ x_k = 0, 1}} \left\{ c_k x_k + \hat{F}_{k+1}(y - q_k x_k) \right\}. \quad (3.9)$$

Схема решения задачи по уравнению (3.9) не требует пояснений. Объем вычислений для определения  $\hat{F}_k(y)$  равен 3 операциям, и трудоемкость алгоритма  $T(n, Q) \leq 3n(Q+1) = O(nQ)$ .

Метод легко обобщается на более общий случай целочисленных переменных. Пусть для каждого предмета  $P_j$  задана величина  $m_j$  - имеющееся количество предметов данного типа. В этом случае решение  $x_k$ , принимаемое на  $k$ -м шаге, лежит в диапазоне  $\{0, 1, \dots, m_j\}$  и расчеты по рекуррентному уравнению принципиально не меняются.

Рассмотрим теперь задачу о рюкзаке с целочисленными переменными и с большим числом ограничений.

$$\sum_{j=1}^n c_j x_j \longrightarrow \max_{x_1, \dots, x_n},$$

$$\sum_{j=1}^n q_j x_j \leq Q, \quad \sum_{j=1}^n v_j x_j \leq V, \quad x_j \in \mathbb{Z}^+, \quad j = 1, \dots, n.$$

Величины  $v_j$  можно интерпретировать как объемы предметов  $P_j$ , а  $V$  - как объем рюкзака. Метод динамического программирования применим и для этой задачи. Только

теперь состояние пошагового процесса принятия решения определяется двумя величинами:  $y$  – оставшийся запас по весу и  $z$  – оставшийся запас по объему. Функция условного максимума  $\hat{F}_k$  будет функцией двух переменных  $\hat{F}_k = \hat{F}_k(y, z)$ , а рекуррентное уравнение для нее примет следующий вид:

$$\hat{F}_k(y, z) = \max_{\substack{q_k x_k \leq y, \\ v_k x_k \leq z, \\ x_k \in \mathbb{Z}^+}} \left\{ c_k x_k + \hat{F}_{k+1}(y - q_k x_k, z - v_k x_k) \right\}.$$

Нетрудно оценить трудоемкость алгоритма в этом случае:  $T(n, Q, V) = O(nQV)$ . Если же в задаче будет  $L$  линейных ограничений с правыми частями  $Q_1, \dots, Q_L$ , то оценка трудоемкости становится показательной по  $L$ :  $T \geq nQ_{\min}^L$ , где  $Q_{\min} = \min_l \{Q_l\}$ <sup>1</sup>. Этот рост трудоемкости в методе динамического программирования называется «проклятием размерности»; он делает практически невозможным использование метода в реальных задачах при  $L \geq 10$ .

В заключение этой главы приведем задачи для самостоятельного решения.

**Задача 1.** Пусть в задаче (3.1) – (3.2) функции доходов всех потребителей  $f_j(x)$  одинаковы:  $f_1(x) = f_2(x) = \dots = f_n(x) = f(x)$ . Если  $f(x)$  – вогнутая неубывающая функция,  $f(0) = 0$ , то оптимальное распределение ресурса таково: все  $A$  единиц ресурса надо отдать одному (любому) потребителю. Доказать.

**Задача 2** (С.Г. Волченков). Пусть дана произвольная последовательность чисел  $a_1, \dots, a_n$ . Требуется удалить из этой последовательности минимальное число элементов так, чтобы оставшаяся подпоследовательность была монотонно убывающей. Предложить алгоритм динамического программирования для решения задачи.

**Задача 3.** Дано уравнение  $b_1 x_1 + \dots + b_n x_n = B$ ,  $b_1, \dots, b_n, B \in \mathbb{Z}^+$ ,  $x_j \in \{0, 1\}$ . Предложить алгоритм, определяющий, существует или нет решение  $x_1^*, \dots, x_n^*$  этого уравнения<sup>2</sup>.

**Задача 4.** Разработать алгоритм решения задачи

$$F(x_1, \dots, x_n) = \sum_{j=1}^{n-1} f_j(x_j, x_{j+1}) \longrightarrow \min, \quad x_j \in \{0, 1, \dots, k\}, \quad j = 1, \dots, n.$$

<sup>1</sup>Заметим, что и затраты на память растут с такой же скоростью.

<sup>2</sup>Обратим внимание читателя, что в задаче не требуется находить максимум или минимум какой-либо целевой функции. Тем не менее здесь можно применить схему динамического программирования.

# Глава 4

## Элементы теории массового обслуживания

### 4.1. Задачи теории массового обслуживания

При исследовании операций часто приходится сталкиваться с работой своеобразных систем, называемых системами массового обслуживания (СМО). Примеры СМО – предприятие, выполняющее заказы; станок, обрабатывающий детали; компьютер, решающий задачи; магазин, обслуживающий покупателей, и т.д. Заявки, поступающие на обслуживание в СМО (заказы, детали, задачи, покупатели и т.д.), образуют *поток заявок*. Каждая СМО состоит из какого-то числа обслуживающих единиц, которые называются каналами обслуживания. Каналами могут быть: линии связи, рабочие точки, кассиры, продавцы, лифты, компьютеры и др. СМО могут быть одноканальными и многоканальными. Рассмотрим основные задачи, связанные с исследованием СМО<sup>1</sup>.

Всякая СМО предназначена для обслуживания какого-то потока заявок, поступающих в случайные моменты времени. Обслуживание заявки продолжается, вообще говоря, случайное время, после чего канал освобождается и готов к приему следующей заявки. Случайный характер потока заявок и времен обслуживания приводит к тому, что в какие-то периоды времени на входе СМО скапливается излишне большое число заявок (они либо становятся в очередь, либо покидают СМО необслуженными); в другие же периоды СМО будет работать с недогрузкой или вообще простаивать.

Процесс работы СМО представляет собой случайный процесс с дискретными состояниями и непрерывным временем; состояние СМО меняется скачком в моменты появления каких-то событий (или прихода новой заявки, или окончания обслуживания, или момента, когда заявка, которой надоело ждать, покидает очередь).

Предмет теории массового обслуживания – построение математических моделей, связывающих заданные условия работы СМО (число каналов, их производительность, правила работы, характер потока заявок) с интересующими нас характеристиками (показателями эффективности) СМО, описывающими с той или другой точки зрения ее способность справляться с потоком заявок. В качестве таких показателей могут при-

---

<sup>1</sup>Изложение материала этой главы в основном следует работе [2].



меняться разные величины, например: среднее число заявок, обслуживаемых СМО в единицу времени; среднее число занятых каналов; среднее число заявок в очереди и среднее время ожидания обслуживания; вероятность того, что число заявок в очереди превысит какое-то значение, и т.д. В этой главе рассматриваются задачи определения в аналитическом виде некоторых из указанных характеристик для относительно простых СМО и практически не изучаются задачи, связанные с принятием решений по оптимизации самих систем (например, по числу каналов обслуживания, режимов работы СМО и т.п.).

Системы массового обслуживания делятся на типы (или классы) по ряду признаков. Первое деление: СМО с отказами и СМО с очередью. В СМО с отказами заявка, поступившая в момент, когда все каналы заняты, получает отказ, покидает СМО и в дальнейшем процессе обслуживания не участвует. Примеры СМО с отказами встречаются в телефонии: заявка на разговор, пришедшая в момент, когда все каналы связи заняты, получает отказ и покидает СМО необслуженной. В СМО с очередью заявка, пришедшая в момент, когда все каналы заняты, не уходит, а становится в очередь и ожидает возможности быть обслуженной. На практике чаще встречаются (и имеют большее значение) СМО с очередью; недаром теория массового обслуживания имеет второе название: «теория очередей».

СМО с очередью подразделяются на разные виды в зависимости от того, как организована очередь, ограничена она или не ограничена. Ограничения могут касаться как длины очереди, так и времени ожидания (так называемые «СМО с нетерпеливыми заявками»). При анализе СМО должна учитываться также и «дисциплина обслуживания»: заявки могут обслуживаться либо в порядке поступления (раньше пришла, раньше обслуживается), либо в случайном порядке. Нередко встречается так называемое обслуживание с приоритетом – некоторые заявки обслуживаются вне очереди. Приоритет может быть как абсолютным – когда заявка с более высоким приоритетом «вытесняет» из-под обслуживания заявку с низким (например, пришедший в парикмахерскую клиент высокого ранга прогоняет с кресла обыкновенного клиента), так и относительным – когда начатое обслуживание доводится до конца, а заявка с более высоким приоритетом имеет лишь право на лучшее место в очереди.

Существуют СМО с так называемым многофазовым обслуживанием, состоящим из нескольких последовательных этапов или фаз (например, покупатель, пришедший в магазин, должен сначала выбрать товар, затем оплатить его в кассе, затем получить на контроле).

Кроме этих признаков, СМО делятся на два класса: «открытые» и «замкнутые». В открытой СМО характеристики потока заявок не зависят от того, в каком состоянии сама СМО (сколько каналов занято), а в замкнутой СМО зависят. Например, если один

рабочий обслуживает группу станков, время от времени требующих наладки, то интенсивность потока «требований» со стороны станков зависит от того, сколько их уже неисправно и ждет наладки. Это пример замкнутой СМО. Классификация СМО далеко не ограничивается приведенными их разновидностями, но мы ограничимся ими.

Оптимизация работы СМО может производиться под разными углами зрения: с точки зрения организаторов (или владельцев) СМО или с точки зрения обслуживаемых клиентов. С первой точки зрения желательно «выжать все, что возможно» из СМО и добиться того, чтобы ее каналы были предельно загружены. С точки зрения клиентов желательно всемерное уменьшение очередей. При решении задач оптимизации в теории массового обслуживания существенно необходим «системный подход», полное и комплексное рассмотрение всех последствий каждого решения. Например, с точки зрения клиентов СМО, желательно увеличение числа каналов обслуживания; но ведь работу каждого канала надо оплачивать, что удорожает обслуживание. Построение математической модели позволяет решить оптимизационную задачу о разумном числе каналов с учетом всех факторов «за» и «против».

## 4.2. Формула Литтла

*Потоком событий* называется последовательность однородных событий, следующих одно за другим в случайные моменты времени. Событиями в СМО являются поступления заявок в систему и окончания обслуживания заявок каналами.

Рассмотрим произвольную СМО и связанные с ней два потока: поток заявок, прибывающих в СМО, и поток заявок, покидающих СМО.

Жизнь каждой заявки можно схематично описать так: в некоторый момент времени заявка поступает в очередь, стоит в очереди на обслуживание до момента  $t$ , обслуживается и уходит из очереди. Пусть  $A(t)$  – число заявок, поступивших в систему за промежуток времени  $[0, t]$ ,  $B(t)$  – число заявок, покинувших систему за тот же промежуток  $[0, t]$ . Будем считать, что в любой момент времени может поступить или быть обслужена только одна заявка и  $B(t) < A(t)$ , так как не может быть обслужено заявок больше, чем поступило. Функции  $A(t)$  и  $B(t)$  являются случайными и меняются скачком (увеличиваются на единицу) в моменты прихода заявок ( $A(t)$ ) и ухода заявок ( $B(t)$ ). Вид этих функций показан на рис. 4.1. Очевидно, что для любого момента  $t$  их разность  $c(t) = A(t) - B(t)$  есть не что иное, как *число заявок, находящихся в СМО*. Когда линии  $A(t)$  и  $B(t)$  сливаются, в системе нет заявок.

Рассмотрим работу СМО на достаточно большом интервале времени  $[0, T]$ . Нас будут интересовать следующие характеристики системы на этом промежутке времени:

$L_{\text{сист}}(T)$  – среднее число заявок в системе в произвольный момент времени;

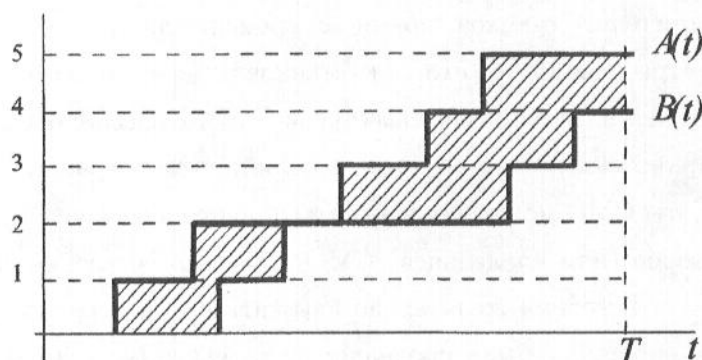


Рис. 4.1. Изменение числа заявок в СМО

$W_{\text{сист}}^{\Sigma}(T)$  — суммарное время нахождения всех заявок в системе;

$W_{\text{сист}}(T)$  — среднее время пребывания одной заявки в системе;

$\lambda_{\text{ин}}(T)$  — среднее число заявок, поступавших за единицу времени (интенсивность потока заявок) на интервале  $[0, T]$ . Среднее число заявок, находящихся в СМО в произвольный момент времени, равно:

$$L_{\text{сист}}(T) = \frac{1}{T} \int_0^T c(t) dt. \quad (4.1)$$

Интеграл в (4.1) есть площадь фигуры между графиками  $A(t)$  и  $B(t)$  на рис. 4.1. Фигура состоит из прямоугольников единичной высоты и с основанием, равным времени пребывания в системе соответствующей заявки. Следовательно, площадь равна суммарному времени  $W_{\text{сист}}^{\Sigma}$  нахождения всех заявок в системе (т. е. в очереди или на обслуживании). Тогда

$$L_{\text{сист}}(T) = \frac{1}{T} W_{\text{сист}}^{\Sigma}(T) = \lambda_{\text{ин}}(T) \frac{W_{\text{сист}}^{\Sigma}(T)}{\lambda(T)T}.$$

Но величина  $\lambda(T)T$  есть среднее число заявок, пришедшее за время  $T$ . Тогда отношение суммарного времени пребывания всех заявок в системе к среднему числу заявок, поступивших в систему есть не что иное, как среднее время пребывания заявки в системе, т. е. величина  $W_{\text{сист}}(T)$ . Таким образом получаем равенство  $L_{\text{сист}}(T) = \lambda(T)W_{\text{сист}}(T)$  или  $W_{\text{сист}}(T) = L_{\text{сист}}(T)/\lambda_{\text{ин}}(T)$ .

