Раздел 8. Математические методы исследования операций

8.1.Введение

Сегодня теория исследования операций является важным инструментом при принятии решения в различных отраслях промышленности. Одной из первых отечественных работ, которую можно отнести к исследованию операций, была работа Л.В.Канторовича «Математические методы организации и планирования производства» (1939г). В зарубежной литературе начало развития исследования операций связывают с работой Дж. Данцинга, посвященной решению линейных экстремальных задач (1947г.). Так или иначе, но Вторая мировая война заставила обратить особое внимание на задачи, которые сейчас относят к области исследования операций. После окончания войны идеи были перенесены в гражданскую сферу, но, конечно, не потеряли своей актуальности и для военных.

Под термином «***исследование операций»*** мы будем понимать *комплексную математическую дисциплину, которая занимается построением, анализом и применением математических моделей принятия оптимальных решений*. Следует отметить, что на практике всегда учитываются и другие разнообразные факторы, не имеющие числового выражения, однако информация, полученная в процессе исследования математических моделей, является основой для принятия окончательного решения.

Предполагая наличие базовых знаний по курсам «Высшая математика», «Дискретная математика» и «Теория оптимизации», в настоящем изложении мы будем рассматривать следующие разделы теории исследования операций: «Линейное программирование», «Целочисленное программирование», «Транспортная задача», «Сетевые модели» и «Основы теории игр». Задачей настоящего раздела является ***ознакомление*** студентов ***с основами курса*** «Математические методы исследования операций». Освоение данного материала позволит студентам претендовать на минимальные положительные оценки на экзамене.

***

Может сложиться впечатление, что исследование операций является чисто математической дисциплиной, однако оно не совсем верно. Несмотря на все разнообразие задач, традиционно реализация методов исследования операций на практике включает следующие этапы:

1. Формализация
2. Построение математической модели
3. Решение задачи, сформулированной на основе математической модели
4. Проверка адекватности
5. Реализация решения

Центральное внимание в данной главе будет уделено именно третьему этапу, поскольку только этот этап может быть точно определен, а действия будут следовать строгой математической теории. На остальных этапах в реальных задачах кроме математики зачастую применяют дополнительно психологию, социологию и различные поведенческие науки.

Очевидно, что исследование операций неотрывно связано с математическим моделированием. Построение математической модели произвольной задачи исследования операций всегда начинается с определения переменных, описания множества допустимых

решений и критериев оптимальности, т.е. признаков по которым проводится сравнительная оценка допустимых решений и выбор наилучшего решения.

Пример 1. (Х. Таха). *Рассмотрим работу предприятия, производящего краску для внутренних и наружных работ из сырья двух типов:* M_1 *и* M_2. *Данные, влияющие на выпуск продукции, представлены в виде следующей таблицы*:

	Расход сырья (в тоннах) на тонну краски		Максимально возможный ежедневный расход
	для наружных работ	для внутренних работ	
Сырье M_1	6	4	24
Сырье M_2	1	2	6
Доход (в тыс. дол.) на тонну краски	5	4	

Отдел маркетинга компании ограничил ежедневное производство краски для внутренних работ до 2т. (из-за отсутствия надлежащего спроса), а также поставил условие, чтобы ежедневное производство краски для внутренних работ не превышало более чем на тонну аналогичный показатель производства краски для внешних работ. Компания хочет определить оптимальное (наилучшее) соотношение между видами выпускаемой продукции для максимизации общего ежедневного дохода. Необходимо составить математическую модель рассматриваемой задачи.

Первый шаг создания модели – определение переменных задачи. В задаче необходимо определить объемы дневного производства краски разных типов, поэтому и обозначим эти объемы в качестве переменных: x_1 - ежедневный объем производства краски для наружных работ, x_2 - ежедневный объем производства краски для внутренних работ.

Следующим шагом является определение целевой функции, в качестве которой целесообразно выбрать ежедневный доход, получаемый от реализации x_1 и x_2 тонн краски: $f(x) = 5x_1 + 4x_2$. Компания хочет получить наибольший доход, поэтому перед нами стоит задача максимизации целевой функции.

Остается теперь сформулировать в математической форме ограничения, которых придерживается компания при выпуске краски. Очевидно, что объем ежедневного использования сырья каждого вида не может превышать соответствующих запасов: $6x_1 + 4x_2 \le 24$ (сырье M_1) и $x_1 + 2x_2 \le 6$ (сырье M_2).

Существует еще два условия, налагаемые отделом маркетинга. Одно из них ограничивает выпуск краски для внутренних работ: $x_2 \le 2$, другое – вводит соотношение между объемами выпускаемых видов краски: $x_2 \le 1 + x_1$.

В данной задаче есть еще одно неявное условие – объемы выпуска продукции не могут быть отрицательной величиной, т.е. переменные неотрицательные: $x_i \le 0, i = 1,2$.

Окончательно, построенная математическая модель задачи имеет следующий вид:

$$f(x) = 5x_1 + 4x_2 \to \max$$
$$6x_1 + 4x_2 \le 24,\ x_1 + 2x_2 \le 6$$
$$x_2 \le 2,\ x_1 - x_2 \ge -1$$
$$x_i \le 0, i = 1,2$$

8.2. Линейное программирование

Линейное программирование объединяет в себе методы оптимизации моделей, в которых целевые функции строго линейны, а область допустимых решений – выпуклый многогранник. Благодаря своей простоте и компьютерной эффективности алгоритмы линейного программирования нашли свое применение как самостоятельно, так и в составе более сложных типов моделей и задач.

8.2.1 Постановка задач

Определение. Задачей линейного программирования ***в общей форме*** мы будем называть следующую задачу:

$$f = (c, x) \to extr$$
$$D = \begin{cases} A_1 x \le b_1 \\ A_2 x = b_2 \\ A_3 x \ge b_3 \\ x \ge 0 \end{cases} \quad (1)$$

где $A_1 : m \times n$, $A_2 : l \times n$, $A_3 : k \times n$, $x \in \Re^n$, $c \in \Re^n$, $b_1 \in \Re^m$, $b_2 \in \Re^l$, $b_3 \in \Re^k$.

Определение. Задачей линейного программирования ***в стандартной форме*** мы будем называть

$$f = (c, x) \to extr$$
$$D = \begin{cases} Ax = b, b \ge 0 \\ x \ge 0 \end{cases} \quad (2)$$

Определение. Под ***размерностью задачи*** будем понимать сумму числа неизвестных и числа ограничений системы задачи линейного программирования.

Теорема. *Любую задачу линейного программирования в общей форме можно привести к эквивалентной задаче в стандартной форме, при этом размерность задачи может увеличиться.*

Пример 2. *Рассмотрим следующую задачу линейного программирования в общей форме, которую мы построили в примере 1:*

$$f = 5x_1 + 4x_2 \to \max$$
$$D = \begin{cases} 6x_1 + 4x_2 \le 24 \\ x_1 + 2x_2 \le 6 \\ x_2 \le 2 \\ x_1 - x_2 \ge -1 \\ x_1, x_2 \ge 0 \end{cases}$$

Необходимо привести данную задачу к задаче линейного программирования в стандартной форме.

Для приведения задачи к стандартной форме в первую очередь необходимо преобразовать правую часть ограничений, чтобы все компоненты правой части были неотрицательные. Для этого умножим 4е неравенство на (-1):

$$f = 5x_1 + 4x_2 \to \max$$
$$D = \begin{cases} 6x_1 + 4x_2 \le 24 \\ x_1 + 2x_2 \le 6 \\ x_2 \le 2 \\ -x_1 + x_2 \le 1 \\ x_1, x_2 \ge 0 \end{cases}$$

Для того, чтобы неравенства преобразовать в равенства необходимо ввести вспомогательные переменные, имеющие экономический смысл, например, как объем невыработанного за день остаток сырья того или иного вида. Все неравенства системы имеют знак «$\le$», тогда в каждом из основных ограничений введем свою новую переменную $y_k, k=\overline{1,4}$. Очевидно, что $y_k \ge 0, k=\overline{1,4}$.

$$f = 5x_1 + 4x_2 \to \max$$

$$D = \begin{cases} 6x_1 + 4x_2 + y_1 = 24 \\ x_1 + 2x_2 + y_2 = 6 \\ x_2 + y_3 = 2 \\ -x_1 + x_2 + y_4 = 1 \\ x_1, x_2 \ge 0 \\ y_k \ge 0, k = \overline{1,4} \end{cases}$$

Рассмотрим новый вектор $z = (x_1, x_2, y_1, y_2, y_3, y_4) \in \Re^6$, $z \ge 0$, тогда целевую функцию можно представить в виде $f = (c, z) = 5x_1 + 4x_2 + 0y_1 + 0y_2 + 0y_3 + 0y_4$, а основные ограничения – как равенство

$$Az = b,$$

где $b = (24,6,2,1) \in \Re^4$,

$$A = \begin{pmatrix} 6 & 4 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Следовательно, мы свели исходную задачу в общей форме к задаче линейного программирования в стандартной форме. Отметим, что данное преобразование не прошло бесследно, и размерность нашей задачи возросла до 10.

8.2.2 Графический метод

Для решения задач размерностью $m \times 2$ зачастую применяется графический метод, который позволяет относительно просто найти оптимальное решение. В то же время графический метод позволяет понять качественные особенности задачи линейного программирования, характерные для любой размерности пространства переменных и лежащие в основе численных методов ее решения.

Рассмотрим следующую задачу линейного программирования:

$$f = (c, x) = c_1 x_1 + c_2 x_2 \to extr$$

$$D = \begin{cases} Ax \le b \\ x_1, x_2 \ge 0 \end{cases}$$

Как видно, каждое из основных ограничений может быть представлено в виде $\alpha x_1 + \beta x_2 \le \gamma$, а множество точек $x \in \Re^2$, удовлетворяющих каждому из этих ограничений есть одна из полуплоскостей, на которые прямая $\alpha x_1 + \beta x_2 = \gamma$ разбивает плоскость $\Re^2$. Чтобы определить нужную полуплоскость, достаточно в проверить справедливость неравенства для произвольной точки (x_1^0, x_2^0), не лежащей на указанной прямой.

Очевидно, что при данной постановке задачи допустимое множество $D \subset \Re^2$ представляет собой многоугольник (не обязательно замкнутый), образованный пересечением построенных полуплоскостей. Линии уровня функции $f(x) = (c, x)$ (напомним, что линией уровня называется множество $\{\lambda \in \Re : f(x) = \lambda\}$) образуют семейство параллельных прямых. При этом градиент целевой функции $\operatorname{grad} f(x) = c$, т.е. всюду одинаков и является нормалью каждой из данных прямых.

Таким образом, поиск решения задачи сводится к нахождению максимального (или минимального) значения λ^* среди всех таких λ, для которых линия уровня имеет непустое пересечение с D. При этом все точки $\{x \in D : f(x) = \lambda^*\}$ образуют множество решений задачи. Заметим, что в случае неограниченной задачи множество решений задачи может не содержать ни одной точки.

Из графического представления становится очевидна характерная особенность задачи линейного программирования (при $c \neq 0$): *если ее решение существует, то оно достигается обязательно на границе*.

Пример 3. *При помощи графического метода решить задачу линейного программирования, построенную в примере 1*:

$$f = 5x_1 + 4x_2 \to \max$$

$$D = \begin{cases} 6x_1 + 4x_2 \le 24 \\ x_1 + 2x_2 \le 6 \\ x_2 \le 2 \\ -x_1 + x_2 \le 1 \\ x_1, x_2 \ge 0 \end{cases}$$

Вследствие прямых ограничений $x_1, x_2 \ge 0$ все построения мы производим в первом квадранте. Построим необходимые полуплоскости и выделим множество точек, удовлетворяющие всем неравенствам (на рисунке – заштрихованный многоугольник). После этого построим прямую, направляющий вектор которой есть $\vec{c} = (5,4)$. Заметим, что линии уровня являются перпендикулярами к данной прямой. Для того, чтобы найти оптимальное решение данной задачи нам необходимо определить максимальное λ^*, как бы перемещая прямую (на рисунке 8.1. изображена пунктиром) в направлении, которое задает вектор $\vec{c} = (5,4)$.

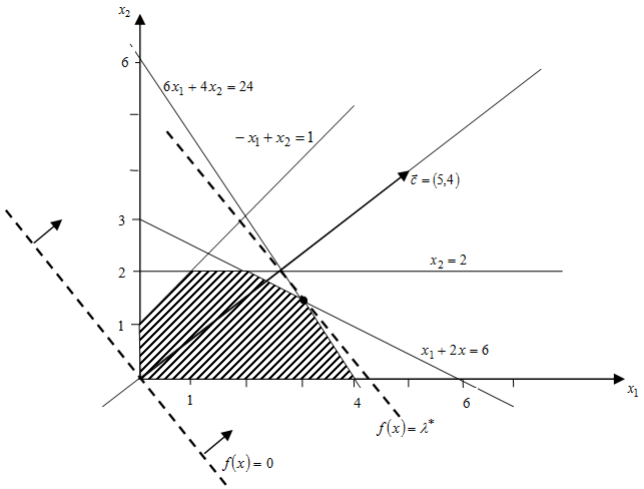


Рис. 8.1. Графический метод решения задачи линейного программирования из примера 1.

