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

Инженерно-экономический факультет

Кафедра экономической информатики

Т. Г. Пинчук, С. А. Поттосина

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ В ЭКОНОМИКЕ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для направлений специальности 1-40 05 01-02 «Информационные системы и технологии (в экономике)», 1-40 05 01-08 «Информационные системы и технологии (в логистике)»

Минск БГУИР 2017

УДК 519.8:33(076)
ББК (22.18+65)я73
П32

Р е ц е н з е н т ы:

кафедра математического и информационного обеспечения экономических систем учреждения образования «Гродненский государственный университет имени Я. Купалы» (протокол №6 от 26.05.2016);

профессор кафедры теории вероятностей и математической статистики Белорусского государственного университета, доктор физико-математических наук, профессор Г. А. Медведев

Пинчук, Т. Г.

П32 Исследование операций в экономике : учеб.-метод. пособие / Т. Г. Пинчук, С. А. Поттосина. – Минск : БГУИР, 2017. – 115 с. : ил.

ISBN 978-985-543-334-8.

Содержит основные разделы исследования операций: модели поведения потребителей и производителей как модели нелинейного программирования, модели и методы целочисленного программирования, модели динамического программирования, оптимизационные задачи на графах и сетях, игровые модели исследования операций, марковские модели с доходами.

Рассматриваются прикладные задачи по некоторым разделам с описанием постановки задачи и алгоритма их решения.

УДК 519.8:33(076)
ББК (22.18+65)я73

ISBN 978-985-543-334-8

СОДЕРЖАНИЕ

Введение

Исследование операций – наука о выборе разумных, научно обоснованных решений во всех областях целенаправленной человеческой деятельности. Под операцией понимают совокупность действий, направленных на достижение определенной цели. Действующими факторами операции называются объективные условия и обстоятельства, непосредственно влияющие на ее исход. Факторы, которыми распоряжается оперирующая сторона для достижения цели операции, называются контролируемыми. Для их обозначения будем использовать букву x. Факторы операции, которые не контролируются оперирующей стороной, называются неконтролируемыми. Среди них можно выделить неопределенные факторы y (известно лишь множество значений факторов) и случайные факторы y (функция распределения величин z известна полностью либо с точностью до неизвестных параметров).

Критерием эффективности операции называется показатель $F(x,y,z)$ требуемого соответствия между результатом предпринимаемых оперирующей стороной действий и целью операций. Состоянием операции в некоторый момент времени t называется совокупность ее характеристик в этот момент. Всякая операция представляет собой процесс (детерминированный или вероятностный), протекающий во времени, проходящий различные этапы развития и завершающийся получением конечного результата.

Математической моделью операции называются формальные взаимоотношения, устанавливающие связь принятого критерия эффективности с действующими факторами операции. Чтобы построить математическую модель, необходимо оценить количественно воздействия рассматриваемых факторов и определить совокупность параметров, формально представляющих эти факторы. Решением, связанным с выбранной математической моделью, называется конкретный набор значений управляемых (контролируемых) параметров.

Основная задача исследования операций – найти в рамках принятой модели те решения, которым отвечают F. Создание математической модели – самая важная и ответственная часть операционных исследований, требующая глубокого знания не только математики, но и существа моделируемых явлений. Как детерминированные, так и вероятностные модели содержат подлежащую оптимизации целевую функцию и некоторую совокупность ограничений. В задачах производственно-экономического характера целевая функция чаще всего представляет собой подлежащую максимизации прибыль или подлежащие минимизации затраты. При решении практических операционных задач находят применение различные классы оптимизационных моделей (модели математического программирования, имитационные модели, графовые и сетевые модели) и методы оптимизации, основанные на использовании математического программирования и теории графов.

1 Модели поведения потребителей и производителей как модели нелинейного программирования

1.1 Постановка задачи нелинейного программирования

Задача нелинейного программирования (НЛП) – это задача выбора таких неотрицательных значений некоторых переменных, подчиненных системе ограничений в форме неравенств, при которых достигается максимум или минимум данной функции.

Формальная постановка задачи НЛП:

$$F(x_1, \ldots, x_n) \to \max \qquad (1.1)$$

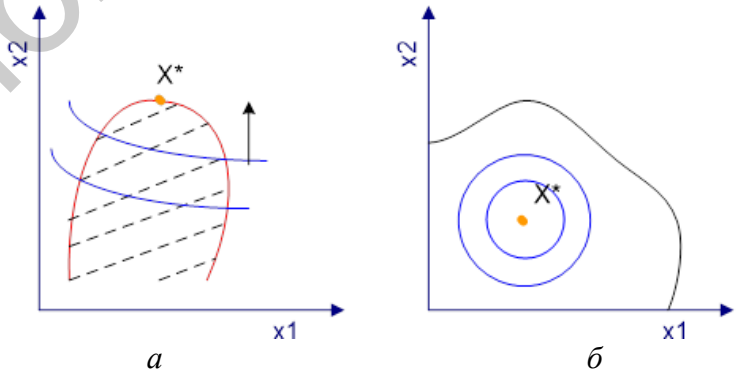
при условии, что

$$g_1(x_1, \ldots, x_n) \le b_1,$$
$$\ldots$$
$$g_i(x_1, \ldots, x_n) \le b_i,$$
$$\ldots$$
$$g_m(x_1, \ldots, x_n) \le b_m,\ i = 1, \ldots, m$$
$$x_1, \ldots, x_n \ge 0.$$

Величины $x_1, \ldots, x_n$ представляют собой инструментальные переменные. Функция $F(x_1,\ldots, x_n)$ – целевая функция, $g_i(x_1,\ldots,x_n)$, $i = 1, \ldots, m$ – функция ограничений, b_i – константы ограничений.

На переменные накладывается условие неотрицательности, которое является необходимым практически во всех экономических задачах.

Геометрически задача НЛП состоит в отыскании точки или множества точек из допустимого множества, где достигается поверхность наибольшего уровня. Решение (глобальный максимум) существует, если допустимое множество не пустое и ограниченное. Решение может принадлежать либо границе, либо внутренней части допустимого множества (рисунок 1.1).



а *б*

а – граничное решение; *б* – внутреннее решение

Рисунок 1.1 – Нелинейное программирование

Общая задача НЛП очень сложна и до сих пор не имеет полного решения. Она становится легче, если ограничимся случаем, когда область ограничений выпукла, а целевая функция выпукла или вогнута. В этом случае задача НЛП становится задачей выпуклого программирования. Условия выпуклости играют важную роль. Дело в том, что в этом случае локальный максимум целевой функции, находящийся внутри допустимого множества или на его границе, является глобальным максимумом, а множество точек, на котором достигается глобальный максимум, выпукло. Если дополнительно предполагается, что целевая функция строго вогнута, то задача имеет единственное решение.

Подход Куна – Таккера к общей задаче НЛП состоит в том, что вводится вектор-строка множителей Лагранжа $\lambda = (\lambda_1,\ldots,\lambda_m)$, число которых равно числу ограничений-неравенств, и определяется функция Лагранжа:

$$L(x, \lambda) = F(x) + \lambda(b - g(x)). \tag{1.2}$$

В этом случае условия Куна – Таккера в векторной форме записываются

$$\begin{gathered}
\frac{\partial L}{\partial x}(x^*,\lambda^*) = \frac{\partial F}{\partial x}(x^*) - \lambda\frac{\partial g}{\partial x}(x^*) \le 0,\\
\frac{\partial L}{\partial x}(x^*,\lambda^*)x^* = \left(\frac{\partial F}{\partial x}(x^*) - \lambda\frac{\partial g}{\partial x}(x^*)\right)x^* = 0,\ x^* \ge 0,\\
\frac{\partial L}{\partial x}(x^*,\lambda^*) = b - g(x^*) \ge 0,\\
\lambda^*\frac{\partial L}{\partial x}(x^*,\lambda^*) = \lambda^*\left(b - g(x^*)\right) = 0,\ \lambda^* \ge 0.
\end{gathered} \tag{1.3}$$

Эти условия являются необходимыми и достаточными для существования (строгого) локального максимума (минимума), если целевая функция (строго) вогнутая (выпуклая), а функции ограничений выпуклые, т. е. задача вида (1.1) является задачей выпуклого программирования.

Точка (x^*,λ^*) является седловой точкой функции Лагранжа, т. е. точкой, максимизирующей функцию по совокупности всех неотрицательных переменных x и минимизирующей ее по совокупности всех неотрицательных множителей Лагранжа λ:

$$L(x, \lambda^*) \le L(x^*, \lambda^*) \le L(x^*, \lambda) \text{ при всех } x \ge 0,\ \lambda \ge 0. \tag{1.4}$$

Задача отыскания неотрицательных векторов (x^*,λ^*), удовлетворяющих (1.4), называется задачей о седловой точке.

По теореме Куна – Таккера x является решением задачи НЛП, если (x^*,λ^*) является решением задачи о седловой точке, и при некоторых условиях x^* является решением задачи НЛП лишь в том случае, когда существует такой вектор λ, что (x^*,λ^*) является решением задачи о седловой точке.

Теорема Куна – Таккера позволяет перейти от задачи выпуклого программирования с ограничениями к нахождению седловой точки функции без ограничений. Единственным условием является требование неотрицательности переменных, которое не мешает применять классические методы нахождения седловой точки.

1.2 Модель поведения потребителя

Конкретное решение потребителя о покупке определенного набора товаров математически можно представить как выбор конкретной точки в пространстве товаров C. Пусть n – конечное число товаров, $x = (x_1, \ldots, x_n)^{\mathrm{T}}$ – вектор-столбец товаров, приобретенных потребителем за определенный срок при заданных ценах и доходе за тот же срок.

Пространство товаров – множество всевозможных наборов товаров x с неотрицательными компонентами:

$$C = \{x: x \geq 0\}.$$

Отношение предпочтения, характерное для теории потребления, задается с помощью непрерывной действительной функции полезности $U(x)$, определенной на пространстве товаров C.

Предельной полезностью товара называется предел отношения приращений полезности к вызвавшему этот прирост приращению товара:

$$\lim_{\Delta x_i \to 0} \frac{\Delta U(x)}{\Delta x_i} = \frac{\partial U(x)}{\partial x_i} \ . \qquad (1.5)$$

С точки зрения потребителя, наличие множества товаров, обладающих одинаковой полезностью, означает возможность замены одного набора товаров другим равноценным в смысле одинаковой степени предпочтения, в том числе возможность замены одного товара другим.

Предельная норма замены одного товара другим равна отношению предельных полезностей этих товаров. Норма замены показывает, сколько требуется единиц второго товара, чтобы заменить выбывшую малую единицу первого товара:

$$-\frac{dx_2}{dx_1}=\frac{\partial U(x)/\partial x_1}{\partial U(x)/\partial x_2}. \qquad (1.6)$$

Бюджетное множество – множество тех наборов товаров, которое может приобрести потребитель, имея доход I:

$$X=\{x: px\le I\},$$

где $p=(p_1, \ldots, p_n)$ – вектор-строка цен.

В теории потребления предполагается, что потребитель всегда стремится максимизировать свою полезность, и единственное, что его сдерживает, – это ограниченность дохода I. Таким образом, имеет место следующая модель нелинейного программирования:

$$\begin{gathered}\max U(x)\\ px\le I\\ x\ge 0.\end{gathered} \qquad (1.7)$$

Считается, что потребитель покупает все виды товаров и услуг. В противном случае можно уменьшить размерность пространства товаров, исключив из рассмотрения непокупаемые товары. Считая, что некоторые товары были куплены, получим, что оптимальный множитель Лагранжа должен быть положительным, т. е. весь доход должен быть израсходован. Поэтому условия Куна – Таккера принимают следующий вид:

$$\begin{cases}\dfrac{\partial U(x)}{\partial x}-\lambda p=0,\\ I-px=0.\end{cases} \qquad (1.8)$$

Решениями этой системы уравнений являются:

$$x^*=x^*(p, I) \text{ и } \lambda^*=\lambda^*(p, I). \qquad (1.9)$$

Первое уравнение называется функцией спроса $x_j^*=x_j^*(p_1, p_2, \ldots, p_n; I)$ как функция от цен на товары и дохода. Второе уравнение (1.9) отражает оптимальный множитель Лагранжа как функцию цен и дохода. Причем из (1.8) следует, что λ^* представляет собой предельную полезность денег – это количество, на которое увеличивается оптимальный уровень полезности, если произойдет малое приращение дохода, а именно:

$$\lambda^*=\frac{\partial U^*(x^*)}{\partial I}=\frac{\partial U(x^*)}{\partial I}. \qquad (1.10)$$

1.3 Модель поведения производителя (фирмы)

Фирма определяется как некоторая организация, производящая затраты экономических факторов для производства продукции и услуг, которые она продает потребителям или другим фирмам.

Пусть фирма производит только один вид продукции, используя несколько видов затрат. Если x_j есть количество затрат j-го вида (j = 1, 2, …, n), то вектор затрат представляет собой вектор-столбец $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$. Считается, что $x_j \geq 0$ и принимают любые вещественные значения (безгранично делимы).

Каждой точке пространства затрат соответствует единственный максимальный выпуск продукции. Связь между максимальным выпуском продукции и затратами называется технологической функцией:

$$q = f(x) = f(x_1, x_2, \ldots, x_n). \tag{1.11}$$

Существует подмножество пространства затрат, называемое экономической областью, в которой увеличение любого вида затрат не приводит к уменьшению выпуска продукции. То есть если $x^{(1)}$ и $x^{(2)}$ – две точки этой области, то

$$x^{(1)} \geq x^{(2)} \Rightarrow f(x^{(1)}) \geq f(x^{(2)}) .$$

Такая область характеризуется неотрицательностью всех первых частных производных производственной функции, которые называются предельными продуктами. Отсюда следует, что в экономической области

$$\frac{\partial f(x)}{\partial x_j} = MP_j(x) \geq 0, j = 1, 2, \ldots, n . \tag{1.12}$$

Пусть p – цена единицы продукции, выпускаемой фирмой, и w_j – цена единицы затрат j-го типа, так что $w = (w_1, w_2, \ldots, w_n)^{\mathrm{T}}$ – вектор-столбец цен затрат. Теория фирмы построена на предположении, что цель фирмы заключается в максимизации прибыли путем выбора видов и количества затрат при заданной производственной функции и ценах p и w.

Прибыль фирмы очевидно равна годовому доходу за вычетом издержек производства, где годовой доход вычисляется как готовая продукция, умноженная на цену выпуска:

$$\max_x \ \Pi(x) = pf(\mathrm{x}) - w^{\mathrm{T}} x. \tag{1.13}$$

Эта задача представляет собой задачу нелинейного программирования, в которой в качестве инструментальных переменных выступает вектор x, целевая функция выражается функцией прибыли.

Для случая $x_j > 0$, получим следующее решение задачи:

$$p\frac{\partial f(x)}{\partial x_j} - w_j = 0,\ j = 1, 2, \ldots, n. \qquad (1.14)$$

Решение этой системы называется функциями спроса на затраты:

$$x^* = x^*(p, w) = x_j^*(p, w_1, w_2, \ldots, w_n),\ j = 1, 2, \ldots, n. \qquad (1.15)$$

Подставляя это решение в $f(x)$, получим функцию предложения выпуска $q^* = f(x^*(p, w)) = q^*(p, w)$.

1.4 Монополия и монопсония

Случай совершенной конкуренции – это ситуация, когда на рынке бесконечно много фирм, производящих одинаковый товар, и бесконечно много продавцов и покупателей товаров, необходимых фирме. Именно вследствие этого цены на товар, производимый фирмой, и на товары, необходимые фирме, фиксированы.

Другим крайним случаем является ситуация, когда имеется лишь одна фирма, производящая какой-то товар. В этом случае говорят о монополии этой фирмы. Монополист имеет возможность произвольно устанавливать цену на товар. Но кривая спрос – цена существует объективно, и каждой цене p, установленной фирмой-монополистом на товар, соответствует свой спрос q, зависящий от цены p, т. е. $q = q(p)$. Можно эту зависимость написать и наоборот, в виде $p = p(q)$. При этом $dp/dq < 0$. Таким образом, монополист влияет на цену своего товара, варьируя свой выпуск (рисунок 1.2).

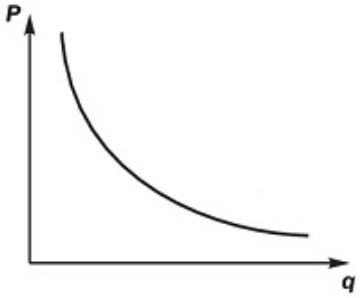


Рисунок 1.2 – Кривая спрос – цена для фирмы-монополиста

Другой крайний случай – это когда фирма является единственным потребителем какого-то фактора. В этом случае говорят о монопсонии. Таким образом, монополия – это один производитель (продавец), а монопсония – это один потребитель (покупатель). Имея в виду кривую цена – предложение, можно сказать, что монопсонист влияет на цену затрат путем варьирования величиной своих покупок данного вида затрат. Для него $w_j = w_j(x_j)$. Фирма может заставить поставщика производить больше товара, повышая цену на него и поэтому $dw_j / dx_j > 0$ (рисунок 1.3).

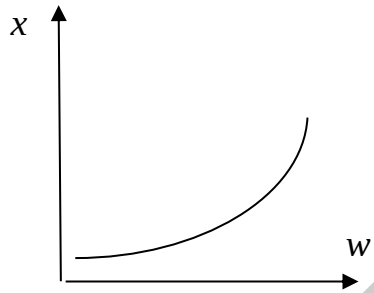


Рисунок 1.3 – Кривая цена – предложение для фирмы-монопсониста

Случаи монополии-монопсонии объединяются одним соотношением, определяющим прибыль фирмы:

$$\Pi = p(q) \cdot q - \sum_{j=1}^{n} w_j(x_j) \cdot x_j,$$

$$q = f(x_1, x_2, \ldots, x_n),$$

и задача выглядит так:

$$\begin{cases} p(q) \cdot q - \sum\limits_{j=1}^{n} w_j(x_j) \cdot x_j \Rightarrow \max\limits_{x}, \\ q = f(x_1, x_2, \ldots, x_n), \\ x_j \geq 0, \quad j = \overline{1, n}. \end{cases} \tag{1.16}$$

Запишем функцию Лагранжа:

$$L(q, x_1, x_2, \ldots, x_n, \lambda) = p(q) \cdot q - \sum_{j=1}^{n} w_j(x_j) x_j + \lambda(f(x_1, x_2, \ldots, x_n) - q).$$

Тогда необходимые условия экстремума приобретают вид

$$\frac{\partial L}{\partial q} = p(q) + q\frac{dp(q)}{dq} - \lambda = 0,$$

$$\frac{\partial L}{\partial x_j} = -w_j(x_j) + x_j\frac{dw_j}{dx_j} + \lambda\frac{\partial f}{\partial x_j} = 0,\ j = 1, \ldots, n, \qquad (1.17)$$

$$\frac{\partial L}{\partial \lambda} = f(x_1, x_2, \ldots, x_n) - q = 0.$$

Решение этой задачи и определяет стратегию фирмы-монополиста или фирмы-монопсониста.

Заметим, что из первого уравнения системы (1.17) можно выразить λ в явном виде и подставить во второе уравнение. Тогда система (1.17) примет вид

$$\left(p(q) + q\frac{dp}{dq}\right)\frac{\partial f}{\partial x_j} = w_j(x_j) + x_j\frac{dw_j}{dx_j},\ j = 1, \ldots, n, \qquad (1.18)$$

$$q = f(x_1, x_2, \ldots, x_n),$$

и λ исчезает из выкладок.

1.5 Модель взаимодействия производителей и потребителей

На рынке действует очень много фирм и очень много потребителей, причем фирма, как правило, выступает и как производитель товаров, и как потребитель товаров, созданных другими фирмами. Общую схему товарных потоков можно пояснить рисунком 1.4.

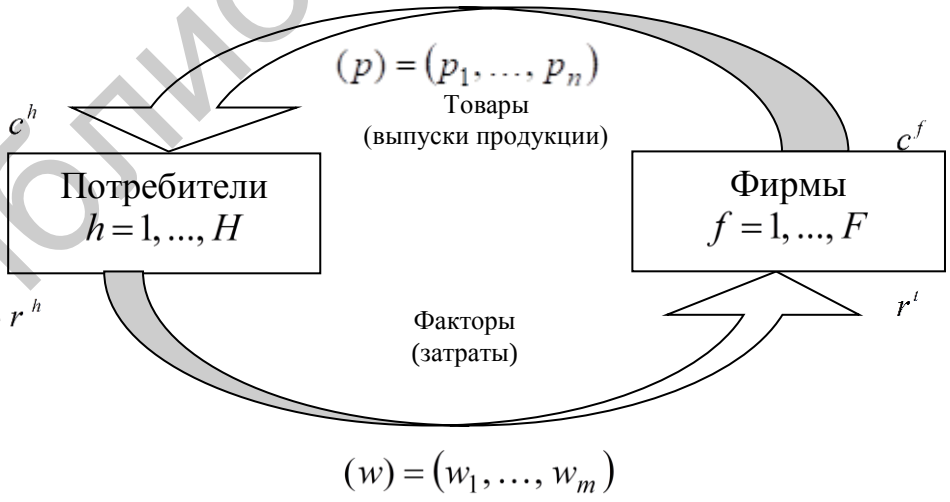


Рисунок 1.4 – Круговая диаграмма потоков

Рынок называется конкурентным, если его участники не могут влиять на цены. От фирм потребителей идут товары и услуги. Фирмы используют факто-

ры для производства товаров. Потребители, располагающие набором некоторых факторов, в том числе и рабочей силой, получают доход от их продажи на рынке факторов и используют его для приобретения товаров на рынке товаров.

Задача общего конкурентного равновесия заключается в анализе взаимосвязи двух основных элементов микроэкономики – потребителей и фирм, в терминах цен, объемов товаров и затраченных ресурсов. Каждый из перечисленных выше участников преследует свои цели, поэтому возникают конфликтные ситуации. Для того чтобы экономическая система функционировала нормально, индивидуальные действия различных участников должны быть согласованы.

В модели Вальраса, включающей конечное число потребителей и производителей, такое разрешение конфликтов достигается использованием некоторой системы цен, при которой индивидуальные планы участников становятся совместимыми. Такая равновесная ситуация и называется конкурентным равновесием.

Рассмотрим экономику, производящую N товаров (выпусков) и M факторов (видов затрат). Вектор $p=(p_1, p_2,...,p_n)$ характеризует цены товаров (выпусков), $w=(w_1, w_2,...,w_m)$ характеризует цены факторов (затрат). Предполагаем, что экономика является конкурентной в том смысле, что все потребители и фирмы действуют по заданным ценам.

В экономике существует F фирм, каждая из которых производит затраты на рынке факторов, для изготовления продуктов, продаваемых на рынке товаров. Пусть $r^f=(r_1{}^f, r_2{}^f,...,r_m{}^f)$ – вектор-столбец, составленный из объектов факторов, «купленных» фирмой f; $r_i{}^f$ – количество первичных затрат i-го вида фирмы f.

Пусть $c^f=(c_1{}^f, c_2{}^f,...,c_n{}^f)$ – вектор-столбец, составленный из объемов товаров, проданных фирмой f; $c_j{}^f$ – объем выпуска j-го продукта, продаваемого фирмой f. Тогда прибыль фирмы f может быть выражена в следующем матричном соотношении:

$$\Pi^f = pc^f - wr^f,\ f=1,2,...,F.$$

Каждая фирма максимизирует сумму своей прибыли при условии выполнения ограничений в форме производственной функции, которую можно записать в общем виде неявной функции $\Phi^f(c^f, r^f)=0$.

Сформулируем задачу для фирмы f:

$$\max_{c^f, r^f} (\Pi^f = pc^f - wr^f) \text{ при условии } \Phi^f(c^f, r^f)=0. \qquad (1.19)$$

В экономике есть H потребителей, каждый из которых владеет определенным фактором, который он может продать на рынке факторов по заданным ценам и получить доход. Кроме того, каждый потребитель может иметь свою долю в фирме как собственник и получатель чистой прибыли.

Пусть $c^h = (c_1{}^h, c_2{}^h, ..., c_n{}^h)$ – вектор-столбец товаров, потребляемых потребителем h; $c_j{}^h$ – количество j-го товара, потребленного потребителем h; $r^h = (r_1{}^h, r_2{}^h, ..., r_m{}^h)$ – вектор-столбец факторов, предложенных потребителем h, $r_i{}^h$ – количество i-го фактора, предложенного потребителем h. Тогда полезность, получаемая потребителем h, будет выражена функцией $U^h(c^h, r^h)$, где $h = 1, 2, ..., H$.

Пусть $s^{h}{}_{f}$ – доля участия потребителя h в фирме f. Участие потребителя h в фирмах обобщается вектором-строкой $s^h = (s^{h_1}, s^{h_2}, ..., s^{h_F})$.

Прибыль фирм записываем вектором-столбцом $\Pi = (\Pi^1, \Pi^2, ..., \Pi^F)^{\mathrm{T}}$.

Бюджетное ограничение для потребителя h запишем как

$$wr^h + s^h\Pi = pc^h.$$

Сформулируем задачу для потребителя h:

$$\max_{c^h, r^h} U^h(c^h, r^h) \text{ при условии, что } wr^h + s^h\Pi = pc^h. \qquad (1.20)$$

Необходимые условия максимизации прибыли в задаче (1.19) приводят к $(n+m+1)$ уравнениям:

$$\begin{cases} \dfrac{\partial L^f}{\partial c^f} = p + \lambda^f \dfrac{\partial \Phi^f}{\partial c^f} = 0, \quad (n) \\ \dfrac{\partial L^f}{\partial r^f} = -w + \lambda^f \dfrac{\partial \Phi^f}{\partial r^f} = 0, \quad (m) \\ \dfrac{\partial L^f}{\partial \lambda^f} = \Phi^f(c^f, r^f) = 0, \quad (1) \end{cases} \qquad \begin{cases} \lambda^f \dfrac{\partial \Phi^f}{\partial c^f} = -p, \\ \lambda^f \dfrac{\partial \Phi^f}{\partial r^f} = w, \\ \Phi^f(c^f, r^f) = 0. \end{cases} \qquad (1.21)$$

Поскольку эти уравнения справедливы для каждой из фирм, в итоге получим $F(n+m+1)$ уравнений.

Необходимое уравнение максимизации полезности при выполнении бюджетного ограничения может быть выражено системой $H(n+m+1)$ уравнений:

$$\begin{cases} \dfrac{\partial L^h}{\partial c^h} = \dfrac{\partial U^h}{\partial c^h} - \lambda^h p = 0, & (n) \\ \dfrac{\partial L^h}{\partial r^h} = \dfrac{\partial U^h}{\partial r^h} + \lambda^h w = 0, & (m) \\ \dfrac{\partial L^h}{\partial \lambda^h} = wr^h + s^h \Pi - pc^h = 0, & (1) \end{cases} \quad \begin{cases} \dfrac{\partial U^h}{\partial c^h} = \lambda^h p, \\ \dfrac{\partial U^h}{\partial r^h} = -\lambda^h w, \\ wr^h + s^h \Pi - pc^h = 0. \end{cases} \tag{1.22}$$

Равновесие на товарных рынках порождает *n* уравнений:

$$\sum_{h=1}^{H} c_j{}^h = \sum_{f=1}^{F} c_j{}^f \text{ , где } j = 1, 2, \ldots, n. \tag{1.23}$$

А равновесие на рынках факторов порождает *m* уравнений:

$$\sum_{h=1}^{H} r_i{}^h = \sum_{f=1}^{F} r_i{}^f \text{ , где } i = 1, 2, \ldots, m. \tag{1.24}$$

Таким образом, мы имеем $(n+m+1)(F+H)+(n+m)$ уравнений для описания рыночного механизма в задаче общего равновесия.

Закон Вальраса или основное равенство теории общего равновесия утверждает, что общая величина спроса должна быть равна общей величине предложения при какой-либо системе цен.

2 Модели и методы целочисленного программирования

2.1 Постановка задачи целочисленного программирования

По смыслу значительной части экономических задач, относящихся к задачам линейного программирования (ЛП), компоненты решения должны выражаться в целых числах. Например, задачи, в которых переменные означают количество единиц неделимой продукции; задачи, в которых переменные принимают два значения – 0 или 1. В большинстве случаев сложность, вызываемая наличием условий в системе ограничений оптимизационной задачи, состоит в том, что нельзя дискретную задачу заменить ее непрерывным аналогом, и,

найдя соответствующее решение, округлить его результат до ближайших целых значений. При округлении оптимального плана x^* может получиться точка, не принадлежащая области допустимых планов задачи D. Может оказаться и так, что округление будет происходить до целых значений результата, план окажется допустим, целевая функция может вести себя так, что ее рост будет хуже, чем на оптимальном плане.

Общий вид модели целочисленного программирования (ЦЛП):

$$\begin{cases} \sum_{j=1}^{n} c_j \cdot x_j \to \text{extr}, \\ \sum_{j=1}^{n} a_{ij} \cdot x_j \le b_i,\ i = 1, 2, \ldots, m, \\ x_j \ge 0,\ j = 1, 2, \ldots, n, \\ x_k - \text{целое}, k = 1, 2, \ldots, p (p \le n). \end{cases} \qquad (2.1)$$

Ниже представлена постановка и формализация *задачи о рюкзаке*.

Турист, собирающийся в поход, может нести груз весом не более b кг. Этот груз может состоять из набора предметов n типов, каждый из которых характеризуется определенной полезностью c_j и весит a_j кг. Сколько предметов каждого вида нужно положить в рюкзак, чтобы его суммарная полезность была максимальной?

$$\sum_{j=1}^{n} c_j \cdot x_j = \max,$$

$$\sum_{j=1}^{n} a_j \cdot x_j \le b,$$

$$x_j \ge 0, x_j - \text{целое}, j = 1, \ldots, n,$$

где x_j – количество предметов j-го типа, укладываемых в рюкзак;

$c_j x_j$ – полезность предметов j-го типа.

Методы целочисленной оптимизации можно разделить на три основные группы:

- методы отсечений (метод Гомори);
- комбинаторные методы (методы ветвей и границ);
- приближенные методы.

2.2 Методы отсечений. Метод Гомори

Метод состоит в следующем. Временно отбросив условия целочисленности, отыскиваем оптимальный опорный план. Если окажется, что он не удовле-

творяет условию целочисленности, то формируем дополнительное ограничение, называемое правильным отсечением, которому заведомо удовлетворяет любое целочисленное и не удовлетворяет найденное оптимальное нецелочисленное решение. Правильное ограничение обладает следующими свойствами:

- оно должно быть линейным;
- должно отсекать найденный оптимальный нецелочисленный план;
- не должно отсекать ни одного целочисленного плана.

Далее задачу решают с учетом нового ограничения. После этого в случае необходимости добавляют еще одно ограничение и т. д.

Один из алгоритмов решения задачи целочисленного линейного программирования (ЦЛП), предложенный Гомори, основан на симплексном методе и использует достаточно простой способ построения правильного отсечения.

Пусть задача ЛП имеет конечный оптимум и на последнем шаге ее решения симплексным методом получены уравнения, выражающие основные (базисные) переменные $x_1, \ldots, x_j, \ldots, x_m$ через неосновные переменные $x_{m+1}, \ldots, x_{m+i}, \ldots, x_n$ оптимального решения:

$$\begin{cases} x_1 = \beta_1 - \alpha_{1m+1}x_{m+1} - \ldots - \alpha_{1n}x_n, \\ x_2 = \beta_2 - \alpha_{2m+1}x_{m+1} - \ldots - \alpha_{2n}x_n, \\ \ldots \\ x_m = \beta_m - \alpha_{mm+1}x_{m+1} - \ldots - \alpha_{mn}x_n. \end{cases}$$

Оптимальным решением задачи в таком случае является $x^* = (\beta_1, \beta_2, \ldots, \beta_m, 0, 0, \ldots, 0)$.

Пусть β_i – нецелая компонента. Можно доказать, что неравенство

$$\{\beta_i\} - \{\alpha_{im+1}\}x_{m+1} - \ldots - \{\alpha_{in}\}x_n \le 0, \qquad (2.2)$$

сформулированное по i-му уравнению системы, обладает всеми свойствами правильного отсечения.

Алгоритм метода Гомори:

1 Решить задачу симплекс-методом без учета целочисленности. Если все компоненты оптимального плана целые, то он является оптимальным и для задачи ЦЛП. Если задача ЛП неразрешима, то и задача ЦЛП неразрешима.

2 Если среди компонент оптимального решения есть нецелые, то выбрать компоненту с наибольшей целой частью и для нее сформулировать правильное отсечение.

3 Полученное неравенство (2.2) введением дополнительной неотрицательной целочисленной переменной преобразовать в равносильное уравнение

$$\{\beta_i\} - \{\alpha_{im+1}\}x_{m+1} - \ldots - \{\alpha_{in}\}x_n + x_{n+1} = 0$$

и включить его в систему ограничений задачи ЛП. Двойственный симплекс-метод позволяет отыскать оптимальный целочисленный план, добавляя дополнительное ограничение не к исходной системе, а к полученному оптимальному нецелочисленному решению.

4 Расширенную задачу решить симплекс-методом. Если найденный оптимальный план целочисленный, то задача ЦЛП решена, в противном случае вернуться к шагу 2.

Если задача разрешима в целых числах, то после некоторого числа шагов оптимальное целый план будет найден.

Для того чтобы иметь право после формирования правильного отсечения пользоваться двойственным симплекс-методом, исходную задачу необходимо привести к задаче на максимум. Это связано с тем, что двойственный симплекс-метод применим, если оценки в индексной строке положительны.

2.3 Метод ветвей и границ

Другим широко применяемым для решения задач ЦЛП является метод ветвей и границ. Термин «метод ветвей и границ» является собирательным и включает в себя целое семейство методов, объединяемых общими принципами. Реализация этого метода состоит в последовательном ветвлении исходного множества решений на дерево подмножеств с нахождением решений на всех подмножествах, пока не получим искомое.