Будем считать, что при  $T \rightarrow \infty$  в СМО установился предельный, стационарный режим, т. е. характеристики  $L_{\text{сист}}(T)$ ,  $W_{\text{сист}}(T)$  имеют предельные значения  $N_{\text{сист}}$  и  $W_{\text{сист}}$  соответственно. Тогда  $\lim_{T \rightarrow \infty} \lambda_{\text{ин}}(T) = \lambda_{\text{ин}}$  и окончательно получаем равенство

$$W_{\text{сист}} = L_{\text{сист}}/\lambda_{\text{ин}}. \quad (4.2)$$

Это и есть формула Литтла: для любой СМО, при любом характере потока заявок, при любом распределении времени обслуживания, при любой дисциплине обслуживания

среднее время пребывания заявки в системе равно среднему числу заявок в системе, деленному на среднюю интенсивность потока заявок.

Точно таким же образом выводится и вторая формула Литтла, связывающая среднее время пребывания заявки в очереди  $W_{out}$  и среднее число заявок в очереди  $N_{out}$ :

$$W_{out} = L_{out} / \lambda_{out}. \quad (4.3)$$

Заметим, что в установившемся стационарном режиме  $\lambda_{out} = \lambda_{in} = \lambda$ .

## 4.3. Потоки событий в СМО

### 4.3.1. Простейший поток

Для расчета характеристик СМО требуется формальное описание потока заявок, поступающих в нее. Как правило, достаточно точный расчет характеристик СМО возможен только в случаях, когда поток заявок представляет собой так называемый *простейший поток*. Простейшим называется поток событий (под событием в данном случае понимается поступление заявки), обладающий свойствами *стационарности*, *ординарности*, *отсутствия последствий*. Приведем определения этих понятий.

Рассмотрим случайное событие  $Z_k(t_1, t_2)$ , которое состоит в том, что за интервал времени  $(t_1, t_2)$  в систему поступит ровно  $k$  заявок.

*Ординарность*. Поток событий является ординарным, если вероятность появления двух или более событий за очень короткий, близкий к нулевому интервал времени очень мала по сравнению с вероятностью появления за этот же период одного события. Другими словами, поток заявок является ординарным, если заявки поступают на обслуживание не группами, а по одной. Формально это можно записать следующим образом:

$$\mathcal{P}\{Z_1(t, t+h)\} = \lambda(t)h + o(h).$$

Тогда вероятность поступления на интервале  $(t, t+h)$  более чем одной заявки равна  $o(h)$ .

*Стационарность*. Поток событий является стационарным, если  $\lambda(t) = \text{const}$ . Фактически это означает, что вероятностные характеристики потока не зависят от времени.

*Отсутствие последствий*. Это свойство потока формально выражается следующим образом: для непересекающихся интервалов времени  $(t_1, t_2)$  и  $(t_3, t_4)$  события  $Z_k(t_1, t_2)$  и  $Z_m(t_3, t_4)$  независимы. Иными словами, число заявок, попадающих на один интервал, не зависит от того, сколько заявок попало на другой.

Легко показать, что для простейшего потока интервал  $T$  между соседними событиями имеет **показательное распределение** с плотностью

$$f_T(t) = \lambda e^{-\lambda t} \quad (t > 0).$$



Для случайной величины  $T$ , распределенной по показательному закону с параметром  $\lambda$ , математическое ожидание  $M[T]$  и среднее квадратичное  $\sigma[T]$  равны  $1/\lambda$ .

При показательном распределении интервала между соседними событиями количество событий (в данном случае поступление заявок), происходящих за некоторый интервал времени, представляет собой случайную величину  $\xi$ , распределенную по закону Пуассона: вероятность события  $Z_k(t_1, t_2)$  определяется как

$$p_k(t_2 - t_1) = \frac{\lambda^k (t_2 - t_1)^k}{k!} e^{-\lambda(t_2 - t_1)}.$$

Математическое ожидание случайной величины  $\xi$ , распределенной по закону Пуассона, равно

$$M[\xi] = \sum_{k=0}^{\infty} k p_k(t) = \sum_{k=0}^{\infty} k \frac{\lambda^k t^k}{k!} e^{-\lambda t} = \lambda t,$$

где  $t = t_2 - t_1$  — длина рассматриваемого интервала времени. Следовательно, параметр  $\lambda$  есть среднее число заявок, приходящееся на единицу времени, т. е. интенсивность потока.

Простейший поток играет среди других потоков особую роль, в чем-то подобную роли нормального распределения среди других потоков распределения, а именно: при суммировании (наложении) достаточно большого числа независимых, стационарных и ординарных потоков (сравнимых между собой по интенсивности) получается поток, близкий к простейшему. Отметим также, что сумма  $k$  простейших потоков с интенсивностями  $\lambda_1, \dots, \lambda_k$  является простейшим потоком с интенсивностью  $\lambda = \lambda_1 + \dots + \lambda_k$ .

Математический анализ работы СМО значительно облегчается, если все потоки событий, переводящие систему из состояния в состояние (потоки заявок, потоки «обслуживаний»), являются простейшими (или, что то же самое, стационарными пуассоновскими). Если это свойство нарушается, то математическое описание процесса становится гораздо сложнее и довести его до явных, аналитических формул удастся лишь в редких случаях. Однако все же аппарат «простейшей» теории массового обслуживания может пригодиться для приближенного описания работы СМО даже в тех случаях, когда потоки событий не простейшие. Во многих случаях для принятия разумного решения по организации работы СМО вовсе и не требуется точного знания всех ее характеристик, зачастую достаточно и приближенного, ориентировочного.

### 4.3.2. Процесс «гибели и размножения»

Рассмотрим некоторую систему, которая может в каждый момент времени  $t$  находиться в одном из  $n$  состояний  $S_1, \dots, S_n$ . Пусть  $E_k(t)$  — случайное событие, состоящее в том, что в момент времени  $t$  система находится в состоянии  $S_k$  и  $p_k(t) = \mathcal{P}\{E_k(t)\}$  — веро-



ятность этого события. Будем предполагать, что потоки событий, вызывающие переход системы из одного состояния в другое – простейшие со следующими параметрами:

1.  $\mathcal{P}\{E_{k+1}(t+h) \mid E_k(t)\} = \lambda_k h + o(h)^2$ .
2.  $\mathcal{P}\{E_{k-1}(t+h) \mid E_k(t)\} = \mu_k h + o(h)$ .
3.  $\mathcal{P}\{E_m(t+h) \mid E_k(t)\} = o(h)$ , если  $|m-k| > 1$ .
4.  $\mathcal{P}\{E_k(t+h) \mid E_k(t)\} = 1 - (\lambda_k + \mu_k)h + o(h)$ .

На рис. 4.2 показан граф возможных переходов из одного состояния в другое.

Определим вероятность  $p_k(t)$ . Придадим  $t$  малое приращение  $h$  и посчитаем  $p_k(t+h)$ . По формуле полной вероятности

$$\begin{aligned} p_k(t+h) &= p_{k-1}(t)\mathcal{P}\{E_k(t+h) \mid E_{k-1}(t)\} + p_k(t)\mathcal{P}\{E_k(t+h) \mid E_k(t)\} + \\ &+ p_{k+1}(t)\mathcal{P}\{E_k(t+h) \mid E_{k+1}(t)\} = p_{k-1}(t)(\lambda_{k-1}h + o(h)) + p_k(t)[1 - (\lambda_k + \mu_k)h + o(h)] + \\ &+ p_{k+1}(t)(\mu_{k+1}h + o(h)) = p_{k-1}\lambda_{k-1}h + p_k(t)[1 - (\lambda_k + \mu_k)h] + p_{k+1}(t)\mu_{k+1}h + o(h). \end{aligned}$$

Преобразуем последнее равенство к виду

$$\frac{p_k(t+h) - p_k(t)}{h} = p_{k-1}(t)\lambda_{k-1} - p_k(t)(\lambda_k + \mu_k) + p_{k+1}(t)\mu_{k+1} + \frac{o(h)}{h}.$$

Переходя к пределу при  $h \rightarrow 0$ , вместо разностного уравнения получим дифференциальное

$$\frac{dp_k(t)}{dt} = p_{k-1}(t)\lambda_{k-1} - p_k(t)(\lambda_k + \mu_k) + p_{k+1}(t)\mu_{k+1}. \quad (4.4)$$

Справедлива следующая теорема теории случайных процессов: если число состояний

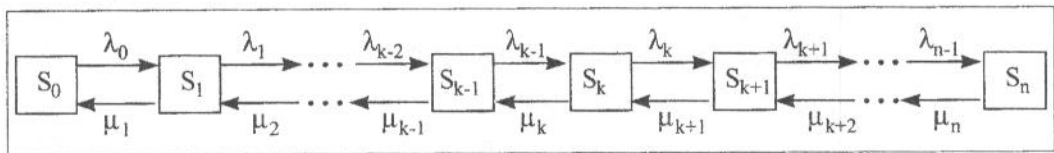


Рис. 4.2. Граф переходов

системы конечно и из каждого из них можно за конечное число шагов перейти в любое другое, то существуют пределы

$$\lim_{t \rightarrow \infty} p_k(t) = p_k.$$

Эти предельные значения называются *финальными вероятностями*. Очевидно, что  $\sum_k p_k = 1$ . Величина  $p_k$  показывает, какую долю времени система проводит в состоянии  $S_k$ .

<sup>2</sup> $\mathcal{P}(A \mid B)$  означает вероятность события  $A$  при условии, что произошло событие  $B$ .

Условия этой теоремы очевидным образом выполнены в нашем случае. Таким образом, при  $t \rightarrow \infty$  уравнение (4.4) переходит в алгебраическое уравнение

$$0 = p_{k-1}\lambda_{k-1} - p_k(\lambda_k + \mu_k) + p_{k+1}\mu_{k+1}$$

или

$$\underbrace{(\lambda_{k-1}p_{k-1} - \mu_k p_k)}_{z_{k-1}} - \underbrace{(\lambda_k p_k - \mu_{k+1} p_{k+1})}_{z_k} = 0. \quad (4.5)$$

Из (4.5) очевидно следует

$$z_k = z_{k-1} = z_{k-2} = \dots = z_0 = 0 \Rightarrow \mu_{k+1} \cdot p_{k+1} = \lambda_k \cdot p_k, \quad k \geq 1.$$

Тогда

$$p_{k+1} = \frac{\lambda_k}{\mu_{k+1}} p_k = \frac{\lambda_k \lambda_{k-1}}{\mu_{k+1} \mu_k} p_{k-1} = \dots = \frac{\lambda_k \lambda_{k-1} \dots \lambda_0}{\mu_{k+1} \mu_k \dots \mu_1} p_0. \quad (4.6)$$

Значение  $p_0$  определяется из условия  $\sum_{k=0}^n p_k = 1$ . Опуская промежуточные выкладки, приведем окончательный результат

$$p_0 = \left( 1 + \frac{\lambda_0}{\mu_1} + \frac{\lambda_1 \lambda_0}{\mu_2 \mu_1} + \dots + \frac{\lambda_{n-1} \dots \lambda_0}{\mu_n \dots \mu_1} \right)^{-1}. \quad (4.7)$$

Все остальные финальные вероятности  $p_k$  выражены через  $p_0$  в (4.6).

Приведенная схема процесса «гибели и размножения» полезна при решении многих задач массового обслуживания. Заметим, что процесс подобного типа относится к так называемым *марковским случайным процессам с дискретными состояниями и непрерывным временем*. Нестрого это понятие можно сформулировать следующим образом. Случайный процесс, протекающий в системе, называется марковским, если для любого момента  $t_0$  вероятностные характеристики процесса в будущем зависят только от его состояния в данный момент  $t_0$  и не зависят от того, когда и как система пришла в это состояние.

## 4.4. Расчет марковских СМО

### 4.4.1. Многоканальная СМО с отказами

Рассмотрим  $n$ -канальную СМО с отказами, т. е. пришедшая в систему заявка, застав все  $n$  каналов занятыми, получает отказ в обслуживании и покидает систему необслуженной. Будем рассматривать эту систему при следующих предположениях:

- 1) входящий поток заявок – простейший с интенсивностью  $\lambda$ ;
- 2) для потока заявок, обслуживаемых одним каналом, время обслуживания показательное с параметром  $\mu$  ( $\mu$  – интенсивность обслуживания, т. е. среднее число заявок, обслуживаемых каналом в единицу времени при условии, что канал никогда не простаивает из-за отсутствия заявок).

Требуется найти финальные вероятности состояний СМО, а также характеристики ее эффективности:

$Q_a$  – абсолютную пропускную способность, т. е. среднее число заявок, обслуживаемых в единицу времени;

$Q_o$  – относительную пропускную способность, т. е. среднюю долю пришедших заявок, обслуживаемых системой;

$P_{отк}$  – вероятность отказа, т. е. того, что заявка покинет СМО необслуженной;

$\bar{k}$  – среднее число занятых каналов.

Работа такой СМО укладывается в схему процесса «гибели и размножения»: «размножение» – это поступление заявок в систему, «гибель» – обслуживание заявок, состояния системы нумеруются по числу заявок, находящихся в системе (оно совпадает с числом занятых каналов). Очевидно, что переход  $S_k \rightarrow S_{k+1}$  для любого  $k = 1, 2, \dots, n-1$  вызывается поступлением в СМО одной заявки, поэтому  $\lambda_0 = \lambda_1 = \dots = \lambda_n = \lambda$ .