В рассматриваемой задаче $\lambda^* = 21$, а множество оптимальных решений будет состоять лишь из одной угловой точки $\left(3, \frac{3}{2}\right)$, которая и задаст оптимальное соотношение выпускаемой продукции.

8.2.3 Экстремальные свойства угловых точек. Базисные решения

Определение. Решение системы (2) называется ***базисным***, если множество вектор-столбцов $\{A_i\}_{x_i > 0}$ матрицы A, отвечающих неотрицательным компонентам вектора x, линейно независимы.

Определение. Решение системы (2) называется ***невырожденным базисным*** решением, если множество вектор-столбцов $\{A_i\}_{x_i>0}$ образуют базис пространства $\Re^m$.

Теорема (о базисности). *Для того, чтобы точка $\tilde{x} \in D$ была угловой точкой множества допустимых решений D задачи (2), необходимо и достаточно, чтобы $\tilde{x}$ было базисным решением.*

Определение. Задачей линейного программирования ***в канонической форме*** мы будем называть такую задачу в стандартной форме, у которой матрица $A=(E,\gamma)$, $E=\{e_i\}_{i=1}^m$ с точностью до перестановки строк и (или) столбцов:

$$\begin{aligned} &f=(c,x)\to extr \\ &D=\begin{cases} Ax=b, b\ge 0 \\ x\ge 0 \end{cases} \\ &A=(E,\gamma),\ \gamma:(n-m)\times m,\ E:m\times m \end{aligned} \tag{3}$$

Определение. ***Каноническим решением*** задачи (3) будем называть допустимое решение $\bar{x}=(b,0)$.

Теорема. *Каноническое решение задачи (3) является невырожденным базисным решением.*

8.2.4 Метод Жордана-Гаусса

Метод Жордана-Гаусса предназначен для последовательного нахождения канонических решений задачи линейного программирования, следовательно, и угловых точек области допустимых решений D.

Определение. ***Базисными переменными*** будем называть те компоненты x_i, которым соответствуют базисные вектор-столбцы матрицы A, т.е. $A_i=e_i\in\Re^m$. Остальные переменные мы будем называть ***небазисными***.

Переход к новой канонической форме по методу Жордана-Гаусса представляет собой процедуру последовательного введения в базис некоторой небазисной переменной.

Алгоритм (метод Жордана-Гаусса).

Для задачи линейного программирования в канонической форме (3) алгоритм состоит из следующих этапов:

1. Определить множества индексов, которые соответствуют базисным и небазисным переменным: $Б=\{i: x_i - базисная\}$, $H=\{i: x_i - не\ базисная\}$.
2. Выбрать некоторое $k\in H$ такое, что в соответствующем вектор-столбце A_k имеется хотя бы один положительный элемент $a_{i_k}>0$.
3. Определить ведущий элемент $a_{l_k}>0$ преобразования Жордана-Гаусса из соотношения:

$$\theta=\min_{a_{i_k}>0}\left\{\frac{b_i}{a_{i_k}}\right\}=\frac{b_l}{a_{l_k}}.$$

4. Исключить переменную x_k из базиса по методу Гаусса с ведущим элементом a_{l_k}:

$$\tilde{a}_{l_k}=\frac{a_{l_j}}{a_{l_k}};\ \tilde{b}_l=\frac{b_l}{a_{l_k}},\ i=l,k=1..n$$

$$\tilde{a}_{i_j}=a_{i_j}-\frac{a_{l_j}}{a_{l_k}}a_{i_k};\ \tilde{b}_i=b_i-\frac{b_l}{a_{l_k}}a_{i_k},\ i\ne l,\ j=1..m.$$

5. Найти каноническое решение полученной задачи по формуле

$$\bar{x}_i = \begin{cases} \tilde{b}_j, i \in Б, A_i = e_j \\ 0, i \in H \end{cases}.$$

Пример 4. *При помощи алгоритма Жордана-Гаусса и связи базисного решения с угловыми точками найти 2 угловые точки области допустимых решений задачи из примера 1*:

$$f = 5x_1 + 4x_2 \to \max$$

$$D = \begin{cases} 6x_1 + 4x_2 \le 24 \\ x_1 + 2x_2 \le 6 \\ x_2 \le 2 \\ -x_1 + x_2 \le 1 \\ x_1, x_2 \ge 0 \end{cases}$$

В примере 2 было показано, что рассматриваемая задача может быть сведена к задаче в стандартной форме с матрицей

$$A = \begin{pmatrix} 6 & 4 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Данная матрица системы содержит в себе единичную матрицу необходимой размерности, поэтому полученная форма задачи является канонической и к задаче может быть применен метод Жордана-Гаусса. Для нахождения угловых точек необходимо найти различные канонические формы задачи и выписать их канонические решения. Запишем задачу в виде расширенной матрицы системы (матрицы и вектора правой части):

$$\left(\begin{array}{cccccc|c} 6 & 4 & 1 & 0 & 0 & 0 & 24 \\ 1 & 2 & 0 & 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & 0 & 1 & 0 & 2 \\ -1 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

Каноническое решение, отвечающее данной канонической форме, есть $\bar{x}^0 = (0,0,24,6,2,1)$ и ему соответствует угловая точка $(0,0)$. Для получения следующей канонической формы определим множества $Б = \{3,4,5,6\}$, $H = \{1,2\}$ и выберем один из небазисных вектор-столбцов матрицы, например, $k = 1 \in H$ (выбор осуществляется произвольно с тем условием, чтобы среди элементов столбца был хотя бы один положительный элемент). Для определения ведущего элемента вычислим $\theta = \min\limits_{a_{i1} > 0} \left\{ \frac{24}{6}, \frac{6}{1} \right\} = \frac{b_1}{a_{11}} = 4$, следовательно, $l = 1$ и ведущим элементом будет $a_{11} = 6 > 0$.

Преобразовываем матрицу, используя метод Гаусса с ведущим элементом a_{11} - делим первую строку на ведущий элемент и при помощи элементарных преобразований с этой строкой делаем первый столбец базисным:

$$\left(\begin{array}{cccccc|c} 6 & 4 & 1 & 0 & 0 & 0 & 24 \\ 1 & 2 & 0 & 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & 0 & 1 & 0 & 2 \\ -1 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right) \sim \left(\begin{array}{cccccc|c} 1 & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 4 \\ 1 & 2 & 0 & 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & 0 & 1 & 0 & 2 \\ -1 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right) \sim \left(\begin{array}{cccccc|c} 1 & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 4 \\ 0 & \frac{4}{3} & -\frac{1}{6} & 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 & 1 & 0 & 2 \\ 0 & \frac{5}{3} & \frac{1}{6} & 0 & 0 & 1 & 5 \end{array}\right)$$

Полученная задача так же имеет каноническую форму, но с другим распределением базисных столбцов. Каноническое решение есть $\bar{x}^1 = (4,0,0,2,2,5)$ и ему соответствует угловая точке $(4,0)$.

Следовательно, используя метод Жордана-Гаусса и связь базисного решения с угловыми точками, мы нашли 2 угловые точки области допустимых решений: $(0,0)$ и $(4,0)$.

8.2.5 Теоремы линейного программирования

Теорема (экстремальное свойство угловых точек). *Если целевая функция $f(x)$ принимает максимальное значение в некоторой точке множества допустимых решений D, то она принимает это значение и в некоторой угловой точке данного множества.*

Теорема. *Если целевая функция $f(x)$ принимает максимальное значение в нескольких точках множества D, то она принимает это значение в любой точке, являющейся их выпуклой комбинацией.*

Теорема (о приращении линейного функционала). *В задаче линейного программирования в канонической форме (3) имеет место формула приращений:*

$$f(\bar{x}_1) = f(\bar{x}_0) + \theta \cdot \Delta_k , \qquad (4)$$

где $\bar{x}_0$ - угловая точка D, заданная исходной канонической формой;

$\bar{x}_1$ - угловая точка, полученная в результате перехода к новой канонической форме;

Δ_k - оценка k-й небазисной переменной, введенной в базис;

$\theta \geq 0$ - определяется алгоритмом Жордана-Гаусса.

8.2.6 Симплекс-метод

Симплексный алгоритм позволяет производить целенаправленный перебор вершин многогранного множества ограничений.

Алгоритм симплекс-метода в задаче на min (max)

Для задачи линейного программирования в канонической форме (3) алгоритм состоит из следующих этапов:

1. Составить вектор $\tilde{c} \in \Re^m$, который представляется теми координатами вектора цены c, которые соответствуют базисным векторам-столбцам $A_i = e_i$, составленным в порядке следования векторов канонического базиса пространства $\Re^m$.

Вычислить оценки небазисных переменных:

$$\Delta_k = c_k - (\tilde{c}, A_k), \ k \in H$$

2. Проверить, существует ли $\Delta_k < 0$ $(\Delta_k > 0)$, такое, что $A_k \leq 0$. Тогда задача линейного программирования неограниченна:

$\inf\limits_D f(x) = -\infty$ $(\sup\limits_D f(x) = +\infty)$.

3. Проверить условие $\Delta_k \geq 0$ $(\Delta_k \leq 0)$ для всех небазисных переменных. Тогда данная каноническая форма определяет экстремум:

$f(\bar{x}) = \min\limits_D f(x)$ $(f(\bar{x}) = \max\limits_D f(x))$,

где $\bar{x} = (\tilde{c}, b)$

4. Определить номер $k \in H$ ведущего столбца в преобразовании Жордана-Гаусса:

$\max\limits_{\Delta_k < 0} |\Delta_i| = |\Delta_k|$ $(\max\limits_{\Delta_k > 0} |\Delta_i| = |\Delta_k|)$

5. Применить преобразование Жордана-Гаусса с k-м ведущим столбцом и перейти к выполнению пункта 1.

Пример 5. *При помощи алгоритма симплекс-метода найти оптимальное решение следующей задачи:*

$$f = 2x_1 + 7x_2 \to \max$$
$$D = \begin{cases} -2x_1 + 3x_2 \le 14 \\ x_1 + x_2 \le 8 \\ x_1, x_2 \ge 0 \end{cases}$$

Преобразуем задачу к стандартной форме, введя дополнительные переменные:

$$f = 2x_1 + 7x_2 + 0y_1 + 0y_2 \to \max$$
$$D = \begin{cases} -2x_1 + 3x_2 + y_1 \le 14 \\ x_1 + x_2 + y_2 \le 8 \\ x_1, x_2, y_1, y_2 \ge 0 \end{cases}$$

Расширенная матрица системы имеет вид:

$$\left(\begin{array}{cccc|c} -2 & 3 & 1 & 0 & 14 \\ 1 & 1 & 0 & 1 & 8 \end{array}\right)$$

Нетрудно убедиться, что полученная стандартная форма является канонической, и мы можем применить к данной задаче симплекс-метод. Вычислим вектор $\tilde{c}$ и оценки небазисных переменных:

1. базисными столбцами являются 3 и 4, следовательно $\tilde{c} = (0,0)$.
2. $\Delta_1 = 2 - (-2 \cdot 0 + 1 \cdot 0) = 2 > 0$
   $\Delta_2 = 7 - (3 \cdot 0 + 1 \cdot 0) = 7 > 0$

Так как рассматриваемая задача является задачей максимизации, то номер ведущего столбца выбирается как индекс оценки небазисной переменной с максимальным положительным значением: $\Delta_k = \max(2,7) = \Delta_2$, $k = 2$. Ведущая строка находится из условия $\theta = \min\limits_{a_{ik}>0}\left\{\frac{14}{3}, \frac{8}{1}\right\} = \frac{b_1}{a_{12}} = \frac{14}{3}$, следовательно, $l = 1$ и ведущим элементом будет $a_{12} = 3 > 0$. Преобразовываем матрицу при помощи Жордана-Гаусса и получаем:

$$\left(\begin{array}{cccc|c} -2 & 3 & 1 & 0 & 14 \\ 1 & 1 & 0 & 1 & 8 \end{array}\right) \sim \left(\begin{array}{cccc|c} -\frac{2}{3} & 1 & \frac{1}{3} & 0 & \frac{14}{3} \\ 1 & 1 & 0 & 1 & 8 \end{array}\right) \sim \left(\begin{array}{cccc|c} -\frac{2}{3} & 1 & \frac{1}{3} & 0 & \frac{14}{3} \\ \frac{5}{3} & 0 & -\frac{1}{3} & 1 & \frac{10}{3} \end{array}\right)$$

Для полученной новой канонической формы вычисляем вектор $\tilde{c}$ и оценки небазисных переменных:

1. базисными столбцами являются 2 и 4, следовательно $\tilde{c} = (7,0)$.
2. $\Delta_1 = 2 - \left(-\frac{2}{3} \cdot 7 + \frac{5}{3} \cdot 0\right) = \frac{20}{3} > 0$
   $\Delta_3 = 0 - \left(\frac{1}{3} \cdot 7 - \frac{1}{3} \cdot 0\right) = -\frac{7}{3} < 0$