Пусть стоит задача максимизации целевой функции $f(x)$, $x \in D$, D – конечное множество, $D = \{d_1, d_2, \ldots, d_r\}$. Алгоритм является итеративным, и на каждой итерации происходит работа с некоторым подмножеством множества D. Назовем это подмножество «текущим» и будем обозначать $D^{(k)}$, где k – индекс итерации. Перед началом первой итерации в качестве текущего выбирается все множество D, т. е. $D^{(1)} = D$ и для него некоторым способом вычисляется значение верхней оценки для целевой функции $\max\limits_{x \in D} f(x) \le \xi(D^{(1)})$ (*).

Стандартная итерация алгоритма состоит из следующих этапов:

1 Если можно указать план $x^{(k)} \in D^{(k)}$, для которого $f(x^{(k)}) = \xi(D^{(k)})$, то $x^{(k)} = x^*$ – решение задачи (*).

2 Если такой план не найден, то область определения $D^{(k)}$ некоторым образом разбивается на подмножества $D_1^{(k)}$, $D_2^{(k)}$, …, $D_{l_k}^{(k)}$, удовлетворяющие условиям

$$\bigcup_i D_i^{(k)} = D^{(k)},$$

$$D_i^{(k)} \cap D_j^{(k)} = 0, i \ne j.$$

Для каждого подмножества находятся оценки сверху (верхние границы) для целевой функции $\xi(D_1^{(k)})$, $\xi(D_2^{(k)})$, …, $\xi(D_{l_k}^{(k)})$, уточняющие ранее полученную оценку $\xi(D^{(k)})$, т. е. $\xi(D_i^{(k)}) \le \xi(D^{(k)})$, $i = 1, 2, \ldots, l_k$.

Возможно одно из двух:

2.1 Если существует такой план $x^{(k)}$, то $f(x^{(k)}) = \xi(D_i^{(k)}), i \in 1, 2, \ldots, l_k$, то этот план оптимальный.

2.2 Если такой план не найден, то выбирается одно из множеств $D_i^{(k)}$ (как правило, имеющее наибольшую оценку) $\xi(D_m^{(k)}) = \max \xi(D_i^{(k)}), i \in 1, 2, \ldots, l_k$.

Все имеющиеся к текущему моменту концевые подмножества, т. е. те подмножества, которые еще не подверглись процессу дробления, переобозначаются как $D_1^{(n+1)}$, …, $D_{l_{k+1}}^{(n+1)}$, после чего процесс повторяется.

Схема дробления множества D представлена на рисунке 2.1 в виде графа. Существуют и более сложные системы индексации подмножеств, при которых не требуется их переобозначение на каждом шаге.

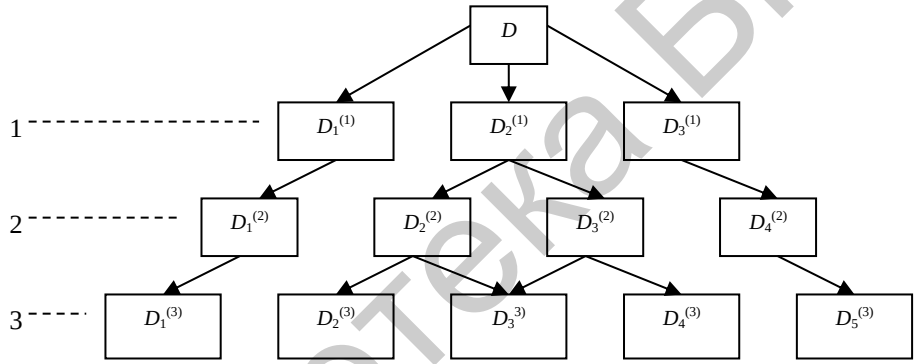


Рисунок 2.1 – Схема дробления исходного множества

Конкретная реализация метода ветвей и границ связана с правилами разбиения на подмножества (правила ветвления) и построения оценок значений целевых функций на данных подмножествах (определение границ).

Перед началом первой итерации ($k = 1$) принимаем $D = D^{(1)}$, после чего решается задача ЛП (без учета целочисленности), являющаяся непрерывным аналогом исходной задачи ЦЛП. Если найденный оптимальный план $x^{(1)}$ содержит только целочисленные компоненты, то он является и оптимальным планом для задачи ЦЛП: $x^{(1)} = x^*$. В противном случае значение $f(x^{(1)})$ становится оценкой (верхней границей) значения целевой функции на множестве $D^{(1)}$ и переходим к выполнению стандартной итерации алгоритма. Опишем входящие в нее этапы:

1 Выбирается некоторая нецелочисленная компонента плана $x_m^{(k)}$. Поскольку в оптимальном плане она должна быть целой, можно наложить ограничения $x_m \le [x_m^{(k)}]$ и $x_m \ge [x_m^{(k)}]+1$. $D^{(k)}$ разбивается на подмножества:

$$\tilde{D}_1^{(k)} = \{ x \mid x \in D^{(k)} \wedge (x_m \le [x_m^{(k)}] \},$$

$$\tilde{D}_2^{(k)} = \{ x \mid x \in D^{(k)} \wedge (x_m \ge [x_m^{(k)}]+1 \}.$$

На рисунке 2.2 приведена графическая интерпретация такого разбиения.

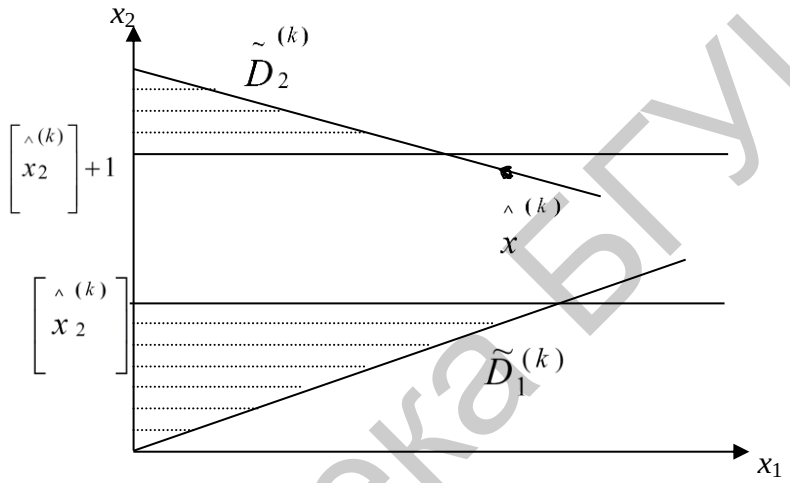


Рисунок 2.2 – Схема разбиения исходного множества на подмножества

2 Решают задачи ЛП $\max\limits_{x \in \tilde{D}_1^{(k)}} f(x)$ и $\max\limits_{x \in \tilde{D}_2^{(k)}} f(x)$. Соответствующие максимальные значения целевой функции принимают как ее нули на этих множествах:

$$\xi(\tilde{D}_1^{(k)}) = \max\limits_{x \in \tilde{D}_1^{(k)}} f(x) \text{ и } \xi(\tilde{D}_2^{(k)}) = \max\limits_{x \in \tilde{D}_2^{(k)}} f(x).$$

Если оптимальный план $\tilde{x}$ для одной из решенных задач удовлетворяет условию $f(\tilde{x}) = \max \xi(\overline{D}_i^{(k)})$ и содержит только целые компоненты, то найдено решение исходной задачи ЦЛП. В противном случае среди всех концевых подмножеств, полученных как на предыдущих $D_i^{(n)}$, так и на текущем $\left(\overline{D}_1^{(k)}, \overline{D}_2^{(k)}\right)$ этапе, выбирается область с наибольшей оценкой $\xi(D_{i*}^{(k)})$. Она становится текущим рассматриваемым подмножеством $D^{(k+1)}$. Далее производится перенумерация концевых множеств и вычислительный процесс итеративно повторяется.

2.4 Задача об оптимальном назначении и венгерский алгоритм

Пусть имеется n должностей и m претендентов на эти должности. Заданы «технологические» параметры или «полезность» i-го претендента на j-й должности. Как провести назначения претендентов на должности, чтобы добиться максимального суммарного эффекта? При решении задачи учитывать, что дробление должностей не допускается и каждый претендент может быть назначен только на одну должность.

$$\begin{aligned} &\sum_{j=1}^{n}\sum_{i=1}^{m} a_{ij} \cdot x_{ij} \to \max, \\ &\sum_{i=1}^{m} x_{ij} = 1,\ j = 1, \ldots, n, \\ &\sum_{j=1}^{n} x_{ij} = 1, i = 1, \ldots, m, \end{aligned} \qquad (2.3)$$

где $x_{ij} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$; X_{ij} – матрица назначений.

Для решения задачи об оптимальном назначении и ее модификаций используется венгерский метод.

Венгерский метод состоит из ряда операций, цель которых путем эквивалентных преобразований матрицы $A = \|a_{ij}\|$ привести ее к матрице D, содержащей n независимых нулей. Независимые нули матрицы – такая совокупность нулей, из которых никакие два нуля не стоят ни в одной строке, ни в одном столбце матрицы. Расположение независимых нулей соответствует оптимальному назначению.

Алгоритм состоит из предварительного этапа, конечного числа итераций, на каждой из которых число независимых нулей увеличивается на единицу.

Предварительный этап состоит из трех шагов:

1 В каждом столбце матрицы из максимального элемента вычитают все его элементы, в результате чего получаем матрицу A' с элементами $a_{ij}' = \max a_{ij} - a_{ij}$.

2 В каждой строке матрицы A' вычитаем минимальный элемент строки из всех ее элементов, в результате чего получаем матрицу $A'' = \|a_{ij}''\|$ с элементами $a_{ij}'' = a_{ij}' - \min a_{ij}'$.

3 Отмечаем произвольный нуль первого столбца A'' звездочкой (*). Просматриваем второй столбец и, если в нем имеется неотмеченный нуль в строке, не содержащий нуля со звездочкой, отмечаем его звездочкой. Осуществляем эту процедуру для всех столбцов матрицы, в результате чего получаем исходную совокупность независимых нулей.

Приведем описание произвольной итерации алгоритма:

1 В матрице *A*'' отмечаем крестиком (+) столбцы, содержащие нули со звездочкой. Элементы этих столбцов считаются выделенными. Если число таких столбцов равно *n*, задача решена. Если таких столбцов меньше *n*, переходим к шагу 2.

2 Проверяем, нет ли в матрице невыделенных нулей. Если такие нули есть, отмечаем любой из них значком штрих (') и переходим к шагу 3. Если таких нулей нет, переходим к шагу 6.

3 Проверяем число нулей, отмеченных звездочкой в строке, где находится нуль, отмеченный штрихом на предыдущем шаге. Если число этих нулей меньше 1, переходим к шагу 5, иначе – к шагу 4.

4 Строку, в которой стоит нуль, отмеченный штрихом, отмечаем знаком (+). Уничтожаем выделение над теми столбцами, в которых содержатся нули со звездочкой, стоящие в отмеченной строке. Переходим к шагу 2.

5 Строим в полученной матрице следующую последовательность элементов: исходный нуль, отмеченный штрихом, затем нуль, отмеченный звездочкой, лежащий в одном столбце с первым нулем (если такой имеется), затем нуль, отмеченный штрихом и лежащий в одной строке со вторым нулем и т. д. Последовательность обязательно закончится нулем, отмеченным штрихом, после которого в том же столбце нельзя найти нуль, отмеченный звездочкой. В полученной последовательности нулей убираем звездочки, штрихи заменяем на звездочки, в результате чего число независимых нулей увеличивается на единицу. Убираем все штрихи над элементами матрицы, а также значения, которыми отмечены ее строки и столбцы. Переходим к шагу 1 (начало следующей итерации).

6 Среди невыделенных элементов матрицы находим минимальный по величине. Вычитаем его из элементов невыделенных строк и прибавляем к элементам выделенных столбцов. Получаем новую матрицу, в которой есть невыделенные нули и переходим к шагу 2.

Пример. Воспользуемся предложенным алгоритмом при решении задачи об оптимальном назначении с матрицей эффективности.

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

На предварительном этапе приведем матрицу *A* к матрице *A*'.

$$A' = \begin{pmatrix} 0 & 2 & 4 & 0 & 0 \\ 4 & 0 & 3 & 5 & 1 \\ 3 & 1 & 0 & 4 & 2 \\ 7 & 0 & 7 & 7 & 4 \\ 5 & 2 & 2 & 0 & 5 \end{pmatrix}.$$

На шаге 1 выясняем, что число независимых нулей $k = 4 < 5$. Выделим столбцы, содержащие независимые нули (нули, отмеченные звездочкой) крестиками. На шаге 2 выделяем невыделенный нуль штрихом и переходим к шагу 3. Поскольку число нулей со звездочкой, лежащих в первой строке, равно единице, переходим к шагу 4, на котором выделяем крестиком первую строку и снимаем выделение с первого столбца, после чего переходим к шагу 2. Поскольку невыделенные нули отсутствуют, переходим к шагу 6, в результате выполнения которого получаем матрицу

$$\begin{pmatrix} + & + & + & + & \\ 0^* & 3 & 5 & 1 & 0' \\ 3 & 0^* & 3 & 5 & 0' \\ 2 & 1 & 0^* & 4 & 1 \\ 6 & 0' & 7 & 7 & 3 \\ 4 & 2 & 2 & 0^* & 4 \end{pmatrix} \begin{matrix} + \\ + \\ . \\ \\ \\ \end{matrix}$$

В полученной матрице находим невыделенный нуль во второй строке, выделяем эту строку, снимаем выделение со второго столбца. Вновь ищем среди невыделенных элементов нули. Такой нуль находится в четвертой строке. Выделив его, переходим к шагу 5, на котором число независимых нулей увеличивается на единицу и матрица принимает вид

$$\begin{pmatrix} 0^* & 3 & 5 & 1 & 0 \\ 3 & 0 & 3 & 5 & 0^* \\ 2 & 1 & 0^* & 4 & 1 \\ 6 & 0^* & 7 & 7 & 3 \\ 4 & 2 & 2 & 0^* & 4 \end{pmatrix}.$$

Начиная следующую итерацию, убеждаемся, что число независимых нулей $k = n = 5$, т. е. задача решена. Оптимальное решение задачи определяется расположением независимых нулей в матрице, а максимальный суммарный эффект от назначения равен $F_{\text{опт}} = 9 + 6 + 7 + 9 + 8 = 39$.

Заметим, что помимо традиционной экономической интерпретации (назначение работников на рабочие места, распределение технологических операций по станкам) задача о назначении находит применение в конструкторском проектировании цифровой аппаратуры при размещении конструктивно-функциональных узлов в монтажном пространстве.

3 Модели динамического программирования

3.1 Общая постановка задачи динамического программирования

Динамическое программирование (ДП) – метод оптимизации, приспособленный к операциям, в которых процесс принятия решения может быть разбит на этапы (шаги). Такие операции называются многошаговыми.

Если модели линейного программирования можно использовать в экономике для принятия крупномасштабных решений, то модели динамического программирования применяются при решении задач значительно меньшего масштаба. Например, при разработке правил управления запасами, при разработке принципов календарного планирования производства и выравнивания занятости в условиях колеблющегося спроса на продукцию, при составлении календарных планов текущего и капитального ремонта оборудования и его замены, при разработке долгосрочных правил замены выбывающих из эксплуатации основных фондов и т. п.

Приведем общую постановку задачи ДП. Рассматривается управляемый процесс, например, процесс распределения средств между предприятиями, замены оборудования, использования ресурсов в течение ряда лет, пополнения запасов и т. п. В результате управления система (объект управления) S переводится из начального состояния s_0 в состояние €. Предположим, что управление можно разбить на n шагов, т. е. решения принимаются последовательно на каждом шаге, а управление, переводящее систему S из s_0 в €, представляет собой совокупность n пошаговых управлений.

Обозначим через X_k управление на k-м шаге (k = 1, 2, …, n). Переменные X_k удовлетворяют некоторым ограничениям и в этом смысле называются допустимыми.

Пусть X (X_1, X_2, …, X_n) – управление, переводящее систему S из состояния s_0 в состояние €. Обозначим через s_k состояние системы после k-го шага управления. Получаем последовательность состояний s_0, s_1, …, s_{n-1}, s_n = €, которую изобразим кружками (рисунок 3.1).

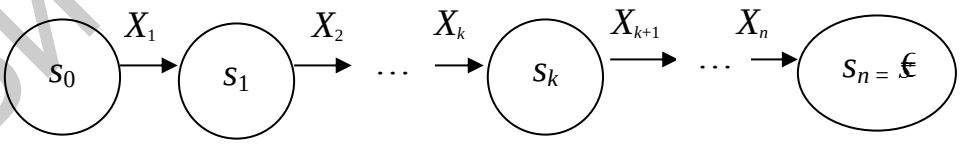


Рисунок 3.1 – Последовательность состояний

Показатель эффективности данной управляемой операции – целевая функция – зависит от начального состояния и управления:

$$Z = F(s_0, X). \qquad (3.1)$$

Сделаем несколько предположений.

1 Состояние s_k системы в конце k-го шага зависит только от предшествующего состояния s_{k-1} и управления на k-м шаге X_k (и не зависит от предшествующих состояний и управлений). Это свойство называется «отсутствием последействия». Это положение записывается в виде уравнений состояний:

$$s_k = \varphi_k(s_{k-1}, X_k),\ k = 1, 2, \ldots, n\ . \qquad (3.2)$$

2 Целевая функция является аддитивной от показателей эффективности Z_k каждого шага:

$$Z_k = f_k(s_{k-1}, X_k),\ \ k = 1, 2, \ldots, n, \qquad (3.3)$$

$$Z = \sum_{k=1}^{n} f_k(s_{k-1}, X_k)\ . \qquad (3.4)$$

Задача пошаговой оптимизации (задача ДП) формулируется так: определить такое допустимое управление X, переводящее систему S из состояния s_0 в состояние €, при котором целевая функция Z принимает наибольшее (наименьшее) значение.

Выделим особенности модели ДП:

1 Задача оптимизации интерпретируется как n-шаговый процесс управления.

2 Целевая функция равна сумме целевых функций каждого шага.

3 Выбор управления на k-м шаге зависит от состояния системы к этому шагу и не влияет на предшествующие шаги (отсутствие обратной связи).

4 Состояние s_k после k-го шага управления зависит только от предшествующего состояния s_{k-1} и управления X_k (отсутствие последействия).

5 На каждом шаге управления X_k зависит от конечного числа управляющих переменных, а состояние s_k – от конечного числа параметров.

3.2 Принцип оптимальности и уравнения Беллмана

Вычислительная схема ДП связана с принципом оптимальности Беллмана и использует рекуррентные соотношения.

Каково бы ни было состояние s системы в результате какого-либо числа шагов, на ближайшем шаге нужно выбирать управление так, чтобы оно в совокупности с оптимальным управлением на всех последующих шагах приводило к оптимальному выигрышу на всех оставшихся шагах, включая данный.

На каждом шаге любого состояния системы s_{k-1} решение X_k нужно выбирать «с оглядкой», т. к. как этот выбор влияет на последующее состояние s_k и

дальнейший процесс управления, зависящий от s_k. Это следует из принципа оптимальности.

Последний шаг планируется локально-оптимально, исходя только из соображений этого шага. Рассмотрим n-й шаг: s_{n-1} – состояние системы к началу n-го шага; $s_n = €$ – конечное состояние; X_n – управление на n-м шаге, $f_n(s_{n-1},X_n)$ – целевая функция (выигрыш) n-го шага; $Z_n^*(s_{n-1})$ – показатель эффективности n-го шага при условии, что к началу последнего шага система S была в состоянии s_{n-1}, а на последнем шаге управления было оптимальным:

$$Z_n^*(s_{n-1}) = \max_{\{X_n\}} f_n(s_{n-1}, X_n). \tag{3.5}$$

$Z_n^*(s_{n-1})$ называется условным максимумом целевой функции на n-м шаге. Решение $X_n^*(s_{n-1})$, при котором достигается $Z_n^*(s_{n-1})$, зависит от s_{n-1} и называется условным оптимальным управлением на n-м шаге.

Решив одномерную задачу локальной оптимизации по управлению (3.5), найдем для всех возможных состояний s_{n-1} две функции $Z_n^*(s_{n-1})$ и $X_n^*(s_{n-1})$.

Рассмотрим двухшаговую задачу.

Для любых состояний s_{n-2}, управлений X_{n-1} и оптимальном управлении на n-м шаге значение целевой функции на двух последних шагах равно

$$f_{n-1}(s_{n-2}, X_{n-1}) + Z_n^*(s_{n-1}). \tag{3.6}$$

Согласно принципу оптимальности для любых s_{n-2} решение X_{n-1} нужно выбирать так, чтобы оно вместе с оптимальным решением на последнем n-м шаге приводило бы к максимуму целевой функции на двух последних шагах:

$$\begin{gathered} Z_{n-1}^*(s_{n-2}) = \max_{\{X_{n-1}\}} \{ f_{n-1}(s_{n-2}, X_{n-1}) + Z_n^*(s_{n-1}) \}, \\ s_{n-1} = \varphi_{n-1}(s_{n-2}, X_{n-1}). \end{gathered} \tag{3.7}$$

В результате максимизации только по одной переменной X_{n-1} вновь получаем две функции:

$$Z_{n-1}^*(s_{n-2}) \text{ и } X_{n-1}^*(s_{n-2}).$$

Далее рассматривается трехшаговая задача: к двум последним шагам присоединяется $(n-2)$-й шаг и т. д.

Целевая функция на $(n-k)$ последних шагах (рисунок 3.2) при произвольном управлении X_k на k-м шаге и оптимальном управлением на последующих $(n-k)$ шагах равна $f_k(s_{k-1}, X_k) + Z_{k+1}^*(s_k)$.

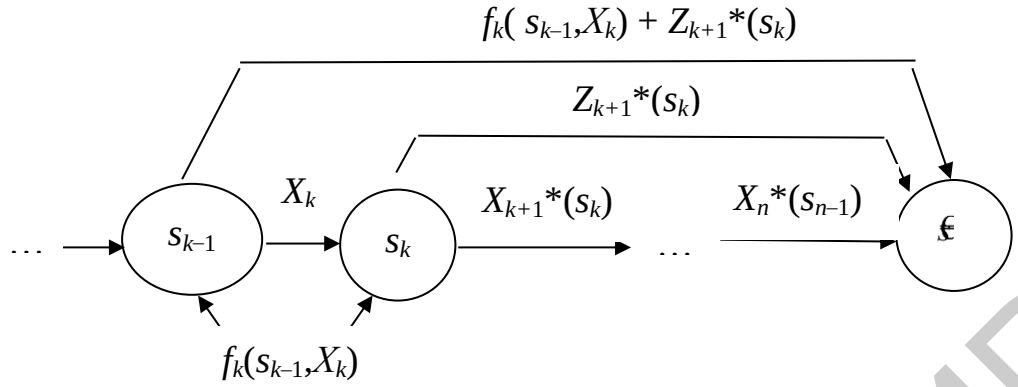


Рисунок 3.2 – Целевая функция на $(n - k)$ последних шагах

Согласно принципу оптимальности, X_k выбирается из условия максимума этой суммы, т. е.

$$Z_k^*(s_{k-1}) = \max_{\{X_k\}} \{ f_k(s_{k-1}, X_k) + Z_{k+1}^*(s_k) \}, \quad k = n-1, n-2, ..., 2, 1. \tag{3.8}$$

В результате решения (3.8) получим условно оптимальное управление на k-м шаге $X_k^*(s_{k-1})$. Уравнения (3.8) называются уравнениями Беллмана. Это рекуррентные соотношения, позволяющие найти предыдущее значение функции, зная последующие. Процесс решения уравнений Беллмана называется условной оптимизацией (по «обратной схеме»). В результате условной оптимизации получаются две последовательности:

1) $Z_n^*(s_{n-1})$, $Z_{n-1}^*(s_{n-2})$, …, $Z_2^*(s_1)$, $Z_1^*(s_0)$ – условные максимумы целевой функции на последнем, на двух последних, …, на n шагах;

2) $X_n^*(s_{n-1})$, $X_{n-1}^*(s_{n-2})$, …, $X_2^*(s_1)$, $X_1^*(s_0)$ – условные оптимальные управления на n-м, $(n - 1)$-м, …, 1-м шагах.

Используя эти последовательности, можно найти решения задачи ДП при данных n и s_0: $Z_{\max} = Z_1^*(s_0)$.

При фиксированном s_0 получим $X_1^* = X_1^*(s_0)$. Затем из уравнения (3.2) находим $s_1^* = \varphi_1(s_0, X_1^*)$, подставляем это выражение в последовательность условных оптимальных уравнений: $X_2^* = X_2^*(s_1)$ и т. д. по цепочке:

$$\begin{gathered} X_1^* = X_1^*(s_0) \to s_1^* = \varphi_1(s_0, X_1^*) \Rightarrow X_2^* = X_2^*(s_1^*) \to \\ s_2^* = \varphi_2(s_1^*, X_2^*) \Rightarrow X_3^* = X_3^*(s_2^*) \to \ldots \to \\ s_{n-1}^* = \varphi_{n-1}(s_{n-2}^*, X_{n-1}^*) \Rightarrow X_n^* = X_n^*(s_{n-1}^*). \end{gathered} \tag{3.9}$$

Получаем безусловное оптимальное решение задачи ДП $X^* = (X_1^*, X_2^*, \ldots, X_n^*)$.

3.3 Задача о замене оборудования

Задача о замене оборудования заключается в определении оптимальных сроков замены старого оборудования. Критерием оптимальности являются, как правило, либо прибыль от эксплуатации оборудования (задача максимизации), либо суммарные затраты на эксплуатацию в течение планируемого периода (задача оптимизации).

Рассматривается плановый период из n лет, в начале которого имеется одна машина фиксированного возраста. В процессе работы машина приносит доходы, требует эксплуатационных затрат и имеет остаточную стоимость.

Введем следующие обозначения:

1) t – возраст машины ($t = 0, 1, 2, \ldots, n$);

2) $r(t)$ – стоимость продукции, производимой за год, на момент возраста t;

3) $u(t)$ – годовые эксплуатационные затраты на машину возраста t;

4) $s(t)$ – остаточная стоимость машины возраста t;

5) p – цена новой машины.

В любой год машину можно либо сохранить, либо продать по остаточной стоимости и купить вместо нее новую. Требуется найти оптимальную по суммарной прибыли за плановый период политику замены и сохранения машины.

Пусть $k = 0, 1, 2, \ldots, n$ – количество лет до окончания планового периода. Если за k лет до окончания планового периода имеем машину возраста t, то через год, т. е. за $(k – 1)$ лет до окончания планового периода будем иметь:

а) при сохранении машины – машину возраста $t + 1$;

б) при замене (покупается новая машина) – машину возраста $t = 1$.

$f_n^*(t)$ – суммарная прибыль за последние k лет планового периода из n лет при условии, что в начале этого периода имеется машина возраста t и в эти последние k лет используется оптимальная политика «замены».

$f_1^*(t)$ – прибыль за последний год планового периода.

$f_2^*(t)$ – прибыль за последние два года планового периода.

…

$f_n^*(t)$ – прибыль за весь плановый период.

Если мы сумеем вычислить $f_n(t)$ при $t = t_0$ и найти политику замены, обеспечивающую доход $f_n(t_0)$, то это и будет решением задачи.

Развернем процесс планирования с конца, справа налево, в соответствии с ростом параметра n.

Рассмотрим случай, когда до конца планового периода остается один год. В зависимости от наших (пока не определенных) решений в предыдущие годы планового периода мы можем прийти к началу последнего года с машиной произвольного возраста (от машины возраста $t = 1$, купленной в предыдущем году, до очень старой, купленной в начале планового периода).

Пусть к началу последнего года имеем машину возраста t.

Если мы сохраняем машину, то прибыль рассчитывается как доход за вычетом эксплуатационных затрат $(r(t) – u(t))$.

Если мы машину меняем, то прибыль рассчитывается по формуле

$$s(t) - p + r(0) - u(0), \tag{3.10}$$

где $s(t)$ – остаточная стоимость старой машины;

p – цена новой машины;

$r(0)$ – доход от новой машины;

$u(0)$ – затраты на содержание новой машины.

Очевидно, что решение о замене (З) машины следует принять, если

$$s(t) - p + r(0) - u(0) > r(t) - u(t), \tag{3.11}$$

и о сохранении (С) машины, если

$$s(t) - p + r(0) - u(0) \le r(t) - u(t) \tag{3.12}$$

(при равенстве дохода сохраняем старую машину, т. к. мы к ней привыкли, знаем все ее плюсы и минусы). Значит,

$$f_1^*(t) = \max\begin{cases} r(t) - u(t) \to \text{С}, \\ s(t) - p + r(0) - u(0) \to \text{З}. \end{cases} \tag{3.13}$$

Для любого значения t можно найти $f_1(t)$ и решение (З или С), при котором достигается это значение прибыли.

Рассмотрим ситуацию, когда до конца планового периода остается k лет. Этот период можно разбить на две части.

Пусть после первого года в этом периоде мы знаем оптимальную политику З и С. Поэтому разумно процесс нахождения оптимальной политики разделить на два этапа: рассмотреть все возможные решения в «первом году» для машины возраста t и для каждого получившегося состояния системы найти оптимальную политику в оставшейся чисти из $(n - 1)$ последних лет.

Все возможные решения в «первом году» исчерпываются двумя:

а) сохранив машину возраста t, мы получим доход $r(t) - u(t)$, в результате чего в конце года будем иметь машину возраста $(t + 1)$ (система придет в состояние $(t + 1)$) и оптимальная политика в последние $(n - 1)$ лет обеспечит доход $f_{n-1}^*(t + 1)$;

б) замена машины приведет к затратам на покупку новой машины $s(t) - p$, доходу от появившейся новой машины за «первый год» $r(0) - u(0)$, причем к концу года новая машина «постареет» и будет иметь возраст 1, т. е. (система перейдет в состояние 1) и оптимальная политика в последующие $(n - 1)$ лет обеспечит доход $f_{n-1}^*(1)$.

Предположим, что мы имеем машину в возрасте t.

Если мы сохраняем машину, то суммарная прибыль за k последних лет будет вычисляться как

$$r(t) - u(t) + f_{k-1}^*(t+1),$$

где $r(t) - u(t)$ – доход за «первые годы»;

$f_{k-1}^*(t+1)$ – прибыль за оставшиеся $(k-1)$ лет.

Если мы заменяем машину в этом году, то получаем доход за первый год от новой машины, равный

$$s(t) - p + r(0) - u(0) + f_{k-1}^*(1).$$

Тогда

$$f_k^*(t) = \max \begin{cases} r(t) - u(t) + f_{k-1}^*(t+1) \to \text{С}, \\ s(t) - p + r(0) - u(0) + f_{k-1}^*(1) \to \text{З}. \end{cases} \quad (3.14)$$

Соотношения (3.14), устанавливающие связь между выражениями для f_{n-1}, называются рекуррентными соотношениями Беллмана. Эти соотношения позволяют развернуть процесс нахождения оптимальной политики с конца, последовательно находя $f_1^*, f_2^*, \ldots, f_k^*, \ldots, f_n^*$ для различных значений t.

Пример. Ограничимся машинами возраста не более 10 лет, т. е. плановый период равен $N = 10$ лет. Пусть ликвидационная стоимость машины не зависит от возраста машины и равна нулю, т. е. $s(t) = 0$; цена новой машины со временем не меняется и равна $p = 10$; длина планового периода $N = 10$.

Функции $r(t)$, $u(t)$ и $r(t) - u(t)$ задаются таблицей 3.1 (из таблицы видно, что машина старше 10 лет невыгодна).

Таблица 3.1 – Стоимость продукции и годовые эксплуатационные затраты для машины возраста t лет

t	0	1	2	3	4	5	6	7	8	9	10
$r(t)$	20	20	20	19	19	18	18	17	17	16	15
$u(t)$	10	11	12	12	13	13	14	14	15	15	15
$r(t) - u(t)$	10	9	8	7	6	5	4	3	2	1	0

Используя (3.14) и подставляя данные $s(t) = 0$, $p = -10$, $r(0) - u(0) = 10$, получим, что

$$s(t) - p + r(0) - u(0) = 0.$$

Заметим, что с ростом t выражения $r(t) - u(t) = f_1(t)$ убывают, поскольку, чем «старее» машина, тем меньший доход она дает в год.

Поэтому если при $t = 6$

$$[9 = f_1(1)] > [r(6) - u(6) + f_1(7) = 7],$$

то при $t > 6$ и подавно $f_1(1) > r(t) - u(t) + f_1(t+1)$.

Значит, при $t \geq 6$

$$f_2^*(t) = \max\begin{cases} r(t) - u(t) + f_1^*(t+1) \to \text{C} \\ f_1^*(1) \to 3 \end{cases} = f_1^*(1) = 9 \geq 3.$$

В таблице 3.2 приведены расчеты по рекуррентным формулам.

Таблица 3.2 – Расчетные значения прибыли за плановый период

	0	1	2	3	4	5	6	7	8	9	10
$f_1(t)$	10	9	8	7	6	5	4	3	2	1	0
$f_2(t)$	19	17	15	13	11	9	9	9	9	9	9
$f_3(t)$	27	24	21	18	17	17	17	17	17	17	17
$f_4(t)$	34	30	26	24	24	24	24	24	24	24	24
$f_5(t)$	40	35	32	31	30	30	30	30	30	30	30
$f_6(t)$	45	41	39	37	36	35	35	35	35	35	35
$f_7(t)$	51	48	45	43	41	41	41	41	41	41	41
$f_8(t)$	58	54	51	48	48	48	48	48	48	48	98
$f_9(t)$	64	60	56	55	54	54	54	54	54	54	54
$f_{10}(t)$	70	65	63	61	60	60	60	60	60	60	60

Элемент таблицы $f_5(4) = 30$ означает, если у нас есть машина возраста 4 года и нам осталось хозяйствовать 5 лет, то максимальный доход, который можно получить за эту пятилетку, равен 30, причем для его получения надо в первом году этой пятилетки сохранить имеющуюся машину.