Несколько сложнее определяются  $\mu_k$ . Пусть система находится в состоянии  $S_1$  (работает один канал). Он производит  $\mu$  обслуживаний в единицу времени, поэтому  $\mu_1 = \mu$ . Предположим, что система находится в  $S_2$  (работают два канала). Для перехода в  $S_1$  нужно, чтобы либо закончил обслуживание первый канал, либо второй; суммарная интенсивность их обслуживания равна  $2\mu$ . Таким образом,  $\mu_2 = 2\mu$ . Суммарный поток обслуживания  $k$  каналами равен  $k\mu$  и, следовательно,  $\mu_k = k\mu$ .

Воспользуемся теперь формулами (4.6) – (4.7) для финальных вероятностей в схеме «гибели и размножения».

$$p_0 = \left( 1 + \frac{\lambda}{\mu} + \frac{\lambda^2}{2\mu^2} + \frac{\lambda^3}{2 \cdot 3\mu^3} + \dots + \frac{\lambda^k}{k!\mu^k} + \dots + \frac{\lambda^n}{n!\mu^n} \right)^{-1}, \quad (4.8)$$

$$p_k = \frac{\lambda^k}{k!\mu^k} p_0, \quad k = 1, \dots, n. \quad (4.9)$$

Обозначим  $\rho = \lambda/\mu$ . Эта величина называется предельной интенсивностью потока заявок. Ее смысл – среднее число заявок, приходящее за среднее время обслуживания одной заявки. Формулы (4.8) – (4.9) удобнее записывать в виде

$$p_0 = \left( 1 + \rho + \frac{\rho^2}{2!} + \dots + \frac{\rho^k}{k!} + \dots + \frac{\rho^n}{n!} \right)^{-1}, \quad (4.10)$$

$$p_k = \frac{\rho^k}{k!} p_0, \quad k = 1, \dots, n. \quad (4.11)$$

Определим теперь характеристики СМО. Найдем сначала вероятность отказа  $P_{отк}$ .

Очевидно

$$P_{отк} = p_n = \frac{\rho^n}{n!} p_0. \quad (4.12)$$

Относительная пропускная способность  $Q_o$  – вероятность того, что заявка будет обслужена:

$$Q_o = 1 - P_{отк} = 1 - \frac{\rho^n}{n!} p_0. \quad (4.13)$$

Абсолютная пропускная способность  $Q_a$  есть произведение интенсивности потока заявок  $\lambda$  на  $Q_o$ :

$$Q_a = \lambda Q_o = \lambda \left( 1 - \frac{\rho^n}{n!} p_0 \right). \quad (4.14)$$

Среднее число занятых каналов  $\bar{k}$  определим следующим способом. Вместо громоздкого вычисления математического ожидания  $\sum_{k=0}^n k \cdot p_k$  заметим, что  $Q_a$  есть интенсивность потока обслуженных системой заявок. Каждый занятый канал в единицу времени обслуживает в среднем  $\mu$  заявок. Следовательно, среднее число занятых каналов равно  $Q_a/\mu$ , или, учитывая (4.14),

$$\bar{k} = \rho \left( 1 - \frac{\rho^n}{n!} p_0 \right). \quad (4.15)$$

**Пример.** Имеется станция связи с тремя каналами, интенсивность потока заявок  $\lambda = 1.5$  заявки в минуту; среднее время обслуживания одной заявки  $t_{обсл} = 2$  (мин.), все потоки простейшие. Требуется найти финальные вероятности состояний и характеристики СМО.

Интенсивность обслуживания одним каналом  $\mu = 1/t_{обсл} = 1/2$  заявки в минуту. Параметр  $\rho = \lambda/\mu = 3$ . Вычисления по формулам (4.10) – (4.11) дают следующие результаты:  $p_0 = (1 + 3 + 9/2 + 27/6)^{-1} = 1/13$ ,  $p_1 = 3/13$ ,  $p_2 = 9/26$ ,  $p_3 = 9/26 \approx 0.346$ ,  $Q_a \approx 0.981$ ,  $Q_o \approx 0.654$ ,  $P_{отк} \approx 0.346$ ,  $\bar{k} \approx 1.96$ .

Видно, что СМО сильно перегружена: из трех каналов занято в среднем около двух, а из прибывающих заявок около 35% остаются необслуженными. Естественно, возникает задача: сколько надо каналов, чтобы обслуживать не менее 80% поступающих заявок? Какая доля каналов при этом будет простаивать?

Решение указанных вопросов – это уже в некотором смысле *оптимизация* СМО. Надо только учесть, что содержание каждого канала стоит определенную сумму денег. Вместе с тем каждая обслуженная заявка приносит некоторый доход. Умножая этот доход на среднее число заявок, обслуживаемых в единицу времени, мы получим средний доход от СМО в единицу времени как функцию  $f(n)$  от числа каналов. Пусть  $g(n)$  – стоимость обслуживания  $n$  каналов в единицу времени. Тогда критерием выбора оптимального числа каналов будет  $\max_n \{f(n) - g(n)\}$ .

### 4.4.2. СМО с неограниченной очередью

Пусть имеется одноканальная СМО с очередью, на которую не наложено никаких ограничений (ни по длине очереди, ни по времени ожидания). На эту СМО поступает поток заявок с интенсивностью  $\lambda$ ; поток обслуживаний простейший с интенсивностью  $\mu$ , обратной среднему времени  $t_{\text{сред}}$  обслуживания заявки. Требуется найти финальные вероятности состояний СМО, а также характеристики ее эффективности:

$L_{\text{сист}}$  – среднее число заявок в системе;

$W_{\text{сист}}$  – среднее время пребывания заявки в системе;

$L_{\text{оч}}$  – среднее число заявок в очереди;

$W_{\text{оч}}$  – среднее время пребывания заявки в очереди;

$P_{\text{зан}}$  – вероятность того, что канал занят (степень загрузки канала).

Что касается абсолютной пропускной способности  $Q_a$  и относительной  $Q_o$ , то вычислять их нет необходимости: в силу того, что длина очереди не ограничена, каждая заявка рано или поздно будет обслужена, поэтому  $Q_a = \lambda$  и  $Q_o = 1$ .

Состояния системы, как и раньше, будем нумеровать по числу заявок, находящихся в СМО:  $S_0$  – канал свободен,  $S_k$  – канал занят,  $k - 1$  заявок в очереди ( $k = 1, 2, \dots$ ). Число состояний теоретически бесконечно. Граф переходов приведен на рис. 4.3.

В силу потенциальной бесконечности числа состояний возникает вопрос о существовании финальных вероятностей  $p_k$ . Действительно, финальные вероятности для такой СМО существуют не всегда, а только тогда, когда система не перегружена. Можно показать, что если  $\rho = \lambda/\mu < 1$ , то финальные вероятности существуют, а при  $\rho = \lambda/\mu \geq 1$  очередь при  $t \rightarrow \infty$  растет неограниченно.

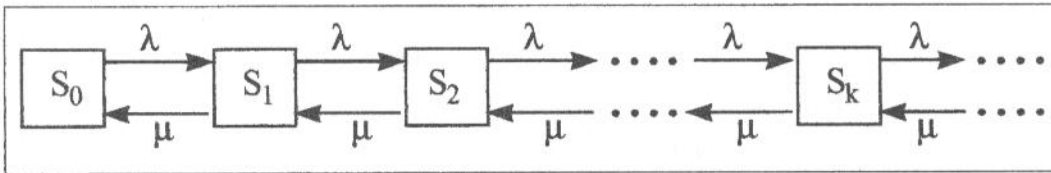


Рис. 4.3. Граф переходов в одноканальной СМО с неограниченной очередью

Воспользуемся формулами (4.6) – (4.7) общей схемы «гибели и размножения» для расчета финальных вероятностей, заменяя конечную сумму бесконечной и полагая  $\lambda_1 = \dots = \lambda_k = \dots = \lambda$  и  $\mu_1 = \dots = \mu_k = \dots = \mu$ . Получаем при  $\rho < 1$

$$p_0 = (1 + \rho + \rho^2 + \dots + \rho^k + \dots)^{-1} = \left( \frac{1}{1 - \rho} \right)^{-1} = 1 - \rho, \quad (4.16)$$

$$p_k = \rho^k (1 - \rho), \quad k = 1, 2, \dots \quad (4.17)$$



Определим среднее число заявок в СМО  $L_{сист}$ .

$$\begin{aligned} L_{сист} &= \sum_{k=0}^{\infty} k p_k = k \rho_k (1 - \rho) = \rho(1 - \rho) \sum_{k=0}^{\infty} k \rho^{k-1} = \rho(1 - \rho) \sum_{k=0}^{\infty} \frac{d}{d\rho} \rho^k = \\ &= \rho(1 - \rho) \frac{d}{d\rho} \sum_{k=0}^{\infty} \rho^k = \rho(1 - \rho) \frac{d}{d\rho} \left( \frac{1}{1 - \rho} \right) = \frac{\rho}{1 - \rho}. \end{aligned} \quad (4.18)$$

Для нахождения среднего времени пребывания заявки в системе  $W_{оч}$  применим формулу Литтла (4.2)

$$W_{сист} = \frac{\rho}{\lambda(1 - \rho)}. \quad (4.19)$$

Вероятность занятости канала  $P_{зан}$  определяется как

$$P_{зан} = 1 - p_0 = \rho. \quad (4.20)$$

Число заявок в системе под обслуживанием может быть или нулем (канал свободен), или единицей (канал занят). Математическое ожидание этой величины равно  $L_{обсл} = \rho$ . Тогда среднее число заявок в очереди  $L_{оч}$  равно

$$L_{оч} = L_{сист} - L_{обсл} = \frac{\rho}{1 - \rho} - \rho = \frac{\rho^2}{1 - \rho}. \quad (4.21)$$

По формуле Литтла (4.3) найдем среднее время пребывания заявки в очереди:

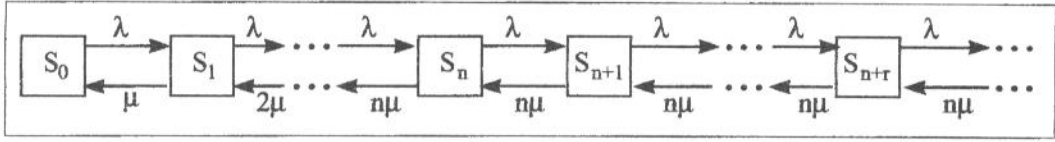
$$W_{оч} = \frac{\rho^2}{\lambda(1 - \rho)}. \quad (4.22)$$

Для многоканальной СМО без ограничения на длину очереди задача определения характеристик системы решается аналогично. Состояния системы обозначаются следующим образом:  $S_k$  ( $k \leq n$ ) – занято  $k$  каналов, остальные свободны;  $S_n$  – заняты все  $n$  каналов (очереди нет);  $S_{n+r}$  ( $r = 1, 2, \dots$ ) – заняты все  $n$  каналов,  $r$  заявок стоит в очереди. Граф состояний показан на рис. 4.4. Приведем без доказательства естественное условие существования финальных вероятностей:  $\rho/n < 1$ . Если  $\rho/n \geq 1$ , очередь растет до бесконечности. Будем предполагать, что  $\rho/n < 1$  и финальные вероятности существуют. Тогда

$$p_0 = \left( 1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \frac{\rho^n}{n!} + \frac{\rho^{n+1}}{n!(n - \rho)} \right)^{-1}, \quad (4.23)$$

$$p_1 = \frac{\rho}{1!} p_0, \dots, p_k = \frac{\rho^k}{k!} p_0, \dots, p_n = \frac{\rho^n}{n!} p_0, \quad (4.24)$$

$$p_{n+1} = \frac{\rho^{n+1}}{n \cdot n!} p_0, \dots, p_{n+r} = \frac{\rho^{n+r}}{n^r \cdot n!} p_0, \dots \quad (4.25)$$

Рис. 4.4. Граф переходов в  $n$ -канальной СМО с неограниченной очередью

Определим характеристики эффективности СМО. Среднее число занятых каналов  $\bar{k} = \lambda/\mu$ . Среднее число заявок в очереди  $L_{оч}$  вычисляется как сумма ряда  $L_{оч} = \sum_{r=1}^{\infty} r p_{n+r}$ . Выполняя по аналогии с (4.18) соответствующие преобразования, получим:

$$L_{оч} = \frac{\rho^{n+1} p_0}{n \cdot n! (1 - \rho/n)^2}. \quad (4.26)$$

Прибавляя к  $L_{оч}$  среднее число заявок под обслуживанием (это среднее число занятых каналов)  $\bar{k} = \rho$ , получим

$$L_{сист} = \frac{\rho^{n+1} p_0}{n \cdot n! (1 - \rho/n)^2} + \rho. \quad (4.27)$$

Используя формулы Литтла, получим средние времена пребывания заявки в очереди и в системе:

$$W_{оч} = \frac{1}{\lambda} L_{оч}, \quad W_{сист} = \frac{1}{\lambda} L_{сист}. \quad (4.28)$$

Еще раз подчеркнем, что все полученные формулы справедливы для случая простейшего потока прибывающих заявок и показательного закона распределения времени обслуживания (марковские СМО). Если эти условия не выполняются (чаще всего это касается времени обслуживания), то формулы носят приближенный характер. Правда, в некоторых случаях удастся установить и точные зависимости. Так, например, для одноканальной СМО с неограниченной очередью, простейшим потоком заявок и **произвольным** распределением времени обслуживания с математическим ожиданием  $\bar{t}_{обсл} = 1/\mu$  справедливы следующие равенства для среднего числа заявок в очереди  $L_{оч}$  и среднего числа заявок в системе  $L_{сист}$ :

$$L_{оч} = \frac{\rho^2 (1 + \nu_\mu^2)}{2(1 - \rho)}, \quad L_{сист} = \frac{\rho^2 (1 + \nu_\mu^2)}{2(1 - \rho)} + \rho, \quad (4.29)$$

где, как и ранее,  $\rho = \lambda/\mu$ , а  $\nu_\mu$  – коэффициент вариации времени обслуживания, равный отношению среднего квадратичного отклонения времени обслуживания к его математическому ожиданию:  $\nu_\mu = \sigma_{обсл}/\bar{t}_{обсл}$ . Формулы (4.29) носят название *формул Полячека-Хинчина*.