Так как рассматриваемая задача является задачей максимизации, то номер ведущего столбца выбирается как индекс оценки небазисной переменной с максимальным положительным значением: $\Delta_k = \Delta_1$, $k = 1$. Ведущая строка находится из условия $\theta = \min\limits_{a_{ik}>0}\left\{\frac{10}{5}\right\} = \frac{b_2}{a_{21}}$, следовательно, $l = 2$ и ведущим элементом будет $a_{21} = \frac{5}{3} > 0$. Преобразовываем матрицу при помощи Жордана-Гаусса и получаем:

$$\left(\begin{array}{cccc|c} -\frac{2}{3} & 1 & \frac{1}{3} & 0 & \frac{14}{3} \\ \frac{5}{3} & 0 & -\frac{1}{3} & 1 & \frac{10}{3} \end{array}\right) \sim \left(\begin{array}{cccc|c} -\frac{2}{3} & 1 & \frac{1}{3} & 0 & \frac{14}{3} \\ 1 & 0 & -\frac{1}{5} & \frac{3}{5} & 2 \end{array}\right) \sim \left(\begin{array}{cccc|c} 0 & 1 & \frac{1}{5} & \frac{2}{5} & 6 \\ 1 & 0 & -\frac{1}{5} & \frac{3}{5} & 2 \end{array}\right)$$

Для полученной новой канонической формы вычисляем вектор $\tilde{c}$ и оценки небазисных переменных:

1. базисными столбцами являются 2 и 1, следовательно $\tilde{c} = (7,2)$.
2. $\Delta_3 = 0 - \left(\frac{1}{5} \cdot 7 - \frac{1}{5} \cdot 2\right) = -1 < 0$

$$\Delta_4 = 0 - \left(\frac{2}{5}\cdot 7 + \frac{3}{5}\cdot 2\right) = -4 < 0$$

В рассматриваемой задаче максимизации мы нашли каноническую форму, в которой все оценки небазисных переменных неположительные, следовательно, эта форма задает оптимальное решение:

$$x^* = (2,6), \quad f^* = f(x^*) = 2\cdot 2 + 7\cdot 6 = 46 .$$

8.2.7 Метод искусственных переменных

Для успешного применения алгоритма симплекс-метода необходима задача в канонической форме. Метод искусственных переменных позволяет построить каноническую форму задачи, которая имеет стандартную форму.

Рассмотрим задачу линейного программирования в стандартной форме (2) и следующую вспомогательную задачу:

$$\begin{aligned} &v = \sum_{i=1}^{m} u_i \to \min \\ &\mathrm{D} = \begin{cases} Ax + u = b, b \ge 0 \\ x, u \ge 0 \end{cases} \\ &\widetilde{A} = (A,E), u \in \Re^m, E{:}m \times m \end{aligned} \tag{5}$$

Очевидно, что в силу особенностей построения, задача (5) имеет каноническую форму, а, следовательно, к ней может быть применен симплекс-метод.

Теорема (метод искусственных переменных). *Рассмотрим пару задач линейного программирования (2) и (5):*

1) *если* $\mu = \min\limits_{\mathrm{D}} v > 0$, *то задача (2) недопустимая.*
2) *если* $\mu = \min\limits_{\mathrm{D}} v = 0$, *тогда оптимальная каноническая форма задачи (5), полученная при помощи симплекс-метода, задает каноническую форму задачи (2).*

8.2.8 Двойственные задачи

Определение. ***Симметричной парой*** двойственных задач называется следующая пара задач линейного программирования:

$$\begin{aligned} &f = (c,x) \to \max && g = (b,y) \to \min \\ &D = \begin{cases} Ax \le b \\ x \ge 0 \end{cases} && \mathrm{D} = \begin{cases} A^t y \ge c \\ y \ge 0 \end{cases} \end{aligned} \tag{6}$$

Правило построения симметричной пары двойственных задач:

1. Вектор цены исходной задачи задает вектор правой части двойственной задачи.
2. Вектор правой части исходной задачи задает вектор цены двойственной задачи.
3. Матрица коэффициентов двойственной задачи является транспонированной матрицей коэффициентов исходной задачи.
4. Каждая неотрицательная переменная исходной задачи задает ограничение вида $\ge$ двойственной задачи.
5. Каждое ограничение $\ge$ исходной задачи задает неотрицательную переменную двойственной задачи.

Достаточно часто на практике возникают и несимметричные пары двойственных задач:

$$\begin{aligned} &f = (c,x) \to \max && g = (b,y) \to \min \\ &D = \begin{cases} Ax = b \\ x \ge 0 \end{cases} && \mathrm{D} = \left\{A^t y \ge c\right\} \end{aligned} \tag{7}$$

и

$$f=(c,x)\to\max \qquad g=(b,y)\to\min$$
$$D=\{Ax\le b\} \qquad \mathrm{D}=\begin{cases}A^t y=c\\ y\ge 0\end{cases} \tag{8}$$

Теория двойственности в линейном программировании основывается на нескольких основных теоремах.

Теорема (основное неравенство). *Рассмотрим симметричную пару (6) двойственных задач. Для любых допустимых* $x\in D$ *и* $y\in\mathrm{D}$ *прямой и двойственной задач линейного программирования справедливо следующее неравенство:*

$$f(x)\le g(y).$$

Теорема (малая теорема двойственности). *Рассмотрим симметричную пару (6) двойственных задач. Если для некоторых допустимых* $x^*\in D$ *и* $y^*\in\mathrm{D}$ *прямой и двойственной задач выполняется* $f(x^*)=g(y^*)$, *тогда* x^* *и* y^* *- оптимальные решения соответствующих задач.*

Теорема (первая основная теорема). *Рассмотрим симметричную пару (6) двойственных задач. Если одна из задач пары имеет оптимальное решение, то и другая имеет оптимальное решение, причем экстремальные решения целевых функций равны; если же целевая функция одной из задач не ограничена, то система ограничений другой задачи недопустима.*

Теорема (вторая основная теорема). *Рассмотрим симметричную пару (6) двойственных задач. Для того, чтобы допустимые решения* $x^*\in D$ *и* $y^*\in\mathrm{D}$ *были оптимальны, необходимо и достаточно выполнение условий:*

$$\begin{aligned} x_j^*\cdot\left(\sum_{i=1}^{m}a_{ij}y_j^*-c_j\right)&=0,\quad j=1..n,\\ y_i^*\cdot\left(\sum_{j=1}^{n}a_{ij}x_j^*-b_i\right)&=0,\quad i=1..m\end{aligned} \tag{9}$$

Вторая основная теорема двойственности, позволяет использовать известное оптимальное решение одной из задач двойственной пары для нахождения оптимального решение другой задачи. Такая связь оптимальных решений часто позволяет упростить решение задачи линейного программирования.

8.3. Целочисленное линейное программирование

Общая постановка задачи целочисленного программирования отличается от общей постановки задачи линейного программирования наличием дополнительного условия – требованием целочисленности решений.

8.3.1 Метод ветвей и границ

На первый взгляд наиболее естественным методом решения задач целочисленного программирования является метод округления. Однако такой метод в большинстве случаев дает неоптимальное решение, а иногда попросту неприменим, поэтому разработаны специальные методы решения таких задач.

Наиболее известным комбинаторным методом решения задач целочисленного программирования является метод ветвей и границ (а точнее – целое семейство методов).

В основе метода ветвей и границ лежит процедура решения задачи линейного программирования с ослабленными ограничениями, соответствующей исходной задачи целочисленного программирования.

Алгоритм метода ветвей и границ общего случая для задачи целочисленного программирования. Для произвольной задачи целочисленного программирования можно выделить несколько основных этапов метода ветвей и границ:

1. *Задание начальных границ.* Задаем границу оптимального значения целевой функции (для задачи максимизации формально это будет нижняя граница -∞, а для задачи минимизации - верхняя граница +∞). Положим $i = 0$ (нулевая итерация).

2. *Зондирование и определение границ.* Выбираем для исследования i-ю подзадачу линейного программирования (i-ю ветвь, если представить процесс решения задачи как дерево) и решаем ее как обычную задачу линейного программирования. Возможно три ситуации:

   а) оптимальное решение целевой функции i-й подзадачи не может улучшить значение текущей нижней (верхней) границы;

   б) задача не имеет допустимых решений;

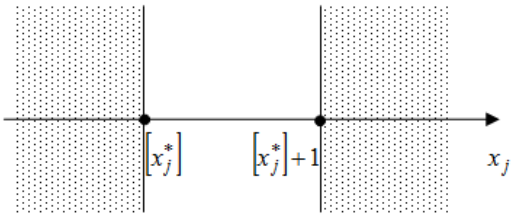
   в) решение i-й подзадачи приводит к лучшему допустимому целочисленному решению, которое улучшает текущую нижнюю (верхнюю) границу.

Считаем подзадачу прозондированной, если решение задачи линейного программирования является целочисленным.

Если i-я подзадача прозондирована (решение целочисленно), тогда нижняя (верхняя) граница изменяется только при условии, что найдено лучшее значение целочисленной задачи.

Если все подзадачи прозондированы, то оптимальное решение задачи целочисленного программирования найдено (им является решение, соответствующее текущей нижней (верхней) границе), иначе - переходим к ветвлению.

3. *Ветвление.* Рассматриваем оптимальное решение i-й подзадачи x^* и выбираем одну из не целочисленные компонент, например, x_j^*.



Из области допустимых решений исключаем область $\left([x_j^*],[x_j^*]+1\right)$, формируя тем самым две подзадачи линейного программирования, соответствующие ограничениям $x_j \le [x_j^*]$ и $x_j \ge [x_j^*]+1$. Далее переходим к пункту 2, выбрав одну из сформированных задач в качестве текущей.

8.3.2 Задача коммивояжера

Одной из известных задач, которую можно решать как задачу целочисленного линейного программирования, есть задача перевозки груза по критерию минимума пробега – задача коммивояжера.

Пусть имеется n городов $A_1, A_2, \cdots A_n$. Для каждого города i известная стоимость проезда $c_{ij} \ge 0$ из него в каждый город j. Необходимо составить план объезда городов таким образом, чтобы стоимость объезда городов была минимальная, и выполнялось условие, что в каждый город необходимо один раз въехать и из каждого выехать нужно только один раз. Вместо стоимости проезда можно задать расстояние между городами и тогда задача состоит в поиске плана объезда всех городов с минимальным расстоянием.

Очевидным способом решения такой задачи является метод перебора, заключающийся в исследовании всех $(n-1)!$ вариантов. Однако при достаточно больших n такая работа становится чрезвычайно трудной, поэтому для решения задачи коммивояжера применяют специальные методы.

Данную задачу можно записать как задачу целочисленного линейного программирования, если определить неизвестные следующим образом:

$$x_{ij} = \begin{cases} 1, \text{если город } j \text{ достижим из города } i \\ 0, \text{если город } j \text{ н достижим из города } i \end{cases}$$

Задав стоимости проезда c_{ij} из i-го города в j-й, математическая постановка имеет вид:

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

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = 1, i = 1..n \\ \sum_{i=1}^{n} x_{ij} = 1, j = 1..n \\ x_{ij} \in \{0,1\}, \text{решение должно быть циклом} \end{cases} .$$

Определение. ***Циклом*** называется маршрут объезда всех городов с условием въезда и выезда из каждого города только один раз. Цикл будем обозначать $1 \to 2 \to \cdots n \to 1$.

Определение. ***Подциклом*** называется маршрут объезда части городов с условием въезда и выезда из каждого города только один раз.

Данные задачи могут быть сведены в матрицу расстояний C, где в ячейке (i,j) записана величина стоимости проезда $c_{ij} \geq 0$ из i-го города в j-й (в ячейках (i,i) стоит ∞):

	A_1	A_2	…	A_n
A_1	∞	c_{12}	…	c_{1n}
A_2	c_{21}	∞	…	c_{2n}
…	…	…	∞	…
A_n	c_{n1}	c_{n2}	…	∞

Определение. Маршруты (циклы) называются ***аналогичными***, если в матрицах расстояний одинаковой размерности указанные циклы составлены путем выбора клеток, стоящих на одинаковых местах.

Определение. ***Константой приведения по строке (столбцу)*** матрицы расстояний называется минимальный элемент этой строки (столбца).

Для каждой строки матрицы расстояний C найдем константы приведения по строке $h_i, i = 1..n$ и сформируем матрицу $C' = \{c'_{ij}\}$, где $c'_{ij} = c_{ij} - h_i$. Для каждого столбца матрицы C' вычисляем константы приведения по столбцам $g_j, j = 1..n$ и сформируем матрицу $C'' = \{c''_{ij}\}$, где $c''_{ij} = c'_{ij} - g_j = c_{ij} - h_i - g_j$. Матрица C'' имеет в каждом столбце и каждой строке по крайней мере один нулевой элемент и называется ***приведенной матрицей***. Число $H_0 = \sum_{i=1}^{n} h_i + \sum_{j=1}^{n} g_j$ называется ***константой приведения матрицы***.