Ниже (рисунок 3.3) представлена траектория политики замены и сохранения машины в ситуации, когда в начале планового периода мы имеем машину возраста 4 лет, а до конца планового периода остается 5 лет.

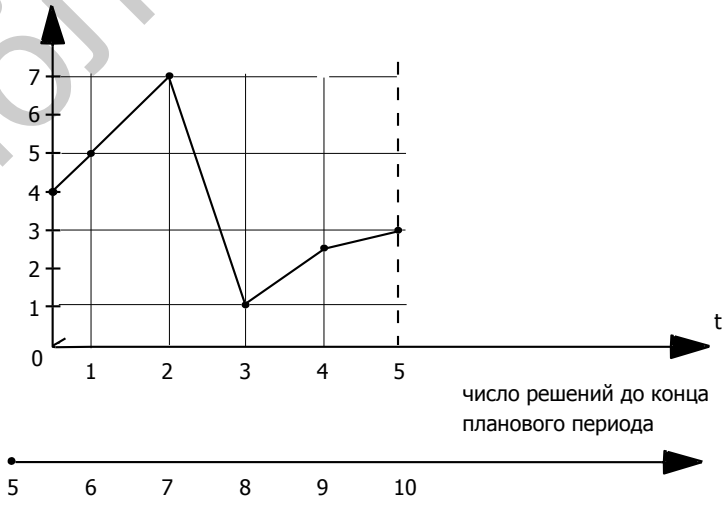


Рисунок 3.3 – Траектория политики замены и сохранения машины возраста 4 лет за 5 лет до конца планового периода

Данная траектория составлена на основе табличного решения задачи. Линия, направленная вверх, соответствует политике сохранения машины, линия, направленная вниз, – политике замены.

На рисунке 3.4 представлена траектория политики замены и сохранения машины, когда в начале планового периода мы имеем машину возраста 6 лет и до конца планового периода остается 10 лет.

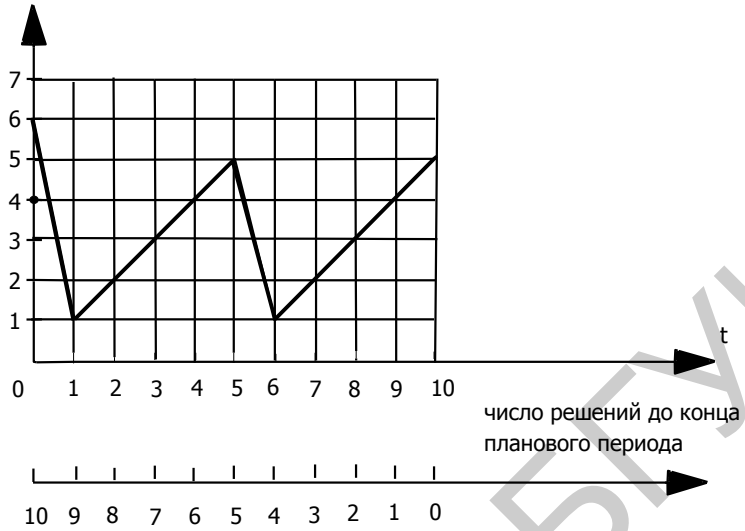


Рисунок 3.4 – Траектория политики замены и сохранения машины возраста 6 лет за 10 лет до конца планового периода

3.4 Динамические оптимизационные модели

Рассмотрим особенности динамических процессов управления запасами. Важно отметить, что внедрение модификаций рассматриваемой модели рядом промышленных фирм дало заметный экономический эффект.

3.4.1 Простейшая задача управления запасами

Рассмотрим следующий пример [2]. Фирма «Надежный поставщик» должна разработать календарную программу выпуска некоторого вида изделий на плановый период, состоящий из N отрезков. Предполагается, что для каждого из этих отрезков имеется точный прогноз спроса на выпускаемую продукцию.

Время изготовления партии изделия настолько мало, что им можно пренебречь, соответственно продукция, изготавливаемая в течение отрезка t, может быть использована для полного или частичного покрытия спроса в течение этого отрезка. Для разных отрезков спрос неодинаков, кроме того, на экономические показатели производства влияют размеры изготавливаемых партий. Поэтому фирме нередко бывает выгодно изготовлять в течение некоторого месяца продукцию в объеме, превышающем спрос в переделах этого отрезка, и хра-

нить излишки, используя их для удовлетворения последующего спроса. Вместе с тем хранение возникающих при этом запасов связано с определенными затратами. В зависимости от обстоятельств затраты обусловлены такими факторами, как проценты на капитал, взятый взаймы для создания запасов, арендная плата за складские помещения, страховые взносы и расходы по содержанию запасов. Эти затраты необходимо учитывать и при установлении программы выпуска.

Цель фирмы «Надежный поставщик» – разработать такую программу, при которой общая сумма затрат на производство и на содержание запасов минимизируется при условии полного и своевременного удовлетворения спроса на продукцию. Анализ этой задачи начнем с преобразования качественного ее описания в математическую модель.

Введем следующие переменные:

1) x_t – выпуск продукции в течение отрезка t;

2) i_t – уровень запасов на конец отрезка t.

Спрос на продукцию для отрезка t обозначим D_t; предполагается, что величины D_t для всех t отображаются неотрицательными целыми числами и что к началу планового периода все D_t известны.

Предположим также, что для каждого отрезка t затраты зависят от выпуска продукции x_t, уровня запасов i_t на конец отрезка t и, кроме того, возможно, от значения t. Обозначим затраты на отрезке t как $C_t(x_t;i_t)$. Тогда целевую функцию можно записать в следующем виде:

$$\min \sum_{t=1}^{N} C_t(x_t; i_t). \qquad (3.15)$$

На значения переменных x_t и i_t наложено несколько ограничений. Во-первых, предполагается целочисленность объемов выпуска:

$$x_t = 0, 1, 2, 3\ldots (t = 1, 2, \ldots, N). \qquad (3.16)$$

Во-вторых, предполагается, что для администрации фирмы желателен нулевой уровень запасов на конец отрезка N:

$$i_N = 0 \text{ (конечный запас равен нулю)}. \qquad (3.17)$$

В-третьих, ставится условие полного и своевременного удовлетворения спроса в пределах каждого месяца. Выполнение этого условия можно обеспечить, введя два ограничения. Первое из них назовем «балансовым», поскольку в нем утверждается, что

уровень запасов на конец отрезка t = (уровень запасов на начало отрезка t) + + (выпуск продукции на отрезке t) – (спрос на отрезке t).

Если воспользоваться принятыми условными обозначениями, то это ограничение можно будет записать:

$$i_t = i_{t-1} + x_t - D_t$$

или в более удобном для нас виде:

$$D_t = i_{t-1} + x_t - i_t \quad (t = 1, 2, \ldots, N), \tag{3.18}$$

где i_t – заданный уровень запасов на начало планового периода.

Согласно второму вводимому ограничению, обеспечивающему своевременное выполнение фирмой «Надежный поставщик» своих обязательств, уровень запасов на начало каждого отрезка и объем выпуска продукции в течение этого отрезка должны быть достаточно велики для того, чтобы уровень запасов на конец отрезка был бы неотрицательным. На самом же деле требуется не только неотрицательность, но и целочисленность уровней запасов (правда, если предположить целочисленность объемов спроса и выпуска продукции, то предположение о целочисленности уровней запасов не создает дополнительных трудностей). Таким образом, требуется, чтобы

$$i_t = 0, 1, 2, 3\ldots \ (t = 1, 2, \ldots, N - 1). \tag{3.19}$$

Отметим, что ограничение (3.18) является линейным. Если бы все величины затрат $C_t(x_t; i_t)$ линейно зависели от значений переменных, то, как показано далее, полученная модель была бы эквивалентна сетевой модели и для нее легко удалось бы найти решение с помощью соответствующих методов. Однако в большинстве практических случаев применения производственных моделей функция затрат нелинейна. Так, для выпуска партии изделий могут потребоваться дорогостоящие подготовительные операции (переналадка), из-за которых затраты на производство первой единицы партии изделий превышают дополнительные затраты на производство остальных единиц. В тех же случаях, когда объем производства в течение некоторого периода превышает нормальную мощность производственного участка, дополнительные затраты на единицу изделия могут возрастать из-за использования сверхурочных работ.

Для того чтобы решить задачу при нелинейности каждой из величин $C_t(x_t; i_t)$, сформулируем ее в терминах динамического программирования.

Пусть $N = 4$. Составим балансовые уравнения (3.18) для $t = 1, 2, 3, 4$. Матрица этой системы ограничений представлена в таблице 3.3.

Таблица 3.3 – Матрица ограничений

		x_1	i_1	x_2	i_2	x_3	i_3	x_4	
Периоды	1	1	–1						$= D_1 – i_0$
	2		1	1	–1				$= D_2$
	3				1	1	–1		$= D_3$
	4						1	1	$= D_4$

Построим пятое уравнение, просуммировав четыре уравнения, а затем составим систему из пяти уравнений, содержащую наряду с пятым четыре исходных уравнения, умноженных на минус 1. Построенная система адекватна сети, изображенной на рисунке 3.5.

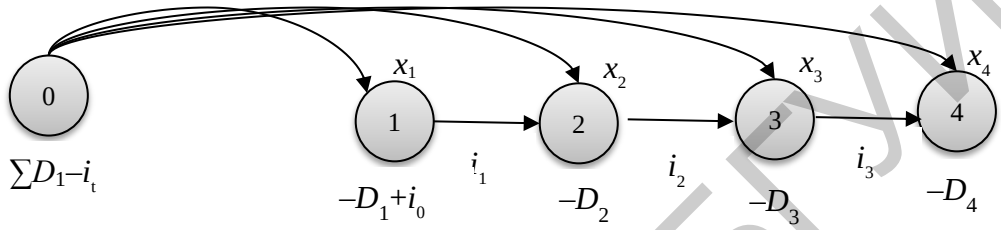


Рисунок 3.5 – Сеть, отображающая производство продукции и движение запасов

Согласно задаче о дилижансах вычислительный процесс строится от конечного состояния (сделаны все шаги многошагового процесса) к исходному. Такой же подход будет использован и в данной задаче. Здесь конечным состоянием будет начало последнего отрезка планового периода, а исходным – начальный момент первого отрезка (впереди еще N отрезков).

При составлении математической модели удобно использовать систему индексов, при которой подстрочный индекс «1» соответствует конечному, а «N» – начальному состоянию. Применим следующие обозначения:

1) d_n – спрос на продукцию на отрезке n, отстоящем от конца планового периода на n отрезков (включая рассматриваемый);

2) $c_n(x, j)$ – затраты на отрезке n, связанные с выпуском x единиц продукции и с содержанием запасов, уровень которых на конец отрезка равен j единиц.

В этой системе обозначений $d_1 \equiv D_N$ и $d_N \equiv D_1$, а $c_1(x, j) \equiv C_N(x, j)$.

Пусть $N = 4$, а плановый период начинается с января. Тогда D_1 есть январский спрос, D_4 – апрельский. В модели же используется «обратная система индексов»: январский спрос обозначен d_4, апрельский – d_1.

Что же определяет состояние системы в начале любого отрезка? Можно считать, что уровень запасов на начало отрезка. Для принятия текущего решения об объеме выпуска не нужно знать, каким образом достигнут начальный уровень. Учитывая это обстоятельство, введем следующие обозначения:

1) $f_n(i)$ – стоимость, отвечающая стратегии минимальных затрат на n оставшихся отрезков при начальном уровне запасов i;

2) $x_n(i)$ – выпуск, обеспечивающий достижение $f_n(i)$.

Согласно (3.17) уровень запасов на конец планового периода равен нулю, поэтому можно записать, что

$$f_0(0) = 0, \quad n = 0. \tag{3.20}$$

Затем перейдем к $n = 1$. Начальный уровень запасов i может определяться любым неотрицательным целым числом, не большим, чем d_1, вне зависимости от значения i для полного удовлетворения потребности в пределах последнего отрезка объем выпуска должен быть равен $(d_1 - i)$. Следовательно,

$$f_1(i) = c_1(d_1 - i, 0), \quad i = 0, 1, \ldots, d_1. \tag{3.21}$$

Перейдем к $n = 2$. Заметим, что если начальный уровень запасов равен i, а объем выпуска x, то общие затраты для двух месяцев составляют

$$c_2(x, i + x - d_2) + f_1(i + x - d_2),$$

причем предполагается, что выбранная стратегия для $n = 1$ была оптимальной. Заметим, что величина $(i + x - d_2)$ есть попросту уровень запасов на *конец* отрезка 2. Величина i может принимать любые неотрицательные целочисленные значения, не превышающие $(d_1 + d_2)$. При заданном i целочисленное значение x должно быть не меньше, чем $(d_2 - i)$, что обеспечивает полное удовлетворение потребности на отрезке 2, но не больше, чем $(d_1 + d_2 - i)$, т. к. конечный запас равен нулю. Оптимальному объему выпуска соответствует такое значение x, при котором минимизируется указанная выше сумма. Выполненный анализ ситуации для $n = 2$ можно выразить следующим общим выражением:

$$f_2(i) = \min_2[c_2(x, \ i + x - d_2) + f_1(i + x - d_2)],$$

где $i = 0, 1, \ldots, d_1 + d_2$, причем для отыскания минимума перебираются все неотрицательные целые значения x, заключенные в пределах

$$d_2 - i \leq x \leq d_1 + d_2 - i\,.$$

Значения $f_3(i)$ можно вычислить, если известны значения $f_2(i)$. В конце концов в данной задаче можно вычислить $f_N(i_0)$, где i_0 – уровень запасов на начало планового периода. Общее рекуррентное соотношение записывается в следующем виде:

$$f_n(i) = \min_x[c_n(x, \ i + x - d_n) + f_{n-1}(i + x - d_n)], \ n = 1, 2, \ldots N, \tag{3.22}$$

где $i = 0, 1, \ldots, d_1 + \ldots + d_n$, причем для отыскания минимума перебираются все неотрицательные целые значения x, заключенные в пределах

$$d_n - i \leq x \leq d_1 + d_2 + \ldots + d_n - i\,.$$

Заметим, что, поскольку начальный уровень запасов i рассматривается как переменная величина, полностью характеризующая состояние системы, единственной независимой управляющей переменной в рекуррентном соотношении (3.22) является x, т. к. уровень запасов на конец отрезка равен $(i + x - d_n)$. Заметим также, что, поскольку $f_0(0)$ и $f_1(i)$ без труда вычисляются по формулам (3.20) и (3.21), можно непосредственно и поочередно вычислить значения $f_2(0), f_2(1), \ldots, f_2(d_1)$, а затем $f_3(0), f_3(1), \ldots, f_3(d_1 + d_2)$. Последовательно переходя ко все большим значениям n, мы дойдем до вычисления $f_{N-1}(0), f_{N-1}(1), \ldots, f_{N-1}(d_1 + d_2 + \ldots + d_{N-1})$ и, наконец, до $f_N(i_0)$.

Для отыскания оптимальной производственной программы определим, какой объем выпуска $x_N(i_0)$ позволяет достичь полученного значения $f_N(i_0)$; соответствующее решение о выпуске является оптимальным решением для начального отрезка планового периода. Уровень запасов на начало следующего отрезка равен $i_0 + x_N(i_0) - d_N$; найдем объем выпуска, позволяющий достичь полученного нами значения $f_{N-1}[i_0 + x_N(i_0) - d_N]$, и т. д.

3.4.2 Модель распределения усилий

Рассмотрим нижеследующий гипотетический, но тем не менее поучительный пример так называемой *модели распределения усилий* [2]. Владелец торговой фирмы «Свежие продукты» должен распределить имеющийся у него недельный запас в количестве *N* яиц по *s* магазинам. Из опыта известно, что если направить y_j яиц в магазин j, то прибыль составит $R_j(y_j)$. Владелец фирмы считает, что для максимизации общей прибыли не следует направлять все яйца в один магазин и стремится найти оптимальное распределение имеющихся в наличии яиц по магазинам.

Эту задачу можно сформулировать следующим образом:

$$\max \sum_{j=1}^{s} R_j(y_j) \tag{3.23}$$

при ограничениях

$$\sum_{j=1}^{s} y_j = N \text{ (наличный запас яиц)}, \tag{3.24}$$

$$y_j = 0, 1, 2, \ldots \text{ для любого } j \text{ (учитывая только целые яйца)}. \tag{3.25}$$

Для преобразования постановки задачи (3.23)–(3.25) в задачу динамического программирования примем:

1) $g_j(n)$ – прибыль при оптимальном распределении *n* яиц по магазинам 1, 2, …, *j*;

2) $y_j(n)$ – количество яиц, направленных в магазин j и дающих прибыль $g_j(n)$.

Буква g (goal) определяет значение целевой функции фирмы, т. е. прибыли. Буква n (number) относится к числу яиц, которые нужно распределить. Индекс j (just) обозначает просто номер магазина.

Отметим, что этот пример не относится к разряду динамических задач в смысле многошагового принятия решения. Многошаговое свойство задачи относится, скорее, к методу оптимизации, основанному на рассмотрении одного дополнительного магазина на каждом шаге.

Рекуррентное соотношение динамического программирования можно записать в виде

$$g_j(n) = \max_{y_i}[R_j(y_j) + g_{j-1}(n - y_j)], j = 1, 2, \ldots, s, \quad (3.26)$$

$$g_0(n) \equiv 0, \ \ j = 0\,, \quad (3.27)$$

где n = 0, 1, …, N и максимум берется только по неотрицательным целочисленным значениям y_j, удовлетворяющим условиям $y_j \le n$.

При заданных числовых значениях функций прибыли $R_j(y_j)$ и известном значении N вычисления при решении задачи начинаются с определения значений $g_1(0), g_1(1), \ldots, g_1(N)$ при j = 1. Затем находятся значения $g_2(0), g_2(1), \ldots, g_2(N)$. Вычисления продолжаются по той же схеме при последовательном возрастании значений j, пока в конце концов не определяются значения $g_s(N)$. Оптимальное распределение в принципе находится путем осуществляемого в обратном порядке выбора значений y_j, дающих в совокупности $g_s(N)$.

Пример торговой фирмы «Свежие продукты» относится к частному случаю *задачи распределения усилий*. Этот пример имеет частный характер в силу того, что единственное ограничение (3.24) является линейным. Точно такой же динамический подход в той же мере справедлив и в случае, когда единственное ограничение нелинейно. Предположим для определенности, что модель описывается следующими соотношениями:

$$\max \sum_{j=1}^{s} R_j(y_j) \quad (3.28)$$

при ограничениях

$$\sum_{j=1}^{s} H_j(y_j) = N\,, \quad (3.29)$$

$$y_j = 0, 1, 2, \ldots \text{ для любого } j. \quad (3.30)$$

Допустим, что каждая функция $H_j(y_j)$ есть неубывающая функция, принимающая целочисленные значения при любом $y_j = 0, 1, 2 \ldots$ и удовлетворяющая условию $H_j(0) = 0$. Для упрощения рассуждений примем, что $H_1(y_1) = y_1$, вследствие чего допустимое решение существует при любом значении N. На каждую величину y_j можно также наложить ограничение сверху.

Рекуррентное соотношение динамического программирования, соответствующее задаче (3.28)–(3.30), имеет следующий вид:

$$g_j(n) = \max_{y_i}[R_j(y_j) + g_{j-1}(n - H_j(y_j))],\ j = 1, 2, \ldots, s, \quad (3.31)$$

$$g_0(n) \equiv 0,\ \ j = 0, \quad (3.32)$$

где $n = 0, 1, \ldots, N$, а максимум берется только по неотрицательным целочисленным значениям y_j, удовлетворяющим условиям $H_j(y_j) \le n$. Отыскивается значение $g_s(N)$. Для выполнения вычислений нужно определить значения каждой $g_j(n)$ при $n = 0, 1, \ldots, N$, начиная с $j = 1$ и заканчивая $j = s$.

Рассмотрим еще один пример *задачи распределения усилий* с двумя ограничениями. Фирма «Вечный двигатель» планирует проведение радиорекламной кампании по сбыту своих автомобилей в одном из районов страны. В рекламных радиопередачах в течение двух недель будут рекламироваться новые модели фирмы. В зоне расположено s радиостанций. Для каждой j-й радиостанции отдел исследования сбыта фирмы определил *оценку* функции реакции сбыта, связывающей чистый доход $R_j(y_j)$ (объем продаж за вычетом расходов на рекламу) с выделенной суммой y_j в дол. США на оплату рекламных передач этой радиостанции. Общая сумма ассигнований на рекламную кампанию составляет N дол. США. Руководитель отдела рекламы фирмы определил, что число рекламных объявлений по радио в дневное время не должно превышать M. Радиостанция j включает в свою дневную программу $K_j(y_j)$ рекламных объявлений при условии, что фирма выплатит ей общую сумму y_j дол. США за рекламные передачи.

Оптимизационную модель задачи, возникшей перед руководителем отдела рекламы, можно записать следующим образом:

$$\max \sum_{j=1}^{s} R_j(y_j) \quad (3.33)$$

при ограничениях

$$\sum_{j=1}^{s} y_j \le N \text{ (общая сумма ассигнований)}, \quad (3.34)$$

$$\sum_{j=1}^{s} K_j(y_j) \le M \text{ (число рекламных объявлений в дневное время)}, \quad (3.35)$$

$$y_j = 0, 1, 2, \ldots \text{ для любого } j. \quad (3.36)$$

Предположим, что N, M и $K_j(y_j)$ – неотрицательные целые числа.

Поскольку ранее для записи одного ограничения типа (3.35) в рекуррентном виде требовалась одномерная (скалярная) переменная состояния, теперь можно легко и безошибочно догадаться, что нужна двумерная (векторная) переменная состояния. Соответствующее рекуррентное соотношение имеет вид

$$g_j(n,m) = \max_{y_i}\{R_j(y_j) + g_{j-1}[n - y_j;\ m - K_j(y_j)]\}, j = 1, \ldots, s, \quad (3.37)$$

где n = 0, 1, …, N и m =0, 1, …, M, а максимизация производится только по неотрицательным значениям y_j, удовлетворяющим условиям $y_j \le n$ и $K_j(y_j) \le m$.

Другим примером рассмотренной модели может служить распределение капиталовложений. Ежегодно фирма «Большая энергетика», выпускающая тяжелое энергетическое оборудование, рассматривает значительное число s независимых вариантов капиталовложений в различные виды основных фондов. Для осуществления варианта j требуется выделить K_j дол. США. Общий объем годовых капиталовложений M строго фиксирован. Ожидаемый доход от капиталовложений по j-му варианту составляет R_j дол. США. Фирма, естественно, стремится максимизировать общий доход по всем капиталовложениям. Руководители фирмы пришли к выводу, что вследствие ограниченных возможностей организационного управления общее число проектов, реализуемых в рамках любого года, не должно превышать N. Каждый вариант уникален, и нужно решить, принимать ли его на данный год или отвергнуть. Для отображения этого условия примем, что переменная y_j, сопоставляемая j-му варианту, может принимать только значения $y_j = 0$ (вариант отвергнут) или $y_j = 1$ (вариант принят). Перечисленные условия можно формализовать в виде следующей задачи:

$$\max \sum_{j=1}^{s} R_j(y_j) \quad (3.38)$$

при ограничениях

$$\sum_{j=1}^{s} y_j \le N \text{ (общее число проектов)}, \quad (3.39)$$

$$\sum_{j=1}^{s} K_j(y_j) \le M \text{ (общий объем капиталовложений)}, \quad (3.40)$$

$$y_j = \begin{cases} 1, \text{если вариант принят для } \forall j, \\ 0, \text{если вариант отвергнут для } \forall j. \end{cases} \quad (3.41)$$

В этой задаче $g_j(n,m)$ есть доход от оптимального распределения *m* дол. США капиталовложения по *n* проектам, выбранный из вариантов от 1 до *j*.

4 Оптимизационные задачи на графах

4.1 Понятия и определения теории графов

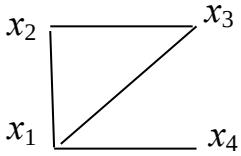
Граф есть упорядоченная пара $G = (X, E)$, где X – непустое множество, называемое множеством вершин, E – неупорядоченное бинарное отношение на X, т. е. множество неупорядоченных пар (x_i,x_j), называемых ребрами. Говорят, что ребро инцидентно вершинам, которые оно соединяет. Ребра, инцидентные одной и той же вершине, называются смежными. Вершины, принадлежащие одному и тому же ребру, называются смежными.

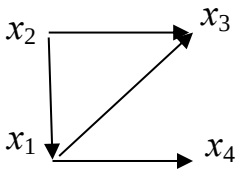
Ориентированный граф (орграф) есть упорядоченная пара $(X,E)=G$, где X – множество вершин, E – упорядоченное бинарное отношение на X, называемое множеством дуг. Первая и вторая вершины дуги называются соответственно начальной и конечной вершинами.

Граф $G' = (X', E')$ называется подграфом графа $G = (X, E)$, если X', E' содержатся соответственно в X, E, т. е. $X' \subseteq X$, $E' \subseteq$ E. Граф $G = (X, E)$ называется нагруженным, если каждому его ребру (x_i,x_j) приписано число $l(x_i, x_j) = l_{ij}$, называемое длиной этого ребра. Граф является полным, если каждая его вершина смежна со всеми другими вершинами графа. Граф называется двудольным, если его вершины могут быть разбиты на два таких подмножества X_1 и X_2 $(X_1 \cup X_2 = X_1,\ X_1 \cap X_2 = \varnothing)$, что каждое ребро графа соединяет вершину одного подмножества с вершиной другого подмножества.

В неориентированном (ориентированном) графе последовательность из m ребер $e_1, e_2, \ldots, e_m$ называется маршрутом (путем) длины m, если существует соответствующая последовательность из $(m + 1)$ (необязательно различных) вершин $x_0, \ldots, x_i, \ldots, x_m$, таких, что ребро e_i инцидентно вершинам x_{i-1}, x_i $(i = 1, 2, \ldots, m)$. Маршрут называется замкнутым (незамкнутым), если $x_0 = x_m$ $(x_0 \neq x_m)$. Если $e_i \neq e_j$ для всех $i \neq j$, то он называется цепью (орцепью). Если в цепи все вершины различны, она называется простой цепью. Замкнутая цепь (орцепь) называется циклом (контуром).

Любой граф $G = (X, E)$ можно задать бинарным отношением инцидентности между множествами X и E и отношением смежности на множестве X. Поскольку любое бинарное отношение задается матрицами, то и граф можно задать матрицей инцидентности и матрицей смежности. На рисунке 4.1 представлены граф G, орграф G, их матрицы смежности и инцидентности.





	x_1	x_2	x_3	x_4
x_1	0	1	1	1
x_2	1	0	1	0
x_3	1	1	0	0
x_4	1	0	0	0

	x_1x_2	x_1x_3	x_1x_4	x_2x_3
x_1	1	1	1	0
x_2	1	0	0	1
x_3	0	1	0	1
x_4	0	0	1	0

а

	x_1	x_2	x_3	x_4
x_1	0	0	1	1
x_2	1	0	1	0
x_3	0	0	0	0
x_4	0	0	0	0

	x_1x_2	x_1x_3	x_1x_4	x_2x_3
x_1	1	1	–1	0
x_2	0	0	1	1
x_3	–1	0	0	–1
x_4	0	–1	0	0

б

а – неориентированный граф; *б* – ориентированный граф

Рисунок 4.1 – Граф, орграф, их матрицы смежности и инцидентности

Граф называется связным, если каждая пара его вершин может быть соединена цепью. Граф не являющийся связным, может быть разбит на конечное число связных подграфов, называемых компонентами связности. Разрезом графа называется множество ребер, удаление которых увеличивает число компонент связности графа. Разрез называется простым, если никакое его собственное подмножество не является разрезом. Ориентированный граф является сильно связным, если для любых двух его вершин x_i и x_j существует путь от x_i к x_j и от x_j к x_i.

Деревом называется связный граф, не имеющий циклов. Если все вершины графа принадлежат дереву, оно называется покрывающим деревом, или остовом графа. Ребра графа, принадлежащие покрывающему дереву, называются ветвями, остальные ребра – хордами.

Ориентированное дерево имеет корень в вершине *v*, если существует путь от *v* к каждой из других вершин графа. В ориентированном дереве никакие две дуги не заходят в одну и ту же вершину. Корнем дерева может быть только та вершина, в которую не заходит ни одна дуга.

4.2 Эйлеровы циклы и задача китайского почтальона

Связный граф называется эйлеровым, если он содержит эйлеров цикл, т. е. цикл, проходящий по каждому ребру ровно один раз. Для того чтобы граф

был эйлеровым, необходимо и достаточно, чтобы он был связным и каждая его вершина имела четную степень (теорема Эйлера).

Флери дал простой алгоритм построения эйлерова цикла (если такой цикл существует):

- начинать с некоторой вершины p и каждый раз вычеркивать пройденное ребро;

- не проходить по ребру, если удаление этого ребра приводит к разбиению графа на две несвязные компоненты (не считая изолированных вершин).

Задача китайского почтальона. Требуется найти цикл, проходящий через каждое ребро нагруженного графа G по крайней мере один раз и такой, что для него общий вес (сумма величин $n_j * c(e_j)$) был минимален, где n_j показывает, сколько раз проходится ребро e_j, а $c(e_j)$ – вес ребра.

Если граф G содержит эйлеров цикл, то любой такой цикл будет оптимален и вес его равен сумме $\sum_{j=1}^{r} c(e_j)$, где r – число ребер графа.

Решение задачи китайского почтальона имеет много приложений, например, задачи о сборе мусора, доставке молока и др.

Задача о сборе мусора. Допустим, что определенный район города обслуживает единственная машина. Ребра графа G – это дороги, а вершины – пересечения дорог. Величина $c(e_i)$ – вес ребра – будет соответствовать длине соответствующей дороги. Проблема сбора мусора в данном районе сводится к нахождению цикла в графе G, проходящего по каждому ребру G по крайней мере один раз с наименьшим километражем. Поскольку емкость машины и продолжительность рабочего дня накладывают ограничения на число улиц, которое может обслужить машина за день, то реально потребуется найти минимальное число Q таких циклов, обслуживаемых по одному в день.

Задача о доставке. Эта задача, в которой требуется определить на заданном нагруженном графе маршрут, проходящий по каждой из улиц хотя бы один раз, возникает всякий раз, когда минимизируется общий километраж (время или стоимость) при доставке молока или почты.

4.3 Гамильтоновы циклы и задача коммивояжера

Связный граф называется гамильтоновым, если в нем существует цикл, проходящий по каждой вершине ровно один раз (гамильтонов цикл). Критерий существования гамильтонова цикла является слишком общим и непригодным для произвольных графов на практике. Алгебраические методы определения гамильтоновых циклов не могут быть применены к задачам с более чем несколькими десятками вершин, т. к. требуют слишком много времени и большой памяти компьютера. Более приемлемым является алгоритм Робертса и Флореса.

Алгоритм Робертса и Флореса [7] относится к методам перебора. Он имеет дело с цепью, непрерывно продлеваемой вплоть до момента, когда либо получается гамильтонов цикл, либо становится ясно, что эта цепь не может при-

вести к гамильтонову циклу. Тогда цепь модифицируется определенным способом, после чего продолжается поиск гамильтонова цикла. При реализации алгоритма предварительно удобно построить матрицу $M = [m_{ij}]$, где элемент m_{ij} есть i-я вершина, например x_q, для которой в графе G существует дуга (x_j, x_q). Число строк k матрицы M равно наибольшей полустепени исхода вершины, число столбцов n равно числу вершин графа.

Метод состоит в следующем. Некоторая начальная вершина, например, x_1, выбирается в качестве отправной и образуется первый элемент множества S, которое каждый раз будет хранить уже найденные вершины строящейся цепи. К S добавляется первая вершина, например, a в столбце x_1. Затем к множеству добавляется первая возможная вершина, например, b в столбце a и т. д. Под «возможной» вершиной понимается вершина, еще не принадлежащая S. Пусть $S = \{x_1, a, b, \ldots, x_{r-b}, x_r\}$. Существует две причины, препятствующие включению некоторой вершины на шаге r к множеству S:

а) либо в столбце x_r нет возможной вершины;

б) либо цепь, определяемая последовательностью вершин в S, имеет длину $(n - 1)$, т. е. является гамильтоновой цепью. В этом случае:

1) в графе G существует дуга (x_r, x_1) и поэтому найден гамильтонов цикл;

2) или дуга (x_r, x_1) не существует и не может быть получен никакой гамильтонов цикл.

В случаях 1 и 2 следует прибегнуть к возвращению. Возвращение состоит в удалении последней включенной вершины x_r из S и добавлении к S первой возможной вершины в столбце x_{r-1} матрицы M. Если не существует никакой возможной вершины, делается следующий шаг возвращения и т. д.

Поиск заканчивается в том случае, когда множество S состоит только из вершины x_1 и не существует никакой возможной вершины, которую можно добавить к S так, что шаг возвращения делает множество S пустым. Гамильтоновы циклы, найденные к этому методу, являются всеми гамильтоновыми циклами, существующими в графе.

Задача коммивояжера. Дан полный граф G, дугам которого приписаны произвольные веса $C = \|c(i, j)\|$. Найти гамильтонов цикл, который имеет наименьший общий вес. Если орграф G неполный, его можно рассматривать как полный орграф, приписывая отсутствующим дугам бесконечный вес.

Алгоритм решения задачи коммивояжера и ее вариантов имеет большое число практических приложений в различных областях человеческой деятельности.

В качестве примера можно привести задачу о доставке, в которой грузовик выезжает с центральной базы для доставки товаров данному числу потребителей и возвращается на базу. Стоимость перевозки пропорциональна пройденному грузовиком расстоянию. При заданной матрице расстояний между потребителями маршрут с наименьшими транспортными затратами определяется как решение задачи коммивояжера.