Деля  $L_{оч}$  и  $L_{сист}$  на  $\lambda$ , получим, согласно формулам Литтла, среднее время пребывания заявки в очереди и среднее время пребывания в системе:

$$W_{оч} = \frac{\rho^2 (1 + \nu_\mu^2)}{2\lambda(1 - \rho)}, \quad W_{сист} = \frac{\rho^2 (1 + \nu_\mu^2)}{\lambda 2(1 - \rho)} + \frac{1}{\mu}. \quad (4.30)$$

В частном случае, когда время обслуживания – показательное,  $\nu_\mu = 1$  и формулы (4.29) – (4.30) совпадают с (4.18) – (4.22).

Приведем еще один пример. Рассматривается одноканальная СМО с неограниченной очередью, на которую поступает произвольный поток заявок с интенсивностью  $\lambda$  и коэффициентом вариации интервалов между заявками  $\nu_\lambda$ . Предполагается, что  $0 < \nu_\lambda < 1$ . Время обслуживания  $t_{обсл}$  также имеет произвольное распределение вероятностей с математическим ожиданием  $\bar{t}_{обсл} = 1/\mu$  и коэффициентом вариации  $\nu_\mu$ ,  $0 < \nu_\mu < 1$ . В этом случае удастся лишь оценить сверху и снизу среднюю длину очереди:

$$\frac{\rho^2 (\nu_\lambda^2 + \nu_\mu^2)}{2(1 - \rho)} - \frac{\rho(1 - \nu_\lambda^2)}{2} \leq L_{оч} \leq \frac{\rho^2 (\nu_\lambda^2 + \nu_\mu^2)}{2(1 - \rho)} + \frac{(1 - \rho)(1 - \nu_\lambda^2)}{2}. \quad (4.31)$$

Для полностью немарковских многоканальных СМО точных аналитических методов не существует. В этом случае используется *имитационное моделирование* таких систем, которое позволяет численно получить необходимые характеристики с достаточно высокой точностью [1].

В заключение этой главы читателю предлагается решить следующую задачу сравнительного анализа двух СМО [2].

**Задача.** Железнодорожная касса по продаже билетов с двумя окошками представляет собой двухканальную СМО с неограниченной очередью, устанавливающейся сразу к двум окошкам (если одно окошко освобождается, первый в очереди пассажир его занимает). Касса продает билеты на два направления А и В. Интенсивность потока заявок (пассажиров, желающих купить билет) для обоих направлений одинакова:  $\lambda_A = \lambda_B = 0.45$  пассажира в минуту, а в сумме они образуют общий поток с интенсивностью  $\lambda = \lambda_A + \lambda_B = 0.9$ . Кассир тратит на обслуживание пассажира в среднем две минуты. Опыт показывает, что у кассы скапливается очередь, пассажиры жалуются на медлительность обслуживания. Поступило рационализаторское предложение: вместо одной кассы, продающей билеты и по направлению А и по направлению В, создать две специализированные кассы (по одному окошку в каждой), продающие билеты одна – только по направлению А, другая только по направлению В. Разумность этого предложения вызывает споры – кое-кто утверждает, что очереди останутся прежними. Требуется проверить полезность предложения расчетами. Предполагается, что все потоки событий – простейшие (на качественной стороне выводов это не скажется).

# Глава 5

## Теория игр

### 5.1. Принципы рационального поведения

Под игрой в исследовании операций понимается математическая модель конфликтной ситуации. Чтобы описать конфликтную ситуацию, надо указать:

- 1) участников конфликта (их должно быть, по меньшей мере, двое);
- 2) какими вариантами поведения (стратегиями) в конфликте располагают участники;
- 3) какую цель преследует каждый участник конфликта.

В теории игр участники конфликта называются игроками. Пусть  $I = \{1, 2, \dots, n\}$  – множество игроков. Обозначим через  $X_i$ ,  $i \in I$ , множество стратегий поведения  $i$ -го игрока. Элемент множества  $X_i$  будем называть индивидуальной стратегией  $i$ -го игрока и обозначать строчной буквой  $x_i$ . Для описания цели отдельного игрока в классической теории игр вводится понятие «выигрыша». Выигрыш  $i$ -го игрока зависит как от его индивидуального выбора  $x_i \in X_i$ , так и от выбора  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$  остальных игроков. Будем предполагать, что заданы скалярные функции  $f_i(x_1, \dots, x_i, \dots, x_n)$  выигрыша игроков  $f_i : X_1 \times \dots \times X_n \rightarrow \mathbb{R}^1$ ,  $i \in I$ . Цель  $i$ -го игрока заключается в достижении максимально возможного в условиях игры значения функции  $f_i$ . Тройка  $\Gamma = \langle I, \{X_i\}_{i \in I}, \{f_i\}_{i \in I} \rangle$  называется *бескоалиционной игрой в нормальной форме*. Здесь слово «бескоалиционная» означает, что субъектом принятия решения является индивид, т. е. отдельный игрок, а не группа или коалиция игроков. Набор индивидуальных стратегий  $x = (x_1, \dots, x_n) \in X_1 \times \dots \times X_n$  называется *ситуацией* игры, или совокупной стратегией (далее эти термины будут использоваться как синонимы). Если игроки приняли решение разыгрывать ситуацию  $x$ , то каждый игрок получит выигрыш  $f_i(x)$ ,  $i \in I$ .

Термины «выигрыш» и «игрок» сложились исторически, когда анализировались преимущественно азартные игры. Эти термины не совсем точные. Например, если значение  $f_i(x) < 0$ , то «выигрыш»  $i$ -го игрока является фактически его проигрышем. Кроме того, изучают игры, где  $f_i$  является не денежным выигрышем, а, скажем, вероятностью поражения цели.

Рассмотрим теперь вопрос о выборе оптимальной (в определенном смысле) ситуа-

ции, или, иными словами, решения игры. Этот вопрос не является простым. Очевидно, что существование ситуации  $x^* = (x_1^*, \dots, x_n^*)$ , при которой достигается максимальный выигрыш  $f_i(x^*) = \max_{x \in X_1 \times \dots \times X_n} f_i(x)$  сразу для всех игроков  $i \in I$ , маловероятно (по существу, в этом случае и конфликта-то никакого нет). Поэтому для определения оптимальной ситуации рассмотрим сначала возможные принципы рационального поведения в игре.

**Принцип Парето.** Ситуация  $\bar{x} = (\bar{x}_1, \dots, \bar{x}_n)$  называется оптимальной по Парето, если в  $X_1 \times \dots \times X_n$  не существует ситуации  $x$  такой, что  $\forall i \in I : f_i(x) \geq f_i(\bar{x})$  и  $\exists i_0 : f_{i_0}(x) < f_{i_0}(\bar{x})$ . Смысл ситуации  $\bar{x}$  состоит в том, что ее нельзя улучшить сразу для всех игроков. Множество оптимальных по Парето ситуаций обозначим через  $P$ .

**Принцип наилучшего гарантированного результата.** Рассмотрим случай, когда игроку совершенно неизвестны предпочтения остальных игроков. В этой ситуации он не может сделать никаких предсказаний о выборе стратегий остальных и должен ориентироваться на самый плохой для себя исход. Выбирая свою стратегию  $x_i \in X_i$ ,  $i$ -й игрок, не зная целей других игроков или подозревая, что они хотят навредить ему, может гарантировать себе лишь величину

$$\hat{f}_i(x_i) = \min_{x_j \in X_j, j \neq i} f_i(x_1, \dots, x_i, \dots, x_n).$$

Иными словами, игрок рассчитывает на самый плохой для него вариант поведения остальных участников конфликта. Эта величина называется гарантированным результатом  $i$ -го игрока при выборе им стратегии  $x_i$ . Исходя из принципа наилучшего гарантированного результата игрок  $i$  должен выбрать  $x_i = \hat{x}_i$  так, чтобы получить максимальный гарантированный выигрыш:

$$\hat{f}_i(\hat{x}_i) = \max_{x_i \in X} \hat{f}_i(x_i) = \max_{x_i \in X} \min_{x_j \in X_j, j \neq i} f_i(x_1, \dots, x_i, \dots, x_n). \quad (5.1)$$

Стратегия  $\hat{x}_i$  называется *максиминной* или *осторожной* стратегией  $i$ -го игрока. Сам принцип выбора  $\hat{x}_i$  иногда называют *принципом максимина*. Если все игроки осторожны в своем выборе, то в результате сложится ситуация  $\hat{x} = (\hat{x}_1, \dots, \hat{x}_n)$ . Множество максиминных совокупных стратегий обозначим через  $Z$ .

**Принцип оптимальности Нэша (принцип равновесия).** Совокупная стратегия  $x^* = (x_1^*, \dots, x_n^*)$  оптимальна по Нэшу (равновесна), если каждому игроку  $i$  невыгодно отступать от своей индивидуальной стратегии  $x_i^*$ , входящей в совокупную стратегию  $x^*$ , при условии, что остальные игроки  $j$  не меняют свои индивидуальные стратегии  $x_j^*$ , т. е.

$$\forall i \in I : f_i(x_1^*, \dots, x_i^*, \dots, x_n^*) \geq f_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_n^*) \quad \forall x_i \in X_i. \quad (5.2)$$

Обозначим через  $\varphi_i(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$  множество точек, в которых достигается максимум по  $x_i \in X_i$  функции выигрыша  $i$ -го игрока  $f_i(x_1, \dots, x_i, \dots, x_n)$ :

$$\varphi_i(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = \text{Arg max}_{x_i \in X_i} f_i(x_1, \dots, x_i, \dots, x_n) \subset X_i.$$



Следовательно, ситуация равновесия является решением системы включений

$$\begin{aligned} x_1 &\in \varphi_1(x_2, \dots, x_n), \\ x_2 &\in \varphi_2(x_1, x_3, \dots, x_n), \\ &\dots\dots\dots \\ x_n &\in \varphi_n(x_1, \dots, x_{n-1}). \end{aligned} \quad (5.3)$$

Множество ситуаций равновесия – всех решений системы (5.3) – обозначим через  $N$ .

*Замечание.* Система (5.3), вообще говоря, может не иметь решения, что означает отсутствие в игре ситуаций равновесия ( $N = \emptyset$ ).

Поясним указанные принципы рационального поведения на следующих примерах.

**Пример 1** (биматричная игра). В игре участвуют два игрока. Множество стратегий игроков конечны, поэтому стратегии каждого игрока можно просто пронумеровать:  $X_1 = \{1, 2, \dots, m\}$ ,  $X_2 = \{1, 2, \dots, n\}$ . Пара  $(i, j)$ ,  $i \in \{1, \dots, m\}$ ,  $j \in \{1, \dots, n\}$ , образует ситуацию игры. В этом случае функции выигрыша игроков удобно представить в виде двух матриц  $A = [a_{ij}]_{i=1, \dots, m}^{j=1, \dots, n}$  и  $B = [b_{ij}]_{i=1, \dots, m}^{j=1, \dots, n}$ , где  $a_{ij} = f_1(i, j)$ ,  $b_{ij} = f_2(i, j)$ . Стратегии первого игрока – номер строки, стратегии второго игрока – номер столбца матриц. Пусть для определенности  $m = 3$ ,  $n = 2$ , а матрицы выигрышей  $A$  и  $B$  представлены в следующих таблицах:

$A =$		1	2	$\hat{f}_1(i)$	$B =$		1	2
	1	15	11	$\min_j = 11$		1	0.3	0.3
	2	12	16	$\min_j = 12$		2	0.4	0.5
	3	11	14	$\min_j = 11$		3	0.6	0.2
						$\hat{f}_2(j)$	$\min_i = 0.3$	$\min_i = 0.2$

Гарантированные выигрыши игроков указаны при соответствующих матрицах. Осторожные стратегии  $\hat{i}$  и  $\hat{j}$  соответственно для игрока 1 и игрока 2 определяются как

$$\begin{aligned} \hat{i} &= \arg \max_{i=1,2,3} \min_{j=1,2} a_{ij} = \arg \max_{i=1,2,3} \{11, 12, 11\} = 2, \\ \hat{j} &= \arg \max_{j=1,2} \min_{i=1,2,3} b_{ij} = \arg \max_{j=1,2} \{0.3, 0.2\} = 1. \end{aligned}$$

В ситуации  $(\hat{i}, \hat{j}) = (2, 1)$  выигрыш игрока 1 равен 12, а игрока 2 – 0.4.

Для определения оптимальных по Парето ситуаций рассмотрим плоскость с системой координат  $(f_1, f_2)$ . Поставим в соответствие каждой ситуации игры  $(i, j)$  точку плоскости с координатами  $(f_1(i, j), f_2(i, j)) = (a_{ij}, b_{ij})$ . В каждой точке построим конус, образованный прямым углом, с вершиной в данной точке (рис. 5.1). Легко заметить, что если для точки  $(a_{ij}, b_{ij})$  соответствующий конус не содержит других точек, то пара  $(i, j)$  является оптимальной по Парето. В рассматриваемом примере множество оптимальных по Парето ситуаций будет  $P = \{(3, 1), (2, 2)\}$ .