Теорема. *Длина цикла D матрицы C равна сумме длины аналогичного цикла D'' приведенной матрицы C'' и константы приведения матрицы:*

$$D = D'' + H_0$$

Из приведенной выше теоремы следует, что длина любого цикла $D \geq H_0$.

Как мы уже отмечали, в приведенной матрице есть, по крайней мере, один нулевой элемент в каждой строке и столбце. Предположим, что элемент $c''_{ij} = 0$.

Определение. ***Оценкой нуля по строке (столбцу)*** приведенной матрицы расстояний называется минимальный элемент этой строки (столбца) за исключением непосредственно оцениваемого нуля. ***Оценкой нуля*** называется величина равная сумме оценок этого нуля по строке и столбцу.

Вычислим оценки нуля $c''_{ij} = 0$ по i-й строке α_i и j-му столбцу β_j, после чего вычислим оценку нуля: $\theta_{ij} = \alpha_i + \beta_j$. Смысл оценки нуля в том, что θ_{ij} составляют минимально возможные потери из-за запрещения проезда из i-го города в j-й. Следовательно, при выборе шагов объезда городов необходимо брать клетку приведенной матрицы, в которой стоит нуль с наибольшей оценкой (или один из них), минимизируя возможные потери.

После выбора первого шага маршрута, например $i_0 \to j_0$, мы можем вычеркнуть строку i_0 (т.к. из города A_{i_0} мы уже выехали, а выехать из него можно только один раз) и столбец j_0 (т.к. в город A_{j_0} мы уже въехали, а въехать в город можно только один раз). После этих операций получается урезанная матрица размером $(n-1)\times(n-1)$. Заметим, что в новой матрице мы не можем выбирать клетку (j_0, i_0), потому что она приводит к выделению подцикла $i_0 \to j_0 \to i_0$ и нарушению требования объезда всех городов. Поэтому на место элемента $c''_{i_0 j_0}$ ставим ∞ и получаем новую матрицу C_1. Продолжаем решать задачу, принимая за исходную матрицу расстояний матрицу C_1. Причем константы приведения мы все время складываем, сохраняя информацию о длине найденного маршрута.

Когда получаем матрицу расстояний 2×2, выбора уже не остается, поэтому оставшиеся два шага включаем в цикл. Финишную матрицу можно и не приводить, а ее элементы прибавить к последней константе приведения. Значение окончательной константы приведения и будет являться длиной цикла D^*.

Найденный маршрут может быть не оптимален, поэтому необходимо проводить дополнительную проверку на оптимальность.

8.3.3 Метод ветвей и границ в задаче коммивояжера

Для проверки на оптимальность найденного маршрута задачи коммивояжера целесообразно воспользоваться методом ветвей и границ, применение которого позволяет перебирать для такой проверки не все $(n-1)!$ *возможных маршрутов.*

Запрещаем первый выбранный шаг $i_0 \to j_0$ проверяемого маршрута, для чего в клетку (i_0, j_0) матрицы C'' ставим ∞ и приводим матрицу. Очевидно, что константа приведения матрицы увеличится на оценку нуля, запрещенного нами: $H'_0 = H_0 + \theta_{i_0 j_0}$.

Возможны два случая:

1. $H'_0 \geq D^*$. Тогда от первого шага $i_0 \to j_0$ отказываться нельзя, т.к. любой маршрут, не содержащий этого шага, имеет длину, не меньшую H'_0, а следовательно не меньшую, чем длина полученного маршрута D^*.
2. $H'_0 < D^*$. В этом случае переходим к решению задачи с запрещенным шагом $i_0 \to j_0$. На каждом из этапов сравниваем получаемую константу приведения с D^* и если $H \geq D^*$, то проверка данной ветви закончена и первый выбранный маршрут лучше.

Далее переходим к запрещению второго шага (производим ветвление) и продолжаем проверку.

Если при проверке находится маршрут меньшей длины, чем D^*, то переходим к проверке уже этого маршрута. Проверка заканчивается, если проверены все ветви (все шаги маршрута) и более лучшего маршрута не найдено.

Пример 6. *Решить задачу коммивояжера для случая 5 городов со следующей матрицей расстояний:*

$$C = C_0 = \begin{pmatrix} \infty & 10 & 3 & 6 & 9 \\ 5 & \infty & 5 & 4 & 3 \\ 4 & 9 & \infty & 7 & 8 \\ 7 & 1 & 3 & \infty & 4 \\ 3 & 2 & 6 & 5 & \infty \end{pmatrix}.$$

Начнем с приведения матрицы, вычислив соответствующие константы приведения по строкам: $h_1 = 3$, $h_2 = 3$, $h_3 = 4$, $h_4 = 1$, $h_5 = 2$.

$$C_0' = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \end{array} \begin{array}{c} \begin{array}{ccccc} A_1 & A_2 & A_3 & A_4 & A_5 \end{array} \\ \begin{pmatrix} \infty & 7 & 0 & 3 & 6 \\ 2 & \infty & 2 & 1 & 0 \\ 0 & 5 & \infty & 3 & 4 \\ 6 & 0 & 2 & \infty & 3 \\ 1 & 0 & 4 & 3 & \infty \end{pmatrix} \end{array}$$

Вычисляем константы приведения по столбцам и формируем приведенную матрицу:
$g_1 = 0$, $g_2 = 0$, $g_3 = 0$, $g_4 = 1$, $g_5 = 0$,

$$C_0'' == \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \end{array} \begin{array}{c} \begin{array}{ccccc} A_1 & A_2 & A_3 & A_4 & A_5 \end{array} \\ \begin{pmatrix} \infty & 7 & 0 & 2 & 6 \\ 2 & \infty & 2 & 0 & 0 \\ 0 & 5 & \infty & 2 & 4 \\ 6 & 0 & 2 & \infty & 3 \\ 1 & 0 & 4 & 2 & \infty \end{pmatrix} \end{array},$$

$$H_0 = 14$$

Переходим к оценке нулей в приведенной матрице: $\theta_{13} = 4$, $\theta_{24} = 2$, $\theta_{25} = 3$, $\theta_{31} = 3$, $\theta_{42} = 2$, $\theta_{52} = 1$. Нуль с максимальной оценкой стоит в ячейке матрицы $(1,3)$, которой соответствует пара городов $A_1 \to A_3$, поэтому маршрут $1 \to 3$ включаем в наш цикл. Выполняя требование о въезде и выезде из каждого города только 1 раз, исключаем из матрицы первую строку (т.к. из города A_1 мы уже выехали) и третий столбец (в город A_3 мы уже въехали). В полученной матрице может возникнуть подцикл $1 \to 3 \to 1$, поэтому запрещаем переход из города A_3 в город A_1 (ставим в ячейку $(2,1)$ урезанной матрицы значение ∞):

$$C_1 = \begin{array}{c} \\ A_2 \\ A_3 \\ A_4 \\ A_5 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_4 & A_5 \end{array} \\ \begin{pmatrix} 2 & \infty & 0 & 0 \\ \infty & 5 & 2 & 4 \\ 6 & 0 & \infty & 3 \\ 1 & 0 & 2 & \infty \end{pmatrix} \end{array}.$$

Вычисляем константы приведения по строкам и столбцам и приводим матрицу:

$$C_1' = \begin{array}{c} \\ A_2 \\ A_3 \\ A_4 \\ A_5 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_4 & A_5 \end{array} \\ \begin{pmatrix} 2 & \infty & 0 & 0 \\ \infty & 3 & 0 & 2 \\ 6 & 0 & \infty & 3 \\ 1 & 0 & 2 & \infty \end{pmatrix} \end{array}, \quad C_1'' = \begin{array}{c} \\ A_2 \\ A_3 \\ A_4 \\ A_5 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_4 & A_5 \end{array} \\ \begin{pmatrix} 1 & \infty & 0 & 0 \\ \infty & 3 & 0 & 2 \\ 5 & 0 & \infty & 3 \\ 0 & 0 & 2 & \infty \end{pmatrix} \end{array}, \quad H_1 = 3.$$

Переходим к оценке нулей в приведенной матрице: $\theta_{24} = 0$, $\theta_{25} = 2$, $\theta_{34} = 2$, $\theta_{42} = 3$, $\theta_{42} = 2$, $\theta_{51} = 1$, $\theta_{52} = 0$. Нуль с максимальной оценкой стоит в ячейке, которой соответствует пара городов $A_4 \to A_2$, поэтому маршрут $4 \to 2$ включаем в наш цикл, получая $1 \to 3, 4 \to 2$. Выполняя требование о въезде и выезде из каждого города только 1 раз, исключаем из матрицы третью строку (т.к. из города A_4 мы уже выехали) и второй столбец (в город A_2 мы уже въехали). В полученной матрице может возникнуть подцикл $4 \to 2 \to 4$, поэтому запрещаем переход из города A_2 в город A_4 (ставим в ячейку $(2,4)$ урезанной матрицы значение ∞):

$$C_2 = \begin{array}{c} \\ A_2 \\ A_3 \\ A_5 \end{array} \begin{array}{c} \begin{array}{ccc} A_1 & A_4 & A_5 \end{array} \\ \begin{pmatrix} 1 & \infty & 0 \\ \infty & 0 & 2 \\ 0 & 2 & \infty \end{pmatrix} \end{array}.$$

Аналогично предыдущим случаям вычисляем константы приведения по строкам и столбцам и приводим матрицу:

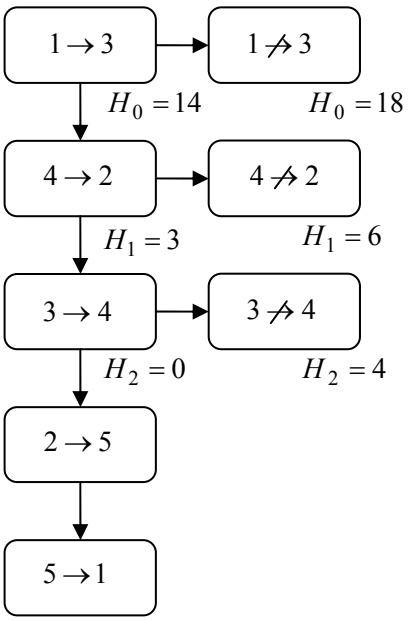
$$C_2'' = C_2' = \begin{array}{c} \\ A_2 \\ A_3 \\ A_5 \end{array} \begin{array}{c} \begin{array}{ccc} A_1 & A_4 & A_5 \end{array} \\ \begin{pmatrix} 1 & \infty & 0 \\ \infty & 0 & 2 \\ 0 & 2 & \infty \end{pmatrix} \end{array}, \quad H_2 = 0.$$

Переходим к оценке нулей в приведенной матрице: $\theta_{25} = 3$, $\theta_{34} = 4$, $\theta_{51} = 3$. Нуль с максимальной оценкой стоит в ячейке, которой соответствует пара городов $A_3 \to A_4$, поэтому маршрут $3 \to 4$ включаем в наш цикл, получая $1 \to 3 \to 4 \to 2$. Выполняя требование о въезде и выезде из каждого города только 1 раз, исключаем из матрицы вторую строку (т.к. из города A_3 мы уже выехали) и второй столбец (в город A_4 мы уже въехали). В полученной матрице может возникнуть подцикл $1 \to 3 \to 4 \to 2 \to 1$, поэтому запрещаем переход из города A_2 в город A_1 (ставим в ячейку $(2,1)$ урезанной матрицы значение ∞):

$$C_3 = \begin{array}{c} \\ A_2 \\ A_5 \end{array} \begin{array}{c} \begin{array}{cc} A_1 & A_5 \end{array} \\ \begin{pmatrix} \infty & 0 \\ 0 & \infty \end{pmatrix} \end{array}.$$

Получив матрицу размерности 2×2 нет смысла вычислять константы приведения по строкам и столбцам и приводить матрицу. Константа приведения в данном случае равна сумме конечных элементов матрицы: $H_3 = 0$. Оставшиеся два маршрута $2 \to 5$ и $5 \to 1$ включаем в цикл, получая цикл $1 \to 3 \to 4 \to 2 \to 5 \to 1$ длины $D^* = H_0 + H_1 + H_2 + H_3 = 17$.

Полученное решение может являться не оптимальным, поэтому выполним проверку оптимальности решения, отказываясь (запрещая) найденные маршруты и оценивая получаемую длину. Если длина какого-либо маршрута, полученного вследствие этой операции, будет меньшей чем D^*, тогда переходим к проверке оптимальности этого нового маршрута.



Как видно из дерева проверки решения, любое отклонение от найденного маршрута приводит к увеличению констант приведения и общей длины маршрута. Поэтому найденный нами маршрут $1 \to 3 \to 4 \to 2 \to 5 \to 1$ длины $D^* = 17$ является оптимальным.