К решению задачи коммивояжера сводятся задачи упорядочения или планирования операций. В качестве примера приведем основную задачу планиро-

вания, в которой нужно произвести n продуктов, используя единственный тип аппаратуры. Аппаратура должна или не должна быть перенастроена после того, как произведен продукт p_i (но до того, как началось производство продукта p_j) в зависимости от комбинаций (p_i, p_j). Стоимость перенастройки (переналадки) аппаратуры постоянна и не зависит от продукта, который только что произведен, или от продукта, следующего за ним. Если переналадка аппаратуры не нужна, не требуется никаких затрат. Предположим, что эти продукты производятся в непрерывном цикле. Может ли быть найдена циклическая последовательность производства продуктов p_i (i = 1, …, n), не требующая перенастройки аппаратуры? Каков порядок производства продуктов, требующий наименьшего возможного числа перенастроек аппаратуры?

Для решения этой задачи рассматривается некоторый орграф $G = (X, E)$, вершины которого представляют продукты p_i (i = 1, …, n), а дуга (x_i, x_j) показывает, что продукт p_j можно производить вслед за продуктом p_i без переналадки аппаратуры.

Ответ на первый из поставленных выше вопросов зависит от того, существует ли в этом орграфе гамильтонов цикл. Ответ на второй вопрос можно получить с помощью итеративного применения алгоритма нахождения гамильтонова цикла в орграфе. В результате выполнения этой процедуры получается последовательность производства продуктов вида $P_{i_{r+1}}, P_{i_{r+2}}, ..., P_{i_n}, P_{i_1}, P_{i_2}, ..., P_{i_r}$, где количество меток соответствует числу переналадок аппаратуры.

Если оптимальное решение задачи содержит m переналадок, то нужно сделать (m – 1) попытку отыскать гамильтонов цикл в графах $C_{(1)}$, $C_{(2)}$, …, $C_{(m)}$ и лишь последняя из них окажется успешной и приведет к решению задачи.

Граф $C_{(m)} = (X_{(m)}, E_{(m)})$ строится из графа G путем введения фиктивных вершин (y_1, y_2, …, y_m) вместе с дугами, идущими в них из действительных вершин графа и исходящими из них в каждую действительную вершину графа, т. е.

$$X_{(m)} = X \cup (\cup y_j),$$

$$E_{(m)} = E \cup (\cup\ (x, y_j) \mid x \in X\ ) \cup \{\Sigma((y_j, x) \mid x \in X)\}.$$

Если в графе G существует гамильтонов цикл, то он имеет вид

$$x_{i_1}, ..., x_{i_\alpha}, [y_1], x_{i_{\alpha+1}}, ..., x_{i_\beta}, [y_2], ..., x_{i_\gamma}, ..., [y_3], x_{i_{\delta+1}}, ..., x_{i_n},$$

а продукты должны производиться в последовательности

$$P_{i_{\alpha+1}}, ..., P_{i_\beta}, P_{i_{\beta+1}}, ... P_{i_\gamma}, ..., P_{i_{\delta+1}}, ..., P_{i_n}, P_{i_1}, ..., P_{i_\alpha}.$$

4.4 Покрывающие деревья и леса

Рассмотрим взвешенный (нагруженный) связный неориентированный граф $G = (X, E)$, вес ребра (x_i,x_j) обозначим через C_{ij}. Определим вес дерева как сумму весов ребер его составляющих. Задача построения кратчайшего дерева графа является одной из немногих задач теории графов, которую можно считать полностью решенной.

Алгоритм Краскала:

1 Начать со вполне несвязного графа T, содержащего n вершин.

2 Упорядочить ребра графа G в порядке неубывания их весов (получить список ребер).

3 Начав с первого ребра этого списка, добавлять ребра в графе T, соблюдая условие, что такое добавление не должно приводить к появлению цикла в T.

4 Повторять шаг 3 до тех пор, пока число ребер в T не станет равным $(n - 1)$. Получившееся дерево является покрывающим (остовным) деревом минимального веса.

Данный алгоритм лучше подходит для графов с небольшим числом ребер. Для полных или близких к ним графов целесообразнее использовать алгоритм Прима [3].

Если алгоритм не заканчивается построением покрывающего дерева, то исходный граф не содержит ни одного покрывающего дерева. В этом случае граф оказывается несвязным и для него можно искать покрывающий лес, т. е. совокупность покрывающих деревьев для каждой компоненты связности графа.

Алгоритм построения максимального ориентированного леса.

Предположим, что управляющему сбытом некоторой компании требуется передать какое-либо сообщение своим агентам на местах. Для решения подобной задачи следует создать определенную систему передачи сообщений, в которой каждый агент передает сообщение агентам после того, как сам его получит. Эта система должна обеспечивать наименьшие затраты при выполнении своих функциональных задач. Любое лицо, входящее в систему, должно получить сообщение ровно один раз. Этой системе передачи сообщений можно поставить в соответствие некоторый граф G, каждая вершина которого соответствует лицу, входящему в систему передачи сообщений, а каждая дуга представляет собой возможность передачи сообщений от одного лица к другому.

Поиск минимального ориентированного леса этого графа и составит решение задачи [9]. Максимальным (минимальным) ориентированным лесом графа G назовем ориентированный лес графа G с максимальным (минимальным) возможным весом составляющих его дуг.

В алгоритме Эдмондса, предназначенном для построения максимального ориентированного леса, используют два подмножества («букета») – букет вершин и букет дуг. Первый букет содержит только те вершины, которые были просмотрены в процессе выполнения алгоритма, а второй – дуги, условно включаемые в максимальный ориентированный лес в ходе процедуры просмот-

ра вершин. На протяжении всей процедуры дуга второго букета образуют лес для соответствующего графа. Перед началом процедуры оба букета пусты.

Первый этап алгоритма состоит в просмотре вершин графа и формировании букетов. Последовательно в произвольном порядке просматриваются вершины графа. Просмотр вершины состоит в том, что из дуг, заходящих в данную вершину, выбирается дуга с максимальным весом. Если добавление этой дуги к дугам, уже вошедшим в букет, не нарушает свойства букета образования леса, то выбранная дуга вводится в букет. В противном случае формируется новый уменьшенный граф путем «стягивания» дуг и вершин выявленного контура в одну вершину. В новом графе веса некоторых дуг соответствующим образом корректируются. Кроме того, для нового графа корректируется состав букета вершин и букета дуг: в них остаются только те элементы (вершины или дуги), которые присутствуют в новом графе.

После проведенной корректировки процедура просмотра вершин продолжается аналогичным образом и заканчивается тогда, когда просмотрены все вершины графа. По окончании процедуры просмотра вершин дуги последнего сформированного букета образуют ориентированный лес в графе.

Второй этап алгоритма состоит в следующем. Граф, полученный по окончании первого этапа, расширяется путем замены контура фиктивной вершиной, в которую он стягивается при соответствующем преобразовании графа. В новый расширенный граф, а также в соответствующий ему букет включаются все дуги указанного контура, за исключением одной. Исключенная дуга выбирается так, чтобы дуги, составляющие новый букет, образовывали лес в соответствующем графе. Этот процесс последовательного расширения графа продолжается до тех пор, пока не будет восстановлен исходный граф.

Пример. Используем алгоритм для получения максимального ориентированного леса в графе G_0, изображенного на рисунке 4.2. Вес каждой дуги этого графа указан на рисунке рядом с соответствующей дугой. В процессе выполнения алгоритма вершины просматриваются в порядке возрастания их номеров. Результаты просмотра первых четырех вершин приведены в таблице 4.1.

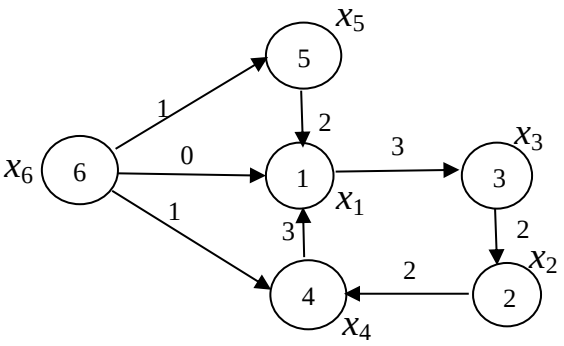


Рисунок 4.2 – Граф для получения максимального ориентированного леса

Таблица 4.1 – Результаты просмотра первых четырех вершин графа

Просмотренные вершины	Букеты	
	V_0	A_0
x_1	1	(4,1)
x_2	1, 2	(4,1), (3,2)
x_3	1, 2, 3	(4,1), (3,2), (1,3)
x_4	1, 2, 3, 4	(4,1), (3,2), (1,3), (2,4)

После просмотра вершины 4 дуги букета A_0 уже не определяют ориентированный лес, поскольку они образуют контур (1,3), (3,2), (2,4), (4,1). На данном этапе выполнения алгоритма осуществляется стягивание этого контура в одну вершину x_0. Полученный в результате этого стягивания граф G_1 изображен на рисунке 4.3. Здесь же рядом с каждой дугой приведены их веса с указанием в соответствующих случаях формул пересчета. Граф G_1 состоит из трех вершин x_0, x_5, x_6. Результаты просмотра этих вершин приведены в таблице 4.2.

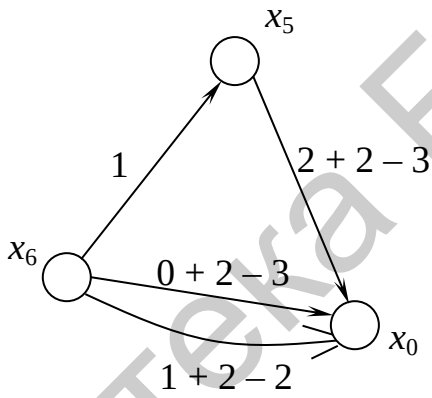


Рисунок 4.3 – Граф в результате стягивания контура

Таблица 4.2 – Результаты просмотра трех вершин x_0, x_5, x_6

Просмотренные вершины	Букеты	
	V_1	A_1
x_5	5	(6,5)
x_6	5,6	(6,5)
x_0	–	(6,5), (5, x_0)

После завершения просмотра всех трех вершин графа G_1 алгоритм формирует в графе максимальный ориентированный лес, состоящий из дуг (6,5) и (5, x_0). В ходе выполнения алгоритма вершина x_0 заменяется соответствующим контуром, при этом к дугам (6,5) и (5, x_0) добавляются дуги (1,3), (3,2), (2,4), (4,1). Далее дуга (4,1) удаляется из рассматриваемой совокупности дуг, так что в ней остается лишь одна дуга, заходящая в вершину 1, – дуга (5,1). Полученная совокупность дуг (6,5), (5,1), (1,3), (3,2) и (2,4) определяет максимальный ориентированный лес в графе G_0, вес этого леса равен 10 и является максимально возможным. Заметим, что построенный лес оказался в графе G_0 покрывающим ориентированным деревом с корнем в вершине 6.

4.5 Поиск экстремальных путей

Исследование задач, связанных с нахождением кратчайших путей, ассоциируется с решением проблем, требующих экономии трудовых затрат. Понятие пути максимальной длины фигурирует в задачах определения критического пути, заключающихся в том, что для осуществления некоторого производственного процесса требуется установить согласованную ко времени последовательность выполнения различных этапов этого процесса. Этапы, требующие наибольшего времени, должны постоянно и надежно обеспечиваться ресурсами. Завершение менее продолжительных этапов может быть на соответствующий срок отложено.

4.5.1 Задача о кратчайшем пути в графе

Пусть задан граф $G = (X, E)$, дугам которого приписаны веса (длины, стоимости, затраты), задаваемые матрицей весов $A = \|a_{ij}\|$. Если в графе отсутствует некоторая дуга, то соответствующий элемент матрицы равен ∞. Требуется найти кратчайший путь от заданной начальной вершины $S \in X$ до заданной конечной вершины $t \in X$, при условии, что такой путь существует. Если $R(s)$ – множество вершин, достижимых из вершины S, то условием существования (s,t)-пути является факт $t \in R(s)$.

Наиболее эффективный алгоритм решения этой задачи принадлежит Дейкстре. Алгоритм основан на приписывании вершинам вершинных пометок, причем пометка вершины дает верхнюю границу длины пути от s к этой вершине. Эти пометки (их величины) постепенно уменьшаются с помощью некоторой итерационной процедуры, и на каждом шаге итерации точно одна из временных пометок становится постоянной. Постоянная пометка дает точную длину кратчайшего пути от s к рассматриваемой вершине.

Алгоритм Дейкстры ($a_{ij} \geq 0$).

Перед началом выполнения алгоритма все вершины и дуги не окрашены (не помечены). Каждой вершине в ходе выполнения алгоритма присваивается число $\lambda(x)$, равное длине кратчайшего пути из s в x. Алгоритм Дейкстры заключается в следующем:

1 Положить $\lambda(s) = 0$ и $\lambda(x) = \infty$ для всех $x \neq s$. Окрасить вершину s и положить $y = s$, где y – последняя из окрашенных вершин.

2 Для каждой неокрашенной вершины x пересчитать величину $\lambda(x)$ по формуле

$$\lambda(x) = \min\{\lambda(x), \lambda(y) + a(x,y)\}. \qquad (4.1)$$

Если $\lambda(x) = \infty$ для всех неокрашенных вершин x, закончить процедуру: в исходном графе не существует пути из вершины s в неокрашенные вершины. В

противном случае окрасить ту из вершин x, для которой величина $\lambda(x)$ является наименьшей. Кроме того, окрасить дугу, ведущую в выбранную на данном шаге вершину x. Положить $y = x$.

3 Если $y = t$, закончить процедуру, кратчайший путь из вершины s в вершину t найден. В противном случае перейти к шагу 2.

Отметим, что окрашивание дуги, заходящей в вершину x, происходит тогда, когда эта вершина получает постоянную пометку. Окрашенные дуги образуют в исходном графе ориентированное дерево с корнем в вершине s. Алгоритм можно рассматривать как процедуру наращивания ориентированного дерева с корнем в вершине s.

Алгоритм Дейкстры может быть обобщен на случай, когда некоторые из дуг имеют отрицательные веса. Необходимая модификация алгоритма состоит в следующем:

1 На шаге 2 алгоритма пересчет величины $\lambda(x)$ с помощью соотношения (4.1) производится для всех вершин, а не только для неокрашенных.

2. Если для некоторой окрашенной вершины x происходит уменьшение величины $\lambda(x)$, то с этой вершины и инцидентной ей окрашенной дуги окраска снимается.

3 Процедура алгоритма заканчивается только тогда, когда все вершины окрашены и когда после выполнения шага 2 ни одно из чисел $\lambda(x)$ не меняется.

Во многих практических приложениях требуется, чтобы кратчайший путь из s в t обладал некоторыми дополнительными свойствами. В этом случае целесообразно просто найти K кратчайших путей из s в t и выбрать из них тот, который обладает нужными свойствами. В работе [9] изложен метод, предложенный Йеном и позволяющий находить K кратчайших простых цепей.

4.5.2 Задача о кратчайшем пути между всеми парами вершин

Пусть требуется найти кратчайшие пути между всеми парами вершин нагруженного графа. Алгоритм, предложенный Флойдом, применим к графам с произвольной матрицей весов и базируется на использовании последовательности из n преобразований (итераций) начальной матрицы весов $A = \|a_{ij}\|$. При этом на k-й итерации матрица представляет длины кратчайших путей между каждой парой вершин с тем ограничением, что путь между x_i и x_j содержит в качестве промежуточных только вершины из множества $\{x_1, x_2, \ldots, x_{k-1}, x_k\}$.

Алгоритм Флойда:

1 Перенумеровать вершины исходного графа целыми числами от 1 до n. Определить матрицу $W^{(0)}$, задав величину каждого ее элемента w_{ij} равной длине кратчайшей дуги, соединяющей вершину x_i с вершиной x_j ($W^{(0)} = A$).

Если в исходном графе указанные вершины не соединяются дугами, положить $w_{ij}^{(0)} = \infty$. Для всех i положить $w_{ii}^{(0)} = 0$.

2. Для целого m, последовательно принимающего значения 1, 2, ..., n,

определить по величине элементов матрицы $W^{(m-1)}$ величины элементов матрицы $W^{(m)}$, используя рекуррентное соотношение

$$w_{ij}^{(m)} = \min(w_{im}^{(m-1)} + w_{mj}^{(m-1)}, w_{ij}^{(m-1)}) . \qquad (4.2)$$

При определении величины каждого элемента матрицы $W^{(m)}$ фиксировать соответствующий кратчайший путь.

По окончании данной процедуры величина элемента $w_{ij}^{(n)}$ матрицы $W^{(n)}$ определяет длину кратчайшего пути, ведущего из вершины x_i в вершину x_j.

Отметим, что для всех i и m должно быть $d_{ij}^{(m)} = 0$, $d_{im}^{(m-1)} = d_{ij}^{(m)}$, $d_{mi}^{(m-1)} = d_{mi}^{(m-1)}$. Данный алгоритм экономит почти половину того времени, которое потратилось бы на решение этой задачи *n*-кратным применением алгоритма Дейкстры.

4.5.3 Задачи, близкие к задаче о кратчайшем пути

В алгоритме Флойда нахождения кратчайшего пути между всеми парами вершин графа была использована трехместная операция, которая является частным случаем более общей трехместной операции

$$z_{ij}^{(k)} = \mathrm{opt}(z_{ik}^{(k-1)} \otimes z_{kj}^{(k-1)}, z_{ij}^{(k-1)}) , \qquad (4.3)$$

где z – функция пути, подлежащая оптимизации;

$\otimes$ – некоторая общая операция. Для этой операции должно выполняться условие: если путь от x_i к x_j проходит через промежуточную вершину x_k и z – некоторая характеристика, то $z_{ij} = z_{ik} \otimes z_{kj}$.

Наиболее надежный путь. Пусть «вес» дуги представляет ее надежность, т. е. вероятность ее существования в графе (для физических систем – вероятность того, что дуга находится в работоспособном состоянии). Надежность пути от s к t, составленного из дуг, принадлежащих множеству P, находится по формуле

$$\rho(P) = \prod_{(x_i, x_j) \in P} \rho_{ij} ,$$

где ρ_{ij} – надежность дуги (x_i, x_j).

Задачу о нахождении наиболее надежного пути от s к t можно свести к задаче о кратчайшем пути, взяв в качестве веса a_{ij} дуги (x_i, x_j) величину $a_{ij} = -\log \rho_{ij}$.

Для определения наиболее надежного пути между всеми парами вершин можно ввести трехместную операцию $p_{ij} = \max(p_{ik}p_{kj}, p_{ij})$ и использовать алгоритм Флойда. В качестве начальной $[z_{ij}^{(0)}]$-матрицы берется матрица надежностей дуг, причем нулевые элементы указывают на отсутствие соответствующих дуг.

Путь с наибольшей пропускной способностью. Пусть каждая дуга (x_i, x_j) графа имеет пропускную способность q_{ij} и требуется найти пути с наибольшей пропускной способностью между всеми парами вершин. Для решения этой задачи методом Флойда вводится трехместная операция $q_{ij} = \max(\min(q_{ik}, q_{kj}), q_{ij})$. Начальной матрицей $[q_{ij}^{(0)}]$ является исходная матрица пропускных способностей дуг с нулевыми элементами на месте отсутствующих дуг.

Задача об «узких местах» – это задача поиска такого пути между двумя вершинами, в котором длина кратчайшей дуги максимальна (узким местом пути называется кратчайшая из входящих в него дуг). Если в алгоритмах Флойда и Дейкстры операции сложения и сравнения на минимум заменить операциями сравнения на минимум и на максимум, то их можно использовать для решения задачи об узких местах. Отсутствие какой-либо дуги необходимо фиксировать, полагая вес несуществующей дуги равным $-\infty$. Кроме того, длина кратчайшей дуги нулевого пути (т. е. пути, не содержащего дуг) полагается равной $-\infty$.

Задача о путях с усилением. Поставим в соответствие каждой дуге некоторое действительное число, называемое коэффициентом усиления дуги. Назовем усилением пути величину произведения коэффициентов усиления всех дуг, составляющих данный путь. Если при этом некоторая дуга входит в путь многократно, то коэффициент усиления этой дуги повторяется в качестве сомножителя соответствующее число раз. Задача о путях с усилением формулируется как задача нахождения такого пути между двумя вершинами, который дает наибольшую величину усиления.

Пример. Рассмотрим *задачу о финансисте*.

Финансист крупной компании решил вложить ее свободный капитал в облигации, от которых можно было бы иметь доход в последующий пятилетний период. В распоряжении финансиста имеется несколько видов облигаций. В течение рассматриваемого пятилетнего периода можно реализовывать облигации и высвобождающиеся средства вкладывать в облигации других видов. Как в такой ситуации финансист должен распределить во времени вложения имеющегося капитала?

Поставим в соответствие началу каждого года, когда по предположению производится вложение капитала, вершину графа. Каждому виду облигаций поставим в соответствие дугу, ведущую из вершины, представляющей момент приобретения облигации, в вершину, представляющую момент их реализаций. Коэффициент усиления каждой дуги совпадает с суммой, отнесенной к одному дол. США, которая получается в результате реализации облигаций. Тогда задачу финансиста можно рассматривать как задачу поиска пути, имеющего наибольшую величину усиления.

С помощью алгоритмов Флойда, Дейкстры можно решать задачу поиска путей с максимальной величиной усиления, если длиной дуги графа считать коэффициент усиления в данной дуге и если в этих алгоритмах операции сложения и сравнения на минимум заменить соответственно операциями умножения и сравнения на максимум. Если задача о кратчайшем пути не может быть решена при существовании в исходном графе контуров отрицательной длины, то задача о путях с максимальной величиной усиления не может быть решена при существовании в исходном графе контура с величиной усиления, большей единицы.

Ниже представлена таблица 4.3 задач, родственных задаче о кратчайшем пути в графе.

Таблица 4.3 – Задачи, родственные задаче о кратчайшем пути в графе

Задачи	Вес пути	Операция		$W^{(k)}$
		$\otimes$	opt	
Задача о кратчайшем пути	Длина l_{ij}	+	min	$w_{ij}^{(m)} = \min(w_{im}^{(m-1)} + w_{mj}^{(m-1)}, w_{ij}^{(m-1)})$
Задача о наиболее надежном пути	Надежность ρ_{ij}	*	max	$\rho_{ij}^{(k)} = \max(\rho_{ik}^{(k-1)}\rho_{kj}^{(k-1)}, \rho_{ij}^{(k-1)})$
Путь с наибольшей пропускной способностью	Пропускная способность q_{ij}	min	max	$q_{ij}^{(k)} = \max(\min(q_{ik}^{(k-1)}, q_{kj}^{(k-1)}), q_{ij}^{(k-1)})$
Задача об «узких» местах	Длина l_{ij}	min	max	$l_{ij}^{(k)} = \max(\min(l_{ik}^{(k-1)}, l_{kj}^{(k-1)}), l_{ij}^{(k-1)})$
Задача о путях с усилением	Коэффициент усиления v_{ij}	*	max	$v_{ij}^{(k)} = \max(v_{ik}^{(k-1)}v_{kj}^{(k-1)}, v_{ij}^{(k-1)})$

4.6 Задачи о покрытии и паросочетании в нагруженном графе

Задача о наименьшем вершинном покрытии является обобщением задачи о нахождении числа доминирования графа. Она имеет большое число прямых приложений.

Пусть A^{T} – транспонированная матрица смежности графа G с единичными диагональными элементами. Задача определения наименьшего доминирующего множества графа G эквивалентна задаче нахождения такого наименьшего множества столбцов в матрице A^{T}, при которой каждая строка матрицы содержит единицу хотя бы в одном из выбранных столбцов. Эта задача о поиске наименьшего множества столбцов, накрывающих все строки, и получила название задачи «о наименьшем вершинном покрытии» (ЗНВП).

В общей постановке ЗНВП матрица T, состоящая из 0 и 1, необязательно является квадратной. Кроме того, каждому столбцу j (каждой вершине x_j) ставится в соответствие некоторый вес (стоимость) c_i и требуется выбрать покрытие с наименьшей общей стоимостью.

Таким образом, задача построения доминирующего множества является частной задачей о вершинном покрытии квадратной булевой матрицы T с $t_{ij} = 1$ для всех $j = 1, \ldots, N$.

Общая ЗНВП может быть сформулирована как задача линейного программирования:

$$f = \sum_{j=1}^{N} c_j x_j = \min,$$

$$\sum_{j=1}^{N} t_{ij} x_j \geq 1, i = 1, ..., M, \qquad (4.4)$$

$$c_j \geq 0,$$

где $x_j = \begin{cases} 1, \text{если вершина } x_j \text{ включается в покрытие } P, \\ 0, \text{в противном случае;} \end{cases}$

t_{ij} – элементы матрицы T с M строками и N столбцами.

Паросочетанием графа $G = (X, E)$ называется такое подмножество M его ребер, при котором каждая вершина графа инцидентна не более чем одному ребру этого подмножества. Очевидно, что в любом паросочетании никакие два ребра не являются смежными. Паросочетание может быть названо «независимым множеством ребер».

Реберным покрытием (покрытием) графа $G = (X, E)$ называется такое подмножество Q его ребер, что каждая вершина графа инцидентна по крайней мере одному ребру из Q. Очевидно, что покрытие является «доминирующим множеством ребер».

С отысканием паросочетаний и покрытий часто приходится сталкиваться на практике. Перечислим наиболее распространенные задачи о паросочетаниях и покрытиях для ненагруженного графа ($c_{ij} = 1$, если $(x_i, x_j) \in E$):

- задача о паросочетании максимальной мощности;
- задача о покрытии минимальной мощности.

Нетрудно доказать следующее утверждение: решение задачи о покрытии минимальной мощности дает решение задачи о паросочетании максимальной мощности, и наоборот, т. е. эти задачи эквивалентны друг другу.

Пусть M – произвольное паросочетание. Выберем произвольную вершину x, не инцидентную ни одной из дуг, входящих в M. Присоединим к паросочетанию любую дугу, инцидентную x. Повторяем эту процедуру до тех пор, пока не будут просмотрены все вершины x, не инцидентные ни одной из дуг, входящих в M. Полученное в результате множество дуг Q задается некоторым покрытием графа.

Пусть Q – произвольное покрытие. Выберем произвольную вершину x, инцидентную более чем одной дуге, входящей в P. Исключим из Q любую дугу, инцидентную x. Повторяем эту процедуру до тех пор, пока не останется ни одной вершины, инцидентной более чем одной дуге. Полученное в результате множество дуг M есть искомое паросочетание.

Перечислим наиболее распространенные задачи о паросочетаниях и покрытиях для нагруженного графа (c_{ij} – вес (i,j)-го ребра):

- задача о паросочетании с максимальным весом

$$C_{м} = \sum_{(i,j)\in M} c_{ij} = \max_{(i,j)\in M};$$

- задача о покрытии с минимальным весом

$$C_{м} = \sum_{(i,j)\in Q} c_{ij} = \min_{(i,j)\in Q}.$$

Эквивалентными являются задачи о паросочетаниях с минимальным (максимальным) весом и задачи о покрытии с минимальным (максимальным) весом соответственно. Между задачей о паросочетании с максимальным весом и задачей о покрытии с минимальным весом аналогичных соотношений, по-видимому, не существует.

Задачу о паросочетании с максимальным весом можно выразить в терминах линейного программирования:

$$\begin{gathered} z = \sum_{j=1}^{r} c_j \xi_j = \max, \\ \sum_{j=1}^{r} b_{ij} \xi_j \le 1, \quad i = 1, ..., n, \end{gathered} \tag{4.5}$$

где $\xi_j = \begin{cases} 1, \text{ если ребро } e_j \text{ входит в паросочетание } M, \\ 0 \text{ в противном случае;} \end{cases}$

$[b_{ij}]$ – матрица инцидентности графа G.

Задачу о покрытии с минимальным весом можно сформулировать в терминах линейного программирования:

$$\begin{gathered} z = \sum_{j=1}^{r} c_j \eta_j = \min, \\ \sum_{j=1}^{r} b_{ij} \eta_j \ge 1, \quad i = 1, ..., n, \end{gathered} \tag{4.6}$$

где $\eta_j = \begin{cases} 1, \text{ если ребро } e_j \text{ входит в покрытие } Q, \\ 0 \text{ в противном случае.} \end{cases}$

В специальном случае, когда веса c_j произвольны, а сам граф является двудольным, задача о паросочетании с максимальным весом сводится к задаче о назначениях и к транспортной задаче.

4.7 Прикладные задачи на поиск паросочетаний и покрытий

Задача о гармоничном составе экспедиции. Группа исследователей намерена отправится в экспедицию, причем желательно, чтобы количество участников экспедиции было максимальным. Однако некоторые из них не ладят друг с другом: если одно из конфликтных лиц отправляется в экспедицию, то другое остается дома.

Пусть $x_j = \begin{cases} 1, \text{если } j \text{ – е лицо отправили,} \\ 0, \text{если } j \text{ – е лицо оставили.} \end{cases}$

Тогда оптимизационную задачу можно сформулировать следующим образом:

$$\sum x_j = \max,$$

$$\begin{cases} x_k + x_h \le 1 \text{ для несовместных пар лиц } h \text{ и } k, \\ 0 \le x_j \le 1,\ x_j - \text{целое.} \end{cases}$$

Эта задача эквивалентна задаче о паросочетании максимальной мощности для ненагруженного графа, где

$$x_j = \begin{cases} 1, \text{когда ребро } j \text{ входит в } P, \\ 0, \text{когда ребро } j \text{ не входит,} \end{cases}$$

$x_k + x_h \le 1$ – для тех пар ребер k и h, которые имеют общую точку.

Обобщение:

1 Если ввести оценку полезности каждого лица, то целевая функция $\sum_{j=1}^{n} d_j x_j = \max$.

На графе это равносильно тому, что каждое ребро графа имеет вес и отыскивается паросочетание максимального веса.

2 В задаче о гармонии в экспедиции можно считать, что каждая «несовместная» группа состоит не из двух, а из некоторого множества лиц, например, $|S|$, тогда ограничение задачи имеет вид

$$\sum_{j \in S} x_j \le |S| - 1 .$$

Задача об эффективной экспедиции. Группа исследователей определила минимальное число участников λ. Однако для успеха экспедиции необходимо,

чтобы для каждого типа работ в составе участников было по крайней мере одно лицо, способное к ее выполнению.

Пусть $x_j = \begin{cases} 1, \text{ если } j \text{ – е лицо включено в экспедицию,} \\ 0, \text{ если } j \text{ – е лицо не включено в экспедицию,} \end{cases}$

S_i – множество лиц, способных к выполнению j-го вида работ.

Тогда формализация задачи имеет вид:

$$\sum x_j = \min,$$

$$\begin{cases} \sum\limits_{j \in S_i} x_j \geq 1, \\ 0 \leq x_j \leq 1,\ x_j - \text{целое}. \end{cases}$$

Очевидно, что данная задача эквивалента задаче о покрытии P наименьшей мощности в соответствующем графе, где

$$x_j = \begin{cases} 1, \text{ когда ребро } j \text{ входит в } P, \\ 0, \text{ когда ребро } j \text{ не входит в } P, \end{cases}$$

S_i – множество ребер, инцидентных вершине i.

Обобщение этих задач состоит в приписывании ребрам графа определенных весов и требовании многократного покрытия вершины.

Задача о доставке. Некоторая фирма каждый день доставляет своим клиентам товары на грузовых машинах. Существует множество допустимых маршрутов доставки, каждый из которых позволит обслужить определенное подмножество клиентов и требует использования в течение дня одного транспортного средства. Каждый маршрут характеризуется определенными расходами (стоимость расходуемого топлива и т. д.). Необходимо выбрать такое множество маршрутов, при котором суммарные расходы минимальны и обеспечивается обслуживание каждого из клиентов.

Пусть $x_j = \begin{cases} 1, \text{ если маршрут } j \text{ выбран,} \\ 0, \text{ если маршрут } j \text{ не выбран.} \end{cases}$

Введем в рассмотрение матрицу A с элементами a_{ij} :

$$a_{ij} = \begin{cases} 1, \text{ если клиент } i \text{ обслужился по маршруту } j, \\ 0 \text{ в противном случае.} \end{cases}$$

Пусть c_j – стоимость доставки по маршруту i. Формализация задачи в векторной форме имеет вид

$$CX = \min,$$
$$\begin{cases} Ax = e, \\ 0 \le x_j,\ x_j - \text{целое}, \end{cases}$$

где e – единичный вектор-столбец.

Задача о доставке может содержать и ограничение вида $\sum x_j \le k$, где k – максимально возможное число выбранных маршрутов.

Задача об агенте по продаже недвижимости. Агентство по продаже недвижимого имущества имеет для продажи целый ряд домов и некоторое количество потенциальных покупателей. Каждый такой покупатель может проявлять интерес к более чем одному из домов. Агент по продаже недвижимого имущества может достаточно точно оценить, сколько каждый покупатель заплатит за каждый из представляемых для него интерес дом.

Агент по продаже недвижимости получает 7 % комиссионных отчислений от каждой сделки. Поэтому он заинтересован в максимизации общего объема совершенных им продаж в дол. США. Как ему решить задачу?

Для решения задачи построим двудольный граф. Пусть каждый покупатель и каждый дом представляется вершинами графа; вершины соединены ребром в случае, когда конкретный покупатель желает приобрести определенный дом. Каждая дуга при этом представляет возможную сделку и ей приписан вес, равный размеру комиссионных отчислений.

Решение задачи сводится к нахождению в данном графе паросочетания с максимальным весом.

Задача о службе знакомств. Служба знакомств создает возможность встречи каждому обратившемуся в нее лицу по крайней мере с одним подходящим для него претендентом на знакомство. Размер затрат службы на каждую встречу различен в зависимости от конкретных мероприятий, требуемых на ее организацию. Как с минимальными издержками выполнить все обязательства перед клиентами?

Как и в предыдущем примере, строим граф, в котором каждому клиенту соответствует вершина и каждой совместимой паре – дуга. Любое покрытие (реберное) этого графа представляет собой организацию по крайней мере одной подходящей встречи для каждого клиента. В итоге необходимо найти покрытие с наименьшими общими издержками или покрытие с минимальным весом.