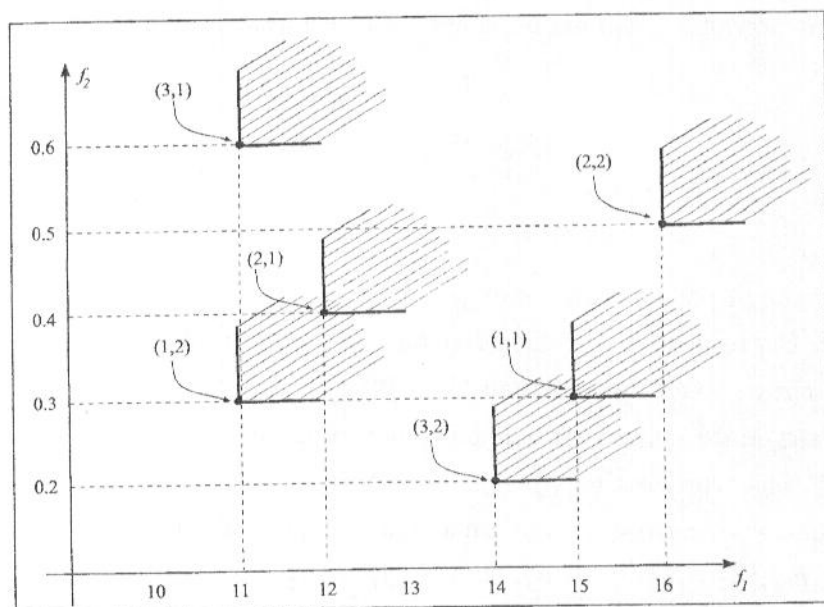


Рис. 5.1. Определение оптимальных по Парето ситуаций

Для нахождения ситуаций равновесия  $(i^*, j^*)$  необходимо решить систему уравнений (5.3), которая в рассматриваемом примере имеет вид  $i = \varphi_1(j)$ ,  $j = \varphi_2(i)$ .

Отображение  $\varphi_1$  определяется по матрице  $A$ :

$$\varphi_1(1) = 1, \varphi_1(2) = 2.$$

Отображение  $\varphi_2$ , определяемое по матрице  $B$ , имеет вид:

$$\varphi_2(1) = \{1, 2\}, \varphi_2(2) = \{2\}, \varphi_2(3) = \{1\}.$$

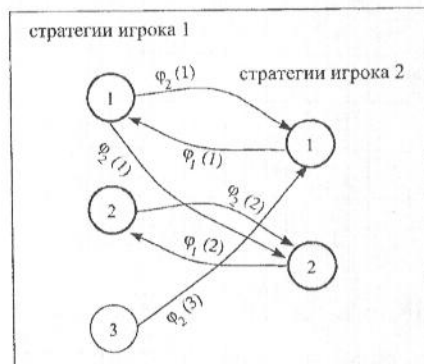


Рис. 5.2. Определение ситуаций равновесия

Представим функции  $\varphi_1(j)$ ,  $\varphi_2(i)$  в виде двудольного ориентированного графа, как показано на рис. 5.2. Дуги, идущие справа налево, соответствуют отображению  $\varphi_1$ , дуги, идущие слева направо, — отображению  $\varphi_2$ . Пара вершин  $(i^*, j^*)$ , соединенная дугами обоих типов, очевидно, является решением системы уравнений (5.3), т. е. ситуацией равновесия. В данном случае это ситуации (1,1) и (2,2).

Таким образом, для рассматриваемой игры три принципа рационального поведения дают следующие результаты:

- 1) множество оптимальных по Парето ситуаций  $P = \{(3, 1), (2, 2)\}$ ;
- 2) множество ситуаций, образованных минимаксными (осторожными) стратегиями игроков  $Z = \{(2, 1)\}$ ;
- 3) множество оптимальных по Нэшу ситуаций (равновесий)  $N = \{(1, 1), (2, 2)\}$ .

Что же понимать под решением игры? Ответ, вообще говоря, неоднозначен. Если у игрока нет никакой информации относительно функций выигрыша других игроков, то разумно «подстраховаться», т. е. считать других игроков «врагами» и использовать в этом случае принцип наилучшего гарантированного результата (принцип максимина). При наличии информации о целях остальных игроков целесообразно использовать принцип оптимальности по Парето и принцип оптимальности Нэша, если, конечно, эти цели не прямо противоположны цели данного игрока. При этом оптимальность по Парето есть необходимое условие для того, чтобы рассматривать некоторую совокупную стратегию  $x$  в качестве претендента на решение. В классической теории игр принято под решением считать ситуацию  $x$ , удовлетворяющую условию  $x \in P \cap N$ . Однако и здесь возможны проблемы, связанные с тем, что для некоторых игр может оказаться  $P \cap N = \emptyset$  либо за счет того, что эти множества не имеют общих ситуаций, либо вследствие того, что  $N = \emptyset$  (множество  $P$  всегда не пусто). Кроме того,  $P \cap N$  может содержать более чем одну ситуацию, причем одна ситуация равновесия выгодна одним игрокам, а другая – другим игрокам. (В рассматриваемом примере  $P \cap N = \{(2, 2)\}$ , т. е. решение определено однозначно).

В заключение этого раздела приведем один пример моделирования реальной ситуации в виде биматричной игры.

**Пример 2.** Покупатель (игрок 2) приходит на рынок за яблоками. Продавец, торгующий яблоками (игрок 1), использует пружинные весы. У него есть две стратегии:

- 1) честно взвесить 1 кг яблок;
- 2) подкрутить пружинку и обвесить покупателя на 200 граммов.

Назовем эти стратегии «честность» и «обман».

Покупатель также имеет две стратегии:

- 1) поверить продавцу, заплатить деньги и уйти;
- 2) взвесить купленные яблоки на контрольных весах и в случае обмана звать кого-то и доказывать, что его обвесили. Назовем эти стратегии «поверить» и «проверить» соответственно.

Определим выигрыши продавца и покупателя в каждой ситуации:

а) продавец честно взвесил, а покупатель поверил. Соответствующие выигрыши обоих, равные 0, выберем в качестве начала отсчета;

б) продавец обманул, а покупатель ему поверил. Выигрыш продавца равен 1, так как он получил дополнительную прибыль. Выигрыш покупателя равен -1, так как он получил меньше яблок;

в) продавец честно взвесил, а покупатель его проверил. Выигрыш продавца равен 1. Выигрыш покупателя равен -1/2: он, во-первых, зря потерял время, во-вторых, глупо себя чувствует;

г) продавец обманул, а покупатель его проверил. Выигрыш продавца равен -1, так как обнаружение обмана грозит ему определенными неприятностями (например, его могут лишиться лицензии на торговлю на этом рынке). Выигрыш покупателя равен  $1/2$ , так как, во-первых, ему возместили обвес, во-вторых, он испытывает моральное удовлетворение от разоблачения обманщика.

Получается следующая биматричная игра:

$A =$		пов	пров	,	$B =$		пов	пров
	честн	0	0			честн	0	-1/2
	обман	1	-1			обман	-1	1/2

Легко проверить, что здесь нет ситуаций равновесия.

**Пример 3.** Множество игроков  $I = \{1, \dots, n\}$ ,  $n \geq 3$ , множества стратегий  $X_i = [0, 1]$ , функции выигрыша  $f_i(x) = x_i + \sum_{i \neq j} (1 - x_j)$ . Оптимальной по Нэшу (равновесием) и принципу максимина является ситуация, при которой все игроки выбирают  $x_i = 1$ ,  $i = 1, \dots, n$ , и получают по единице. Оптимальными по Парето являются стратегии, когда все выбирают 0 и выигрывают  $n - 1$ , или когда  $i$ -й игрок выбирает 1, а остальные - 0; при этом  $i$ -й получает  $n$ , а остальные - по  $n - 2$ . (Доказать!).

## 5.2. Антагонистические игры

В антагонистической игре принимают участие два игрока. Игрок 1 выбирает стратегию  $x$  из множества стратегий  $X$ , игрок 2 выбирает стратегию  $y$  из множества стратегий  $Y$ . Нормальная форма игры подразумевает, что каждый игрок выбирает свою стратегию независимо, не зная выбора партнера. Функции выигрыша игроков удовлетворяют условию  $f_1(x, y) + f_2(x, y) = 0$ . Обозначим  $F(x, y) = f_1(x, y)$ . Тогда  $f_2(x, y) = -F(x, y)$ , т. е. выигрыш  $F(x, y)$  первого игрока является проигрышем для второго, и наоборот. Цель первого игрока состоит в увеличении своего выигрыша  $F(x, y)$ , а цель второго - в уменьшении  $F(x, y)$ . Таким образом, антагонистическая игра задается набором  $\Gamma = \langle X, Y, F(x, y) \rangle$ .

Пусть, как и ранее,  $Z$  означает множество ситуаций, образованных осторожными (максиминными) стратегиями игроков,  $P$  - множество оптимальных по Парето ситуаций,  $N$  - множество оптимальных по Нэшу ситуаций (ситуаций равновесия).

Анализ принципов рационального поведения применительно к этой игре позволяет сформулировать следующие утверждения.

1.  $Z \neq \emptyset$ .
2. Каждая ситуация  $(x, y)$  является оптимальной по Парето, т. е.  $P = X \times Y$ .
3. Если  $N \neq \emptyset$ , то  $N = Z$ .

Первое утверждение очевидно. Заметим только, что принцип максимина для второго игрока сводится к вычислению минимакса функции выигрыша первого  $F(x, y)$ . Действительно,

$$\max_{y \in Y} \min_{x \in X} f_2(x, y) = \max_{y \in Y} \min_{x \in X} (-F(x, y)) = - \min_{y \in Y} \max_{x \in X} F(x, y).$$

Справедливость второго утверждения следует из того, что в антагонистической игре любую ситуацию  $(x, y)$  нельзя улучшить сразу для обоих игроков: если увеличивается выигрыш одного игрока, то настолько же уменьшается выигрыш другого, точнее увеличивается его проигрыш.

Перейдем к третьему утверждению. Предварительно введем следующие обозначения:

$$\underline{v} = \max_{x \in X} \min_{y \in Y} F(x, y), \quad \bar{v} = \min_{y \in Y} \max_{x \in X} F(x, y).$$

Величина  $\underline{v}$  выражает наилучший гарантированный результат игрока 1, а  $\bar{v}$  – наилучший гарантированный результат игрока 2.

**Лемма 1.**  $\underline{v} \leq \bar{v}$ .

*Доказательство.* Возьмем произвольные стратегии игроков  $x$  и  $y$ . Тогда

$$\min_{y \in Y} F(x, y) \leq F(x, y) \leq \max_{x \in X} F(x, y) \Rightarrow \min_{y \in Y} F(x, y) \leq \max_{x \in X} F(x, y).$$

Левая часть последнего неравенства зависит от  $x$ , а правая часть нет. Поэтому

$$\max_x \min_y F(x, y) \leq \max_x F(x, y) \quad \forall y \in Y \Rightarrow \underline{v} \leq \bar{v}. \quad \square$$

Числа  $\underline{v}$ ,  $\bar{v}$  называются соответственно нижним и верхним значениями игры.

Для антагонистической игры определение (5.2) оптимальности по Нэшу (равновесия) ситуации  $(x^*, y^*)$  можно переписать в виде

$$\forall x \in X \quad F(x, y^*) \leq F(x^*, y^*) \leq F(x^*, y) \quad \forall y \in Y. \quad (5.4)$$

В литературе по теории игр ситуацию  $(x^*, y^*)$ , удовлетворяющую системе неравенств (5.4), часто называют *седловой точкой* функции  $F(x, y)$ . Из рис. 5.3 видно, что график функции  $F(x, y)$ , у которой есть седловая точка, имеет в окрестности этой точки вид седла.

**Теорема 5.1.** Для того чтобы ситуация  $(x^*, y^*)$  была оптимальной по Нэшу (равновесием, седловой точкой), необходимо и достаточно, чтобы

$$\underline{v} = \bar{v} = F(x^*, y^*). \quad (5.5)$$



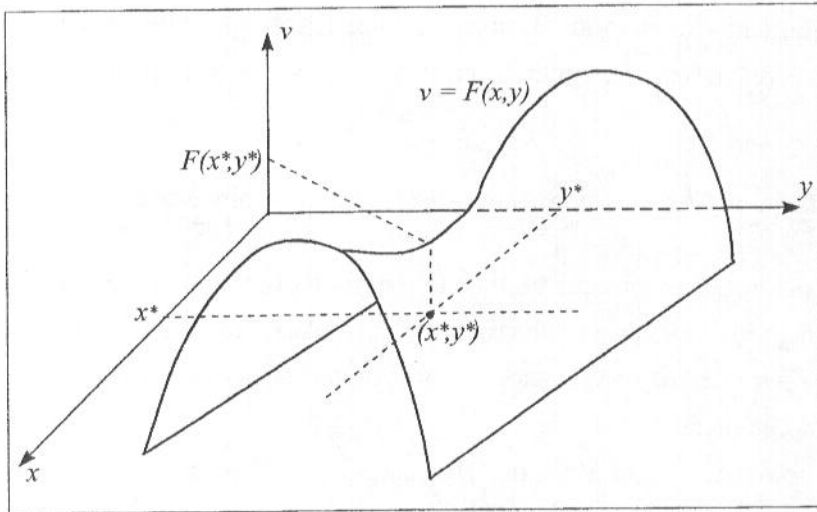


Рис. 5.3. Седловая точка функции выигрыша

*Доказательство. Необходимость.* Пусть  $(x^*, y^*)$  – ситуация равновесия (седловая точка функции  $F(x, y)$ ). Покажем, что выполнено равенство (5.5), а  $(x^*, y^*)$  – максиминная и минимаксная стратегии. Поскольку в определении седловой точки (5.4) левая часть неравенства выполнена при любом  $x \in X$ , а правая – при любом  $y \in Y$ , то справедливо

$$\max_x F(x, y^*) \leq F(x^*, y^*) \leq \min_y F(x^*, y).$$

Но тогда

$$\bar{v} = \min_y \max_x F(x, y) \leq \max_x F(x, y^*) \leq F(x^*, y^*) \leq \min_y F(x^*, y) \leq \max_x \min_y F(x, y) = \underline{v}.$$

Поскольку в силу леммы 1 всегда  $\underline{v} \leq \bar{v}$ , то все неравенства в этой цепочке превращаются в равенства и, следовательно,  $\underline{v} = \bar{v} = F(x^*, y^*)$ .  $\square$

*Достаточность.* Пусть  $\underline{v} = \bar{v} = F(x^*, y^*)$ . Тогда

$$\begin{aligned} F(x, y^*) &\leq \max_x F(x, y^*) = \min_y \max_x F(x, y) = \bar{v} = \underline{v} = \\ &= \max_x \min_y F(x, y) = \min_y F(x^*, y) \leq F(x^*, y), \end{aligned}$$

т. е.  $F(x, y^*) \leq F(x^*, y)$  для всех  $x$  и  $y$ . Полагая сначала  $x = x^*$ , а потом  $y = y^*$ , получим

$$F(x, y^*) \leq F(x^*, y^*) \leq F(x^*, y). \quad \square$$

Из теоремы 5.1 следует, что ситуация равновесия (седловая точка), если она существует, удовлетворяет всем принципам рационального поведения. В антагонистической игре может существовать несколько ситуаций равновесия. Однако это не приводит к неоднозначности выбора решения, так как справедлива следующая

**Теорема 5.2.** (Свойство «прямоугольности» ситуаций равновесия). Пусть  $(x_1, y_1)$  и  $(x_2, y_2)$  – ситуации равновесия. Тогда  $(x_1, y_2)$ ,  $(x_2, y_1)$  также являются ситуациями равновесия, причем

$$F(x_1, y_1) = F(x_2, y_2) = F(x_1, y_2) = F(x_2, y_1) = \underline{v} = \bar{v} = v.$$

(Доказать!)