8.4. Транспортная задача

Транспортная задача (Т-задача) является одной из самых известных специальных задач линейного программирования. Транспортную задачу можно решить симплекс-методом, но в силу ряда особенностей ее можно решить проще (для задач малой размерности). Проблема была впервые формализована французским математиком Гаспаром Монжем в 1781 году. Основное продвижение было сделано во время Великой Отечественной войны советским математиком Леонидом Канторовичем. Поэтому иногда эту задачу называют транспортной задачей Монжа-Канторовича.

8.4.1 Постановка задачи

Рассмотрим транспортную задачу по критерию минимума стоимости перевозок. Пусть имеется некоторое количество m поставщиков $A_i, i = 1..m$ однородного груза, у каждого из которых он сосредоточен в количестве $a_i, i = 1..m$ единиц, и n получателей этого груза $B_j, j = 1..n$, потребность которых в нем соответственно равна $b_j, j = 1..n$. Пусть $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$ (суммарное наличие груза равно суммарной его потребности). Предположим, что известна стоимость перевозки единицы груза c_{ij} от любого поставщика A_i к любому получателю B_j. Требуется так спланировать перевозки груза от поставщиков к получателям, чтобы суммарная их стоимость была наименьшей, причем весь груз от поставщиков должен быть вывезен и потребности всех получателей должны быть удовлетворены.

Определение. Транспортная задача называется ***замкнутой (сбалансированной)***, если выполняется условие $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$, иначе – задача называется ***открытой (несбалансированной).***

Перейдем к составлению математической модели задачи. Обозначим через x_{ij} количество груза, перевозимого от поставщика A_i получателю B_j. Всего от i-го поставщика

перевозится $\sum_{j=1}^{n} x_{ij}$ единиц груза, но по условию задачи необходимо вывезти весь груз от поставщика, следовательно $\sum_{j=1}^{n} x_{ij} = a_i$. Аналогично, к получателю B_j поступает груз в количестве $\sum_{i=1}^{m} x_{ij}$, причем из требования полностью удовлетворить потребности получателя получаем $\sum_{i=1}^{m} x_{ij} = b_j$. Суммарная стоимость перевозок равна $c = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}$. Окончательно, для рассматриваемой транспортной задачи получаем следующую математическую постановку:

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

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i, i = 1..m \\ \sum_{i=1}^{m} x_{ij} = b_j, j = 1..n \\ x_{ij} \ge 0, i = 1..m, j = 1..n \end{cases}$$

Очевидно, что транспортная задача является задачей линейного программирования и ее можно решать симплекс-методом. Однако удобнее ее решать специальными методами, с одним из которых мы познакомимся далее.

Определение. ***Транспортной матрицей*** называется такая матричная форма записи транспортной задачи, в клетках (i, j) которой в верхнем правом углу стоят стоимости перевозок единицы груза c_{ij} от поставщика A_i получателю B_j, а в левом нижнем углу – количество груза x_{ij}, перевозимого от поставщика A_i к получателю B_j. Последняя строка и последний столбец транспортной матрицы задают количество имеющего груза у поставщиков и потребности в грузе потребителей.

	B_1	B_2	…	B_n	
A_1	c_{11} x_{11}	c_{12} x_{12}	…	c_{1n} x_{1n}	a_1
A_2	c_{21} x_{21}	c_{22} x_{22}	…	c_{2n} x_{2n}	a_2
…	…	…	…	…	…
A_m	c_{m1} x_{m1}	c_{m2} x_{m2}	…	c_{mn} x_{mn}	a_m
	b_1	b_2	…	b_n	

Определение. Если $x_{ij} > 0$, то клетку (i, j) будем называть ***загруженной***, если $x_{ij} = 0$ то клетку будем называть ***незагруженной*** и нуль в левом нижнем углу клетки писать не будем, кроме нескольких исключительных случаев, когда необходимо клетку $x_{ij} = 0$ считать загруженной.

Определение. ***Планом задачи*** назовем любой набор загрузок, при котором их сумма в любой сроке равна наличному у поставщика груза, а сумма загрузок по любому столбцу равна потребности получателя.

Определение. ***Циклом*** назовем набор клеток, лежащих в вершинах замкнутой ломаной с вертикальными и горизонтальными звеньями, причем из каждой вершины выходит точно два (перпендикулярных) звена. Любой набор клеток, содержащих хотя бы один цикл, назовем ***циклическим набором*** клеток, в противном случае – ***ациклическим набором***.

Лемма. *Максимальное число клеток, составляющих ациклический набор равно* $m+n-1$.

Из леммы следует, что любой набор из $m+n$ клеток является циклическим, чем мы и воспользуемся позднее.

Теорема (об отыскании оптимального решения). *Оптимальное решение (оптимальный план) транспортной задачи достаточно искать среди ациклических наборов загруженных клеток.*

Определение. ***Опорным планом*** назовем ациклический набор загруженных клеток, содержащий $m+n-1$ клеток, загрузки которых образуют решение (план) задачи.

Свойства опорного плана:

1. Если к опорному плану присоединить произвольную клетку, то получим циклический набор клеток, содержащий единственный цикл.
2. Если после присоединения к опорному плану клетки выделить цикл и произвести перенос по циклу, начиная нумерацию клеток цикла с присоединенной, то снова получим опорный план.

Перенос по циклу заключается в последовательном выполнении следующих этапов:

1. Перенумеруем клетки цикла, начиная с присоединенной. Отметим клетки с четным индексом знаком «–», а с нечетным – знаком «+». Множества клеток цикла обозначим θ^- и θ^+, соответственно.

2. Среди клеток, входящих в θ^-, найдем наименьшую загрузку $x^{\min} = \min\limits_{(i,j)\in\theta^-} x_{ij}$.

3. Пересчитаем загрузки, полагая

$$x_{ij} = \begin{cases} x_{ij} - x^{\min}, (i,j) \in \theta^- \\ x_{ij} + x^{\min}, (i,j) \in \theta^+ \end{cases}$$

4. Клетку множества θ^-, которая обратилась в нуль (или любую из таких клеток, если их несколько), уберем из множества загруженных клеток, разрывая тем самым цикл.

Очевидно, что при такой процедуре получается опорный план, так как суммарная загрузка любого столбца и любой строки не меняется, число загруженных клеток остается равным $m+n-1$ и полученный набор загруженных клеток ациклический.

8.4.2 Метод потенциалов

Предположим, что в рассматриваемой задачи найден опорный план. Введем характеристики строк u_i и столбцов v_j, которые будем называть ***потенциалами строк и столбцов***. Значения потенциалов u_i и v_j находятся из системы уравнений:

$$\begin{cases} u_1 = 0 \\ u_i + v_j = c_{ij} \end{cases} \qquad (10)$$

где клетки (i,j) - загруженные.

Нетрудно показать, что система (10) имеет единственное решение.

Для незагруженных клеток введем характеристики $p_{ij} = u_i + v_j - c_{ij}$, которые будем называть ***потенциалами незагруженных клеток***.

Для транспортной задачи можно сформулировать аналог теоремы о приращении линейного функционала в задаче линейного программирования, а именно – если мы

присоединим к опорному плану некоторую незагруженную клетку (i_0, j_0) и сделаем перенос по циклу, тогда приращение функционала стоимости перевозок равно

$$c^{new} = c^{old} - p_{i_0 j_0} x^{\min},$$

где $p_{i_0 j_0}$ - потенциал незагруженной клетки (i_0, j_0)

Теорема (признак достижения оптимального решения). *Опорный план транспортной задачи оптимален, если все потенциалы незагруженных клеток неположительные, т.е.* $p_{ij} \le 0$, $\forall(i, j)$ *незагруженной.*

Нахождение опорного плана аналогично симплекс-методу: метод потенциалов – это метод последовательного улучшения опорного плана (канонической формы). Значит, надо уметь находить начальный опорный план.

8.4.3 Методы нахождения начального опорного плана

Рассмотрим несколько способов нахождения первого опорного плана.

1. Метод минимума по строке.

Рассмотрим первую строку. Выберем клетку строки с наименьшим c_{1j} (или любую из них, если таких несколько). Пусть это клетка $(1, j_0)$. Загрузим ее максимальным образом. Возможны три случая:

а) $a_1 < b_{j_0}$. Тогда $x_{1j_0} = a_1$ (весь груз от поставщика A_1 посылаем получателю B_{j_0}, но потребность данного получателя в грузе удовлетворена не полностью и составляет $b'_{j_0} = b_{j_0} - a_1$). Далее переходим ко второй строке, рассматривая ее как первую.

б) $a_1 > b_{j_0}$. Тогда $x_{1j_0} = b_{j_0}$ (потребность в грузе для получателя B_{j_0} полностью удовлетворена, а у поставщика A_1 остался груз в количестве $a'_1 = a_1 - b_{j_0}$). Далее переходим к следующей по минимуму c_{1j} клетке первой строки. Столбец j_0 больше не рассматриваем.

в) $a_1 = b_{j_0}$. Тогда $x_{1j_0} = b_{j_0} = a_1$ (потребность в грузе для получателя B_{j_0} полностью удовлетворена, а груз от поставщика A_1 полностью вывезен). Выберем следующую по минимуму c_{1j} клетку строки и загружаем ее фиктивной нагрузкой, равной нулю (фиктивная загрузка не ставится в полностью загруженный столбец и в последней строке). Столбец j_0 больше не рассматриваем и переходим к следующей строке.

2.Метод минимума по столбцу.

Метод аналогичен методу минимума по строке и получается заменой строк столбцами.

3.Метод северо-западного угла.

Рассмотрим первую (верхнюю левую) клетку $(1,1)$ первой строки и загружаем ее максимально. Возможны три случая:

а) $a_1 < b_1$. Тогда $x_{11} = a_1$ (весь груз у поставщика A_1 вывезен, потребность получателя B_1 в грузе удовлетворена не полностью и составляет $b'_1 = b_1 - a_1$). Переходим ко второй строке, рассматривая ее как первую, т.е. к клетке $(2,1)$.

б) $a_1 > b_1$. Тогда $x_{11} = b_1$ (полностью удовлетворена потребность в грузе первого получателя B_1, груз у поставщика A_1 остался в количестве $a'_1 = a_1 - b_1$). Переходим ко второй клетке первой строки, т.е. к клетке $(1,2)$.

в) $a_1 = b_1$. Тогда $x_{11} = a_1 = b_1$ (полностью вывезен груз у поставщика A_1 и потребность получателя B_1 в грузе удовлетворена полностью). В первую клетку второй строки – клетку $(2,1)$ – ставим фиктивную нагрузку, равную

нулю (клетку будем считать загруженной) и переходим ко второй клетке второй строки (которая, если не рассматривать первую строку и столбец, будет снова верхней левой клеткой) – клетке $(2,2)$.

8.4.4 Распределительный метод

Алгоритм распределительного метода

1. Находим начальный опорный план. Можно применить несколько методов и выбрать в качестве начального тот опорный план, который задает меньшую стоимость перевозок.
2. Вычисляем потенциалы строк и столбцов и незагруженных клеток.

Возможны два случая:

- все потенциалы незагруженных клеток неположительные – задача решена согласно признаку достижения оптимального решения. Текущий опорный план является оптимальным.
- среди потенциалов незагруженных клеток есть положительные – переходим к следующему этапу.

3. Выбираем клетку с наибольшим положительным потенциалом (или одну из них, если таких клеток несколько) и присоединяем ее к опорному плану.
4. Выделяем цикл, делаем перенос по циклу. Переходим к пункту 2.

8.4.5 Открытые транспортные задачи

На практике достаточно часто возникают несбалансированные задачи, в которых $\sum_{i=1}^{m} a_i \neq \sum_{j=1}^{n} b_j$, однако любая такая задача может быть сведена к сбалансированной следующим образом:

1. $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$. Введем фиктивного поставщика $\widetilde{A}_{m+1}$, полагая наличие у него однородного груза в количестве $\widetilde{a}_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$ и задавая стоимости перевозки от него к любому получателю равными 0. Полученная задача является сбалансированной.
2. $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$. Введем фиктивного получателя $\widetilde{B}_{n+1}$, полагая, что его потребность в грузе равна $\widetilde{b}_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$ и задавая стоимости перевозки к нему от любого поставщика равными 0. Полученная задача является сбалансированной.