4.8 Минимаксные и минисуммные задачи размещения

В практической деятельности постоянно возникают задачи наилучшего размещения оборудования (или средств обслуживания) на сетях или графах. В частности, если граф представляет сеть дорог, и вершины соответствуют отдельным районам, то можно поставить задачу оптимального размещения больниц, пунктов милиции, пожарных частей и многих других крайне необходимых

предприятий и служб. В таких случаях критерий оптимальности может состоять в минимизации расстояния (или времени проезда) от пункта обслуживания до самой отдаленной вершины графа. В более общем случае требуется разместить несколько пунктов обслуживания, например аварийных служб. Задачи такого типа называются минимаксными задачами размещения. Полученные при решении этих задач места размещения пунктов обслуживания называются центрами графа.

В некоторых задачах размещения желательно минимизировать сумму всех расстояний от вершин графа до центра обслуживания. Такой критерий используется, например, в задаче о размещении склада в сети дорог, где вершины представляют потребителей, обслуживаемых этим складом, или в задаче размещения телефонных станций в телефонной сети, где вершины представляют абонентов. Задачи такого типа относятся к минисуммным задачам размещения. Целевая функция в подобных задачах часто является не просто суммой расстояний, а суммой различных функций от расстояний. Места размещения пунктов обслуживания, получаемых в результате решения минисуммной задачи, называются медианами графа.

4.8.1 Разделения, центр и радиус

Пусть $D = (d_{ij}) = (d(x_i,x_j))$ – матрица кратчайших расстояний между парами вершин графа $G = (X, E)$, v_j – вес вершины $x_i \in X$ графа.

Для каждой вершины x_i графа определим два числа:

$$s_0(x_i) = \max_{x_j \in X}[v_j d(x_i, x_j)], \tag{4.7}$$

$$s_t(x_i) = \max_{x_j \in X}[v_j d(x_j, x_i)]. \tag{4.8}$$

Числа $s_0(x_i)$ и $s_t(x_i)$ называются соответственно числом внешнего разделения и числом внутреннего разделения вершины x_i.

Вершина x_0^* называется внешним центром графа G, для которой

$$s_0(x_0^*) = \min_{x_i \in X}[s_0(x_i)]. \tag{4.9}$$

Вершина x_t^* называется внутренним центром графа G, для которой

$$s_t(x_t^*) = \min_{x_i \in X}[s_t(x_i)]. \tag{4.10}$$

У графа может быть несколько внешних и внутренних центров, т. е. можно говорить о множествах внешних и внутренних центрах.

Число внешнего разделения вершины x_0^*, являющейся внешним центром, называется внешним радиусом: $p_0 = s_0(x_0^*)$; число внутреннего разделения внутреннего центра называется внутренним радиусом: $p_t = s_t(x_t^*)$.

Пример. Рассмотрим ориентированный граф, изображенный на рисунке 4.4, и предположим, что все веса вершин и дуг графа равны единице. Ниже представлена матрица $D = (d_{ij})$ кратчайших расстояний, а также приведены в присоединенных к матрице столбце и строке числа внешних и внутренних разделений соответственно.

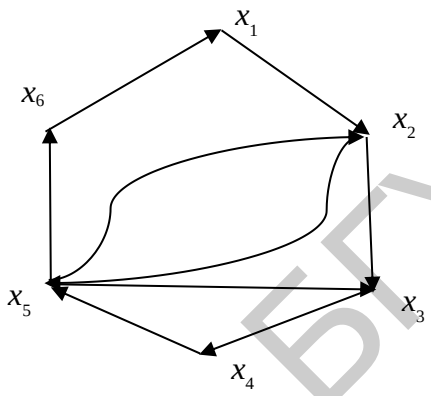


Рисунок 4.4 – Ориентированный граф

$D(G) =$

	x_1	x_2	x_3	x_4	x_5	x_6	$s_0(x_i)$ ↓
x_1	0	1	2	3	2	3	3
x_2	3	0	1	2	1	2	3
x_3	4	3	0	1	2	3	3
x_4	3	2	2	0	1	2	3
x_5	2	1	1	2	0	1	2*
x_6	1	2	3	4	3	0	4
$s_t(x_i) \rightarrow$	4	3*	3*	4	3*	3*	

У данного графа имеется только один внешний центр (вершина x_5) и четыре внутренних центра, образующих множества $\{x_2, x_3, x_5, x_6\}$. Внешний радиус графа равен 2, а внутренний равен 3.

Задача о размещении пункта обслуживания. Рассмотрим задачу обслуживания нескольких жилых районов или населенных пунктов (связанных между собой дорожной сетью) каким-либо одним пунктом обслуживания (например, одной больницей, одним пожарным депо). Данный пункт должен быть размещен в одном из этих районов.

Предположим, что длины c_{ij} дуг графа G (вершины которого соответствуют районам, а дуги дорогам) образуют матрицу времен проезда между этими районами. Эта матрица в общем случае может не быть симметрической.

Каждой вершине приписан вес (v_j), представляющий вероятность потребности каждого района в соответствующем пункте обслуживания. Эти веса, например, могут быть пропорциональны численности населения. В случае размещения пожарного депо или милицейского участка интересуемся временем, необходимым для проезда в наиболее отдаленный из этих районов. Задача размещения состоит в минимизации этого времени. Вершина, которая минимизирует время проезда до самого отдаленного района, является внешним центром графа. В примере это вершина x_5.

В случае размещения больницы интересуются временем, необходимым для проезда машины скорой помощи в самый отдаленный район и возвращения ее в больницу. Если определить число внешне-внутреннего разделения вершины x_i с помощью равенства

$$s_{0t}(x_i) = \min_{x_j \in X} \{v_j [d(x_i, x_j) + d(x_j, x_i)]\}, \tag{4.11}$$

то вершину x_{0t}^*, на которой достигается минимум выражения $s_{0t}(x_i)$, можно назвать внешне-внутренним центром. Для размещения больницы именно его нужно найти на соответствующем графе. В примере внешне-внутренним центром является вершина x_5. Внешне-внутренний радиус равен 5.

4.8.2 Кратные центры (*p*-центры) и размещение нескольких пунктов обслуживания

Пусть X_p – подмножество, содержащее p вершин множества X графа $G = (X, E)$. Через $d(X_p, x_i)$ будем обозначать наикратчайшее из расстояний между вершинами множества X_p и вершиной x_i, т. е.

$$d(X_p, x_i) = \min_{x_j \in X_p} [d(x_j, x_i)], \tag{4.12}$$

$$d(x_i, X_p) = \min_{x_j \in X_p} [d(x_i, x_j)]. \tag{4.13}$$

Можно определить числа разделения для множества вершин x_0:

$$s_0(X_p) = \max_{x_j \in X} [v_j d(X_p, x_j)], \tag{4.14}$$

$$s_t(X_p) = \max_{x_j \in X} [v_j d(x_j, X_p)], \tag{4.15}$$

где $s_0(X_p)$, $s_t(X_p)$ – числа внешнего и внутреннего разделения множества X_p.

Множество X_{p0}^{*}, для которого $s_0(X_{p0}^{*}) = \min\limits_{X_p \subseteq X} [s_0(X_p)]$ называется p-кратным внешним центром графа G; аналогично определяем p-кратный внутренний центр X_{pt}^{*} графа.

Если центры графа легко могут быть найдены из матрицы взвешенных расстояний, то находить таким же способом (полным перебором) p-центры можно лишь для небольших графов и для небольших значений p.

Если одного пункта обслуживания на город оказывается недостаточно, возникает задача о наилучшем размещении нескольких пунктов обслуживания. Например, найти наименьшее число пожарных депо и такое их размещение, чтобы расстояние от каждого жилого района до ближайшего к нему пожарного депо не превышало наперед заданную величину. Если же число пожарных депо известно, то требуется разместить их так, чтобы было минимально возможным расстояние от любого района до ближайшего к нему депо. Если пожарные депо должны размещаться в вершинах соответствующего графа G, то задача будет состоять в нахождении p-центра графа для p = 1, 2, 3, … и т. д. до тех пор, пока число разделения p-центра не станет меньше или равно заданному расстоянию. Последнее полученное значение числа p будет наименьшим числом пожарных депо, а p-центр – их оптимальным размещением, удовлетворяющим предъявляемым требованиям.

4.8.3 Передаточные числа и медиана графа

Пусть дан граф $G = (X, E)$. Для каждой вершины графа определим два числа, которые назовем *передаточными* числами:

$$\sigma_0(x_i) = \sum_{x_j \in X} v_j d(x_i, x_j), \tag{4.16}$$

$$\sigma_t(x_i) = \sum_{x_j \in X} v_j d(x_j, x_i), \tag{4.17}$$

где $d(x_i, x_j)$ – кратчайшее расстояние от вершины x_i до вершины x_j.

Числа $\sigma_0(x_i)$ и $\sigma_t(x_i)$ называются соответственно *внешним* и *внутренним передаточными числами* вершины x_i. Число $\sigma_0(x_i)$ есть сумма элементов строки x_i матрицы, полученной после умножения каждого столбца матрицы расстояний $D(G) = [d(x_i, x_j)]$ на вес соответствующей столбцу вершины, т. е. j-й столбец умножается на v_j, число $\sigma_t(x_i)$ есть сумма элементов столбца x_i матрицы, полученной в результате умножения каждой строки матрицы расстояний $D(G)$ на соответствующий этой строке вес (j-я строка умножается на v_j).

Вершина $\overline{x_0}$, для которой

$$\sigma_0(\overline{x_0}) = \min_{x_i \in X} [\sigma_0(x_i)], \tag{4.18}$$

называется *внешней медианой* графа *G*, а вершина $\overline{x_t}$, для которой

$$\sigma_t(\overline{x_t}) = \min_{x_i \in X}[\sigma_t(x_i)], \tag{4.19}$$

называется *внутренней медианой* графа *G*.

Рассмотрим граф, изображенный на рисунке 4.4, для которого все v_j и c_{ij} приняты равными единице. Вычислим внешние и внутренние передаточные числа вершин. Эти числа приведены в присоединенных к матрице расстояний строке и столбцу.

$D(G) =$

	x_1	x_2	x_3	x_4	x_5	x_6	$\sigma_0(x_i)$ ↓
x_1	0	1	2	3	2	3	11
x_2	3	0	1	2	1	2	9
x_3	4	3	0	1	2	3	13
x_4	3	2	2	0	1	2	10
x_5	2	1	1	2	0	1	7*
x_6	1	2	3	4	3	0	13
$\sigma_t(x_i) \rightarrow$	13	9*	9*	12	9*	11	

По полученным передаточным числам видно, что внешней медианой является x_5 с $\sigma_0(x_5) = 7$ и что существуют три внутренние медианы (x_2, x_3 и x_5), каждая с внутренним передаточным числом, равным 9.

Задача о выборе места для склада. Рассмотрим задачу снабжения ряда потребителей товарами, поступающими с одного склада. Потребителей можно объединить в группы таким образом, чтобы каждую группу обслуживало целое число грузовиков. Машина выезжает со склада, обслуживает некоторую группу потребителей и возвращается на склад. Группы потребителей можно представить вершинами графа, а сеть дорог – его дугами. На практике каждой группе потребителей присваивается вес v_j, представляющий ее «важность» (например, v_j может быть числом, пропорциональным годовому потреблению или частоте, с которой транспорт должен объезжать эту группу потребителей, чтобы удовлетворить их потребности).

В этом случае задача состоит в определении такого места для склада, чтобы общее расстояние, проходимое транспортом, было бы минимально возможным. Если матрица расстояний *D*(*G*) задает действительные расстояния в километрах, то требуемым местом расположения склада будет такая вершина $\overline{x_{0t}}$, для которой сумма внешних и внутренних передаточных чисел будет наименьшей. Вершина $\overline{x_{0t}}$ может быть названа внешне-внутренней медианой и найдена из соотношения, аналогичного (4.11). В следующем разделе будет по-

казано, что для любой точки (вершины или произвольной точки дороги), выбранной для размещения склада, общий километраж, покрываемый транспортом, не может быть меньше, чем для вершины $\overline{x_{0t}}$.

4.8.4 Кратные медианы (*p*-медианы) графа

Обобщая понятие центра, мы ввели понятие *p*-центра. Подобным же образом можно обобщить понятие медианы, определив *p*-медиану.

Пусть X_p – подмножество множества вершин X графа $G = (X, E)$, и предположим, что X_p содержит p вершин. Как и прежде, введем следующие обозначения:

$$d(X_p, x_j) = \min_{x_i \in X_p} [d(x_i, x_j)], \tag{4.20}$$

$$d(x_j, X_p) = \min_{x_i \in X_p} [d(x_j, x_i)], \tag{4.21}$$

Если x_i' – вершина из X_p, на которой достигается минимум в (4.20) или (4.21), то говорят, что вершина x_j прикреплена к x_i'. Передаточные числа множества вершин X_p определяются так же, как и для одиночной вершины.

4.9 Исследование структуры связей в организации

4.9.1 Матрицы достижимостей и контрдостижимостей

Если существует путь, идущий от вершины x_i к вершине x_j, то говорят, что x_j достижима из вершины x_i. Обозначим через $R(x_i)$ множество вершин графа, достижимых из вершины x_i. Определим маршрут достижимостей $R = (r_{ij})$ с элементами

$$r_{ij} = \begin{cases} 1, \text{если } x_j \text{ достижима из } x_i, \\ 0 \text{ в противном случае.} \end{cases}$$

Очевидно, что все диагональные элементы матрицы R равны единице (x_j достижима из x_i).

Пусть через $\Gamma(x_i)$ обозначено множество вершин, которые достижимы за один шаг из x, через $\Gamma^2(x_i)$ – множество вершин, достижимых из x_i с использованием путей длиной 0 + 1 + 2, …; $\Gamma^p(x_i)$ – множество вершин, достижимых из x_i с использованием путей длиной p. Тогда $R(x_i)$ можно представить в виде

$$R(x_i) = \{x_i\} \cup \Gamma(x_i) \cup \Gamma^2(x_i) \cup ... \cup \Gamma^p(x), \tag{4.22}$$

где $\cup$ – операция объединения множеств.

Алгоритм Кристофидеса: множество $R(x_i)$ получается последовательным выполнением (слева направо) операций объединения до тех пор, пока «теку-

щее» множество не перестанет увеличиваться по размеру при очередной операции объединения.

Обозначим через $Q(x_i)$ множество вершин, из которых можно достичь вершину x_i. Определим маршрут контрдостижимостей $Q = (q_{ij})$ с элементами

$$q_{ij} = \begin{cases} 1, \text{если } x_i \text{ достижима из } x_j, \\ 0 \text{ в противном случае.} \end{cases}$$

Пусть через $\Gamma^{-1}(x_i)$ обозначено множество вершин, из которых можно достичь вершину x_i с использованием за один шаг, $\Gamma^{-p}(x_i)$ – множество вершин, из которых можно достичь вершину x_i с использованием путей длиной p. Тогда $Q(x_i)$ можно представить в следующем виде:

$$Q(x_i) = \{x_i\} \cup \Gamma^{-1}(x_i) \cup \Gamma^{-2}(x_i) \cup ... \cup \Gamma^{-p}(x_i), \qquad (4.23)$$

где $\cup$ – операция объединения множеств.

Пример. Построить матрицы достижимостей и контрдостижимостей для графа на рисунке 4.5.

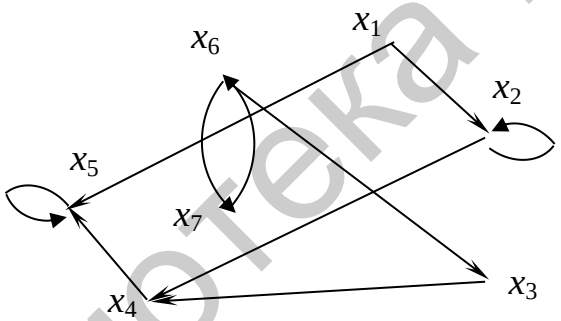


Рисунок 4.5 – Граф для построения матриц достижимостей и контрдостижимостей

Матрица смежности $C = (C_{ij})$.

C	1	2	3	4	5	6	7
1	0	1	0	0	1	0	0
2	0	1	0	1	0	0	0
3	0	0	0	1	0	0	0
4	0	0	0	0	1	0	0
5	0	0	0	0	1	0	0
6	0	0	1	0	0	0	1
7	0	0	0	1	0	1	0

Матрица достижимостей $R = (r_{ij})$.

R	1	2	3	4	5	6	7
1	1	1	0	1	1	0	0
2	0	1	0	1	1	0	0
3	0	0	1	1	1	0	0
4	0	0	0	1	1	0	0
5	0	0	0	0	1	0	0
6	0	0	1	1	1	1	1
7	0	0	1	1	1	1	1

Матрица контрдостижимостей $Q = (q_{ij})$.

Q	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0
2	1	1	0	0	0	0	0
3	0	0	1	0	0	1	1
4	1	1	1	1	0	1	1
5	1	1	1	1	1	1	1
6	0	0	0	0	0	1	1
7	0	0	0	0	0	1	1

4.9.2 Базы, антибазы, сильные компоненты, конденсация графа

Сильной компонентой графа (СК) назовем подграф, в котором любая вершина достижима из другой вершины этого подграфа.

Процедуру поиска сильных компонент графа можно определить с помощью матриц R и Q. Для этого необходимо:

1 Построить матрицу $R \otimes Q$, где $\otimes$ – поэлементное умножение матриц.

2 Путем перестановки строк и столбцов матрицы $R \otimes Q$ привести ее к блочно-диагональному виду. Каждая из диагональных подматриц соответствует сильным компонентам графа.

Определим конденсацию графа G как граф $G^* = (X^*, E^*)$ следующим образом: каждая вершина графа G^* представляет множество вершин некоторой сильной компоненты графа G; дуга (x_i^*, x_j^*) существует в G^* только тогда, когда в графе G существует дуга (x_i, x_j) такая, что $x_i \in x_i^*$, а $x_j \in x_j^*$.

Базой (B) назовем минимальное множество вершин, из которых достижима любая вершина графа, причем никакое собственное подмножество B не обладает такими же свойством:

$$R(B) = \cup R(x_i) = X,\ x_i \in B,$$
$$\forall S \subseteq B, R(S) \neq X. \qquad (4.24)$$

$R(B)$ – множество вершин, достижимых из вершин множества B. Итак, каждая вершина графа достижима хотя бы из одной вершины множества B. Кроме того, в B нет вершины, которая достижима из другой вершины множества B.

Понятием, двойственным понятию база, является антибаза.

Антибаза ($\overline{B}$) есть множество вершин графа таких, что какова бы ни была вершина графа, из нее достижима некоторая вершина из $\overline{B}$:

$$Q(\overline{B}) = \cup Q(x_i) = X,\ x_i \in \overline{B},$$
$$\forall S \subseteq \overline{B}, Q(S) \neq X. \tag{4.25}$$

База графа G^* состоит из таких вершин графа G, полустепени захода которых равны нулю, т. е. ни одна из этих вершин не достижима ни из какой другой вершины графа.

Система связи любой организации может интерпретироваться как граф, в котором люди представлены вершинами, а каналы связи – дугами. Если граф G представляет структуру руководства или влияний некоторой организации, то члены каждой сильной компоненты графа G имеют равную власть или равное влияние друг на друга. Базу графа G можно интерпретировать как «коалицию», включающую наименьшее число лиц, обладающих властью над каждым членом организации.

Один из недостатков определения базы заключается в том, что лицо, не входящее в B, может иметь власть над лицом из B. Этого недостатка можно избежать, если определить «сильную базу» (B_p):

$$R(B_p) = X,$$
$$R(S) \neq X,\ \ \forall S \subseteq B_p, \tag{4.26}$$
$$Q(B_p) = B_p.$$

Последнее условие выражает тот факт, что только лица из B_p могут иметь власть над другими лицами, также принадлежащими B_p.

Сильная база в G есть объединение множеств вершин баз графа G^*, графа конденсации.

Пример. Задача по исследованию структуры организации.

Пусть структура связей (власти, влияния) в организации представлена графом G (рисунок 4.6).

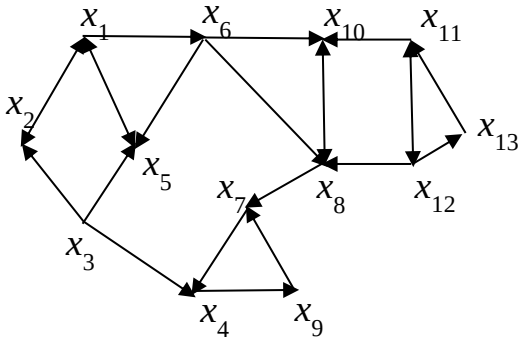


Рисунок 4.6 – Граф структуры связей в организации

Сильными компонентами являются подграфы: $\{x_1, x_2, x_5, x_6\}$, $\{x_4, x_7, x_9\}$, $\{x_8, x_{10}\}$, $\{x_{11}, x_{12}, x_{13}\}$, $\{x_3\}$. В последнем можно убедиться, построив и должным образом преобразовав матрицу графа G:

	x_1	x_2	x_5	x_6	x_8	x_{10}	x_4	x_7	x_9	x_{11}	x_{12}	x_{13}	x_3
x_1	1	1	1	1	0	0	0	0	0	0	0	0	0
x_2	1	1	1	1	0	0	0	0	0	0	0	0	0
x_5	1	1	1	1	0	0	0	0	0	0	0	0	0
x_6	1	1	1	1	0	0	0	0	0	0	0	0	0
x_8	0	0	0	0	1	1	0	0	0	0	0	0	0
x_{10}	0	0	0	0	1	1	0	0	0	0	0	0	0
x_4	0	0	0	0	0	0	1	1	1	0	0	0	0
x_7	0	0	0	0	0	0	1	1	1	0	0	0	0
x_9	0	0	0	0	0	0	1	1	1	0	0	0	0
x_{11}	0	0	0	0	0	0	0	0	0	1	1	1	0
x_{12}	0	0	0	0	0	0	0	0	0	1	1	1	0
x_{13}	0	0	0	0	0	0	0	0	0	1	1	1	0
x_3	0	0	0	0	0	0	0	0	0	0	0	0	1

Конденсацией G^* графа G является граф на рисунке 4.7.

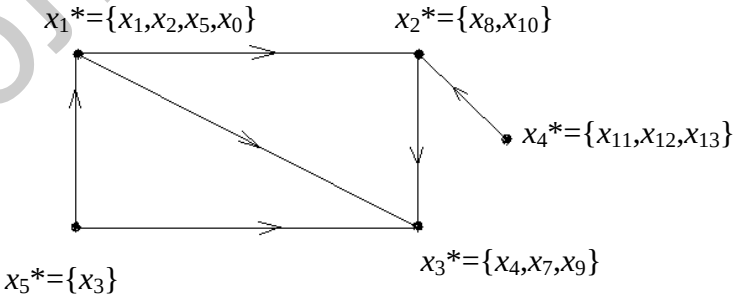


Рисунок 4.7 – Граф конденсации

Базой B^* графа G^* является множество $\{x_4^*, x_5^*\}$. Каждая из этих вершин G^* имеет полустепень захода, равную нулю. Антибазой $\overline{B}^*$ графа G^* является множество $\{x_3^*\}$. Каждая из этих вершин G^* имеет полустепень исхода, равную

нулю. Для графа G базами являются множества: $\{x_3, x_{11}\}$, $\{x_3, x_{12}\}$, $\{x_3, x_{13}\}$, антибазами являются множества: $\{x_4\}$, $\{x_7\}$, $\{x_9\}$. Сильная база B_p графа G имеет следующий вид:

$$B_p = \{x_4^* \cup x_5^*\} = \{x_3, x_{11}, x_{12}, x_{13}\}.$$

Следовательно, вершину x_3 можно рассматривать как руководителя организации, обладающего властью над множеством лиц x_1^*, x_2^*, x_3^*. Множество $x_4^* = \{x_{11}, x_{12}, x_{13}\}$ можно рассматривать как комитет, имеющий власть над двумя множествами лиц x_2^* и x_3^*.

Если рассматривать ограниченную достижимость, т. е. использовать пути ограниченной длины, то можно определить ограниченную базу. В этом случае понятие сильной компоненты и конденсации, упрощающие решение задачи определения баз в случае неограниченной достижимости, теперь использоваться не могут.

В неограниченном случае различные базы имеют одно и то же количество элементов: оно равно числу вершин в конденсации графа с нулевой полустепенью захода. В ограниченном случае разные базы могут иметь различное число элементов и возникает задача нахождения базы с наименьшим числом элементов. Например, пусть достижимость ограничена путями единичной длины (дугами). Тогда ограниченные базы называют минимальными доминирующими множествами вершин графа. Возникает задача поиска наименьшего доминирующего (внешне независимого) множества вершин.

5 Оптимизационные задачи на сетях

5.1 Основные понятия и определения

Под сетью будем понимать пару $(G,C) = S$, где $G = (X,E)$ – произвольный ориентированный граф; C: $E \to R$ – функция, которая каждой дуге (u,v) ставит в соответствие вещественное число $c(u,v) = C_{uv}$, называемое пропускной способность дуги. Множества X и E называются множествами вершин и дуг сети соответственно.

Поток задает способ пересылки некоторых объектов из одной вершины сети в другую по его дугам. Вершина, из которой начинается перемещение объектов, называется источником (s). Вершина, в которой заканчивается перемещение объектов, называется стоком (t). Объекты, которые перемещаются или «протекают» из источника в сток, называются единицами потока. Потоком в сети назовем функцию $\varphi: E \to R$, сопоставляющую каждой дуге (u,v) сети вещественное число $\varphi(u,v)$ и обладающую следующими свойствами:

1 Поток по любой дуге неотрицателен и не превышает пропускную способность дуги:

$$0 \le \varphi(u,v) \le c(u,v). \tag{5.1}$$

2 В любой вершине $v \notin s,t$ поток не может возникнуть и накопиться:

$$\sum_{u:u\to v} \varphi(u,v) = \sum_{u:v\to u} \varphi(u,v). \tag{5.2}$$

3 Количество потока, вытекающего из источника, равно количеству потока, втекающего в сток, и равно величине потока w:

$$\sum_{u:s\to v} \varphi(s,v) = \sum_{u:v\to t} \varphi(v,t) = w. \tag{5.3}$$

Дуга (u,v) называется насыщенной, если поток по этой дуге равен ее пропускной способности, т. е. $\varphi(u,v) = c(u,v)$. Путь из s в t называется насыщенным, если содержит хотя бы одну насыщенную дугу. Поток, насыщающий все пути (цепи) из s в t, называется полным. Наибольший из полных потоков называется максимальным.

Под разрезом $R(A)$ сети S, соответствующим множеству $A \subset X$, будем понимать множество дуг $(u,v) \in E$ таких, что $u \in A, \mathrm{v} \in \mathrm{X} \setminus \mathrm{A}$, т. е. $R(A) = E \cap (A \times (X \setminus A))$.

Пропускная способность разреза $R(A)$ равна сумме пропускных способностей дуг, входящих в этот разрез:

$$C(A, X \setminus A) = \sum_{(u,v)\in R(A)} c(u,v) \cdot \tag{5.4}$$

Разрез с минимальной пропускной способностью называется минимальным разрезом.

Одним из фундаментальных положений теории потоков в сетях является классическая теорема Форда и Фалкерсона о максимальном потоке и минимальном разрезе: величина каждого потока из s в t не превосходит пропускную способность минимального разреза, разделяющего s в t, причем существует поток, достигающий этого значения, и он называется максимальным.

Все известные алгоритмы построения максимального потока основываются на последовательном увеличения потока вдоль цепи из s в t.

Будем говорить, что дуга $e = (u,v)$ сети S является допустимой дугой из u в v относительно потока φ, если:

а) $\varphi(e) < c(e)$, при этом дуга $e = (u,v)$ называется согласованной допустимой (прямой) из u в v;

б) $\varphi(e) > 0$, при этом дуга $e = (u,v)$ называется несогласованной допустимой (обратной) из u в v.

Увеличивающей цепью длиной l для данного потока φ из s в t называется произвольная последовательность попарно различных вершин и дуг:

$$x_0, l_1, x_1, l_2, \ldots, l_l, x_l,$$

такая, что $x_0 = s$, $x_l = t$ и для каждого $i \leq l$ дуга l_i допустима из x_{i-1} в x_i относительно потока φ.

Метод решения задачи о максимальном потоке от s в t был предложен Фордом и Фалкерсоном, и их техника пометок для выделения увеличивающей цепи из s в t составляет основу других алгоритмов решения многочисленных задач, являющихся обобщениями или расширениями указанной задачи. Перечислим некоторые варианты задач о максимальном потоке [2]:

1 Задача нахождения допустимого потока минимальной стоимости от s в t возникает тогда, когда каждой дуге сети приписана не только пропускная способность C_{ij}, дающая верхнюю границу потока через дугу (x_i,x_j), но и пропускная способность r_{ij}, дающая нижнюю границу потока через дугу. Кроме того, заданы стоимости единицы потока, протекающего по каждой дуге.

2 Задача нахождения максимального потока между каждой парой вершин обычно не решается как $n(n-1)/2$ отдельных задач о максимальном потоке от s к t, а требует для своего решения специального метода, как и в случае поиска кратчайших путей между парами вершин.

3 Задача о многопродуктовом потоке предусматривает вместо одного источника и одного стока наличие нескольких заданных источников и стоков. Между различными источниками и стоками протекают потоки различных продуктов. Пропускная способность q_{ij} дуги (x_i,x_j) является ограничением для суммы всех потоков всех видов продуктов через эту дугу. Необходимо максимизировать сумму всех потоков между источниками и стоками.

4 Задача о потоках с выигрышами есть такая модификация задачи о максимальном потоке из s в t, когда допускается, что потоки могут и «порождаться», и «поглощаться» самим графом сети, так что поток, входящий в t, и поток, покидающий s, могут изменяться совершенно независимо. Например, выходной поток дуги равен ее сходному потоку, умноженному на некоторое неотрицательное число.

5.2 Задача о максимальном потоке и алгоритм Форда – Фалкерсона

Рассмотрим сеть $S = <G,C>$, $G = (X,E)$ с пропускными способностями дуг C_{ij}, источником s, стоком t; $s, t \in X$. Поток сети $\varphi: E \to R$ удовлетворяет соотношениям (5.1)–(5.3). Задача состоит в нахождении такого множества потоков по дугам, чтобы величина (5.3) была максимальной.

Идея, лежащая в основе алгоритма Форда и Фалкерсона, довольно проста и состоит в следующем [7].

Выбирается некоторый допустимый начальный поток и осуществляется поиск увеличивающей цели. Если он оказывается успешным, то поток вдоль найденной цены увеличивается до максимально возможного значения. Затем выполняется поиск новой увеличивающей цепи и т. д. Если на каком-то этапе этой процедуры не удается найти увеличивающей цепи из *s* в *t*, выполнение алгоритма заканчивается: текущий поток из *s* в *t* является максимальным.

Удобный способ реализации процедуры увеличения потока сети состоит в назначении каждой вершине сети меток. Метка вершины x_j может иметь вид (i^+, ε_j) или (i^-, ε_j), где i – номер предшествующей вершины ненасыщенного пути из *s* в x_j, а ε_j – число, показывающее величину возможного приращения (знак «+») потока по пути из *s* в x_j или уменьшения (знак «–») потока. Метка вершины s всегда имеет вид $(0, \infty)$.

Алгоритм Форда и Фалкерсона.

Алгоритм состоит из подготовительного этапа (распределение допустимого начального потока w_0) и конечного числа итераций. При реализации подготовительного этапа каждой дуге (x_i, x_j) помимо ее пропускной способности C_{ij} приписывается число $\varphi_0(x_i, x_j)$, удовлетворяющее (5.1)–(5.3) при $w = w_0$.

Описание *k*-й итерации (перед началом итерации поток в сети равен w_k):

1 Определение множества *A* вершин, которые лежат на ненасыщенных путях с начальной вершиной *s*. Выбираем любую помеченную вершину $x_i \in A$. Рассмотрим все непомеченные вершины x_j, связанные с x_i дугой. Вершинам x_j, для которых $\varphi_{ij} < C_{ij}$, назначается метка (i^+, ε_j), где $\varepsilon_j = \min(\varepsilon_j, \varphi_{ij})$. Тем вершинам x_j, для которых величина потока по несогласованной допустимой (обратной) дуге (x_j, x_i) больше нуля, т. е. $\varphi_{ij} > 0$, назначается метка (i^-, ε_j), где $\varepsilon_j = \min(\varepsilon_j, \varphi_{ij})$. Шаг 1 повторяется до тех пор, пока не удается пометить вершину *t* или пока не удается сделать никаких новых пометок. В первом случае существует увеличивающая цепь из *s* в *t* и поток w_k по этой цепи можно увеличить на величину ε_t, т. е. $w_{k+1} = w_k + \varepsilon_t$. После этого перейти к шагу 2 алгоритма. Во втором случае текущий поток максимален и задача решена.

2 Перераспределение потока $w_{k+1} = w_k + \varepsilon_t$ по сети. Для помеченных вершин x_i и x_j, связанных дугой (x_i, x_j), выполнить: по каждой согласованной дуге поток увеличить $(\varphi_{ij} + \varepsilon_t)$, по каждой несогласованной дуге поток уменьшить $(\varphi_{ij} - \varepsilon_t)$. Эту процедуру выполнять, начиная с вершины *t* и повторять до тех пор, пока не будет достигнута вершина *s*. По достижении вершины *s* все старые метки «стираются» и переходят к новой итерации.

5.3 Разновидности задачи о максимальном потоке в сети

Задача о максимальном потоке в сети при наличии нескольких источников и стоков.