Число  $v = \underline{v} = \bar{v} = F(x^*, y^*)$ , где  $(x^*, y^*)$  – ситуация равновесия, называется *ценой* игры.

### 5.3. Матричные игры

Рассмотрим антагонистическую игру  $\Gamma = \langle X, Y, F(x, y) \rangle$ , в которой множества стратегий игроков  $X$  и  $Y$  конечны и пронумерованы  $X = \{1, \dots, m\}$ ,  $Y = \{1, \dots, n\}$ . В этом случае игра полностью задается матрицей выигрышей первого игрока  $A = [a_{ij}]$ ,  $i = 1, \dots, m$ ,  $j = 1, \dots, n$ , где  $a_{ij} = F(i, j)$ . Игры подобного типа называются *матричными играми*.

Определение ситуации равновесия  $(i^*, j^*)$  матричной игры, задаваемой матрицей  $A$ , имеет следующий вид:

$$a_{ij^*} \leq a_{i^*j^*} \leq a_{i^*j} \quad \forall i = 1, \dots, m, j = 1, \dots, n.$$

Наличие ситуации равновесия (седловой точки функции выигрыша) в матричных играх довольно редкий факт. Играть в подобные игры весьма непросто. Проигравшему игроку каждый раз хочется сменить свою стратегию, но он будет бояться это сделать (а вдруг противник догадается?). Теория игр предлагает игрокам использовать *смешанные стратегии*.

Пусть игра задана матрицей  $A = [a_{ij}]$  размера  $m \times n$ , т. е. множества чистых стратегий игроков  $X = \{1, \dots, m\}$ ,  $Y = \{1, \dots, n\}$ . Смешанными стратегиями игроков назовем векторы  $p$  и  $q$ , задающие распределение вероятностей на множествах чистых стратегий игроков, т. е.

$$p \in P = \left\{ p = (p_1, \dots, p_m) : p_i \geq 0, \sum_{i=1}^m p_i = 1 \right\},$$

$$q \in Q = \left\{ q = (q_1, \dots, q_n) : q_j \geq 0, \sum_{j=1}^n q_j = 1 \right\}.$$

Реализация смешанной стратегии  $p$  происходит следующим образом. Интервал  $[0, 1]$  разбивается на  $m$  отрезков с длинами  $p_1, \dots, p_m$ , и датчиком равномерного распределения разыгрывается случайная точка  $\xi \in [0, 1]$ . Если  $\xi$  попала в  $i_0$ -й отрезок, то первый игрок

выбирает в данной партии игры чистую стратегию  $i = i_0$ . Аналогично выбирает свои чистые стратегии второй игрок. Каждый игрок осуществляет свой выбор независимо от результата выбора противника. При таком выборе стратегий выигрыш первого игрока есть случайная величина, зависящая от распределений  $p$  и  $q$ . Посчитаем математическое ожидание  $H(p, q)$  выигрыша первого игрока

$$H(p, q) = \sum_{i=1}^m \sum_{j=1}^n a_{ij} \cdot p_i \cdot q_j = \sum_{i=1}^m H(i, q) p_i = \sum_{j=1}^n H(p, j) q_j,$$

где  $H(i, q) = \sum_{j=1}^n a_{ij} q_j$ ,  $H(p, j) = \sum_{i=1}^m a_{ij} p_i$ . Величина  $H(i, q)$  есть средний выигрыш 1-го игрока, когда он играет только  $i$ -й чистой стратегией, а второй игрок использует смешанную стратегию  $q$ . Соответственно  $H(p, j)$  есть средний выигрыш первого игрока, когда он использует смешанную стратегию  $p$  против чистой  $j$ -й стратегии второго игрока.

Определим осторожные смешанные стратегии  $\hat{p} \in P$  и  $\hat{q} \in Q$  игроков:

$$\hat{p} = \arg \max_{p \in P} \min_{q \in Q} H(p, q), \quad \hat{q} = \arg \min_{q \in Q} \max_{p \in P} H(p, q).$$

Для матричной игры со смешанными стратегиями остается справедливой теорема 1 о связи ситуаций равновесия с максиминной и минимаксной (осторожными) стратегиями:  $(p^*, q^*)$  – ситуация равновесия тогда и только тогда, когда

$$v^* = H(p^*, q^*) = \max_{p \in P} \min_{q \in Q} H(p, q) = \min_{q \in Q} \max_{p \in P} H(p, q).$$

Набор  $(p^*, q^*, v^*)$  называется решением игры.

### 5.3.1. Сведение матричной игры к задаче линейного программирования

Пусть матрица игры такова, что все ее элементы  $a_{ij} > 0$ <sup>1</sup>. Рассмотрим сначала задачу определения максимина функции  $H(p, q) = \sum_{j=1}^n A_j(p) q_j$ , где  $A_j(p) = \sum_{i=1}^m a_{ij} p_i$ . Заметим, что

$$\max_{p \in P} \min_{q \in Q} \sum_{j=1}^n H(p, j) q_j = \max_{p \in P} \min \{H(p, 1), \dots, H(p, n)\}, \quad (5.6)$$

поскольку минимум линейной функции по  $q_1, \dots, q_n$  на многограннике  $Q$  достигается в одной из его вершин  $(1, 0, \dots, 0)$ ,  $(0, 1, \dots, 0)$ ,  $\dots$ ,  $(0, \dots, 0, 1)$ . Введем обозначение

<sup>1</sup>Это предположение необременительно. Если в матрице игры есть элементы  $a_{ij} < 0$ , достаточно прибавить ко всем элементам одно и то же большое число  $M > \min_{ij} a_{ij}$ . Решение игры от этого не меняется, лишь цена игры увеличится на  $M$ .

$t = \min \{H(p, 1), \dots, H(p, n)\}$ . Тогда задача нахождения максимина может быть записана как

$$\begin{aligned} t &\rightarrow \max_{t, p}, \\ t &\leq H(p, 1), \dots, t \leq H(p, n), \\ p_i &\geq 0, \quad i = 1, \dots, m, \quad \sum_{i=1}^m p_i = 1. \end{aligned} \quad (5.7)$$

Введем новые переменные  $x_i = p_i/t$ . В силу положительности величин  $a_{ij}$ , а следовательно, и  $H(p, j)$ , величина  $t > 0$ , поэтому деление на  $t$  корректно. Для переменных  $x_i$  справедливо  $x_i \geq 0$ . Тогда  $\sum_i x_i = \sum_i p_i/t = 1/t$ , откуда  $t = (\sum_i x_i)^{-1}$ . Следовательно, максимизация  $t$  эквивалентна минимизации суммы  $\sum_i x_i$ . Неравенства  $t \leq H(p, j)$  в новых переменных запишутся как  $\sum_{i=1}^m a_{ij} x_i \geq 1$ . Следовательно, задача максимизации (5.7) эквивалентна задаче линейного программирования на минимум следующего вида

$$\begin{aligned} \sum_{i=1}^m x_i &\rightarrow \min \\ \sum_{i=1}^m a_{i1} x_i &\geq 1, \\ &\dots\dots\dots \\ \sum_{i=1}^m a_{in} x_i &\geq 1, \\ x_i &\geq 0, \quad i = 1, \dots, m. \end{aligned} \quad (5.8)$$

Задача отыскания минимаксной стратегии второго игрока  $\min_q \max_p H(p, q)$  точно так же сводится к задаче линейного программирования. Эта задача имеет вид:

$$\begin{aligned} \sum_{j=1}^n y_j &\rightarrow \max \\ \sum_{j=1}^n a_{1j} y_j &\leq 1, \\ &\dots\dots\dots \\ \sum_{j=1}^n a_{mj} y_j &\leq 1, \\ y_j &\geq 0, \quad j = 1, \dots, n. \end{aligned} \quad (5.9)$$

Нетрудно показать, что решения  $x^* = (x_1^*, \dots, x_m^*)$  и  $y^* = (y_1^*, \dots, y_n^*)$  соответствующих задач линейного программирования (5.8) и (5.9) существуют. Эти решения позволяют определить максиминную стратегию  $p^*$  первого игрока и минимаксную стратегию  $q^*$  второго игрока как  $p^* = x^* / \min(\sum x_i)$ ,  $q^* = y^* / \max(\sum y_j)$  соответственно.

Основной вопрос, который возникает при использовании смешанных стратегий: будет ли ситуация  $(p^*, q^*)$ , составленная из максиминной стратегии первого игрока и минимаксной стратегии второго оптимальной по Нэшу, т. е. равновесием? Ответ на этот вопрос дает

**Теорема 5.3 (Основная теорема теории игр).** *Для любой матричной игры существует ситуация  $(p^*, q^*)$  в смешанных стратегиях такая, что*

$$\max_{p \in P} \min_{q \in Q} H(p, q) = \min_{q \in Q} \max_{p \in P} H(p, q) = H(p^*, q^*)$$

и, следовательно,  $(p^*, q^*)$  в силу теоремы 5.1 является оптимальной по Нэшу (равновесием, седловой точкой).

*Доказательство.* Легко заметить, что задачи (5.8) и (5.9) образуют пару двойственных задач линейного программирования. В силу теоремы двойственности минимум в задаче (5.8) равен максимуму в (5.9). Минимум в первой задаче равен  $[\max_{p \in P} \min_{q \in Q} H(p, q)]^{-1}$ , а максимум во второй задаче равен  $[\min_{q \in Q} \max_{p \in P} H(p, q)]^{-1}$ , откуда и следует утверждение теоремы.  $\square$ .

Вторая теорема двойственности («теорема о дополняющей нежесткости») также важна для решения матричных игр. Запишем ее формулировку для задач (5.8)-(5.9):

$$\begin{aligned} x_i^* \left( \sum_{j=1}^n a_{ij} y_j^* - 1 \right) &= 0, \quad i = 1, \dots, m; \\ y_j^* \left( \sum_{i=1}^m a_{ij} x_i^* - 1 \right) &= 0, \quad j = 1, \dots, n. \end{aligned}$$

Но  $x_i^* = p_i^*/v^*$ ,  $y_j^* = q_j^*/v^*$ . Подставляя эти значения в указанные выше равенства и учитывая введенные ранее обозначения  $H(i, q) = \sum_j a_{ij} q_j$ ,  $H(p, j) = \sum_i a_{ij} p_i$ , получим

$$p_i^* [H(i, q^*) - v^*] = 0, \quad i = 1, \dots, m; \quad (5.10)$$

$$q_j^* [H(p^*, j) - v^*] = 0, \quad j = 1, \dots, n. \quad (5.11)$$

Этой теоремой мы воспользуемся ниже для решения матричной игры, в которой у одного из игроков имеется ровно две стратегии.