Пример 7 (М.Перельман). *Решить транспортную задачу, заданную следующей таблицей:*

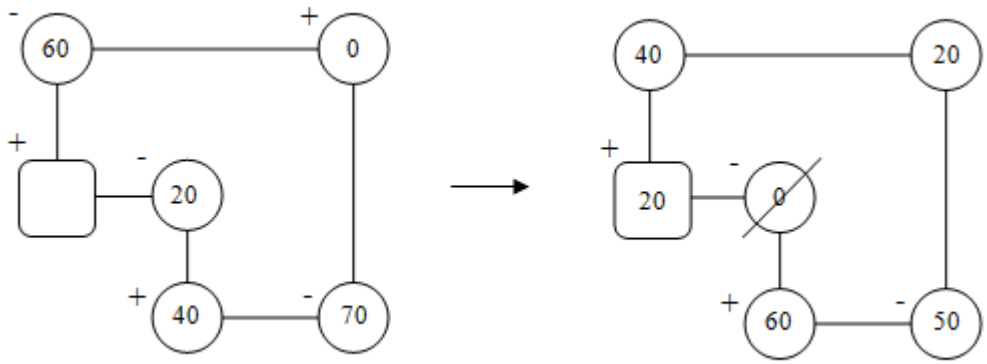
	B_1	B_2	B_3	B_4	B_5	B_6	
A_1	**7**	9	15	13	10	12	150
A_2	8	5	13	11	14	19	120
A_3	10	13	9	14	15	7	170
A_4	12	15	10	15	11	9	110
	90	60	110	100	70	120	550

Данная транспортная задача является сбалансированной, т.к. суммарные возможности поставщиков равны суммарным потребностям потребителей. Для нахождения начального опорного плана воспользуемся методом минимума по строке и получим:

	B_1	B_2	B_3	B_4	B_5	B_6	
A_1	7 90	9 60	15	13	10 0	12	150
A_2	8	5	13 20	11 100	14	19	120
A_3	10	13	9 50	14	15	7 120	170
A_4	12	15	10 40	15	11 70	9	110
	90	60	110	100	70	120	550

Перейдем к вычислению потенциалов строк и столбцов, для чего составим систему уравнений: $u_1 = 0$, $u_1 + v_1 = 7$, $u_1 + v_2 = 9$, $u_1 + v_5 = 10$, $u_2 + v_3 = 13$, $u_2 + v_4 = 11$, $u_3 + v_3 = 9$, $u_3 + v_6 = 7$, $u_4 + v_3 = 10$, $u_4 + v_5 = 11$. Отсюда находим потенциалы строк и столбцов: $u_1 = 0$, $u_2 = 4$, $u_3 = 0$, $u_4 = 1$, $v_1 = 7, v_2 = 9$, $v_3 = 9$, $v_4 = 7$, $v_5 = 10$, $v_6 = 7$. Потенциалы незагруженных клеток равны: $p_{13} = -6$, $p_{14} = -6$, $p_{16} = -5$, $p_{21} = 3$, $p_{22} = 8$, $p_{25} = 0$, $p_{26} = -8$, $p_{31} = -3$, $p_{32} = -4$, $p_{34} = -7$, $p_{35} = -5$, $p_{41} = -4$, $p_{42} = -5$, $p_{44} = -7$, $p_{46} = -1$.

Текущий опорный план не оптимален, т.к. среди потенциалов незагруженных клеток есть положительные значения. Выбираем клетку с наибольшим положительным потенциалом – клетку $(2,2)$, присоединяем ее к опорному плану, выделяем цикл, начиная с присоединенной клетки, осуществляем перенос по циклу:

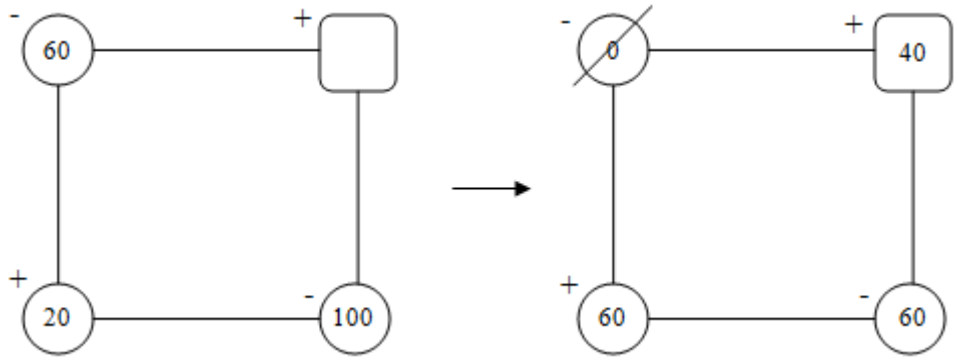


Получаем следующую транспортную матрицу:

	B_1	B_2	B_3	B_4	B_5	B_6	
A_1	7 90	9 40	15	13	10 20	12	150
A_2	8	5 20	13	11 100	14	19	120
A_3	10	13	9 50	14	15	7 120	170
A_4	12	15	10 60	15	11 50	9	110
	90	60	110	100	70	120	550

Потенциалы строк и столбцов находим из условий: $u_1 = 0$, $u_1 + v_1 = 7$, $u_1 + v_2 = 9$, $u_1 + v_5 = 10$, $u_2 + v_2 = 5$, $u_2 + v_4 = 11$, $u_3 + v_3 = 9$, $u_3 + v_6 = 7$, $u_4 + v_3 = 10$, $u_4 + v_5 = 11$. Отсюда находим потенциалы строк и столбцов: $u_1 = 0$, $u_2 = -4$, $u_3 = 0$, $u_4 = 1$, $v_1 = 7$, $v_2 = 9$, $v_3 = 9$, $v_4 = 15$, $v_5 = 10$, $v_6 = 7$. Потенциалы незагруженных клеток равны: $p_{13} = -6$, $p_{14} = 2$, $p_{16} = -5$, $p_{21} = -5$, $p_{23} = -8$, $p_{25} = -8$, $p_{26} = -16$, $p_{31} = -3$, $p_{32} = -4$, $p_{34} = 1$, $p_{35} = -5$, $p_{41} = -4$, $p_{42} = -5$, $p_{44} = 1$, $p_{46} = -1$.

Текущий опорный план снова не оптимален, т.к. среди потенциалов незагруженных клеток есть положительные. Выбираем клетку с наибольшим положительным потенциалом – клетку $(1,4)$, присоединяем ее к опорному плану, выделяем цикл, начиная с присоединенной клетки, осуществляем перенос по циклу:



Получаем следующую транспортную матрицу:

	B_1	B_2	B_3	B_4	B_5	B_6	
A_1	7 90	9	15	13 40	10 20	12	150
A_2	8	5 60	13	11 60	14	19	120
A_3	10	13	9 50	14	15	7 120	170
A_4	12	15	10 60	15	11 50	9	110
	90	60	110	100	70	120	550

Потенциалы строк и столбцов находим из условий: $u_1 = 0$, $u_1 + v_1 = 7$, $u_1 + v_4 = 13$, $u_1 + v_5 = 10$, $u_2 + v_2 = 5$, $u_2 + v_4 = 11$, $u_3 + v_3 = 9$, $u_3 + v_6 = 7$, $u_4 + v_3 = 10$, $u_4 + v_5 = 11$. Отсюда находим потенциалы строк и столбцов: $u_1 = 0$, $u_2 = -2$, $u_3 = 0$, $u_4 = 1$, $v_1 = 7$, $v_2 = 7$, $v_3 = 9$, $v_4 = 13$, $v_5 = 10$, $v_6 = 7$. Потенциалы незагруженных клеток равны: $p_{12} = -2$, $p_{13} = -6$, $p_{16} = -5$, $p_{21} = -3$, $p_{23} = -6$, $p_{25} = -6$, $p_{26} = -14$, $p_{31} = -3$, $p_{32} = -6$, $p_{34} = -1$, $p_{35} = -5$, $p_{41} = -4$, $p_{42} = -7$, $p_{44} = -1$, $p_{46} = -1$.

Так как все потенциалы незагруженных клеток неположительные, то полученный опорный план является оптимальным. Стоимость перевозок по данному плану составляет $c^* = \min c = 4750$.

8.5. Сетевые задачи

Сетевые задачи зачастую являются частными случаями задач линейного программирования, но применение общих методов линейного программирования к ним не целесообразно из-за особенностей их математических характеристик. Эти характеристики позволяют построить более эффективные алгоритмы, а в случае больших размерностей – дают единственный способ решения задачи за разумное время.

8.5.1 Основные понятия и определения

Многие задачи в силу их специфики удобно моделировать при помощи сетей и потоков, используя теорию графов, поэтому напомним некоторые определения.

Определение. ***Ориентированным графом*** называется тройка (I, D, G), в которой I - непустое множество вершин, D - множество дуг, G - отображением, которое каждой дуге $d \in D$ ставит в соответствие упорядоченную пару вершин (i, j), где $i, j \in I$.

Определение. ***Неориентированным графом*** называется тройка (I, D, G), в которой I - непустое множество вершин, D - множество ребер, G - отображение, которое каждому ребру $d \in D$ ставит в соответствие неупорядоченную пару вершин $[i, j]$, где $i, j \in I$.

Определение. Граф называется ***простым*** и может быть задан парой (I, D), если любые две вершины графа соединяются не более чем одним ребром (или дугой).

В простом ориентированном графе каждую дугу $d \in D$ можно определить упорядоченной парой соединяемых вершин (i, j), где i - начало дуги, j - конец дуги, а сама дуга считается инцидентной данным вершинам.

Определение. ***Путь длины*** n в ориентированном простом графе (I, D) это упорядоченная последовательность различных дуг $(d_1, d_2, \cdots, d_n)$, для которых конечная вершина предыдущей дуги совпадает с начальной вершиной последующей дуги. Конечный путь, у которого начальная вершина совпадает с конечной называется **контуром**. Для неориентированного графа аналог пути - ***цепь***, а контура – ***цикл***.

Определение. Неориентированный граф, у которого любые две вершины могут быть соединены цепью, называется ***связным***. Ориентированный граф называется связным, если ему отвечает связный неориентированный граф.

Определение. Конечный граф (I, D, G) называется ***сетью***, если каждой вершине i сопоставлено некоторое число q_i, называемое ***интенсивностью вершины***. Если $q_i > 0$, то вершина i называется ***источником***, если $q_i < 0$, то – ***стоком***, а если $q_i = 0$, то ***нейтральной вершиной***. Множество источников, стоков и нейтральных вершин обозначается I^+, I^-, I^0, соответственно.

Определение. ***Потоком*** называется такая совокупность величин, заданных на множестве дуг $X = \{x_d\}_{d \in D}$, что

$$\sum_{d \in D_i^+} x_d - \sum_{d \in D_i^-} x_d = q_i, \quad i \in I, \quad x_d \geq 0, \quad d \in D, \tag{11}$$

где D_i^+ - множество дуг, исходящих из вершины i, а D_i^- - множество дуг, входящих в нее. Величина x_d называется ***значением потока по дуге*** d и может пониматься как количество груза, пропускаемого по данной дуге.

Из определения потока легко заметить, что для любой вершины сети разность входящего и выходящего потоков равна ее интенсивности.

Используя введенные определения можно сформулировать много различных задач. Например, если для каждой дуги $d \in D$ определить величины $c_d \geq 0$, понимаемые как

стоимости перемещения единицы груза по дуге, тогда ***суммарная стоимость потока*** X равна $f(X)=\sum_{d\in D} c_d x_d$. Задача минимизации суммарной стоимости потока

$$f(X)=\sum_{d\in D} c_d x_d \to \min$$

$$\sum_{d\in D_i^+} x_d - \sum_{d\in D_i^-} x_d = b_i, \quad i\in I, \quad x_d \ge 0, \quad d\in D$$

называется ***линейной сетевой задачей***. Очевидно, что такая задача является и задачей линейного программирования.

Если для каждой дуги сети $d\in D$ добавить величины $r_d \ge 0$, называемые пропускными способностями, тогда, установив ограничения

$$0 \le x_d \le r_d, \quad d\in D, \tag{12}$$

получаем задачу ***о потоке в сети с ограниченными пропускными способностями***.

8.5.2 Транспортная задача в сетевой постановке

Естественной сферой применения сетевых задач является организация грузоперевозок в транспортной сети. В таких задачах вершины i понимаются как пункты, соединенные сетью дорог, и характеризуются потребностями в некотором однородном грузе ($q_i<0$) или его наличием ($q_i>0$). Задачу определения плана, доставляющего минимум стоимости перевозок, которая может быть описана (11), (12), называют ***транспортной задачей в сетевой постановке***.

Определение. Транспортная сеть называется ***сбалансированной***, если $\sum_{i\in I} b_i = 0$.

Теорема (критерий оптимальности). *Для того, чтобы в сбалансированной транспортной сети допустимый поток $X=\{x_d\}_{d\in D}$ был оптимальным, необходимо и достаточно существование для каждой вершины $i\in I$ такого числа v_i, называемого потенциалом, что для всех дуг $d=(i,j)$:*

$$\begin{aligned} &v_j - v_i \le c_d, \text{ если } x_d = 0,\\ &v_j - v_i = c_d, \text{ если } 0 < x_d < r_d,\\ &v_j - v_i \ge c_d, \text{ если } x_d = r_d. \end{aligned} \tag{13}$$

Определение. ***Остовом сети*** (I,D,G) называется любое частичное дерево (частичный неориентированный подграф, не содержащий циклов).

Теорема. *Произвольному остову сети (I,D,G) можно поставить в соответствие базис задачи (11), (12) и наоборот.*

Определение. ***Опорой потока*** $X=\{x_d\}_{d\in D}$ называется частичный подграф $(I,D(X),G)$, *где* $D(X)=\{d\in D: 0<x_d<r_d\}$. Будем говорить, что поток X ***невырожденный***, если его опора $(I,D(X),G)$ является остовом сети (I,D,G).