Рассмотрим сеть с n_s источниками и n_t стоками и предположим, что поток может идти от любого источника к любому стоку. Задачу нахождения максимального общего потока от всех источников ко всем стокам можно преобразовать в задачу о максимальном потоке от s к t путем добавления нового искусственного источника $\tilde{s}$ и нового искусственного стока $\tilde{t}$ с добавлением дуг, ведущих от $\tilde{s}$ к каждому истинному источнику и от истинного стока к $\tilde{t}$.

Пропускные способности дуг, ведущих от $\tilde{s}$ к источникам, могут быть выбраны равными бесконечности или, в случае когда производительность источника S_k ограничена, пропускная способность дуги $(\tilde{S}, S_k)$ может быть выбрана равной этой границе. Аналогично пропускные способности дуг, ведущих от стоков к t, могут быть ограничены некоторой величиной (зависящей от стоков) или положены равными бесконечности, если такой границы нет.

Если в задаче некоторые стоки должны снабжаться определенными источниками и наоборот, то задача уже не будет разновидностью задачи о максимальном потоке от s к t, а становится задачей о многопродуктовом потоке.

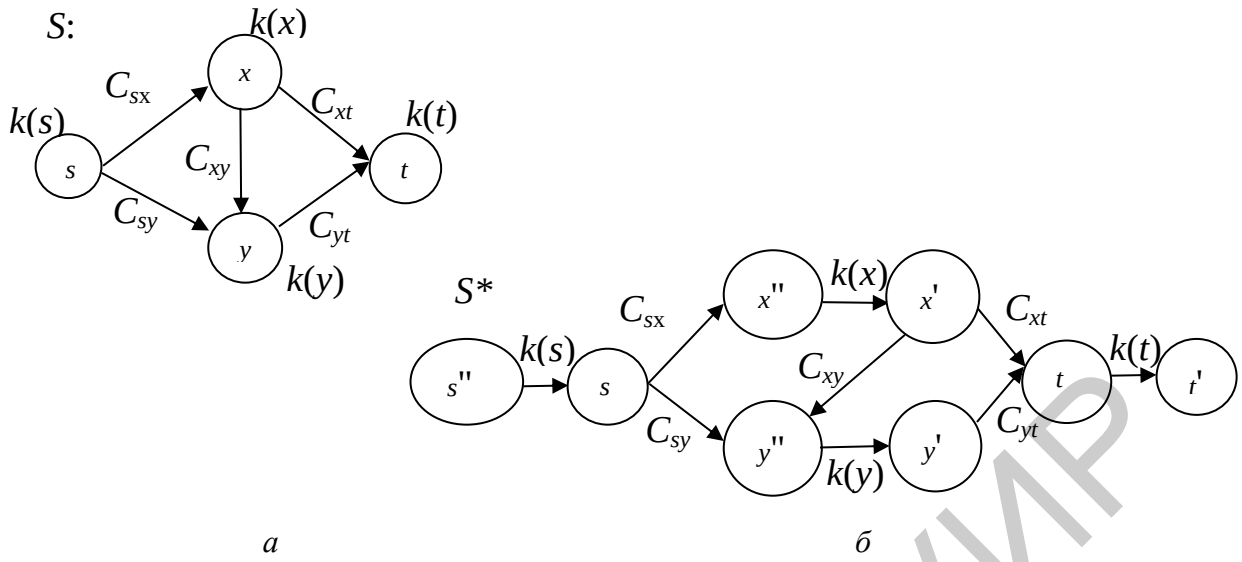
Задача о максимальном потоке в сети при наличии заданных пропускных способностей дуг и вершин.

Пусть, каждый узел (вершина) x сети S имеет пропускную способность $q(x) \geq 0$ и нужно найти максимальный поток от s в t, не превышающий пропускных способностей дуг $\{C_{x,y} = C(x,y)\}$ и узлов $\{q(x)\}$. Чтобы свести эту задачу к уже известной нам задаче о максимальном потоке от s в t, надо построить для сети S новую сеть S^* следующим образом :

- каждому узлу $x \in X$ ставим в соответствие два узла x', $x'' \in X^*$, где X^* – множество вершин сети S^*;
- если $x,y \in E$, то $x',y'' \in E^*$ и $x'',x' \in E^*$ для каждого узла из E. Здесь E – множество дуг сети S, E^* – множество дуг сети S^*;
- пропускные способности C^* дуг множества E^* определяются следующими формулами:

$$\begin{aligned} C^*(x', y'') &= C(x, y), (x, y) \in E, \\ C^*(x'', x') &= k(x), x \in X. \end{aligned} \tag{5.5}$$

На рисунке 5.1, *а* приведен пример сети S с пропускными способностями дуг и вершин, а на рисунке 5.1, *б* представлена эквивалентная ей сеть S^*, в которой пропускные способности имеют только дуги.



а *б*

а – сеть с пропускными способностями дуг и вершин; *б* – сеть с пропускными способностями дуг

Рисунок 5.1 – Сеть с пропускными способностями дуг и вершин и сеть с пропускными способностями дуг

Следует заметить, что если минимальный разрез в сети S^* не содержит дуги вида (x'',x'), то пропускные способности вершин в S пассивны и излишни. Если же минимальный разрез в S^* содержит такие дуги, то соответствующие вершины в S насыщены полученным максимальным потоком.

Задача о наименьшем потоке.

Для решения задачи о наименьшем потоке в заданной сети S, т. е. потоке, удовлетворяющем условию $\varphi(u,v) \geq c(u,v)$, $\forall (u,v) \in E$, можно применить следующую процедуру:

1 Найти поток φ_1 такой, что $\varphi_1(u,v) \geq c(u,v)$ для всех дуг (u,v), последовательно перемещая единицы продукта к дугам, наиболее далеким от насыщенных.

2 Полагая, что $c_1(u,v) = \varphi_1(u,v) - c(u,v)$, построить при помощи алгоритма Форда – Фалкерсона такой максимальный поток φ_2, что $\varphi_2(u,v) \geq c_1(u,v)$ для всех $\forall (u,v) \in E$.

3 Поскольку $\varphi_1(u,v) - \varphi_2(u,v) \geq \varphi_1(u,v) - c_1(u,v) = c(u,v)$, то разность $\varphi = \varphi_1 - \varphi_2$ и есть искомый наименьший поток.

Пример. Рассмотрим задачу компании «Новые просторы» как задачу поиска максимального пути на сетевой модели [2].

Строительная компания разрабатывает план капиталовложений на текущий год. Общий капитал, которым она располагает и который нужно распределить по различным объектам, составляет C сотен тыс. дол. США. Рассматривается n возможных объектов капиталовложений. Компания может вложить свои средства в любые из этих объектов. Единственным ограничением является объем наличного капитала. Минимальный объем капиталовложений, необходимый

для приобретения объекта i, составляет p_i. Компания оценивает соответствующую приведенную к данному моменту прибыль величиной $V_{i,0}$. Однако если в этот объект вложить дополнительно k сотен тыс. дол. США, то в результате приведенная прибыль будет оцениваться величиной $V_{i,k}$ ($V_{i,k+1} > V_{i,k}$). Компания стремится так распределить капиталовложения по объектам, чтобы максимизировать общую приведенную к исходному моменту прибыль.

Приведем математическую формализацию этой задачи и покажем, как ее можно свести к поиску пути максимальной длины на некоторой сети. Пусть x_i – объем капитала, вложенного в объект i, определим неубывающую функцию:

$$h_i(x_i) = \begin{cases} 0, \text{если } x_i < p_i, \\ V_{ik}, \text{если } x_i = p_i + k, k = 0, 1, 2, ... \end{cases} \quad (5.6)$$

Формализованная постановка задачи:

$$\max \sum_{i=1}^{n} h_i(x_i), \quad (5.7)$$

$$\sum_{i=1}^{n} x_i < C, x_i = 0, 1, 2, ...$$

Эту задачу можно интерпретировать как задачу о максимальном пути на некоторой сети.

Сеть строится следующим образом. Для каждого объекта капиталовложений вводится столбец узлов. В обозначении узла (i, C) символ i относится к номеру объекта; величина C определяет объем капитала, который можно вложить в объект i при условии принятия определенных решений относительно вложений в уже рассмотренные объекты. Каждая дуга, исходящая из узла (x, C), соответствует конкретному решению по объекту i. Дуга входит в узел (i+1;C'), где C' вычисляется как имеющийся капитал, который можно вложить в объект i+1 при условии принятия вполне определенного решения по вложениям в объект i.

Рассмотрим построение сети на примере (рисунок 5.2).

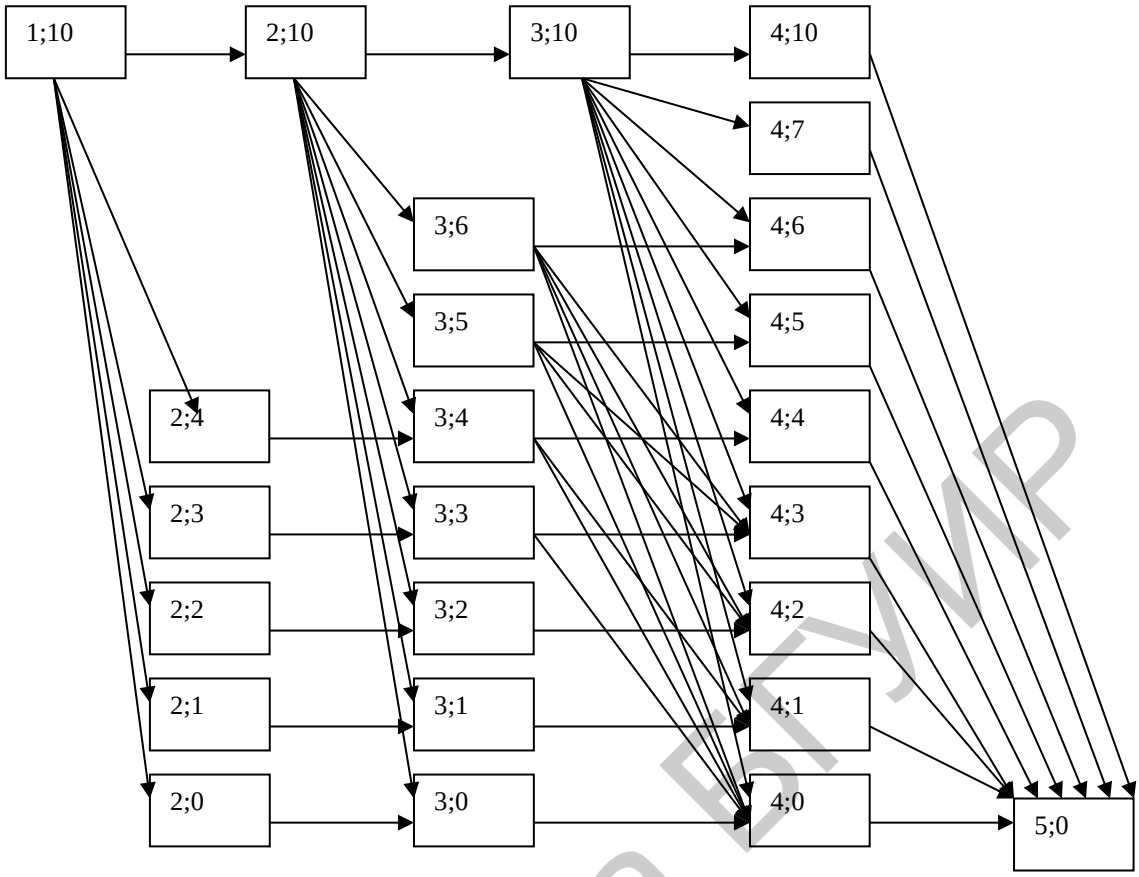


Рисунок 5.2 – План капиталовложений

Пусть $C = 10$, $n = 4$, а минимальные объемы капиталовложений составляют $p_1 = 6$, $p_2 = 4$, $p_3 = 3$, $p_4 = 1$. Пусть дополнительные капиталовложения можно распределять в объемах, кратных 1 тыс. дол. США ($k = 1, 2,\ldots$).

Рассмотрим для примера узел (3;4). Если компания приняла решение вложить минимальный объем капитала в объект 1 и не вкладывать ничего в объект 2 или не вкладывать ничего в объект 1 и вложить $p_2 + 2 = 4 + 2 = 6$ сотен тыс. дол. США в объект 2, то 400 тыс. дол. США можно вложить в объект 3. Эти 2 решения отображаются дугами, входящими в узел (3;4).

Располагая 400 тыс. дол. США, компания может принять одно из следующих решений:

- не вкладывать ничего в объект 3;
- вложить в этот объект минимальный объем капитала $p_3 = 3$;
- вложить в этот объект объем капитала $p_3 + 1 = 4$, что исчерпывает возможность дальнейших вложений.

Эти 3 возможных решения отображаются дугами, исходящими из узла (3;4). Любой допустимый вариант распределения капиталовложений по объектам можно представить в виде проходящего через всю сеть «пути», который начинается в узле (1;10) и заканчивается в узле (5;0). Это означает, что весь наличный объем капитала должен быть полностью распределен между четырьмя объектами.

В этой задаче длине дуги соответствует приведенная прибыль $V_{i,k}$ решения, отображаемого этой дугой. Задача состоит в отыскании в сети пути максимальной длины. Решить эту задачу можно методом динамического программирования как «задачу о дилижансе».

5.4 Задача о потоке минимальной стоимости и ее решение

Пусть задана сеть $S = (G,C)$, в которой каждая дуга (x,y) имеет два связанных с ней числа – пропускную способность $c(x,y)$ и стоимость $a(x,y)$ прохождения единицы потока по этой дуге. Обозначим через $\varphi(x,y) \geq 0$ количество единиц потока, проходящих по дуге (x,y). Обозначим через w количество единиц потока, которое нужно переслать из s в t, т. е. величину потока.

Тогда задача о потоке минимальной стоимости может быть представлена следующей оптимизационной задачей:

$$\min\{\sum_{(x,y)\in E} a(x,y)\cdot\varphi(x,y)\} \tag{5.8}$$

при ограничениях

$$w = \sum\varphi(s,x) = \sum\varphi(x,t), \tag{5.9}$$

$$\sum_{x:x\to y}\varphi(x,y) = \sum_{z:y\to z}\varphi(y,z), \tag{5.10}$$

$$0 \leq \varphi(x,y) \leq c(x,y) \text{ для всех } x,y \in X. \tag{5.11}$$

Сумма, входящая в (5.8), представляет собой общую стоимость потока.

Пусть p – достаточно большое число, тогда (5.8) можно заменить соотношением

$$\max\{pw - \sum_{(x,y)\in E} a(x,y)\cdot\varphi(x,y)\}. \tag{5.12}$$

Если интерпретировать p как доход, полученный от пересылки одной единицы потока от s к t, то (5.12) можно рассматривать как максимально возможный чистый доход от пересылки w единиц потока из s в t за вычетом стоимости их пересылки через сеть. Очевидно, что любой поток, доставляющий максимум в (5.12), одновременно обеспечивает минимум в (5.8). Справедливо и обратное утверждение.

Алгоритм поиска потока минимальной стоимости [9] выполняет следующие действия. Сначала из s в t пересылается как можно больше единиц потока, полная стоимость прохождения по сети каждой из которых равна нулю. Затем из s в t пересылается как можно больше единиц потока, полная стоимость прохождения каждой из которых по сети равна единице и т. д. Работа алгоритма заканчивается либо тогда, когда через сеть удается пропустить заданное число

w единиц потока, либо когда ни одной дополнительной единицы потока по сети пропустить уже нельзя. Другими словами, в алгоритме многократно решается задача, задаваемая соотношениями (5.9)–(5.12) сначала для $p = 0$, затем для $p = 1$, $p = 2$ и т. д.

Способ пересылки новых единиц потока, имеющих общую стоимость прохождения в p единиц, реализуется путем выявления увеличивающей цепи от s к t, стоимость прохождения по которой единицы потока составляет величину p. Этот поиск осуществляется с помощью приписывания каждой вершине x целого числа $p(x)$. Числа выбираются так, чтобы $p(s) = 0$, $p(t) = p$, а для всех вершин x, отличных от стока и источника, выполняется неравенство $0 \le p(x) \le p$. В этой цепи из суммы стоимостей для прямых дуг вычитается сумма стоимостей обратных дуг, и эта разность равна p. Изменение потока допускается только в тех дугах, для которых $p(y) - p(x) = a(x,y)$.

Описание алгоритма:

1 Задание начальных условий: $\varphi(x, y) = 0$; $p(x) = 0$ для $x = s$.

2 Выявление дуг, в которых допускается изменение потока (поиск увеличивающей цепи). На данном шаге определяется множество дуг I, R, N. Здесь I – множество дуг, для которых $p(y) - p(x) = a(x,y)$ и $\varphi(x, y) < c(x,y)$; R – множество дуг, для которых $p(y) - p(x) = a(x,y)$ и $0 < \varphi(x, y)$; N – множество дуг, не вошедших в I и R.

3 Изменение потока в сети.

Применить шаг 2 алгоритма Форда – Фалкерсона к исходной сети при найденных на шаге 2 распределениях ее дуг по множествам I, R, N.

Закончить выполнение алгоритма: а) либо тогда, когда из s в t уже передано w единиц потока (поток имеет минимальную стоимость при заданной величине w); б) либо тогда, когда выясняется, что при текущем разбиении дуг на множество I, R, N полученный поток максимален.

Для проверки последнего утверждения исследуется разрез

$$A = \{(x,y),\ x \in I \cup R,\ y \in X \backslash A\}.$$

Если разрез является насыщенным, то данный поток является максимальным. В этом случае поток имеет минимальную общую стоимость прохождения по сети. Если не выполняются условия окончания алгоритма, перейти к шагу 4.

4 Изменение вершинных чисел $p(x)$. Исходной информацией при реализации этого шага являются результаты распределения меток вершин при поиске увеличивающей цепи. Вершины множества I приобретают метки вида (i^+, ξ_j); вершины множества R – метки вида (i^-, ξ_j).

Для всех непомеченных вершин увеличить вершинное число $p(x)$ на единицу, т. е. $p(x) = p(x) + 1$. При этом обязательно увеличение вершинного числа $p(t)$ стока, т. к. вершина t не является помеченной. Затем перейти к шагу 2.

Применим описанный алгоритм для поиска потока минимальной стоимости на графе, изображенном на рисунке 5.3.

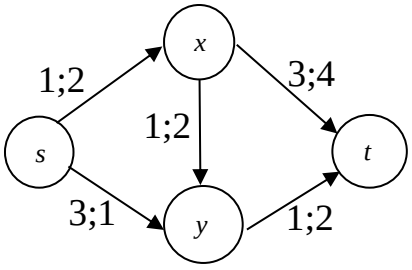


Рисунок 5.3 – Сеть с заданными стоимостями и пропускными способностями дуг

Первое число у каждой дуги представляет собой стоимость, а второе – ее пропускную способность (a;c).

Результаты выполнения алгоритма сведены в таблицу 5.1. Итерация данного алгоритма включает применение к соответствующей сети алгоритма поиска увеличивающей цепи.

Таблица 5.1 – Первый шаг поиска увеличивающей цепи

Итерация	$P(s)$	$P(x)$	$P(y)$	$P(t)$	Помеченные дуги	Помеченные вершины
0	0	0	0	0	–	s
1	0	1	1	1	(s,x)	s, x
2	0	1	2	2	(s,x), (x,y)	s, x, y
3	0	1	2	3	(s,x), (x,y), (y,t)	s, x, y, t

Первоначально все вершинные числа $p(x)$ равны нулю, все дуги являются непомеченными, все вершины за исключением вершины s тоже являются непомеченными.

После третьей итерации вершина t оказалась помеченной. Посылаем три единицы потока из s в t по пути (s,x), (x,y), (y,t). В результате $\varphi(s,x) = 3$, $\varphi(x,y) = 3$, $\varphi(y,t) = 3$. Продолжаем итерации алгоритма, предварительно убрав пометки у дуг и вершин, кроме вершины s. Результаты повторного выполнения алгоритма сведены в таблицу 5.2.

Таблица 5.2 – Второй шаг поиска увеличивающей цепи

Итерация	$P(s)$	$P(x)$	$P(y)$	$P(t)$	Помеченные дуги	Помеченные вершины
4	0	1	2	3	–	s
5	0	2	3	4	(s,y), (x,y)	s, x, y
6	0	2	3	5	(s,x), (x,y), (x,t)	s, x, y, t

После шестой итерации вершина t снова оказалась окрашенной. Посылаем две единицы потока из s в t вдоль цепи (s,y), (x,y), (x,t). В результате $\varphi(s,x) = 3$, $\varphi(s,y) = 2$, $\varphi(x,y) = 3 - 2 = 1$, $\varphi(x,t) = 2$, $\varphi(y,t) = 3$. Продолжаем итерации алгоритма. Результаты третьего шага выполнения алгоритма сведены в таблицу 5.3.

Таблица 5.3 – Третий шаг поиска увеличивающей цепи

7	0	2	3	5	–	*s*

После седьмой итерации оказалось, что ни одна дополнительная единица потока не может быть передана из *s* в *t*, т. к. дуги, ведущие из помеченных вершин (*s*) в непомеченные, а именно дуги (*s*,*x*), (*x*,*y*) являются насыщенными.

На рисунке 5.4 первое число у каждой дуги представляет собой стоимость, второе – пропускную способность дуги, третье – величину потока.

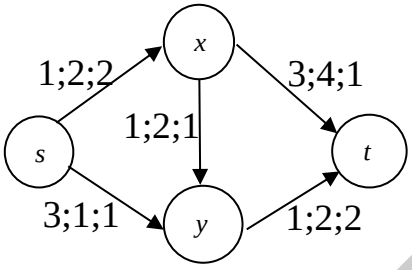


Рисунок 5.4 – Распределение потока минимальной стоимости

Суммарная стоимость потока по сети равна

$$\sum_{(x,y)} a(x,y)\varphi(x,y) = 11\cdot$$

Данный алгоритм, предложенный в [3], позволяет решить задачу только при целочисленных значениях параметра *p* и изменить вершинные числа соответствующих вершин ровно на +1. Однако авторы предложили модификацию, которая позволила использовать его и в том случае, когда стоимости, приписанные дугам, не являются целыми числами.

5.5 Сетевая модель с промежуточными пунктами

Транспортная задача (или задача прикрепления поставщика к потребителям) явилась одним из первых примеров оптимизации на линейных сетях. Сеть транспортной задачи имеет следующий вид:

$$\begin{gathered} \min \sum\sum c_{ij} x_{ij}, \\ \sum_i x_{ij} \le S_i, i = 1, ..., m, \\ \sum_i x_{ij} \ge D_j, j = 1, ..., n, \\ x_{ij} \ge 0, \end{gathered} \qquad (5.13)$$

где S_i – наличные ресурсы;

D_j – спрос потребителей.

Одно из практически важных обобщений классической транспортной задачи связано с учетом возможности доставки товара от *i*-го источника к *j*-му стоку по маршруту, проходящему через некоторый промежуточный пункт (склад). Такие пункты часто встречаются в распределительных системах снабжения компаний, имеющих универсальные магазины во многих городах. Как правило, такие компании имеют зональные оптовые базы, которые снабжают товарами более мелкие региональные склады, откуда товары в свою очередь поступают в розничную торговую сеть. Модель с промежуточными пунктами является эффективным средством выбора оптимального числа и географического размещения складов, поскольку анализ сетевой модели позволяет отыскивать план перевозок товаров, минимизирующий транспортные затраты при любой заданной схеме размещения складов. Рассмотрим пример такой задачи.

Пример. *Задача об оптовых складах компании «Универсал»* [2].

Компания имеет 8 крупных оптовых складов, размещенных в нескольких районах. Отдел сбыта принял решение значительно снизить цену одного дорогостоящего изделия с целью ликвидации образовавшегося запаса этих изделий. Перед началом рекламной кампании руководство намерено разместить имеющиеся в наличии запасы на указанных складах в соответствии с прогнозами сбыта в этих районах. Для осуществления этого плана необходимо перераспределить некоторую часть запасов при минимальных общих транспортных затратах.

На рисунке 5.5 представлена сеть маршрутов перевозок для задачи фирмы «Универсал», где $c_{ij} \geq 0$ соответствует затратам на перевозку одного изделия по указанному маршруту.

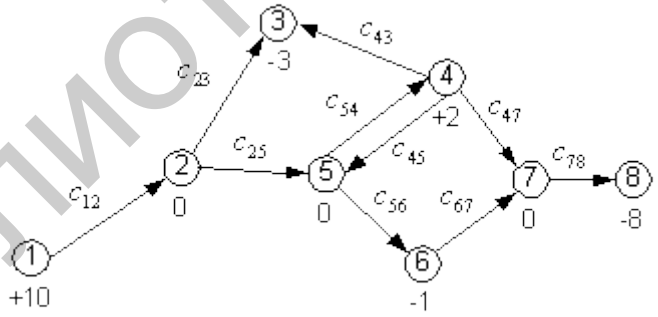


Рисунок 5.5 – Сеть маршрутов перевозок фирмы «Универсал»

Узлы сети соответствуют 8 складам. Положительное число у номера склада обозначает избыточное количество запасов, которое должно быть распределено в системе. Отрицательное число обозначает потребность данного склада в дополнительных запасах.

Изделие может транспортироваться через склады 2, 4, 5, 6 и 7, в связи с чем эти пункты называются промежуточными. Все другие склады называются источниками, если в них имеется избыток запасов и в них не входит ни одна

дуга (склад 1), и стоками, если требуется им дополнительный запас и из данного узла не выходит ни одна дуга (склады 3 и 8).

От задачи с промежуточными пунктами можно перейти к классической транспортной задаче, в которой поиск оптимального плана перевозок методом потенциалов начинается с определения начального допустимого плана перевозок (маршрута перевозок). Рассмотрим один из способов этого перехода.

Строится матрица задачи с промежуточными пунктами для заданной сетевой модели и на этой матрице отыскивается допустимый план перевозок. Построение матрицы осуществляется в следующем порядке:

1 Выделить строку для каждого источника и задать значение S_i как количество поставленных из него изделий.

2 Выделить столбец для каждого стока и задать значение D_j, определяемое потребностью в изделиях.

3 Выделить строку и столбец для каждого промежуточного пункта. Пусть T_k – чистый запас рассматриваемого пункта. Примем, что для любого пункта k выполняется соотношение

$$S_k = T_k + B \text{ и } D_k = B, \qquad (5.14)$$

где B – суммарные запасы, имеющиеся во всех пунктах.

4 Ввести величины x_{ij} $(i \neq j)$, только для дуг, существующих в исходной сети. Для промежуточного пункта k ввести y_{kk} при $c_{kk} = 0$.

Величина B имеет смысл фиктивного буферного запаса в каждом промежуточном пункте.

Матрица задачи с промежуточными пунктами приведена в таблице 5.4.

Таблица 5.4 – Матрица задачи с промежуточными пунктами

	2	3	4	5	6	7	8	Поставки S_i
1	c_{12} x_{12}							10
2	0 x_{22}	c_{23} x_{23}		c_{25} x_{25}				12 = B
4		c_{43} x_{43}	0 x_{44}	c_{45} x_{45}				14 = $B + T_4$
5			c_{54} x_{54}	0 x_{55}	c_{56} x_{56}			12 = B
6					0 x_{66}	c_{67} x_{67}		11 = $B - T_6$
7						0 x_{77}	c_{78} x_{78}	12
Спрос D_j	12	3	12	12	12	12	8	

Допустимый план перевозок представлен в таблице 5.5.

Таблица 5.5 – Допустимый план перевозок

	2	3	4	5	6	7	8	S_i
1	c_{12} 10							10*
2	0 2	c_{23} 3		c_{25} 7				12
4		c_{43}	0 6	c_{45}		c_{47} 8		14
5			c_{54} 6	0 5	c_{56} 1			12
6					0 11	c_{67}		11
7						0 4	c_{78} 8	12
D_j	12	3*	12	12	12	12	8*	

Разберем процесс заполнения таблицы 5.5 подробнее.

Из пункта 1 вывозится 10 ед. изделий через пункт 2. Пункт 2 является промежуточным, в нем присутствует как избыток запасов, так и потребность в них. Поскольку в решение уже входит величина $x_{12} = 10$, необходимо принять $x_{22} = 2$, чтобы выполнялось условие $D_2 = 12$. Это в свою очередь означает, что 10 изделий S_2 можно отгрузить из этого пункта. Они направляются в пункт 3, являющийся стоком, и пункт 5 – промежуточный. Итак, 10 изделий из пункта 1 отгружаются в пункт 3 и пункт 5 и перевозятся через промежуточный пункт 2.

Рассматривая пункт 5, можно увидеть, что 7 изделий, поступающих в этот пункт, отгружаются затем в пункт 4 и пункт 6. В частности, потребность пункта 6 удовлетворяется за счет пункта 5 и далее не отправляется, а остается в этом пункте ($S_6 = 11$, $D_6 = 12$). При этом необходимо положить $x_{66} = 11$.

В пункте 4 был избыток изделий (+2), из пункта 5 в пункт 4 было отправлено 6 ед. изделий. Пункт 4 отправляет этот избыток в пункт 8 через пункт 7. Кроме того, через него 6 ед. изделий в пункт 8 отправляет пункт 1, при этом используя промежуточные пункты 2, 5 и 7.

Итак, мы получили допустимый план перевозки транспортной задачи. Далее начинается этап его оптимизации – методом потенциалов.

6 Игровые модели исследования операций

Задачи, рассмотренные нами ранее, формулировались для ситуаций индивидуального выбора оптимальных решений, т. е. для случая, когда решение принимает отдельно взятый субъект, обладающий единственной целью.

Принципиально иная ситуация возникает при изучении процессов принятия решений несколькими субъектами, интересы которых могут не совпадать, а результаты любого действия каждой из сторон зависят от мероприятий партнера.

Для грамотного решения задач принятия решений в конфликтной ситуации разработана математическая теория конфликтных ситуаций, называемая теорией игр.

Математическая модель конфликтной ситуации называется игрой; стороны, участвующие в конфликтной ситуации называются игроками; исход конфликта – выигрышем; стратегия игрока – его правила действия в каждой из возможных ситуаций игры.

В теории игр не существует установившейся классификации видов игр. Однако по определенным критериям можно выделить некоторые виды игр (таблица 6.1).

Таблица 6.1 – Классификация игр

Классификатор	Виды игр
Количество игроков	Игра двух лиц (парная игра)
	Игра *n* лиц (множественная игра)
Количество стратегий	Конечная игра
	Бесконечная игра
Взаимоотношения сторон	Кооперативные (коалиции заранее определены)
	Коалиционные (игроки могут вступать в коалиции)
	Бескоалиционные
Характер выигрыша	Игра с нулевой суммой (сумма выигрышей всех игроков в каждой партии равна нулю)
	Игра с ненулевой суммой
Вид функции выигрыша	Матричные
	Биматричные
	Выпуклые и т. д.
Количество ходов	Одношаговые игры (игра заканчивается после одного хода каждого игрока)
	Многошаговые (позиционные, стохастические, дифференциальные)
Информированность сторон	Игра с полной информацией
	Игра с неполной информацией
Степень неполноты информации	Стратегические (в условиях полной неопределенности)
	Статистические (в условиях частичной неопределенности)

Игры с природой относят к статистическим играм. В статистической игре имеется возможность получения информации на основе статистического эксперимента, при котором вычисляется распределение вероятностей состояний (стратегий) природы. С теорией статистических игр тесно связана теория принятия экономических решений.

6.1 Игры с нулевой суммой и их решение

Рассмотрим конечную игру двух лиц с нулевой суммой (так называемую антагонистическую игру двух лиц). Удобным способом задания таких игр является платежная матрица $A = (a_{ij})$, где a_{ij} – выигрыш игрока I (проигрыш игрока II), если I игрок применяет стратегию i, а второй игрок – стратегию j

($i = 1, \ldots, m$, m – число чистых стратегий игрока I; $j = 1, \ldots, n$, n – число чистых стратегий игрока II). В подобных играх каждый игрок считает своего противника разумным соперником и при этом стремится к выбору наилучшей для себя стратегии.

I игрок должен получить максимальный гарантированный результат при наихудших условиях, т. е. придерживаться своей максимальной стратегии x_0, т. е. такой, которая гарантирует ему чистую нижнюю цену игры («максимин»):

$$v_1 = \max_i (\min_j a_{ij}). \qquad (6.1)$$

II игрок стремится уменьшить выигрыш игрока I и поэтому должен придерживаться своей минимальной стратегии y_0, т. е. такой стратегии, которая гарантирует ему чистую верхнюю цену игры («минимакс»):

$$v_2 = \min_j (\max_i a_{ij}). \qquad (6.2)$$

Если $v_1 = v_2 = v$, то v – чистая цена игры, а игра называется игрой с седловой точкой (x_0, y_0).

Если игра не имеет седловой точки ($v_1 \neq v_2$), то можно получить оптимальное решение случайным образом, чередуя чистые стратегии.

Смешанной стратегией S_{I}^* игрока I называется применение чистых стратегий $x_1, \ldots, x_m$ с вероятностями $\xi_1, \ldots, \xi_m$, $\sum_{i=1}^{m} \xi_i = 1, \quad \xi_i \geq 0$: $S_{\text{I}} = \begin{pmatrix} x_1 \ldots x_m \\ \xi_1 \ldots \xi_m \end{pmatrix}$.

Аналогичным образом определяется смешанная стратегия S_{II}^* игрока II: $S_{\text{II}} = \begin{pmatrix} y_1 \ldots y_n \\ \eta_1 \ldots \eta_n \end{pmatrix}$, $\sum_{j=1}^{n} \eta_j = 1, \ \eta_j \geq 0$.

Оптимальное решение (решение) игры – это пара оптимальных смешанных стратегий S_{I}^* и S_{II}^*, обладающих следующим свойством: если один из игроков придерживается своей оптимальной стратегии, то другому не может быть выгодно отступать от своей оптимальной стратегии. Выигрыш, соответствующий оптимальному решению, называется ценой игры v. Цена игры удовлетворяет неравенству: $v_1 \leq v \leq v_2$.

Справедлива следующая основная теорема теории игр – теорема Неймана: каждая конечная игра имеет по крайней мере одно оптимальное решение, возможно, среди смешанных стратегий.