### 5.3.2. Графо-аналитический метод решения игр $2 \times n$ и $m \times 2$

Рассмотрим предлагаемый метод на следующем примере игры, в которой у первого игрока две чистые стратегии, а у второго – 3. Пусть матрица выигрышей игрока 1 имеет следующий вид:

$$A = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 1 & 3 & -6 & 2 \\ \hline 2 & 2 & -1 & 4 & -3 \end{array}$$

Сначала необходимо проверить, существует ли ситуация равновесия в чистых стратегиях. Для этого определим  $v_1 = \max_i \min_j \{a_{ij}\}$  и  $v_2 = \min_j \max_i \{a_{ij}\}$

$$v_1 = \max\{-6, -3\} = -6, \quad v_2 = \min_{3,4,2} = 4.$$

Ситуации равновесия в чистых стратегиях нет, так как  $v_1 < v_2$ . Найдем решение в смешанных стратегиях, которое всегда существует в силу теоремы 5.3. Определим сначала



максиминную смешанную стратегию первого игрока, решая задачу (5.6) для данной игры:

$$\begin{aligned} \max_{\substack{p_1+p_2=1, \\ p_1, p_2 \geq 0}} \min\{H(p, 1), H(p, 2), H(p, 3)\} &= \max_{\substack{p_1+p_2=1, \\ p_1, p_2 \geq 0}} \min\{3p_1 - 2p_2, -6p_1 + 4p_2, 2p_1 - 3p_2\} = \\ &= \max_{0 \leq p_1 \leq 1} \min\{3p_1 - 2(1 - p_1), -6p_1 + 4(1 - p_1), 2p_1 - 3(1 - p_1)\}. \end{aligned}$$

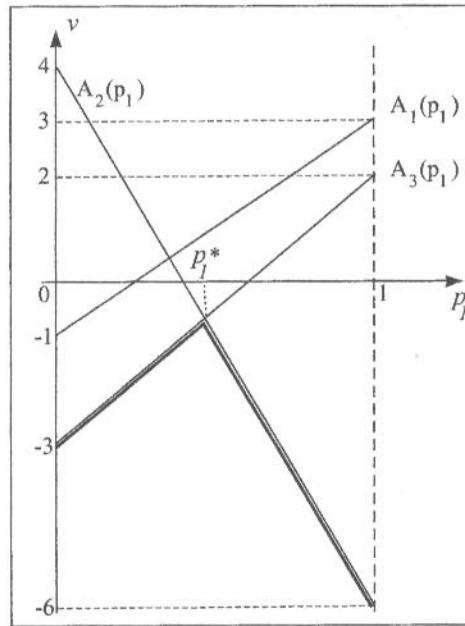


Рис. 5.4. Определение оптимальной стратегии  $p^*$

Графики линейных функций одной переменной  $p_1$   $v = H((p_1, 1 - p_1), j)$  приведены на рис. 5.4. График функции  $\min\{H((p_1, 1 - p_1), 1), H((p_1, 1 - p_1), 2), H((p_1, 1 - p_1), 3)\}$  – это нижняя огибающая указанных графиков. Ее максимум достигается в точке  $p_1^*$  пересечения прямых  $v = H((p_1, 1 - p_1), 2)$  и  $v = H((p_1, 1 - p_1), 3)$ . Решая уравнение  $-6p_1 + 4(1 - p_1) = 2p_1 - 3(1 - p_1)$ , получим  $p_1^* = 7/15$ . Таким образом, оптимальная смешанная стратегия первого игрока  $p^* = (7/15, 8/15)$ . Значение цены игры  $v^* = \max_p \min_q H(p, q) = \max_{p_1} \min_j \{H((p_1, 1 - p_1), j)\}$  получается очевидным образом:

$$v^* = H((7/15, 8/15), 2) = H((7/15, 8/15), 3) = -\frac{2}{3}.$$

Для нахождения оптимальной стратегии  $q^* = (q_1^*, q_2^*, q_3^*)$  второго игрока воспользуемся теоремой о дополняющей нежесткости в форме (5.11). Так как  $H(p^*, 1) - v^* = 13/15 - (-2/3) > 0$ , то необходимо  $q_1^* = 0$ . С другой стороны, поскольку  $p_1^* > 0$  и  $p_2^* > 0$  из

(5.11) следует  $H(1, q^*) - v^* = 0$  и  $H(2, q^*) - v^* = 0$ . Получаем систему уравнений для  $q_j^*$ :

$$\begin{aligned} 3q_1^* - 6q_2^* + 2q_3^* + \frac{2}{3} &= 0, \\ -q_1^* + 4q_2^* - 3q_3^* + \frac{2}{3} &= 0, \\ q_1^* + q_2^* + q_3^* &= 1, \quad q_1^* = 0. \end{aligned}$$

Решая эту систему, окончательно получим  $q_1^* = 0$ ,  $q_2^* = 1/3$ ,  $q_3^* = 2/3$ .

Для игр, в которых две стратегии имеет второй игрок (игры  $m \times 2$ ), все делается аналогично рассмотренному примеру. Необходимо лишь графически определять не  $\max_p \min_j \{H(p, j)\}$ , а  $\min_{\substack{q_1+q_2=1 \\ q_j \geq 0}} \max_i \{H(i, q)\}$ .

### 5.3.3. Доминируемые и активные стратегии

Рассмотрим антагонистическую игру с матрицей  $A = [a_{ij}]$ .

**Определение.** Чистая  $k$ -я стратегия (строка  $k$  матрицы  $A$ ) первого игрока *доминирует* его  $i$ -ю стратегию (строку  $i$ ), если

$$a_{k1} \geq a_{i1}, a_{k2} \geq a_{i2}, \dots, a_{kn} \geq a_{in} \text{ и } \exists j_0 : a_{kj_0} > a_{ij_0}.$$

Доминируемость стратегий будем обозначать как  $k \succ i$ . Аналогично определяется доминируемость чистых стратегий (столбцов матрицы) второго игрока, только знаки неравенств меняются на противоположные:

$$k \succ j \iff a_{1k} \leq a_{1j}, a_{2k} \leq a_{2j}, \dots, a_{mk} \leq a_{mj} \text{ и } \exists i_0 : a_{i_0k} < a_{i_0j}.$$

Так, например, для игры с матрицей

2	-6	3	-5
-4	1	2	5
3	-5	3	-4

для первого игрока выполнено  $3 \succ 1$ , а для второго —  $2 \succ 3$ .

**Теорема 5.4.** Если у первого игрока его  $i_0$ -я чистая стратегия является доминируемой, т. е. существует такая стратегия  $k$ , что  $k \succ i_0$ , то в оптимальную смешанную стратегию  $p^*$  первого игрока  $i_0$ -я стратегия входит с нулевой вероятностью:  $p_{i_0}^* = 0$ .

**Доказательство.** Предположим противное:  $p_{i_0}^* > 0$ . Залишем математическое ожидание выигрыша первого игрока в виде  $H(p^*, q^*) = \sum_i H(i, q^*) p_i^*$ . В силу определения доминируемости

$$H(i_0, q^*) = a_{i_01} q_1^* + \dots + a_{i_0j_0} q_{j_0}^* + \dots + a_{i_0n} q_n^* < a_{k1} q_1^* + \dots + a_{kj_0} q_{j_0}^* + \dots + a_{kn} q_n^* = H(k, q^*).$$

Тогда

$$H(p^*, q^*) = \sum_{i \neq k, j_0} H(i, q^*) p_i^* + H(k, q^*) p_k^* + H(i_0, q^*) p_{i_0}^* < \sum_{i \neq k, j_0} H(i, q^*) p_i^* + H(k, q^*) (p_k + p_{i_0}) = \\ = H(\tilde{p}, q^*),$$

где в новой смешанной стратегии  $\tilde{p}$  первого игрока  $\tilde{p}_k = p_k^* + p_{i_0}^*$ ,  $\tilde{p}_{i_0} = 0$ ,  $\forall i \neq k, i_{i_0} \tilde{p}_i = p_i^*$ . Полученное неравенство  $H(p^*, q^*) < H(\tilde{p}, q^*)$  противоречит определению ситуации равновесия  $(p^*, q^*)$ .  $\square$

Из этой теоремы можно сделать следующий практический вывод: доминируемые строки (столбцы) матрицы выигрышей можно вычеркивать. Это во многих случаях упрощает анализ и решение игры.

**Определение.** Чистая стратегия игрока называется *активной*, если она входит в его оптимальную смешанную стратегию с положительной вероятностью.

**Теорема 5.5.** Если  $i_0$ -я чистая стратегия первого игрока является активной (т. е.  $p_{i_0}^* > 0$ ), то

$$v^* = H(p^*, q^*) = H(i_0, q^*). \quad (5.12)$$

*Доказательство.* Утверждение (5.12) есть не что иное, как запись «теоремы о дополняющей нежесткости» для матричной игры (см. (5.10)).  $\square$

Указанное свойство активной стратегии  $i_0$  создает некоторую иллюзию ее детерминированного, постоянного использования против оптимальной смешанной стратегии противника, ведь по теореме 5.5 результат для игрока будет оптимальным, как если бы он играл оптимальной смешанной стратегией. Однако это не совсем так. Разумный противник через несколько партий игры обнаружит, что игрок использует только одну чистую стратегию  $i_0$ . Тогда он сменит свою оптимальную смешанную стратегию на такую чистую, которая нанесет первому игроку максимальный ущерб против его активной стратегии  $i_0$ .

#### 5.3.4. Непрерывные игры на квадрате

В этом разделе рассматриваются антагонистические игры вида  $\Gamma = \langle X, Y, F(x, y) \rangle$ , где  $X = Y = [0, 1]$ . Нас будут интересовать условия, при которых эта игра имеет ситуацию равновесия (седловую точку)  $(x^*, y^*)$  в чистых стратегиях:  $x^* \in [0, 1]$ ,  $y^* \in [0, 1]$ . Вопрос этот представляет как теоретический, так и практический интерес.

Для одного класса функций выигрыша справедлива следующая

**Теорема 5.6.** Если функция  $F(x, y)$  непрерывна по совокупности переменных, строго вогнута по  $x$  и строго выпукла по  $y$ <sup>1</sup>, то на квадрате  $[0, 1] \times [0, 1]$  существует оптимальная по Нэшу точка  $(x^*, y^*)$ .

*Доказательство.* Определим две функции  $\varphi, \psi : [0, 1] \rightarrow [0, 1]$ :

$$\varphi(y) = \arg \max_{x \in [0, 1]} F(x, y), \quad \psi(x) = \arg \min_{y \in [0, 1]} F(x, y).$$

Эти функции непрерывны (докажите!). Пусть  $(x^*, y^*)$  – решение системы уравнений

$$\begin{aligned} x &= \varphi(y), \\ y &= \psi(x). \end{aligned} \tag{5.13}$$

Покажем, что пара  $(x^*, y^*)$  удовлетворяет определению (5.4) ситуации равновесия для антагонистической игры. Действительно,  $F(x^*, y^*) = F(\varphi(y^*), y^*) \geq F(x, y^*) \forall x \in [0, 1]$ . С другой стороны,  $F(x^*, y^*) = F(x^*, \psi(x^*)) \leq F(x^*, y) \forall y \in [0, 1]$ . Объединяя эти неравенства, окончательно получим

$$\forall x \in [0, 1] \quad F(x, y^*) \leq F(x^*, y^*) \leq F(x^*, y) \quad \forall y \in [0, 1]. \quad \square$$

**Пример 4.** Пусть  $F(x, y) = -2x^2 + y^2 + 3xy - x - 2y$ ,  $x, y \in [0, 1]$ . Очевидно, что условия теоремы 5.6 выполнены (вогнутость по  $x$  следует из  $F''_x = -2 < 0$ , выпуклость по  $y$  – из  $F''_y = 1 > 0$ ). Построим функции  $\varphi(y)$  и  $\psi(x)$ . Для определения  $\varphi(y)$  надо найти максимум  $\max_{x \in [0, 1]} F(x, y)$ . Для этого решим относительно  $x$  уравнение  $F'_x(x, y) = 0$

$$F'_x(x, y) = -4x + 3y - 1 = 0, \quad x = \frac{3y - 1}{4}.$$

Условие  $0 \leq x \leq 1$  выполняется при  $y \in [1/3, 1]$ , при  $y > 1/3$  производная  $F'_x < 0$  и функция  $F(x, y)$  убывает по  $x$ . Поэтому функция  $\varphi(y)$  определяется следующим образом:

$$\varphi(y) = \arg \max_{x \in [0, 1]} F(x, y) = \begin{cases} 0, & \text{если } y \in [0, 1/3); \\ \frac{3y - 1}{4}, & \text{если } y \in [1/3, 1]. \end{cases} \tag{5.14}$$

Аналогичными рассуждениями для  $\psi(x)$  получим

$$\psi(x) = \arg \min_{y \in [0, 1]} F(x, y) = \begin{cases} \frac{2 - 3x}{2}, & \text{если } x \in [0, 2/3); \\ 0, & \text{если } x \in [2/3, 1]. \end{cases} \tag{5.15}$$

<sup>1</sup>Напомним, что функция называется строго вогнутой (строго выпуклой), если отрезок прямой линии, соединяющий две точки графика этой функции, всегда будет целиком лежать под (над) соответствующим участком кривой. У строго вогнутой (строго выпуклой) непрерывной функции на отрезке существует единственная точка максимума (минимума).

Определим теперь  $(x^*, y^*)$ , решая графически систему уравнений (5.13) (рис. 5.5). Непосредственно из рассмотрения графиков функций  $x = \varphi(y)$  и  $y = \psi(x)$ , изображенных в одной системе координат  $(x, y)$ , для  $x^*, y^*$  получаем систему уравнений следующего вида:

$$\begin{cases} x^* = \frac{3y^* - 1}{2}, \\ y^* = \frac{2 - 3x^*}{2} \end{cases} \Rightarrow x^* = \frac{4}{17}, y^* = \frac{11}{17}.$$

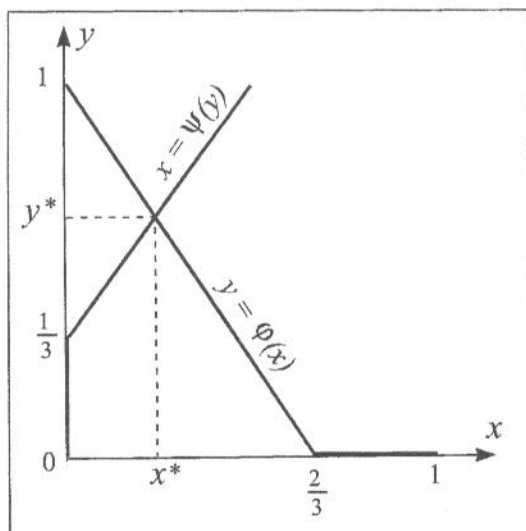


Рис. 5.5. Нахождение ситуации равновесия в непрерывной игре на квадрате

Теорема 5.6 переносится без изменений на случай многомерных стратегий игроков  $x \in M_x \subset \mathbb{R}^m$ ,  $y \in M_y \subset \mathbb{R}^n$ , где  $M_x, M_y$  – выпуклые множества.

## 5.4. Позиционные игры

Многие реальные конфликтные ситуации имеют длительный характер. Их участники действуют неоднократно и с учетом информации о предшествующем развитии конфликта. Для соответствующих моделей общие теоремы существования решения для игр в нормальной форме не позволяют находить или даже конкретизировать оптимальное поведение из-за большого числа возможных стратегий. Для решения динамических игр часто удобно использовать *позиционное* представление игры.