Схема метода потенциалов для транспортной задачи в сетевой подстановке:

1. По имеющемуся на итерации k допустимому невырожденному потоку $X^k=\{x_d^k\}_{d\in D}$ строится система потенциалов. Выбирается произвольную вершину i_0, потенциал которой считаем $v_{i_0}=0$. Множество вершин, смежных с i_0, обозначим через $I(i_0)$. Тогда для $\forall j\in I(j_0)$:

$$v_j = \begin{cases} v_{i_0} + c_{i_0 j}, (i_0, j) - \text{дуга направлена от } i_0 \\ v_{i_0} - c_{j i_0}, (j, i_0) - \text{дуга направлена к } i_0 \end{cases}$$

2. Проверка выполнение критерия оптимальности (13). В случае его невыполнения перейти к следующему пункту.

3. Получение нового потока и совершения переноса по циклу. По аналогии с задачами линейного программирования новый поток получается включением и

исключением дуг. Для ввода в поток выберем дугу $d_l = (s,t)$, для которой не выполняется критерий оптимальности (13) и на которой достигается максимальное отклонение цены от разности потенциалов соединяемых вершин. Если дополнить остов дугой d_l, то образуется цикл, причем единственный. Обозначим через $D^+_{(s,t)}$ множество дуг цикла, ориентация которых совпадает с ориентацией дуги $d_l = (s,t)$, а через $D^-_{(s,t)}$ - множество дуг, имеющих противоположную ориентацию. Определяется величина изменения объемов перевозок:

$$\theta = \min\left\{ \min_{D^-_{(s,t)}} x^k_d , \min_{D^+_{(s,t)}} \left(r_d - x^k_d \right) \right\}.$$

После определения θ проводится пересчет компонент текущего потока:

$$x^{k+1}_d = \begin{cases} x^k_d + \theta, d \in D^+_{(s,t)} \\ x^k_d - \theta, d \in D^-_{(s,t)} \\ x^k_d, d \in D^+_{(s,t)} \cup D^-_{(s,t)} \end{cases}.$$

В результате получаем новый допустимый поток $X^{k+1} = \left\{ x^{k+1}_d \right\}_{d \in D}$ и переходим к пункту 1 алгоритма.

Метод генерации начального допустимого потока:

1. К множеству вершин сети добавляется фиктивная нулевая вершина с нулевой интенсивностью ($q_0 = 0$).

2. Все вершины, имеющие отрицательную интенсивность $q_i < 0$, соединяются с фиктивной вершиной 0 входящими дугами $(0,i)$, а вершины с положительной интенсивностью $q_i > 0$ - исходящими дугами $(i,0)$. Ограничения на пропускные способности для добавляемых дуг отсутствуют.

3. Стоимости перемещения единицы продукта для вновь добавленных дуг полагаются равными 1, а для дуг транспортной сети основной задачи – 0.

8.5.3 Задача о кратчайшем пути

Одной из известных сетевых задач является задача определения кратчайшего пути между вершинами сети.

Пусть задан простой граф (I, D, G), каждой дуге которого ставится в соответствие некоторое число c_d, которые мы назовем ***длиной дуги***. Пусть также каким-то образом выделены две вершины графа $s \in I$ и $t \in I$, и требуется найти путь $l = \left\{ s = i_0, i_1, \cdots i_{p-1}, i_p = t \right\}$ наименьшей длины, ведущий из вершины s в вершину t.

Конечно, задачу о кратчайшем пути можно рассматривать и как задачу линейного программирования и как транспортную сетевую задачу, однако более рациональным является использование свойств задачи и решение её специальными методами, например, методом Минти.

Метод Минти решения задачи о кратчайшем пути в сети представляет собой процесс, в ходе которого последовательно строится путь $l = \left\{ s = i_0, i_1, \cdots i_{p-1}, i_p = t \right\}$.

Алгоритм метода Минти:

1. Формируем массив значений модифицированных длин волн $\tilde{c}_{ij}$, которые перед началом итерации равны $c_{ij} \geq 0$, и отмечаем вершину $i_0 = s$ числом $m_{i_0} = 0$.

2. Отметка вершин сети. Обозначим множество вершин сети, отмеченных на предыдущих итерациях, как $\tilde{I}$ (на первой итерации $\tilde{I} = \{ i_0 \}$). Для каждой вершины $i \in \tilde{I}$ ищутся дуги, соединяющие ее с еще не помеченными вершинами-

потомками j, модифицированная длина которых равна $\widetilde{c}_{ij}=0$. Найденные вершины помечаются числом $m_j=i$, указывающим на родительскую вершину. Если сразу несколько дуг имеет $\widetilde{c}_{ij}=0$ и заканчиваются в j, то значение i для ее пометки выбирается произвольно.

3. Если среди вновь помеченных вершин оказалась вершина t, но искомый путь $l=\{s=i_0,i_1,\cdots i_{p-1},i_p=t\}$ найден, где $i_p=t, i_{p-1}=m_{i_p},\cdots,i_1=m_{i_2},i_0=m_{i_1}=s$, иначе переходим к следующему шагу.

4. Преобразование значений модифицированных длин дуг. Для каждой вершины $i\in\widetilde{I}$ ищутся дуги, соединяющие ее с еще не помеченными вершинами j, и находятся $\widetilde{\Delta}=\min\limits_{j\in\widetilde{I}}\widetilde{c}_{ij}$. Модифицированные длины всех дуг, соединяющих отмеченные вершины с неотмеченными ($i\in\widetilde{I}, j\in\widetilde{I}$), уменьшам на величину $\widetilde{\Delta}=\min\limits_{i\in\widetilde{I}}\widetilde{\Delta}_i$, в результате чего кратчайшие неиспользованные дуги получают модифицированную длину равную 0. Далее переходим к пункту 2 алгоритма.

Следует заметить, что приведенный здесь алгоритм метода Минти пригоден для построения кратчайших путей на неориентированном простом графе.

8.6. Основы теории игр

Если при изучении задач линейного программирования основное внимание уделяется способу эффективного использования ограниченных ресурсов, то в теории игр основное внимание уделяется стратегиям, при помощи которых достигаются интересующие нас цели. Однако между задачами теории игр и линейного программирования существует связь, позволяющая применять известные нам методы.

8.6.1 Основные определения и классификация игр

Определение (Мулен). ***Игра*** – это идеализированная математическая модель коллективного поведения: несколько индивидуумов (***участников, игроков***) влияют на ситуацию (***исход игры***), причем их интересы (и выигрыши при различных возможных ситуациях) различны.

Одной из естественных классификаций игр является деление игр по количеству участников, например, игры двух лиц.

Игроки влияют на исход игры, реализуя или не реализуя свои стратегии – допустимые решения, набором которых обладает участник. В зависимости от количества стратегий различают конечные и бесконечные игры.

Определение. Если хотя бы один из игроков обладает бесконечным множеством решений, то игра называется ***бесконечной***. Если количество стратегий у всех игроков конечное, то игра также ***конечная***.

Между игроками могут возникать предварительные договоренности. В зависимости от таких договоренностей различают кооперативные и некооперативные игры.

Определение. Игра называется ***кооперативной***, если до ее начала игроки договариваются координировать свои стратегии для достижения выгодного всем результата.

Еще один способ классификации – использование информации о структуре выигрыша.

Определение. ***Игрой с нулевой сумой*** называется игра, в которой сумма выигрышей всех игроков равно нулю, т.е. одна часть игроков выигрывает то, что проигрывает другая.

Определение. ***Антагонистическая игра*** это игра двух лиц с нулевой суммой, т.к. выигрыш одного игрока напрямую зависит от проигрыша другого.

Основной моделью анализа некооперативного поведения игроков является игра в нормальной форме. Предположим, что N - фиксированное конечное сообщество игроков.

Определение (Мулен). ***Игрой в нормальной форме*** $G=\left(X_i,u_i \mid i\in N\right)$ называется совокупность, содержащая для каждого игрока $i\in N$:

- множество стратегий X_i, элементы которого обозначаются x_i;
- функцию выигрыша (функцию полезности) $u_i : X_N=\underset{i\in N}{\mathrm{X}} X_i \to \Re$.

Элемент $x=\left(x_i\right)_{i\in N}$ из множества X_N называется ***исходом игры***.

Нормальную форму игры двух лиц фактически можно задать при помощи двух платежных матриц – матриц, содержащих информацию о наборе стратегий, а также суммы выигрышей и проигрышей каждого из игроков для каждой возможной пары стратегий игроков. В частности, объединив элементы этих матриц пару, можно использовать для задания игры лишь одну «единую» матрицу. Если первый игрок обладает множеством стратегий $X_1=\left\{x_i^1\right\}_{i=1}^m$, а второй – множеством $X_2=\left\{x_j^2\right\}_{j=1}^n$, то ***единая платежная матрица*** имеет вид:

$$\begin{array}{c} \\ x_1^1 \\ x_2^1 \\ \\ x_m^1 \end{array} \begin{array}{c} \begin{array}{cccc} x_1^2 & x_2^2 & & x_n^2 \end{array} \\ \begin{pmatrix} \left(U_{11}^1,U_{11}^2\right) & \left(U_{11}^1,U_{11}^2\right) & \cdots & \left(U_{1n}^1,U_{1n}^2\right) \\ \left(U_{21}^1,U_{21}^2\right) & \left(U_{22}^1,U_{22}^2\right) & \cdots & \left(U_{2n}^1,U_{2n}^2\right) \\ \cdots & \cdots & \cdots & \cdots \\ \left(U_{m1}^1,U_{m1}^2\right) & \left(U_{m2}^1,U_{m2}^2\right) & \cdots & \left(U_{mn}^1,U_{mn}^2\right) \end{pmatrix} \end{array},$$

где на пересечении i-й строки и j-го столбца находится пара $\left(U_{ij}^1,U_{ij}^2\right)$, содержащая выигрыши первого и второго игрока при выборе ими стратегий x_i^1 и x_j^2, соответственно.

Считаем, что всем игрокам известна единая платежная матрица еще до начала игры, т.е. рассматриваем игры с полной информацией.

8.6.2 Игры двух лиц с нулевой суммой

Игры двух лиц с нулевой суммой (или антагонистические игры) являются самыми изученными задачами теории игр, поэтому на их примере мы рассмотрим основные вопросы теории игр при некооперативном поведении игроков.

В таких играх выигрыш одного игрока равен проигрышу другого

$$U_{ij}^1=-U_{ij}^2,\ i=1..n,\quad j=1..m.$$

и, следовательно, для задания игр двух лиц с нулевой суммы вместо единой платежной матрицы достаточно задавать платежную матрицу одного из игроков:

$$U^1=\begin{array}{c} \\ x_1^1 \\ x_2^1 \\ \\ x_m^1 \end{array} \begin{array}{c} \begin{array}{cccc} x_1^2 & x_2^2 & & x_n^2 \end{array} \\ \begin{pmatrix} U_{11}^1 & U_{11}^1 & \cdots & U_{1n}^1 \\ U_{21}^1 & U_{22}^1 & \cdots & U_{2n}^1 \\ \cdots & \cdots & \cdots & \cdots \\ U_{m1}^1 & U_{m2}^1 & \cdots & U_{mn}^1 \end{pmatrix} \end{array}$$

Пример. *Рассмотрим игру двух игроков в орлянку, смысл которой заключается в следующем: подбрасывается монета и каждый игрок пытается угадать, какой стороной она упадёт. Если никто не угадывает или угадывают оба, то все остаются при своих, иначе проигравший выплачивает 1 гривну выигравшему. Необходимо задать игру в нормальной форме.*

Перед нами игра двух лиц с нулевой суммой. У каждого из игроков есть две стратегии – варианты ответа «орел» и «решка». Выигрыш и проигрыш обоих игроков равен нулю, если они выбирают один и тот же вариант ответа – используют одинаковые стратегии «решка»-

«решка» или «орел»-«орел» вне зависимости от результата подбрасывания монеты. В случае же выбора разных стратегий одни игрок выигрывает, а другой проигрывает 1. Следовательно, для задания данной игры в нормальной форме достаточно привести платежную матрицу первого игрока (т.к. порядок игроков не важен, считаем первым того, кто выигрывает):

	орел	*решка*
орел	0	1
решка	1	0

Определение. ***Нижней ценой игры в чистых стратегиях*** называется величина $U_* = \max\limits_{i=1..m} \min\limits_{j=1..n} U_{ij}$, а соответствующая ей стратегия – ***максиминной стратегией***.

Придерживаясь своей максиминной стратегии, игрок гарантирует себе выигрыш, больший или равный величине U_*.

Определение. ***Верхней ценой игры в чистых стратегиях*** называется величина $U^* = \min\limits_{j=1..n} \max\limits_{i=1..m} U_{ij}$, а соответствующая ей стратегия – ***минимаксной стратегией***.

Придерживаясь своей минимаксной стратегии, игрок гарантирует, что проигрыш не превысит величины U^*.