Если чистая стратегия входит в оптимальную смешанную стратегию с отличной от нуля вероятностью, то она называется активной.

Справедлива теорема об активных стратегиях: если один из игроков придерживается своей оптимальной смешанной стратегии, то выигрыш остается

неизменным и равным цене игры ν, если второй игрок не выходит за пределы своих активных стратегий.

Пусть B_X – множество всех смешанных стратегий игрока I, B_Y – множество всех смешанных стратегий игрока II. Тогда можно определить средний платеж по следующей формуле:

$$K(x,y)=\sum_{i=1}^{m}\sum_{j=1}^{n}a(x_i,y_j)\xi(x_i)\eta(y_j)\,. \qquad (6.3)$$

Если игра G = (X,Y,A) конечна, то игра Г = (B_X,B_Y,K) будет смешанным расширением игры G.

6.2 Игры с ненулевой суммой и их решение

В игре с ненулевой суммой уже становится необязательным, чтобы один из участников выигрывал, а другой – проигрывал. Напротив, они могут и выигрывать, и проигрывать одновременно. Поскольку интересы игроков теперь не являются полностью противоположными, то имеется возможность угрожать противнику, блефовать, сообщать друг другу о своих намерениях, накапливать опыт игры, координировать свои действия с партнером.

Игры с ненулевой суммой могут быть кооперативными и некооперативными.

В *некооперативных* играх игроки принимают решения независимо друг от друга либо потому, что осуществление соглашения невозможно, либо потому, что оно запрещено правилами игры. Примером первой ситуации могут служить антитрестовские законы, запрещающие некоторые виды соглашений между фирмами, а примером второй ситуации – заключение международных торговых соглашений, навязать которые трудно или невозможно. В качестве примера рассмотрим игру «Дилемма заключенного».

Два преступника ожидают решения судьи за совершенное злодеяние. Адвокат предлагает каждому из них облегчить его участь (и даже освободить), если он сознается и даст показания против сообщника, которому грозит тюрьма на 10 лет. Если никто из них не сознается, то каждому грозит заключение на определенный срок (1 год) по обвинению в незначительном преступлении. Если сознаются оба, то с учетом чистосердечного признания им обоим грозит тюрьма сроком на 5 лет. Каждый заключенный имеет выбор из двух стратегий: не сознаться или сознаться и выдать при этом сообщника. Матрица выигрышей выглядит следующим образом (таблица 6.2).

Таблица 6.2 – Матрица выигрышей игры «Дилемма заключенного»

Первый заключенный	Второй заключенный	
	Сознаться	Не сознаться
Сознаться	–5; –5	0; –10
Не сознаться	–10; 0	–1; –1

В данной игре взаимодействие игроков невозможно по правилам игры, это некооперативная игра.

Один из подходов в решении некооперативных игр состоит в определении (точек) точки равновесия игры, где ни один из игроков не имеет никаких причин отказываться от своей стратегии независимых действий. Эти точки называются «точками равновесия по Нэшу».

Нэш доказал, что всякая некооперативная игра с конечным множеством чистых стратегий имеет по меньшей мере одну уравновешенную (равновесную) пару смешанных стратегий или хотя бы одну ситуацию равновесия $(\xi^*;\eta^*)$:

$$\begin{aligned} &K_1(\xi;\eta^*) \le K_1(\xi^*;\eta^*) \\ &K_2(\xi^*;\eta) \le K_2(\xi^*;\eta^*) \end{aligned} \quad \text{для любых } \xi, \qquad (6.4)$$

где $K_1(\xi;\eta)$, $K_2(\xi;\eta)$ – средние выигрыши игроков при использовании ими смешанных стратегий.

Для игры «Дилемма заключенного» равновесными являются стратегии $\left\{\begin{matrix}\xi_1 = (1;0)\\ \eta_2 = (1;0)\end{matrix}\right\}$ или $\left\{\begin{matrix}\xi_1 = (0;1)\\ \eta_2 = (0;1)\end{matrix}\right\}$, т. е. заключенному 2(1) не выгодно отклоняться от второй (первой) стратегии, если заключенный 1(2) придерживается первой (второй) стратегии.

Замечания:

1 Отметим, что выигрыши в равновесных точках различны.

2 В отличие от игр с нулевой суммой в играх с ненулевой суммой существование ситуаций равновесия в некооперативных играх не определяет их решение и однозначные рекомендации для оперирующих сторон отсутствуют.

Пример. Рассмотрим стратегию, в которой игрок A располагает доминирующей стратегией $X_2 > X_1$, сторона B – доминирующей стратегией $Y_2 > Y_1$ (таблица 6.3).

Таблица 6.3 – Матрица выигрышей игры

A	B	
	Y_1	Y_2
X_1	(6;6)	(0;9)
X_2	(9;0)	(1;1)

Равновесие достигается при $\xi^*_A = (0;1)$, $\eta^*_B = (0;1)$, что дает гарантированный выигрыш по единице. Переход каждой из сторон к «неправильной» стратегии $\xi_A = (1;0)$, $\eta_B = (1;0)$ приводит к выигрышу по 6 единиц. Казалось бы, что такой результат нужно сохранить, однако существование договора запрещено правилами игры. В этих условиях каждой стороне приходится опасаться «маневра» противника, выраженного в неожиданном изменении поведения соперника.

Кооперативной игрой называется игра с ненулевой суммой, в которой игрокам разрешается обсуждать перед игрой свои стратегии и договариваться о совместных действиях, т. е. игроки могут образовывать коалиции. Основная задача в кооперативной игре состоит в дележе общего выигрыша между членами коалиции.

В случае игры двух лиц предполагается, что два игрока не могут воздействовать друг на друга, пока не придут к некоторому соглашению. В случае игры двух лиц предполагается, что игра определяется как множество S в пространстве переменных Π_1 и Π_2, представляющее общие выигрыши. Кроме того, заданы два числа T_1 и T_2, определяющие величины выигрыша, которые каждый из игроков может получить, не вступая в коалицию со своим партнером.

Обычно предполагают, что множество S является замкнутым, выпуклым и ограниченным сверху. Точка $T = (T_1; T_2)$ называется точкой угрозы.

Наконец, на переговорном множестве выделяется точка решения Нэша N, в которой достигается максимум произведения превышения выигрышей каждого из игроков над платежами, которые могут быть получены без вступления в коалицию:

$$\max (\Pi_1 - T_1)(\Pi_2 - T_2). \qquad (6.5)$$

На множестве возможных выигрышей (рисунок 6.1) выделяется множество Парето оптимальных решений, т. е. множество точек, принадлежащих S, для которых увеличение выигрыша одного из игроков возможно только за счет уменьшения выигрыша его партнера («северо-восточная» граница S). Все точки Парето оптимального множества находятся одновременно выше и правее точки угрозы T и образуют так называемое переговорное множество. Очевидно, что игрокам нет смысла договариваться относительно решений, не принадлежащих переговорному множеству, либо потому, что положение одного из игроков может быть улучшено при сохранении положения его партнера и можно договариваться о более выгодных решениях, либо потому, что по крайней мере для одного из игроков теряется смысл вступать в коалицию со своим партнером – нехудших результатов он может добиться и в одиночку.

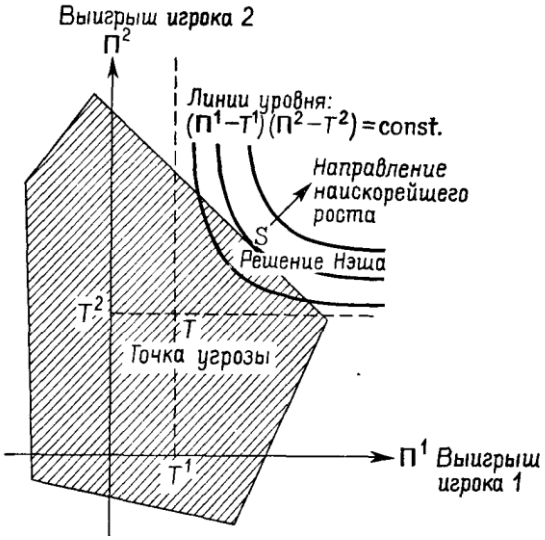


Рисунок 6.1 – Решение Нэша

6.3 Кооперативные игры с побочными платежами

Кооперативными играми с побочными платежами называются игры, в которых допускается заключение взаимообязывающих соглашений о стратегиях, а платежи могут перераспределяться между игроками. Побочными платежами обмениваются конфликтующие стороны в стремлении либо образовать новую коалицию, либо присоединиться к уже действующей, либо противостоять действующей коалиции. Побочный платеж можно интерпретировать как «вступительный взнос», «налог на кооперацию», «штраф за выход из коалиции» и т. д.

Для анализа таких игр пользуются аппаратом характеристических функций игры. С помощью этих функций можно описать возможные коалиции и указать, какой максимальный общий выигрыш может гарантировать себе каждая из коалиций.

Пусть в конфликте участвуют n лиц (сторон). Их совокупность $I_n = (1, \ldots, n)$ – множество всех участников игры. Любое подмножество S множеств I_n является коалицией, всего таких коалиций 2^n.

Если результат, получаемый коалицией S в данной игре, имеет количественное выражение, то возникает проблема выбора наилучшей в каком-то смысле стратегии каждого из игроков. Обычно к этой стратегии предъявляются уже знакомые нам требование: обеспечить максимальный гарантированный средний выигрыш ϑ, сохраняющий неизменное значение при любых допустимых действиях участников игры, оставшихся за пределом коалиции S.

При возможности кооперирования участников неантагонистического конфликта теоретико-игровые принципы оптимальности в зависимости от конкретных условий связываются с устойчивостью поведения (C-ядро) или носят нормативный характер (арбитражные схемы, вектор Шепли). Выбор принципа оптимальности определяется сущностью конфликта.

Характеристическая функция игры – это вещественная функция, область определения которой состоит из 2^n возможных подмножеств множества I_n, т. е. $\vartheta(S)=\vartheta$, $S \in I_n$.

Свойства функции $\vartheta(S)$:

- $\vartheta(\text{пустое})=0$ – у отсутствующей коалиции отсутствует результат;

- для произвольного числа коалиций A_i, $i=1,\ldots,r$, представляющих собой разбиение множества I_n ($\cup A_i = I_n$, $A_i A_j = $ пустое множество, $i \neq j$) выполняется условие супераддитивности:

$$\vartheta(I_n) \geq \sum_{i=1}^{r} \vartheta(A_i) \text{ или } \vartheta(A \cup B) \geq \vartheta(A) + \vartheta(B), \quad A, B \in I. \qquad (6.6)$$

Если $\vartheta(I_n) = \sum_{A_i \subset I_n}^{r} \vartheta(A_i)$, игра называется несущественной, если $\vartheta(I_n) > \sum_{A_i \subset I_n}^{r} \vartheta(A_i)$, игра называется существенной.

Множество недоминируемых дележей кооперативной игры с характеристической функцией $F = \{N, v\}$ называется ее C-ядром. Здесь N – количество игроков; v – функция, сопоставляющая каждому подмножеству S из N некоторый платеж $V(S)$; S – подмножество N. Так как против любого недоминирумого дележа ни у кого из игроков (а также коалиций) не будет возражений, то такой дележ может быть реализован. Поэтому C-ядро можно считать решением игры $\{N, v\}$.

Для того чтобы дележ d принадлежал C-ядру игры, необходимо и достаточно выполнения неравенства $d(S) \geq v(S)$ для любого подмножества S из N.

Пример. Найти C-ядро кооперативной игры с побочными платежами.

Рассмотрим кооперативную игру трех лиц с характеристической функцией. Пусть три игрока поставлены перед необходимостью раздела дохода, равного 1, при этом $v(i) = 0$, $i = 1, 2, 3$; $v(1,2) = v(1,3) = 1/2$, $v(1, 2, 3) = 1$.

Вектор дележей $d = (d_1, d_2, d_3)$ принадлежит C-ядру тогда и только тогда, когда $d_1 + d_2 + d_3 = 1$; $d_i \geq 0$, $i = 1, 2, 3$; $d_1 + d_2 \geq 1/2$; $d_1 + d_3 \geq 1/2$; $d_2 + d_3 \geq 1$.

Решая данную систему неравенств, получаем $d_1 \geq 0$, $d_2 \geq 1/2$, $d_3 \geq 1/2$.

Следовательно, $d = (d_1, d_2, d_3) = (0, 1/2, 1/2)$. C-ядро представляет собой один дележ.

Вектор $\Pi = \Pi^{(1)}, \ldots, \Pi^{(n)}$ n-мерного евклидова пространства, компонентами которого являются суммарные выигрыши каждого отдельного игрока, называется дележом. Эти компоненты должны удовлетворять следующим условиям:

1 $\Pi^{(i)} \geq \vartheta(\{i\})$ $\forall i \in I$ – условие индивидуальной рациональности. (6.7)

Участвуя в коалиции, каждый игрок получает по меньшей мере столько, сколько он мог бы получить, действуя независимо.

2 $\vartheta(I_n) = \sum_{i=1}^{r} \Pi^{(i)}$ – условие групповой рациональности. (6.8)

Величина суммарного выигрыша игроков будет равна выигрышу большей коалиции, включающей всех игроков.

Таким образом, исходом кооперативной игры является дележ как результат соглашения между конфликтующими сторонами. Какой дележ будет решением игры? Однозначный ответ можно дать для случая несущественной игры, когда $\vartheta(I_n) = \sum_{i=1}^{r} \vartheta(\{i\})$ и единственным дележом является $\Pi = \{\vartheta(\{1\}),..., \vartheta(\{n\})\}$.

В любой существенной игре с числом участников $n \geq 2$ множество дележей бесконечно и проблема поиска среди них решения усложняется.

Перечислим четыре правила, которыми может руководствоваться исследователь, изучающий конфликтную ситуацию и принимающий на себя роль арбитра. Для этого рассмотрим дележ $\Pi' = (\Pi'^{(1)}, ..., \Pi'^{(n)})$ в игре с характеристической функцией $\vartheta(S)$:

1 Величина $\Pi'^{(i)}$ сохраняется неизменной при любой перестановке участников игры, $\Pi'^{(i)} = \text{const}$.

2 Сумма выигрышей $\Pi'^{(i)}$ всех участников равна значению характеристической функции игры при $S = I_n$, т. е. $\sum_{i=1}^{r} \Pi'^{(i)} = \vartheta(I_n)$.

3 Участник, присоединенный к любой коалиции $S \in I_n$, но не приносящий ей пользы, ничего не выигрывает, т. е. $\Pi'^{(i)} = 0$ при $\vartheta(S \cup \{i\}) = \vartheta(S)$.

4 Игра, характеристическая функция которой есть сумма характеристических функций двух других самостоятельных игр, должна дать любому участнику $\{i\}$ выигрыш, равный сумме выигрышей, какие он получил бы в указанных самостоятельных играх.

Правила 1–4 позволяют получить в любой игре единственный вектор (дележ) $\Pi' = (\Pi'^{(1)}, ..., \Pi'^{(n)})$, называемый вектором Шепли. Его составляющие равны средней величине вклада участника i во все коалиции, куда он мог бы вступить, и зависят от «силы» i-го игрока ($\vartheta(S \cup \{i\}) = \vartheta(S)$).

Вклад в гарантированный выигрыш некоторой коалиции от включения в нее игрока $\{i\}$, т. е. «сила» этого игрока определяется по следующей формуле:

$$\Pi'^{(i)} = \sum_{S \in I_n} \gamma_n(S)[\vartheta(S \cup \{i\}) - \vartheta(S)], \quad (6.9)$$

где $\gamma_n(S)$ – вероятность присоединения *i*-го игрока к коалиции *S*. Эта вероятность равна: $\gamma_n(S) = \frac{S!(n-S-1)!}{n!}$.

Действительно, коалиция из *n* участников может быть организована *n*! способами. Существует *S*! способов организации для *S* игроков, входящих в коалицию до того, как к ней присоединился *i*-й игрок. Игроки, не входящие в расширенную коалицию, число которых (*n* – *S* – 1), могут быть организованы (*n* – *S* – 1)! способами. Тогда отношение $\frac{S!(n-S-1)!}{n!}$ и есть возможность присоединения *i*-го игрока к коалиции *S*. При вычислении все способы формирования коалиции являются равновероятностными.

Пример. *Модель финансирования строительства*.

Несколько перерабатывающих предприятий собираются строить хранилище дорогостоящего сырья, которое они потребляют для производственных нужд. Затраты на строительство являются некоторой возрастающей функцией *F* вместимости хранилищ. Перед предприятиями ставится вопрос о возможности объединения денежных средств для финансирования строительных работ на договорных началах.

Пусть количество предприятий равно *H*, потребности предприятий в объеме хранилища изменяются во времени по законам $\xi_\gamma(t),\ \gamma = 1, .., n$. За сырьем разрешено обращаться лишь в определенные моменты времени t_1, t_2, …, t_k, не зависящие от будущих способов его хранения и одинаковые для всех предприятий.

Расчетный объем хранилища, предназначенный для обслуживания всех *n* потребителей, есть $\max\{\sum_{\gamma=1}^{n}\xi_\gamma(t_1), ..., \sum_{\gamma=1}^{n}\xi_\gamma(t_k)\}$. Если же какие-то предприятия с номерами $\lambda \in H\ (H \subset \{1, ..., h\})$ отказываются от идеи общего строительства и объединяются с целью создания отдельного хранилища, то оно должно быть рассчитано на объем, равный $\max\{\sum_{\vartheta\in H}\xi_\gamma(t_1), ..., \sum_{\vartheta\in H}\xi_\gamma(t_k)\}$. Можно оценить расходы каждой такой коалиции $H \subset \{1, ..., h\}$ $F(\max_t \sum_{\gamma\in H}\xi_\gamma(t))$ $\forall H \subset [1, 2, ..., n]$.

Требуется найти количество хранилищ и состав коалиций, которые будут их строить, а также распределить соответствующим образом расходы на строительство.

Данную задачу можно рассматривать как кооперативную игру *n* лиц с характеристической функцией вида $\vartheta(I_n) = F[\max_t \sum_{\{\gamma\}\in I_n}\xi_\lambda(t)]$, $I_n = (1, ..., n)$, где $\{\gamma\}$ обозначает ϑ-го участника игры, т. е. предприятие с номером ϑ, а $\sum_{\{\gamma\}\in I_n}\xi_\gamma(t)$ – суммарные потребности коалиции I_n в сырье в момент *t*. По-

скольку $\vartheta(I_n)$ определяет здесь затраты на строительство, понятие выигрыш (доход) нужно связывать с минимизацией ϑ, т. е. экономией средств за счет кооперирования.

Отметим, что перечисленные выше правила 1–4 в нашем случае выполняются при следующих условиях:

- каким бы не был дележ $\Pi = (\Pi_1^{(1)}, ..., \Pi_n^{(n)})$, его составляющие дадут в сумме $\vartheta(I_n)$ независимо от порядка номера и их соответствия тем или иным предприятиям;

- участник $\{\vartheta\}$, для которого $\xi_\gamma(t) = 0$, ничего не вносит в строительство и, следовательно, освобождается от участия в дележе расходов и теряет место в готовом хранилище;

- любое предприятие $\{\vartheta\}$ в игре с характеристической функцией вида $F_1\{iden\} + F_2\{iden\}$ должно сохранить те позиции, которые оно имело бы при прочих равных условиях в играх с характеристической функцией F_1 и F_2, т. е. вносить вклад $\vartheta(\{\gamma\})$, равный сумме вкладов $\vartheta_1(\{\gamma\})$ и $\vartheta_2(\{\gamma\})$. Это равносильно предложению о поэтапному финансированию строительства при неизменной общей структуре игры.

В данных условиях можно попытаться найти решение, основанное на принципе «справедливого дележа», – решение Шепли.

Исходные данные задачи представлены в таблице 6.4.

Таблица 6.4 – Выигрыши коалиций

Возможные коалиции	$\vartheta(S)$		
{1}, {2}, {3}	200	300	260
{1,2}, {1,3}, {2,3}	400	390	500
{1,2,3}	600		

Из таблицы видно, решение кооперации снижает удельную стоимость строительных работ $(\underset{1}{200} + \underset{2}{300} + \underset{3}{260} = 760) > \underset{1,2,3}{600}$. Источники экономии: уменьшение транспортных издержек, улучшение форм организации труда, снижение накладных расходов и т. д.

Если все вышеназванные коалиции «участвуют в игре», то из формулы (6.9) следует

$$\Pi'^{(1)} = \frac{2!0!}{3!}[\vartheta(I_n) - \vartheta(2,3)] + \frac{1!1!}{3!}[\vartheta(1,2) - \vartheta(2)] + \frac{1!1!}{3!}[\vartheta(1,3) - \vartheta(3)] +$$

$$+ \frac{0!2!}{3!}[\vartheta(1) - \vartheta(\text{пустое})] = 140.$$

Аналогично получаем: $\Pi'^{(2)} = 245$, $\Pi'^{(3)} = 215$.

Контроль: $\sum_{i=1}^{3} \Pi'^{(i)} = 600 \cdot \vartheta(I_3)$.

Теперь необходимо оценить размеры взносов, которые придется вносить предприятиям, решившим объединиться в коалиции {1,2}, {1,3}, {2,3}. Используем ту же формулу (6.9).

Для коалиции {1,2}:

$$\Pi'^{(1)}_{(1,2)} = \frac{1!0!}{2!}[\vartheta(1,2) - \vartheta(2)] + \frac{0!1!}{2!}[\vartheta(1) - \vartheta(\text{пустое})] = 150,$$

$$\Pi'^{(2)}_{(1,2)} = \frac{1!0!}{2!}[\vartheta(1,2) - \vartheta(1)] + \frac{0!1!}{2!}[\vartheta(2) - \vartheta(\text{пустое})] = 250.$$

Участник {3}, оставшийся в нерассмотренной коалиции {1,2}, вынужден действовать самостоятельно, расходуя средства в размере $\vartheta(3) = 250 = \Pi^{(3)}_{(1,2)}$.

Аналогичная картина наблюдается и в случае создания коалиции {1,3} без игрока {2}. Здесь $\Pi^{(1)}_{(1,3)} = 170$, $\Pi^{(3)}_{(1,3)} = 220$, $\Pi^{(2)}_{(1,3)} = 300$ (т. к. $\vartheta(2)$ действует самостоятельно).

Для коалиции {2,3}, исключая {1}, получается $\Pi^{(2)}_{(2,3)} = 275$, $\Pi^{(3)}_{(2,3)} = 225$, $\Pi^{(1)}_{(2,3)} = 200$ (т. к. $\vartheta(1)$ действует самостоятельно).

Очевидно, $\Pi'^{(1)} < \Pi^{(1)}_{(1,2)} < \Pi^{(1)}_{(1,3)} < \Pi^{(1)}_{(2,3)}$, поэтому первому предприятию выгодно участие в общей коалиции {1,2,3}. То же самое можно сказать о втором и третьем предприятии. Следовательно, к решению игры для принятых исходных данных приводит первый вариант кооперирования, т. е. {1,2,3} $\Pi'^{(*)} = (140; 245; 215)$.

Полученный выигрыш составляет: 60 тыс. дол. США для первого предприятия; 55 тыс. дол. США для второго; 45 тыс. дол. США для третьего.

Дележи коалиций представлены в таблице 6.5.

Таблица 6.5 – Дележи коалиций

$\vartheta(1)$	$\vartheta(2)$	$\vartheta(3)$	$\vartheta(1,2)$	$\vartheta(1,3)$	$\vartheta(2,3)$	$\vartheta(1,2,3)$
200	300	260	400	390	500	600
$\Pi'^{(1)}$	–	–	150	170	200	140
$\Pi'^{(2)}$	–	–	250	300	375	245
$\Pi'^{(3)}$	–	–	280	220	225	215

Полезно заметить, что изменение данных таблицы 6.5 приводит не только к переоценкам размеров платежей участников игры, но и к распаду коалиций.

6.4 Статистические игры и их применение в экономике

Все модели принятия решений с учетом информации, которой располагает объект, принимающий решение, можно разделить на следующие виды:

- детерминистские (в условиях полной определенности);
- стратегические (в условиях полной неопределенности);
- статистические (в условиях частичной неопределенности).

Характерной чертой статистической игры (игра с природой при использовании статистической информации о состояниях природы) является возможность получения информации в результате проведения некоторого статистического эксперимента для оценивания распределения вероятностей состояния природы.

Рассмотрим две статистические игры:

1) игра двух лиц с нулевой суммой:

$$< X, Y, W >,$$
$$(\theta_j, \delta)$$

где X, Y – чистые стратегии первого и второго игроков;

W – платежная функция;

2) игра с природой:

$$< \theta, A, L >,$$

где θ – множество состояний природы;

A – множество действий статистика (игрок 1) ($a \in A$, $a \in \theta$ – отдельное действие статистика, отдельное состояние природы);

L – функция потерь, при которой любой комбинации (a, θ) ставится в соответствие некоторое число. Функция $L(a, \theta)$ показывает, какие потери несет статистик, если он принимает решение (действие) $a \in A$ при действительном состоянии природы $a \in \theta$.

Назовем эту игру исходной стратегической игрой, соответствующей задаче принятия решений.

Статистические игры и их расширение.

Перед принятием решений статистик может провести эксперимент, цель которого – наблюдение за некоторой переменной ζ, распределение вероятностей которой зависит от состояния природы θ: $F(x \mid \theta)$.

Пусть $X = \{x_1, \ldots, x_n\}$ – выборка объема n или результат эксперимента над величиной ζ, X – множество всех возможных результатов выборки, т. е. $x \in X$.

Получив x, статистик принимает одно из решений $a \in A$. При этом он руководствуется некоторым правилом поведения (функцией решения):

$$d(x): X \to A, \quad d(x) = a, \quad a \in A, x \in X.$$

Функция $d(x)$ называется нерандомизированной (неслучайной) функцией решений статистика. Статистик ищет оптимальную функцию решений d из множества D функций решений ($d \in D$), которая и будет его стратегией.

Для сравнения различных функций решений и выбора из них наилучшей статистик должен знать характеристики этих функций и критерий выбора.

Характеристикой функций решений $d(x)$ является функция риска

$$R = (\theta, d) = M\left[L(\theta, a)\right] = \int L(\theta, a) dF(x \mid \theta)$$

или математическое ожидание функции потерь при некотором состоянии природы θ и заданном законе распределения $F(x \mid \theta)$. Действительно, ведь $a = d(x)$ зависит от результата эксперимента над величиной ζ.

Игра $< \theta, D, R >$ называется статистической игрой, D – множество функций решений, преобразующих X в A, X – множество n-переменных выборок, $R(\theta, d)$ – функция риска.

Данную игру можно обобщить, если ввести смешанные стратегии обоих игроков: статистика и природы.

Рандомизация на стороне статистика может осуществляться двумя способами:

- смешанные решения $a \in A$;
- смешанные чистые функции решений $d \in D$.

Оказалось, что оба эти способа эквивалентны, т. е. связаны с одинаковым риском.

Каждое распределение вероятностей на множестве δ чистых функций решений D назовем рандомизированной (смешанной) функцией статистика. Обозначим множество всех случайных функций решений через D^*, причем $\delta \in D^*$. Очевидно, что $D \subset D^*$.

Если статистик в игре принимает случайную функцию решений δ, то функция риска становится случайной величиной и платеж в игре будет математическим ожиданием функции риска

$R(\theta, d)$, т. е. $R = (\theta, \delta) = M_D\left[R(\theta, d)\right]$.

Игра $< \theta, D^*, R >$ есть расширение статистической игры с рандомизацией на стороне статистика. Априорное распределение $\eta \in \Xi$ постоянной природы представляет собой смешанную стратегию природы в статистической игре. При этом функция риска $R(\theta, \delta)$ становится случайной величиной и платежом в игре теперь выступает байесовский риск функции решений δ, т. е.

$$r(\eta, \delta) = M_\theta\left[R(\theta, \delta)\right] = \int_\theta R(\theta, \delta) d\eta .$$

Игра $< \Xi, D, r >$ – расширение статистической игры с рандомизацией на стороне природы.

Игра $< \Xi, D^*, r >$ – расширение статистической игры с рандомизацией со стороны и природы, и статистика.

Решение статистических игр сводится к получению оптимальной стратегии статистика, т. е. к определению наилучшей функции решений.

Оптимальные функции решений.

Естественно ожидать, что для каждого состояния природы существуют разные наилучшие функции решений. На рисунке 6.2 представлены две функции риска, соответствующие двум различным функциям решений δ_1 и δ_2.

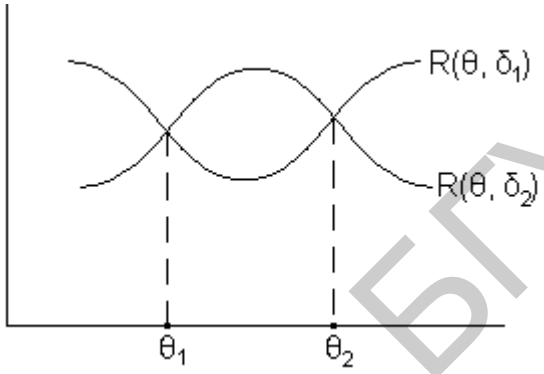


Рисунок 6.2 – Функции риска

Функция $\delta_2 > \delta_1$, если $\theta_1 < \theta < \theta_2$, $\delta_1 > \delta_2$, если $\theta > \theta_2$ и $\theta < \theta_2$.

Ознакомимся с наиболее известными оптимальными функциями решений для игры $< \Xi, D^*, r >$:

1 Байесовская функция решений $\delta_2 \in D^*$ относительно априорного распределения $\eta \in \Xi$ состояний природы. Она минимизирует байесовский риск $r(\eta, \delta)$ на множестве D^*:

$$r(\eta, \delta_0) = \inf_{\delta \in D^*} r(\eta, \delta).$$

2 Наименее выгодным для статистика априорным распределением $\eta_0 \in \Xi$ состояний природы будет maxmin стратегия природы:

$$\inf_{\delta \in D^*} r(\eta_0, \delta) = \sup_{\eta \in \Xi} \inf_{\delta \in D^*} r(\eta, \delta),$$

т. е. стратегия, которая максимизирует байесовский риск в игре.

3 Функцию решений $\delta_0 \in D^*$ назовем *минимаксной*, если в D^* она минимизирует максимальный риск, т. е.

$$\sup_{\theta \in \Theta} R(\theta, \delta_0) = \inf \sup_{\delta \in D^*} R(\theta, \delta).$$

Это наиболее осторожная стратегия статистика.

Заметим, что *байесовская функция* решений статистика относительно самого невыгодного для него состояния природы будет *минимаксной.*

Пессимистический характер минимаксной функции решений приводит к выводу, что эта функция должна быть выбрана статистиком лишь в следующих случаях:

1) невозможно применить байесовскую стратегию из-за полного незнания априорного распределения η состояний природы;

2) с учетом больших потерь статистику целесообразно использовать более осторожную стратегию.

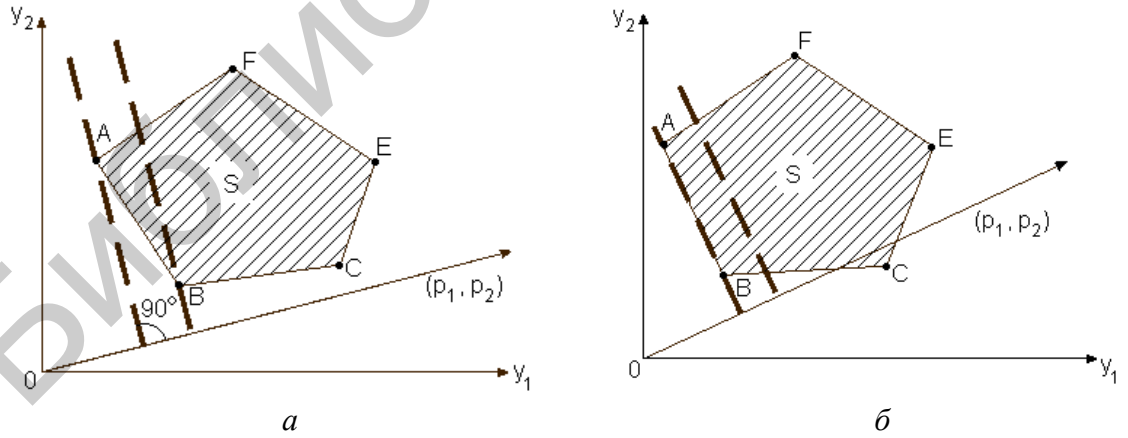
Геометрическая интерпретация функций решения.

Пусть $\theta = (\theta_1, \ldots, \theta_n)$, S – множество рисков в n-мерном эвклидовом пространстве. Это множество рисков $S = \{y_j = R(\theta_j, \delta), j = 1, \ldots, n\}$ выпукло и порождается множеством D чистых функций решений.

Каждой точке этого множества соответствует определенная функция решений $\delta \in D^*$ и наоборот.

Пусть (p_1, p_2) – заданное распределение η состояний природы. Для данного априорного распределения η, заданного вероятностями (p_1, p_2), отыскивается плоскость с уравнением $r = \sum_{j=1}^{n} p_j y_j$, которая в процессе уменьшения значений r максимально перемещается к началу координат до тех пор, пока гиперплоскость еще имеет общую точку с множеством S. Эта точка и соответствует байесовской функции решений δ_0.

На рисунке 6.3 изображены пунктиром линии уровня $p_1y_1 + p_2y_2 = r$ функции риска (средний риск).



а *б*

а – единственное решение; *б* – решение находится на отрезке *AB*

Рисунок 6.3 – Байесовская функция решений

При выборе минимаксной функции решений руководствуются не средним риском $R(\theta, \delta)$, а его максимальным значением.