Под позиционной игрой  $n$  лиц понимается:

1. Ориентированное дерево  $\Gamma$  с выделенным корнем, имеющим номер 0. Вершина 0 – начальная позиция, вершины  $s \neq 0$ , имеющие потомков, – промежуточные позиции, а терминальные вершины  $t$  – заключительные позиции.



2. Для терминальных вершин  $t$  заданы  $n$ -мерные векторы  $V(t) = (v_1(t), \dots, v_n(t))$  выигрышей игроков.
3. Задано разбиение вершин (кроме терминальных) на  $n+1$  подмножеств очередности хода  $S_i$ , и в каждой вершине  $s \in S_i$  ход делает игрок с номером  $i$  ( $i = 0$  означает, что ход делается случайно условным игроком «Природа»).
4. Для позиций из  $S_0$  заданы распределения вероятностей на исходящих дугах.
5. Задано разбиение каждого множества  $S_i$ ,  $i \neq 0$ , на *информационные* подмножества  $I_{ij}$ ,  $S_i = \bigcup_j I_{ij}$ , причем все позиции из одного информационного множества имеют одинаковое количество дочерних позиций, а материнские позиции с дочерними всегда находятся в разных информационных множествах.
6. Для каждой позиции  $s \in I_{ij}$  существует нумерация (или разметка) вариантов ходов, и фактически выбор хода состоит в выборе номера (метки) варианта.

Поясним определение позиционной игры на примере биматричной игры «продавец – покупатель» (пример 2 на стр. 59). Будем теперь рассматривать этот конфликт развивающимся во времени: сначала делает ход игрок 1 (продавец), выбирая одну из своих стратегий (обвес, честное взвешивание), затем игрок 2 (покупатель) делает ход, выбирая свою стратегию из множества {поверить, проверить}, и на этом игра заканчивается. Предположим, что покупатель не знает, обманул его продавец или нет. Дерево игры приведено на рис. 5.6 (а). Поясним его элементы. Каждая терминальная вершина соответствует определенной партии игры, например, если игрок 1 выбрал стратегию «обман», а игрок 2 – «поверил», то игра заканчивается в терминальной вершине с выигрышем игрока 1, равным 1, и выигрышем игрока 2, равным -1. Множество  $S_1$  позиций,

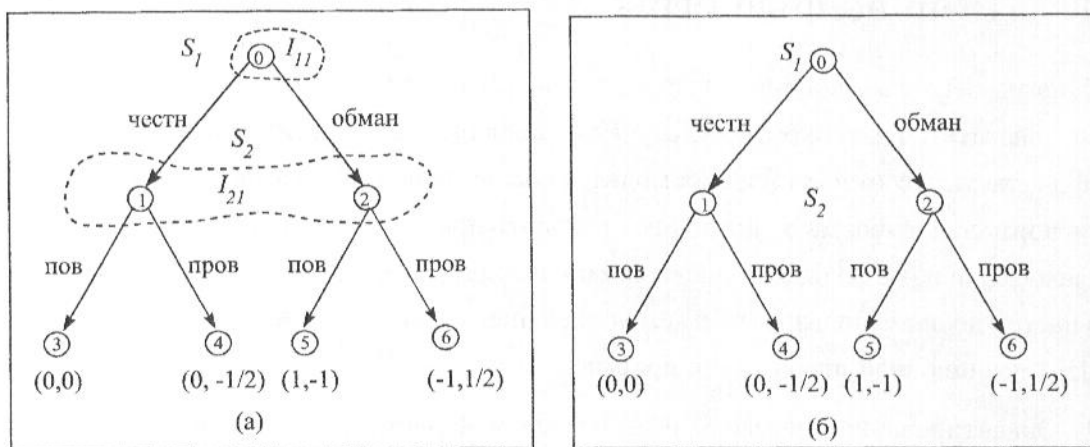


Рис. 5.6. Дерево игры «продавец – покупатель»

в которых делает ход игрок 1, состоит из одной вершины (позиции) с номером 1. Множество  $S_2 = \{2, 3\}$  состоит из позиций с номерами 2 и 3, в которых ход делает игрок 2. Информационное множество  $I_{11}$  игрока 1 состоит из одной позиции 1, т. е. совпадает с  $S_1$ . У игрока 2 тоже только одно информационное множество (оно выделено пунктирной замкнутой линией)  $I_{21} = S_2 = \{2, 3\}$  – это те позиции, в которых его очередь делать ход и которые он *не различает*. Другими словами, второй игрок не знает, в какой именно позиции (1 или 2) он находится после хода игрока 1.

**Определение.** Позиционная игра называется игрой с полной информацией, если каждое информационное множество состоит только из одной позиции, а случайный ход может быть сделан только из корневой вершины.

Рассмотрим ту же игру «продавец – покупатель», только теперь будем предполагать, что покупатель всегда заметит обман продавца в случае обвешивания. Дерево этого варианта игры изменится следующим образом (рис. 5.6 (б)). Теперь игрок 2 всегда знает, в какой именно позиции (2 или 3) он находится после выбора игрока 1. У него два информационных множества, каждое из которых состоит из одной вершины:  $I_{21} = \{2\}$ ,  $I_{22} = \{3\}$ . В соответствии с определением это игра с полной информацией.

Для задания стратегии  $i$ -го игрока нужно для каждого его информационного множества  $I_{ij}$  указать, какой выбор должен сделать игрок, попав в позицию  $s \in I_{ij}$ . Таким образом, чистая стратегия игрока – это детерминированное отображение  $\mu_i(I_{ij})$ , заданное на информационных множествах игрока со значениями в множестве номеров (меток) вариантов его выбора. Смешанная стратегия игрока задается набором распределений вероятностей  $p_i(I_{ij})$  для каждого информационного множества  $I_{ij}$  игрока.

Набор из чистых стратегий всех игроков  $\mu = [\mu_1, \dots, \mu_n]$  называется ситуацией игры в чистых стратегиях. Каждая ситуация  $\mu$  приводит процесс игры в соответствующую терминальную позицию  $t(\mu)$ , в которой  $i$ -й игрок получает свой выигрыш  $V_i(t(\mu))$ .

Ситуацию  $\mu^*$  в чистых стратегиях будем называть оптимальной по Нэшу, или ситуацией равновесия, если  $\forall i = 1, \dots, n$  справедливо:  $i$ -му игроку невыгодно менять свою индивидуальную стратегию  $\mu_i^*$  на любую другую  $\mu_i$  при условии, что остальные игроки  $j$  не меняют свои индивидуальные стратегии  $\mu_j^*$ . Справедлива следующая

**Теорема 5.7.** *Позиционная игра с полной информацией всегда имеет ситуацию равновесия в чистых стратегиях.*

Идея доказательства настолько проста, что ее достаточно пояснить на примере игры «продавец – покупатель». Рассмотрим дерево этой игры, приведенное на рис. 5.6 (б). Припишем каждой нетерминальной позиции  $s$ , в которой делает ход  $i$ -й игрок, гарантированные выигрыши  $\hat{v}_1(s)$ ,  $\hat{v}_2(s)$  игроков, которые они получают в том случае, если бы дерево игры начиналось с позиции  $s$ . Правило расчета  $\hat{v}_1(s)$ ,  $\hat{v}_2(s)$  очевидно: учитывая

ются величины  $\hat{v}_1(s_k)$ ,  $\hat{v}_2(s_k)$ , где  $s_k$  – непосредственные потомки («дочки») позиции  $s$ . Если в позиции  $s$  ход делает игрок 1, то

$$\hat{v}_1(s) = \max_{s_k - \text{«дочки» } s} \{\hat{v}_1(s_k)\} = \hat{v}_1(s_{k_0}), \quad \hat{v}_2(s) = \hat{v}_2(s_{k_0}). \quad (5.16)$$

Если  $s \in S_2$ , т. е. в ситуации  $s$  ход делает игрок 2, то в равенствах (5.16)  $\hat{v}_1$  и  $\hat{v}_2$  меняются местами.

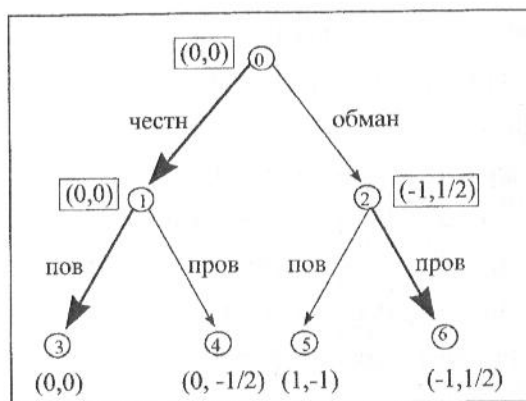


Рис. 5.7. Построение решения игры

Расчет гарантированных выигрышей надо сопровождать указанием оптимального выбора игрока, который делает ход в рассматриваемой позиции  $s$ . На рис. 5.7 выбор хода соответствует выделенной дуге дерева. Начнем этот процесс с вершин, непосредственными потомками («дочками») которых являются терминальные вершины, а далее будем двигаться вверх по дереву, пока не дойдем до корня – вершины с номером 0.

В рассматриваемой игре в позиции 0 ход делает игрок 1. Посчитаем для вершины 0 гарантированные выигрыши  $\hat{v}_1(0)$ ,  $\hat{v}_2(0)$ . На этом процесс расчета фактически заканчивается. Далее надо восстановить гарантированные стратегии игроков. Это легко сделать по отмеченным дугам.

Окончательный результат таков: стратегия игрока 1 задается отображением  $\mu_1^*(0) = \text{«честн»}$ , а стратегия игрока 2 – отображением  $\mu_2^*(1) = \text{«пов»}$ ,  $\mu_2^*(2) = \text{«пров»}$ . Если игроки будут использовать эти стратегии, то процесс игры пойдет по цепочке выделенных дуг, которая закончится в терминальной вершине 3. Очевидно, что каждый игрок не сможет улучшить для себя ситуацию  $\mu^*$  только за счет изменения своей индивидуальной стратегии, если другой игрок не меняет свою оптимальную стратегию, т. е.  $\mu^*$  – ситуация равновесия.  $\square$

Метод построения решения «снизу – вверх» по дереву игры очевидным образом переносится на случай большего числа ходов и большего числа игроков. В этом методе просматривается идея динамического программирования: по существу соотношения (5.16) есть не что иное, как рекуррентные уравнения Беллмана для гарантированных выигрышей игроков.

*Следствие из теоремы 5.7.* В шахматах существует ситуация равновесия в чистых стратегиях.

Действительно, шахматы – это позиционная игра с полной информацией, имеющая конечное дерево. К сожалению (а может быть, и к счастью – иначе игра потеряла бы всю свою привлекательность), дерево этой игры настолько велико, что исследовать его за ра-

зумное время описанным выше приемом не представляется возможным. Это достаточно общая ситуация для реальных позиционных игр.

Если позиционная игра не является игрой с полной информацией, то решения в чистых стратегиях, вообще говоря, не существует. В этом случае надо использовать смешанные стратегии. Смешанной стратегией  $i$ -го игрока будет набор распределений вероятностей  $\{p(I_{ij})\}$ ,  $j = 1, 2, \dots$ , где  $k$ -я компонента распределения  $p(I_{ij})$  – это вероятность выбора  $k$ -й альтернативы в случае, когда  $i$ -й игрок попал в свое  $j$ -е информационное множество  $I_{ij}$ . Алгоритм решения игры в смешанных стратегиях здесь не рассматривается.

**Задача.** Определить выигрыш в игре с полной информацией, в которой первый ход из позиции 0 делает игрок «Природа», случайно выбирая свою стратегию 1 или 2 в соответствии с распределением вероятностей  $p = (0.7, 0.3)$  (дерево игры приведено на рис. 5.8).

Здесь  $S_1 = \{1, 2\}$ ,  $S_2 = \{3, 4, 5, 6, 7\}$ , ходы игрока 1 обозначены латинскими буквами, ходы игрока 2 – греческими.

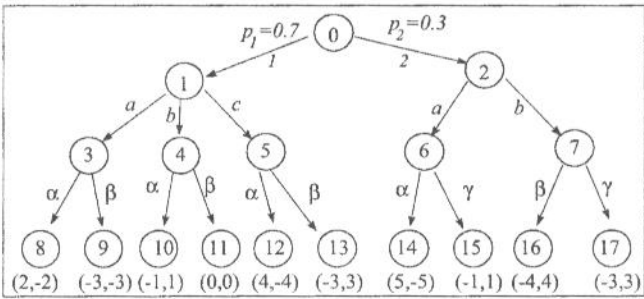


Рис. 5.8. Дерево игры

# Литература

1. Таха, Х. А. Введение в исследование операций / Х. А. Таха. – М.: Издательский дом «Вильямс», 2005.
  2. Вентцель, Е. С. Исследование операций. Задачи, принципы, методология / Е. С. Вентцель. – М.: Наука. Гл. ред физ.-мат. лит., 1988.
  3. Сигал, И. Х. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы: учеб. пособие / И. Х. Сигал, А. П. Иванова. – М.: ФИЗМАТЛИТ, 2003.
  4. Маймика, Э. Алгоритмы оптимизации на сетях и графах / Э. Маймика. – М.: Мир, 1981.
  5. Васин, А. А. Теория игр и модели математической экономики: учеб. пособие / А. А. Васин, В. В. Морозов. – М.: МАКС Пресс, 2005.
- 

Учебное издание

**Короткин Алексей Абрамович**

**Фокин Владимир Григорьевич**

**Модели и алгоритмы исследования операций**

Учебное пособие

Редактор, корректор В. Н. Чулкова

Компоновка и верстка А. А. Короткин

Подписано в печать 26.06.06. Формат 60×84<sup>1</sup>/<sub>8</sub>.

Бумага тип. Усл. печ. л. 8,3. Уч.-изд. л. 5,0.

Тираж 100 экз. Заказ 061/06

Оригинал-макет подготовлен в редакционно-издательском отделе ЯрГУ.

Отпечатано на ризографе.

Ярославский государственный университет

150000 Ярославль, ул. Советская, 14