Пусть S – множество рисков, порождаемых конечным множеством состояний природы $\theta = (\theta_1, \ldots, \theta_n)$:

$$R(\theta_j, \delta) = y_j, j = 1, \ldots, n,$$
$$\max_j R(\theta_j, \delta) = \max y_j.$$

Представим себе множество в виде n-мерного куба Q_c, ребро которого равно $c > 0$, т. е.

$$Q_c = \{(y_1,\ldots, y_n): y_j \le c, j = 1, \ldots, n\}.$$

Максимальные риски, т. е. максимальные y_j для разных функций решений, находятся на гранях множества Q_c, удовлетворяя уравнению $y_j = c$ при $j = 1, \ldots, n$. При отыскании минимаксной функции решений ищется такая функция, которая минимизирует максимальные риски $y_j = c$, т. е. отыскивается наименьшее число $c = c_0 > 0$, при котором множество Q_c имеет с множеством рисков по крайней мере одну общую точку. Это означает, что строится минимальный куб Q_c, пересекающийся с множеством рисков S (рисунок 6.4). Общая точка множества S и пространства Q_c соответствует минимаксной функции решений δ_0.

При этом точка множества S, соответствующая функции решения δ_0, всегда находится на его грани и на стороне минимального куба Q_c.

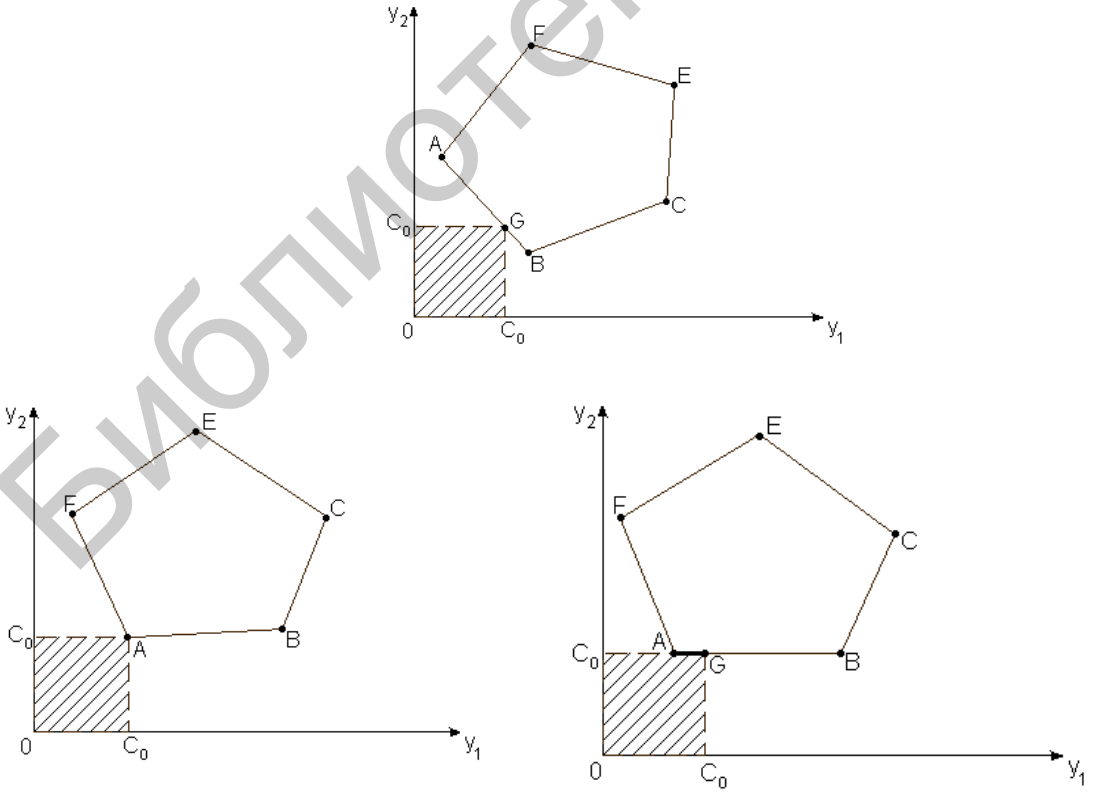


Рисунок 6.4 – Минимальный куб, пересекающийся с множеством рисков

На *AG* находится множество минимаксных решений.

В теории статистических игр действует ряд теорем, из которых вытекает существование обоих видов оптимальных функций решений.

7 Марковские процессы с доходами

7.1 Марковские процессы

При изучении сложных систем полезной математической моделью может служить марковский процесс. Основными для марковских процессов являются понятия состояния системы и перехода из одного состояния в другое.

Случайный процесс, протекающий в системе, называется процессом с дискретным временем, если переходы системы из состояния в состояние могут осуществляться только в заранее определенные моменты времени, называемые шагами этого процесса. В промежутках между соседними шагами система сохраняет свои состояния.

Случайный процесс, протекающий в системе, называется процессом с непрерывным временем, если переходы системы из состояния в состояние возможны в любые, заранее не известные, случайные моменты времени.

Если через $S_i(k)$ ($i = 1, \ldots, n$; $k = 1, 2, \ldots$) обозначить событие, состоящее в том, что с k-го шага до $(k + 1)$-го система S находится в состоянии s_i, то случайный процесс с дискретным временем можно представить случайной последовательностью (по индексу k) этих событий $S_i(k)$, которую называют также цепью.

Основными характеристиками марковских цепей являются вероятности $p_i(k) = p(S_i(k))$ ($i = 1, \ldots, n$; $k = 1, 2, \ldots$) событий $S_i(k)$, которые называют вероятностями состояний.

Так как для каждого шага $k = 1, 2, \ldots$ события несовместны и образуют полную группу, то сумма вероятностей этих событий равна единице:

$$\sum_{i=1}^{n} p_i(k) = 1,\ k = 1, 2, \ldots . \qquad (7.1)$$

Переходной вероятностью $p_{ij}(k)$ из i-го в j-е состояние для k-го шага ($i = 1, \ldots, n$; $k = 1, 2, \ldots$) называется вероятность непосредственного перехода системы S в момент t_k из состояния s_i в состояние s_j.

Если переходные вероятности не зависят от шагов k, то марковская цепь называется однородной. В этом случае переходные вероятности будем обозначать через p_{ij} вместо $p_{ij}(k)$. Если же хотя бы одна вероятность изменяется с изменением шага, то марковскую цепь называют неоднородной.

Запишем переходные вероятности в виде квадратной матрицы n-го порядка, сумма элементов каждой строки которой равна единице:

$$P = (p_{ij})^n = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \cdots & \cdots & \cdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix}.$$

Матрица, каждый элемент которой неотрицателен, а сумма элементов каждой строки равна единице, называется стохастической.

Вероятности задержек можно подсчитать по формуле

$$p_{ii} = 1 - \sum_{j=1,\, j\neq i}^{n} p_{ij}\,,\ i = 1, \ldots, n. \qquad (7.2)$$

Вектор-строка вероятностей состояний $(p_1(0), \ldots, p_n(0))$ в начальный момент времени $t = 0$, непосредственно предшествующий первому шагу, называется вектором начального распределения вероятностей.

Имея в своем распоряжении начальное распределение вероятностей и матрицу переходных вероятностей, можно вычислить вероятности состояний системы от любого k-го до $(k + 1)$-го шага, $k = 1, 2, \ldots$.

Для однородной марковской цепи вектор-строка вероятностей состояний от k-го до $(k + 1)$-го шага равна произведению вектор-строки вероятностей состояний от $(k – 1)$-го до k-го шага на матрицу переходных вероятностей:

$$(p_1(k), \ldots, p_n(k)) = (p_1(k – 1), \ldots, p_n(k – 1))P. \qquad (7.3)$$

При $k = 1$ в правой части равенства стоит произведение вектора начального распределения вероятностей на матрицу P.

Для однородной марковской цепи имеет место следующая формула:

$$(p_1(k), \ldots, p_n(k)) = (p_1(0), \ldots, p_n(0))P^k\,,\ k = 1, 2, \ldots . \qquad (7.4)$$

Пример [10]. Игрушечных дел мастер открывает новое производство игрушек. Он может находиться при этом в одном из двух состояний. Первое состояние ($i = 1$) – если игрушка, которую сейчас делает мастер, получит большой спрос. Второе состояние ($i = 2$) – если игрушка не найдет спрос. Предположим, что если мастер находится в состоянии 1, то в 50 % случаев к концу следующей недели (мастер выпускает каждую неделю новую игрушку) он в нем и останется. В 50 % неудачных случаях он переходит в состояние 2.

Будучи в состоянии 2, мастер экспериментирует с новыми игрушками и с вероятностью 2/5 может вернуться через неделю в состояние 1 или с вероятностью 3/5 остаться в невыгодном состоянии 2. В матричной форме имеем

$$P = [p_{ij}] = \begin{bmatrix} 1/2 & 1/2 \\ 2/5 & 3/5 \end{bmatrix}.$$

Пусть мастер начинает с удачной игрушки, т. е. $p(0) = [1\ 0]$. Пользуясь уравнением (7.3), получим

$$p(1) = p(0)\,P = [1\ \ 0] \begin{bmatrix} 1/2 & 1/2 \\ 2/5 & 3/5 \end{bmatrix} = [1/2\ \ 1/2].$$

После одной недели равновероятно, что мастер добьется успеха или потерпит неудачу.

После двух недель:

$$p(2) = p(1)\,P = [1/2\ 1/2] \begin{bmatrix} 1/2 & 1/2 \\ 2/5 & 3/5 \end{bmatrix} = [9/20\ 11/20].$$

Так что более вероятно, что мастер потерпит неудачу.

После трех недель:

$$p(3) = p(2)\,P = [9/20\ 11/20] \begin{bmatrix} 1/2 & 1/2 \\ 2/5 & 3/5 \end{bmatrix} = [89/200\ \ 111/200].$$

Интересная закономерность выявляется, когда мы вычислим $p_i(n)$ (таблица 7.1).

Таблица 7.1 – Вероятности состояний системы при движении из состояния 1

n	0	1	2	3	4	5
$p_1(n)$	1	0,5	0,45	0,445	0,4445	0,44445
$p_2(n)$	0	0,5	0,55	0,555	0,5555	0,55555

Из этой таблицы видно, что $p_1(n) = 4/9$; $p_2(n) = 5/9$.

Если мастер начинает с неудачной игрушки, т. е. $p(0) = [0\ \ 1]$, то аналогичная таблица имеет следующий вид (таблица 7.2).

Таблица 7.2 – Вероятности состояний системы при движении из состояния 2

n	0	1	2	3	4	5
$p_1(n)$	0	0,4	0,44	0,444	0,4444	0,44444
$p_2(n)$	1	0,6	0,56	0,556	0,5556	0,55556

Очевидно, что $p_1(n) = 4/9$; $p_2(n) = 5/9$.

При финальном стационарном режиме вероятности состояний системы не зависят ни от времени, ни от начального распределения вероятностей.

Вероятности состояний системы в финальном стационарном режиме называются финальными (предельными, стационарными) вероятностями и обозначаются $\pi_1, \pi_2, \ldots, \pi_n$, а вектор $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ называется финальным вектором.

Рассмотрим однородную марковскую цепь, т. е. систему S с дискретными состояниями $s_1, \ldots, s_n$ и дискретным временем.

Пусть $(\pi_1(0), \ldots, \pi_n(0))$ – вектор начального распределения вероятностей, $\Pi = \begin{pmatrix} \pi_{11} & \ldots & \pi_{1n} \\ \ldots & \ldots & \ldots \\ \pi_{n1} & \ldots & \pi_{nn} \end{pmatrix}$ – матрица переходных вероятностей π_{ij}, не зависящих от шагов (моментов времени).

Если существует финальный стационарный режим и, следовательно, существуют финальные вероятности, то, чтобы $\pi_1, \pi_2, \ldots, \pi_n$ были финальными вероятностями, необходимо и достаточно, чтобы существовал m-й шаг, такой, что

$$(\pi_1, \ldots, \pi_n) = (\pi_1(m), \ldots, \pi_n(m)) = (\pi_1(m+1), \ldots, \pi_n(m+1)). \quad (7.5)$$

Финальные вероятности удовлетворяют нормировочному условию:

$$\sum_{i=1}^{n} \pi_i = \sum_{i=1}^{n} \pi_i(m) = 1. \quad (7.6)$$

Если существуют финальные вероятности, то финальный вектор равен

$$\pi = (\pi_1, \pi_2, \ldots, \pi_n) = (\pi_1, \pi_2, \ldots, \pi_n)\Pi, \quad (7.7)$$

где П – матрица переходных вероятностей.

Будем называть эргодическим всякий марковский процесс, для которого предельное распределение вероятностей состояний не зависит от начальных состояний.

Запишем для нашего случая систему уравнений согласно (7.7):

$$\begin{cases} \pi_1 = 1/2\pi_1 + 2/5\pi_2, \pi_2 = 1/2\pi_1 + 3/5\pi_2, \\ \pi_1 + \pi_2 = 1. \end{cases}$$

Эти уравнения имеют единственные решения:

$$\pi_1 = 4/9, \pi_2 = 5/9.$$

Во многих приложениях предельные вероятности являются единственными интересующими нас величинами. Действительно, мастеру достаточно знать, что 4/9 всех выпускаемых им игрушек будут удачны, а оставшиеся 5/9 – неудачными.

Итак, в нашем примере финальное распределение вероятностей существовало, каждое из состояний имело ненулевую вероятность, и эти вероятности постоянно описывают поведение системы в течение длительного времени.

7.2 Рекуррентные соотношения для доходов

Рассмотрим эргодический марковский процесс с конечным множеством состояний $S_1, S_2, \dots, S_N$, которому соответствует матрица вероятностей переходов P размером N на N. Каждому переходу из состояния S_i в состояние S_j поставим в соответствие некоторую оценку r_{ij} $(i, j = \overline{1,N})$, которую в дальнейшем будем называть доходом, за один переход из состояния S_i в состояние S_j. Доход необязательно выражается в денежных единицах и имеет экономический смысл. Это может быть и число единиц продукции, и значение электрического напряжения, и количество набранных очков в спортивной игре, и т. п.

Очевидно, что суммарный доход, получаемый от функционирования такой системы в течение некоторого промежутка времени, будет величиной случайной, зависящей от распределения вероятностей соответствующего марковского процесса.

Осуществим расчет общей величины дохода, которого следует ожидать в результате совершения системой m переходов (под общей величиной дохода здесь понимается математическое ожидание, или среднее значение дохода).

Обозначим через $v_i(m)$ средний ожидаемый доход, получаемый после совершения системой m переходов, если процесс начался из состояния S_i. Доход за m переходов может быть получен как доход за один (первый) переход плюс доход за оставшиеся $m-1$ переходов. Доход за один переход системы из состояния E_i обозначим через q_i:

$$q_i = \sum_{j=1}^{N} p_{ij} r_{ij},\ i = \overline{1,N}. \tag{7.8}$$

Ожидаемый доход за оставшиеся $m-1$ переходов зависит от того, в каком состоянии оказалась система после первого шага. Пусть это будет состояние E_j, тогда средний доход за оставшиеся $m-1$ переходов будет равен $v_i(m-1)$. Так как система из состояния E_i могла попасть в любое состояние E_j $(j = \overline{1,N})$ с соответсвующей вероятностью p_{ij}, средний доход за оставшиеся $m-1$ переходов определяется с помощью выражения

$$\sum_{i=1}^{N} p_{ij} v_i(m-1),\ m = 1, 2, \ldots . \qquad (7.9)$$

Таким образом, с учетом выражений (7.8), (7.9) полный ожидаемый доход за m переходов равен

$$v_i(m) = q_i + \sum_{j=1}^{N} p_{ij} v_j(m-1),\ i = \overline{1, N},\ m = 1, 2, \ldots . \qquad (7.10)$$

Выражение (7.10) в векторном виде записывается следующим образом:

$$V(m) = q + PV(m-1), \qquad (7.11)$$

где $V(m) = \begin{pmatrix} v_1(m) \\ v_2(m) \\ \cdots \\ v_N(m) \end{pmatrix}$;

$q = \begin{pmatrix} q_1 \\ q_2 \\ \cdots \\ q_N \end{pmatrix}$.

С помощью рекуррентного соотношения (7.11) может быть последовательно определен полный доход за любое число переходов. Из выражения (7.10) или (7.11) следует, что для определения ожидаемого полного дохода нет необходимости вводить в рассмотрение матрицу доходов. Достаточно знать матрицу вероятностей переходов P и вектор-столбец q ожидаемых доходов за один переход.

Учитывая эргодичность марковского процесса, ожидаемый доход $v_i(m)$ $(i = \overline{1, N})$, при больших m можно вычислить по формуле

$$v_i(m) = mg + v_i, \qquad (7.12)$$

или в векторной форме

$$V(m) = mg + V,$$

где $g = \sum_{j=1}^{N} p_i q_i$;

p_i – предельные вероятности марковского процесса.

Величина g есть не что иное, как средний ожидаемый доход за один переход, когда система осуществляет достаточно большое число переходов.

Величинам v_i $(i = \overline{1, N})$, которые в дальнейшем будем называть относительными весами, можно дать вполне определенную интерпретацию. Зафиксировав два состояния S_j и S_r и учитывая выражение (7.12), можно записать: $v_j(m) - v_r(m) = v_j - v_r$. Поэтому разность $v_j - v_r$ можно считать величиной

дохода, который будет дополнительно получен, если система начнет функционировать из состояния S_j, а не S_r.

Пример. В условие примера с игрушечных дел мастером добавлена матрица $R = \begin{pmatrix} 9 & 3 \\ 3 & -7 \end{pmatrix}$, элементы которой определяют величину дохода за переход из одного состояния в другое. Тогда на основании соотношения (7.8) доход за один переход системы будет равен:

$$q_1 = 9 \cdot 1/2 + 3 \cdot 1/2 = 6,$$
$$q_2 = 3 \cdot 2/5 + 7 \cdot 3/5 = -3,$$
$$q = \begin{pmatrix} 6 \\ -3 \end{pmatrix}.$$

Полный ожидаемый доход, определяемый с помощью выражения (7.10) или (7.11), может быть представлен в виде таблицы 7.3.

Таблица 7.3 – Полный ожидаемый доход

$v_j(m)$	m					
	0	1	2	3	4	5
$v_1(m)$	0	6	7,5	8,55	9,555	10,5555
$v_2(m)$	0	−3	−2,4	−1,44	−0,444	0,5556

Согласно выражению (7.12) средний ожидаемый доход за один переход составит

$$g = \frac{4}{9} \cdot 6 - \frac{5}{9} \cdot 3 = 1.$$

Из таблицы 7.3 и расчета g следует, что если система начинает функционировать из состояния S_1, то будет получен дополнительный доход 10 ден. ед., а каждая последующая неделя эксплуатации системы обеспечивает 1 ден. ед. дохода.

7.3 Управляемые марковские процессы

Управляемые случайные процессы наблюдаются в различных областях человеческой деятельности. Рассмотрим, например, планирование работы промышленного предприятия. В начале каждого периода планирования, исходя из достигнутого состояния, намечается план на следующий период, причем планирование определяется с учетом имеющихся в наличии активных средств. Возможные способы использования активных средств называются стратегиями.

Пусть деятельность рассматриваемого предприятия, которое в дальнейшем будем называть системой, описывается с помощью марковского случайно-

го процесса. Различным стратегиям соответствуют разные вероятности переходов из одного состояния в другое, а также разные доходы. Для каждой стратегии соответствующие ей вероятности переходов и доходы будем отмечать с буквой k сверху, т. е. p_{ij}^k, r_{ij}^k. Заметим, что число стратегий, соответствующих каждому состоянию системы, должно быть конечным, хотя и различным для каждого из них.

Процесс с множеством стратегий, соответствующих каждому состоянию, называется управляемым марковским процессом.

Задача состоит в том, чтобы для каждого состояния E_i указать номер стратегии $d_i(m)$, которая будет использоваться на m-м шаге и обеспечивать максимальный средний доход g за один переход. Множество этих стратегий образует вектор $d(m)$:

$$d(m) = \begin{pmatrix} d_1(m) \\ d_2(m) \\ \cdots \\ d_N(m) \end{pmatrix}.$$

При последовательном анализе стратегий нетрудно заметить, что оптимальное поведение системы на $(m + 1)$-м шаге возможно лишь в том случае, если предыдущий шаг m был оптимальным. Используя соотношение (7.10), можем выразить это с помощью формулы

$$v_i(m+1) = \max_k (q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(m)), i = \overline{1, N},$$

или с учетом формулы (7.8) можно записать

$$v_i(m+1) = \max_k \sum_{j=1}^{N} p_{ij}^k (r_{ij}^k + v_j(m)) v_j(m), i = \overline{1, N}, \qquad (7.13)$$

где $v_i(m + 1)$, $v_j(m)$ обозначают общий ожидаемый доход для выбранного оптимального поведения.

Последовательное вычисление ожидаемых доходов и оптимальных стратегий с помощью выражения (7.13) заканчивается, когда вектор стратегий $d(m)$ перестает изменяться.

Пример. Рассмотрим условие предыдущего примера, определив для каждого состояния множество стратегий. Если система S находится в состоянии S_1, т. е. выпущена хорошая продукция, то возможны две стратегии: k = 1 – для продажи изделия рекламой не пользоваться; k = 2 – использовать рекламу.

Если система находится в состоянии S_2, т. е. выпущено плохое изделие, то возможны две стратегии: k = 1 – выпускать новую продукцию, не проводя

никаких исследований; $k = 2$ – выпуск новой продукции осуществлять после проведения соответствующих исследований.

Числовые значения вероятностей переходов и доходов представлены в таблице 7.4.

Таблица 7.4 – Числовые значения вероятностей переходов и доходов

Состояние i	Стратегии k	Вероятности переходов		Доходы		$q_i^k = \sum_{j=2}^{2} p_{ij}^k r_{ij}^k$
		p_{i1}^k	p_{i2}^k	r_{i1}^k	r_{i2}^k	
Хорошее изделие	1 – без рекламы	0,5	0,5	9	3	6
	2 – с рекламой	0,8	0,2	4	4	4
Плохое изделие	1 – без исследования	0,4	0,6	3	–7	–3
	2 – с исследованием	0,7	0,3	1	–19	–5

Положим $V(0) = 0$ и определим оптимальные стратегии, используя рекуррентное соотношение (7.13):

$$v_1(1) = \max\left(q_1^1 + \sum_{j=1}^{2} p_{1j}^1 v_j(0), q_1^2 + \sum_{j=1}^{2} p_{1j}^2 v_j(0) \right) = \max(6, 4) = 6,$$

$$v_2(1) = \max\left(q_2^1 + \sum_{j=1}^{2} p_{2j}^1 v_j(0), q_2^2 + \sum_{j=1}^{2} p_{2j}^2 v_j(0) \right) = \max(-3, -5) = -3.$$

Таким образом, на первой итерации оптимальной является стратегия $d = \binom{1}{1}$, и ей соответствует вектор доходов $V(1) = \binom{v_1(1)}{v_2(1)} = \binom{6}{-3}$.

Для $m = 2$ получаем:

$$v_1(2) = \max\left(q_1^1 + \sum_{j=1}^{2} p_{1j}^1 v_j(1), q_1^2 + \sum_{j=1}^{2} p_{1j}^2 v_j(1) \right) =$$

$$= \max(6 + (0{,}5 \cdot 6 + 0{,}5 \cdot (-3)), 4 + (0{,}8 \cdot 6 + 0{,}2 \cdot (-3))) = 8{,}2,$$

$$v_2(2) = \max\left(q_2^1 + \sum_{j=1}^{2} p_{2j}^1 v_j(1), q_2^2 + \sum_{j=1}^{2} p_{2j}^2 v_j(1) \right) = \max(-2{,}4, -1{,}7) = -1{,}7.$$

В этом случае оптимальным решением будет $d = \binom{2}{2}$. Результаты последующих шагов представлены в таблице 7.5.

Таблица 7.5 – Числовые значения вероятностей доходов и стратегий

i	m				
	0	1	2	3	4
$v_1(m)$	0	6	8,2	10,22	12,22
$v_2(m)$	0	–3	–1,2	0,23	2,23
$d_1(m)$	–	1	2	2	2
$d_2(m)$	–	1	2	2	2

После третьего шага процесс протекает так, что оптимальной является стратегия, обеспечивающая прирост дохода $g = 2$ ден. ед.

7.4 Оптимальные стратегии управляемых марковских процессов

7.4.1 Основные соотношения

Недостатком изложенного в подразделе 7.3 подхода является отсутствие критерия, позволяющего определять, получено ли оптимальное решение.

Рассмотрим метод нахождения оптимальных стратеги на основании использования асимптотических формул (7.12) для $v_i(m)$. Используя выражения (7.10) и (7.12), получаем соотношение

$$mg + v_i = q_i + \sum_{j=1}^{N} p_{ij}(v_j + (m-1)g).$$

Раскрывая скобки и учитывая нормировочное условие, получим

$$g + v_i = q_i + \sum_{j=1}^{N} p_{ij} v_j, \; i = \overline{1,N}. \qquad (7.14)$$

Таким образом, получена система N уравнений с $N + 1$ неизвестным $g, v_1, v_2, \ldots, v_N$. В дальнейшем при определении оптимальной стратегии нет необходимости знать абсолютные значения весов v_j $(j = \overline{1,N})$. Достаточно потребовать, чтобы разности $v_j - v_r$ сохраняли постоянное значение. А этого легко добиться, добавив к системе (7.14) одно уравнение, например $v_N = 0$ (можно положить любое из $v_i = 0$). Решая систему (7.14) вместе с уравнением $v_N = 0$, получаем набор $g^0, v_1^0, v_2^0, \ldots, v_N^0$, что можно будет в дальнейшем использовать для оценки различных стратегий. Рассмотрим, как это можно сделать.

Из соотношения (7.13) следует, что если придерживаться оптимального поведения до m-го шага, то для отыскания оптимальной стратегии на $(m + 1)$-м шаге в состоянии S_i достаточно максимизировать выражение

$$q_i^k + \sum_{j=1}^{N} p_{ij} v_j(m) \qquad (7.15)$$

относительно всех стратегий, соответствующих состоянию S_i.

Используя выражение (7.12), на основании соотношения (7.15) можно записать

$$q_i^k + \sum_{i=1}^{N} p_{ij}^k (mg + v_j).$$

Так как $\sum_{j=1}^{N} p_{ij} = 1$ и относительные веса v_i $(i = \overline{1,N})$ отличаются от абсолютных значений на одну и ту же величину, не зависящую от выбираемой стратегии k, из предыдущего выражения следует, что при поиске оптимальной стратегии для состояния S_i достаточно выбрать такую стратегию, которая обеспечивает максимум выражению

$$q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j. \qquad (7.16)$$

Таким образом, выражение (7.16) может служить критерием эффективности при выборе оптимальной стратегии в состоянии S_i.

7.4.2 Алгоритм поиска оптимальных стратегий

Весь процесс решения задачи может быть представлен в виде нижеследующего алгоритма.

Предварительный шаг. Положим все v_i $(i = \overline{1,N})$ и перейдем ко второму шагу.

Первый шаг (определение весов). Используя p_{ij} и q_i, определим доход g и относительные веса $v_1, v_2, \ldots, v_N$, решив систему уравнений (7.14), т. е.

$$\begin{cases} g + v_i = q_i + \sum_{j=1}^{N} p_{ij} v_j, \\ v_N = 0. \end{cases}$$

Переходим ко второму шагу.

Второй шаг (улучшение решения). Используя относительные веса, найденные на первом шаге, для каждого состояния S_i $(i = \overline{1,N})$, находим стратегию k, максимизирующую критерий (7.16):

$$q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j.$$

Затем принимаем эту стратегию за новое вектор-решение в состоянии S_i $(i = \overline{1,N})$. Если новое вектор-решение совпадает со старым, то задача решена, если нет, то заменяем q_i^k на q_i^{k*}, p_{ij}^k на p_{ij}^{k*} и переходим к первому шагу. Конец алгоритма.

Предварительный шаг можно выполнить иначе: зафиксировать некоторое вектор-решение и после этого перейти уже к выполнению первого шага, а не второго.

Для контроля за результатами получаемого решения можно использовать следующие свойства:

- каждое последующее решение (т. е. решение, полученное на последующей итерации) обеспечивает больший доход, чем предыдущие;
- процесс вычисления заканчивается, когда совпадут решения, полученные на двух последовательных итерациях.

Пример. Используя сформулированный алгоритм, определить оптимальные стратегии для предыдущего примера.

Итерация 1. Предварительный шаг. Полагаем $v_1 = v_2 = 0$. Переходим ко второму шагу.

Второй шаг. Вычисляем:

$$q_1^1 + \sum_{j=1}^{2} p_{1j}^1 v_j = 6 + 0 = 6,$$

$$q_1^2 + \sum_{j=1}^{2} p_{1j}^2 v_j = 4 + 0 = 4,$$

$$q_1^k = \begin{pmatrix} 6 \\ 4 \end{pmatrix},$$

$$q_1^2 + \sum_{j=1}^{2} p_{1j}^1 v_j = -3 + 0 = -3,$$

$$q_2^2 + \sum_{j=1}^{2} p_{1j}^2 v_j = -5 + 0 = -5,$$

$$q_1^k = \begin{pmatrix} -3 \\ -5 \end{pmatrix}.$$

Тогда $d = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $q = \begin{pmatrix} 6 \\ -3 \end{pmatrix}$, $P = \begin{pmatrix} 0{,}5 & 0{,}5 \\ 0{,}4 & 0{,}6 \end{pmatrix}$. Переходим к первому шагу.

Итерация 2. Первый шаг. Составляем и решаем систему:

$$\begin{cases} g + v_1 = 6 + 0{,}5v_1 + 0{,}5v_2, \\ g + v_2 = -3 + 0{,}4v_1 + 0{,}6v_2, \\ v_2 = 0. \end{cases}$$

Тогда $g = 1$, $v_1 = 10, v_2 = 0$. Переходим ко второму шагу.

Второй шаг. Для вычисления критерия и определения оптимальной стратегии составим таблицу 7.6.

Таблица 7.6 – Таблица для выбора оптимальных стратегий

i	k	$q_i^k + \sum_{j=1}^{N} p_{ij}^k \cdot v_j$
1	1 2	$6 + 0{,}5 \cdot 10 + 0{,}5 \cdot 0 = 11$ $4 + 0{,}8 \cdot 10 + 0{,}2 \cdot 0 = 12$
2	1 2	$-3 + 0{,}4 \cdot 10 + 0{,}6 \cdot 0 = 1$ $-5 + 0{,}7 \cdot 10 + 0{,}3 \cdot 0 = 2$

Получаем $d = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$. Решение не совпадает с предыдущим, поэтому полагаем $P = \begin{pmatrix} 0{,}8 & 0{,}2 \\ 0{,}7 & 0{,}3 \end{pmatrix}$, $q = \begin{pmatrix} 4 \\ -5 \end{pmatrix}$ и переходим к первому шагу.

Итерация 3. Первый шаг. Составляем и решаем систему:

$$\begin{cases} g + v_1 = 4 + 0{,}8v_1 + 0{,}2v_2, \\ g + v_2 = -5 + 0{,}7v_1 + 0{,}3v_2, \\ v_2 = 0. \end{cases}$$

Тогда $g = 2$, $v_1 = 10, v_2 = 0$. Переходим ко второму шагу.

Второй шаг. Так как v_1, v_2 случайно оказались теми же, что и на предыдущей итерации, в результате оценки стратегий получаем: $d = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$, т. е. результаты выполнения итерации 3 совпадают с результатами итерации 2.

Следовательно, решение $d = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$ является оптимальным, ему соответствует доход $g = 2$.

Список использованных источников

1 Афанасьев, М. Ю. Исследование операций в экономике: модели, задачи, решения : учеб. пособие / М. Ю. Афанасьев, Б. П. Суворов. – М. : ИНФРА-М, 2003. – 444 с.

2 Вагнер, Г. Основы исследования операций. В 3 т. / Г. Вагнер ; пер. с англ. – М. : Мир, 1972. – Т.1 – 336 с. ; Т.2 – 488 с. ; Т.3 – 503 с.

3 Вентцель, Е. С. Исследование операций. Задачи, принципы, методология : учеб. пособие для вузов / Е. С. Вентцель. – 3-е изд., стер. – М. : Дрофа, 2004. – 208 с.

4 Интрилигатор, М. Математические методы оптимизации и экономическая теория / М. Интрилигатор ; пер. с англ. – М. : Айрис-пресс, 2002. – 576 с.

5 Исследование операций в экономике : учеб. пособие для вузов / Н. Ш. Кремер [и др.] ; под ред. проф. Н. Ш. Кремера. – М. : 2005. – 407 с.

6 Конюховский, П. В. Математические методы исследования операций в экономике / П. В. Конюховский. – СПб. : Питер, 2000. – 208 с.

7 Кристофидес, Н. Теория графов. Алгоритмический подход / Н. Кристофидес. – М. : Мир, 1978. – 430 с.

8 Лабскер, Л. Г. Вероятностное моделирование в финансово-экономической области / Л. Г. Лабскер. – М. : Альпина, 2002. – 224 с.

9 Майника, Э. Алгоритмы оптимизации на сетях и графах / Э. Майника. – М. : Мир, 1981. – 322 с.

10 Ховард, Р. А. Динамическое программирование и марковские процессы / Р. А. Ховард ; пер. с англ. В. В. Рыкова ; под ред. Н. П. Бусленко. – М. : Советское радио, 1964. – 189 с.

11 Экономико-математические методы и модели : учеб. пособие для студ. эконом. спец. вузов / Н. И. Холод [и др.] ; под ред. А. В. Кузнецов. – Минск : БГЭУ, 2000. – 412 с.

Св. план 2016, поз. 71

Учебное издание

Пинчук Татьяна Георгиевна
Поттосина Светлана Анатольевна

ИССЛЕДОВАНИЕ ОПЕРАЦИЙ В ЭКОНОМИКЕ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *Е. С. Юрец*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *Е. Г. Бабичева*

Подписано в печать 15.11.2017. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 6,86. Уч.-изд. л. 7,0. Тираж 100 экз. Заказ 413.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6