В общем случае верхняя и нижняя цены игры не совпадают, однако существует целый ряд важных задач, где $U^* = U_*$. Такие игры называют ***играми с седловой точкой*** – играми с платежной матрицей, в которой существует элемент, являющийся одновременно минимальным в строке и максимальным в столбце.

Теорема. *Необходимым и достаточным условием равенства верхней и нижней цен игры в чистых стратегиях является существование в платежной матрице седловой точки.*

Определение. ***Ценой игры в чистых стратегиях (чистой ценой игры)*** υ называется общее значение верхней и нижней цен игры в чистых стратегиях, если такое существует: $\upsilon = U^* = U_*$.

В играх с ценой игры проявляется ***свойство устойчивости***, заключающееся в том, что если один игрок придерживается своей осторожной (минимаксной или максиминной) стратегии, то другой игрок не может увеличить свой выигрыш, отступив от своей осторожной стратегии. Исходя из этих соображений, максиминная стратегия одного игрока и минимаксная стратегия второго игрока являются оптимальными.

Ситуация меняется для игр двух игроков с ненулевой суммой – здесь нижняя цена игры строго меньше верхней, появляется нестабильность, вызванная желанием игроков улучшить свой выигрыш и пара осторожных стратегий уже не является оптимальной.

8.6.3 Смешанное расширение игры

Более общим понятием, чем чистые (максиминные, минимаксные) стратегии в теории игр являются смешанные стратегии, использующие вероятностные распределения. Такой подход позволяет строить более гибкие стратегии и разрешать ситуации с возникшей неустойчивостью в чистых стратегиях.

Определение. ***Смешанной стратегией i-го игрока*** в конечной игре $G = \left(X_i, u_i \mid i \in N\right)$ называется некоторое вероятностное распределение μ^i на множестве чистых стратегий X_i. Такое вероятностное распределение понимается как «конструирование» каждым игроком своего собственного генератора случайных событий, где использование чистой стратегии $x_j^i \in X_i$ определяется вероятностью μ_j^i. Смешанную стратегию i-го игрока мы будем записывать в виде $\mu^i = \left(\mu_1^i, \mu_2^i, \cdots, \mu_n^i\right)$.

Определение. ***Смешанным расширением игры*** $G=\left(X_i,u_i \mid i\in N\right)$ называется такая игра в нормальной форме

$$G_m=\left(M_i,u_i^m \mid i\in N\right), \text{ где } \forall\mu\in\underset{i\in N}{\mathrm{X}}M_i\ :\ u_i^m(\mu)=\sum_{x\in X_N}u_i(x)\prod_{j=1}^{n}\mu_j^i\left(x_j^i\right).$$

Из данных определений следует, что множество чистых стратегий содержится в множестве смешанных стратегий и каждой чистой стратегии $x_j^i\in X_i$ соответствует некоторая смешанная стратегия i-го игрока, выбирающая x_j^i с вероятностью 1.

Возвращаясь к играм двух лиц с нулевой суммой, сформулируем следующую теорему:

Теорема. *Рассмотрим конечную игру двух лиц с нулевой суммой* $G=\left(X_1,X_2,u_1\right)$ *и ее смешанное расширение – игру с нулевой суммой* $G_m=\left(M_1,M_2,u_1^m\right)$. *Тогда игра* G_m *имеет, по крайней мере, одну седловую точку и цену* v_m, *ее мы назовем* ***ценой игры в смешанных стратегиях****, для которой выполняется*

$$\max_{x_1^i\in X_1}\min_{x_2^j\in X_2}u_1\left(x_1^i,x_2^j\right)\le v_m\le\min_{x_2^j\in X_2}\max_{x_1^i\in X_1}u_1\left(x_1^i,x_2^j\right).$$

В случае игр двух лиц, заданных в матричном виде, цена игры в смешанных стратегиях вычисляется по формуле:

$$\overline{v}=\sum_{i=1}^{m}\sum_{j=1}^{n}\overline{\mu}_i^1\overline{\mu}_j^2U_{ij}\ ,$$

где $\overline{\mu}^1$ и $\overline{\mu}^2$ - оптимальные смешанные стратегии игроков.

Приведенная выше теорема позволяет, в частности, утверждать, что в играх двух лиц с нулевой суммой всегда существует оптимальная смешанная стратегия, к методам поиска которой мы и перейдем далее.

8.6.4 Аналитический метод решения матричной игры 2x2

Примером матричной игры 2x2 является игра двух лиц с нулевой суммой $G=\left(X_1,X_2,U\right)$, *где каждый из игроков имеет ровно по две чистые стратегии -* $X_1=\left\{x_1^1,x_2^1\right\}$ *и* $X_2=\left\{x_1^2,x_2^2\right\}$, *соответственно.*

В случае, если в рассматриваемой игре $G=\left(X_1,X_2,U\right)$ существует цена игры в чистых стратегиях, то достаточно найти максиминную и минимаксную стратегии игроков, которые и будут оптимальными. В случае же отсутствия цены игры в чистых стратегиях (т.е. $U_{11}+U_{22}\ne U_{12}+U_{21}$) перейдем к рассмотрению смешанного расширения игры, где каждый игрок будет иметь также ровно две смешанные стратегии, в которые каждая из соответствующих чистых стратегий будет входить с ненулевой вероятностью.

Предположим, что первый игрок определил и придерживается своей оптимальной смешанной стратегии $\overline{\mu}_1=\left(\overline{\mu}_1^1,\overline{\mu}_2^1\right)$, $\overline{\mu}_1^1+\overline{\mu}_2^1=1$, тогда вне зависимости от действий оппонента его выигрыш равен ожидаемой цене игры в смешанных стратегиях $\overline{v}$:

$$\begin{cases}\overline{\mu}_1^1U_{11}+\overline{\mu}_2^1U_{21}=\overline{v}\\ \overline{\mu}_1^1U_{12}+\overline{\mu}_2^1U_{22}=\overline{v}\end{cases}\Rightarrow\overline{\mu}_1^1=\frac{U_{22}-U_{21}}{\Delta},\overline{\mu}_2^1=\frac{U_{11}-U_{12}}{\Delta}, \text{ где } \Delta=U_{11}+U_{22}-\left(U_{12}+U_{21}\right)\ne0.$$

Аналогично, если второй игрок определил и придерживается своей оптимальной смешанной стратегии $\overline{\mu}_2=\left(\overline{\mu}_1^2,\overline{\mu}_2^2\right)$, $\overline{\mu}_1^2+\overline{\mu}_2^2=1$, тогда вне зависимости от действий первого игрока его выигрыш равен ожидаемой цене игры в смешанных стратегиях $\overline{v}$:

$$\begin{cases}\overline{\mu}_1^2U_{11}+\overline{\mu}_2^2U_{12}=\overline{v}\\ \overline{\mu}_1^2U_{21}+\overline{\mu}_2^2U_{22}=\overline{v}\end{cases}\Rightarrow\overline{\mu}_1^2=\frac{U_{22}-U_{12}}{\Delta},\overline{\mu}_2^1=\frac{U_{11}-U_{21}}{\Delta}.$$

Выражение для цены игры в смешанных стратегиях в данной задаче имеет вид:

$$\overline{v} = \frac{U_{11}U_{22} - U_{12}U_{21}}{\Delta}.$$

8.6.5 Графический метод решения матричной игры 2xn

В случае, когда в игре двух лиц с нулевой суммой один из игроков (предположим первый игрок) имеет ровно две чистые стратегии, матрица выигрышей будет иметь размерность 2xn. Как и в линейном программировании к таким задачам можно применить графический метод.

Как и ранее, будем считать, что в задаче нет цены игры в чистых стратегиях, и мы рассматриваем смешанное расширение игры. Предположим, что оптимальная смешанная стратегия первого игрока имеет вид $\overline{\mu}_1 = \left(\overline{\mu}_1^1, \overline{\mu}_2^1\right)$, $\overline{\mu}_1^1 + \overline{\mu}_2^1 = 1$. Тогда, в зависимости от выбора той или иной чистой стратегии x_j^2 вторым игроком, первый игрок может ожидать следующий выигрыш, который линейно зависит от $\overline{\mu}_1^1$:

$$\overline{u}\left(x_j^2, \overline{\mu}^1\right) = \overline{\mu}_1^1 U_{1j} + \overline{\mu}_2^1 U_{2j} = \overline{\mu}_1^1 U_{1j} + \left(1 - \overline{\mu}_1^1\right)U_{2j} = \overline{\mu}_1^1\left(U_{1j} - U_{2j}\right) + U_{2j},\ j = 1..n$$

Придерживаясь минимаксных критериев, игрок выбирает величину $\overline{\mu}_1^1 \in (0,1)$ таким образом, чтобы максимизировать свой минимально возможный выигрыш.

Очевидно, что такая линейная задача оптимизации может быть решена графически. Для этого необходимо на графике изобразить все прямые ($j = 1..n$), соответствующие ожидаемым выигрышам первого игрока, т.е. $y = \left(U_{1j} - U_{2j}\right)x + U_{2j}$, $x \in (0,1)$.

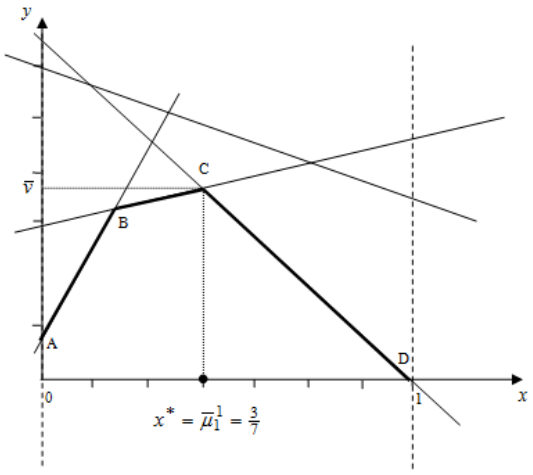


Рис. 8.2. Графический метод решения задачи теории игр.

Согласно минимаксному критерию, ломанная ABCD, дающая минимальный гарантированный выигрыш первого игрока вне зависимости от действий второго игрока, задает оптимальную смешанную стратегию первого игрока. Значение функции выигрыша в точке C задает цену игры в смешанных стратегиях.

Заметим, что рассуждения в случае, когда только второй игрок имеет две чистые стратегии, аналогичны, за исключением использования максиминного критерия выбора.

8.6.6 Связь матричной игры и задачи линейного программирования

В случае большого количество стратегий у игроков и отсутствия седловой точки и цены игры в чистых стратегиях, к сожалению, рассмотренные нами аналитический и графический методы не применимы. В таких ситуациях оправдывает себя применение теории линейного программирования.

Теорема. *Любой конечной игре двух лиц с нулевой суммой* $G=(X_1,X_2,U)$ *соответствует задача линейного программирования, и наоборот.*

Рассмотрим конечную игру $G=(X_1,X_2,U)$, *где* U - матрица выигрышей первого игрока, размерности $m\times n$. Ожидаемая величина $\bar{v}$ цены игры в смешанных стратегиях задает условие поиска оптимальной смешанной стратегии первого игрока

$$\bar{v}=\max_{\mu_i^1}\min_j\sum_{i=1}^m\mu_i^1U_{ij}\,, \tag{14}$$

$$\sum_{i=1}^m\mu_i^1=1\,,\ \mu_i^1\ge 0\,,\ i=1..m\,. \tag{15}$$

Ожидаемый выигрыш $\bar{v}^1$ первого игрока при использовании им смешанной стратегии μ^1 составляет $\bar{v}^1=\min_j\sum_{i=1}^m\mu_i^1U_{ij}$, следовательно $\bar{v}^1\le\sum_{i=1}^m\mu_i^1U_{ij}$, а условие (14) будет иметь вид

$$\bar{v}=\max_{\mu_i^1}\bar{v}^1\,. \tag{16}$$

В итоге задача поиска оптимальной смешанной стратегии первого игрока имеет вид:

$$\begin{cases}\bar{v}^1=\min\limits_j\sum\limits_{i=1}^m\mu_i^1U_{ij}\to\max\\ \bar{v}^1\le\sum\limits_{i=1}^m\mu_i^1U_{ij}\,,j=1..n\\ \sum\limits_{i=1}^m\mu_i^1=1\\ \mu_i^1\ge 0,i=1..m\end{cases} \tag{17}$$

Если предположить, не ограничивая общность, что $\bar{v}>0$, то целесообразно ввести в рассмотрение величины, которые выступят как будущие переменные

$$x_i=\frac{\mu_i^1}{\bar{v}}\ge 0,i=1..n \tag{18}$$

Учитывая (15) и (18), условие (16) можно представить в виде:

$$\bar{v}=\max_{\mu_i^1}\bar{v}^1=\min_{\mu_i^1}\frac{1}{\bar{v}^1}=\min_{\mu_i^1}\frac{1}{\bar{v}^1}\sum_{i=1}^m\mu_i^1=\min_i\sum_{i=1}^m x_i \tag{19}$$

С учетом (18) и (19) из задачи (17) получаем задачу линейного программирования:

$$\begin{cases}\sum\limits_{i=1}^m x_i\to\min\\ \sum\limits_{i=1}^m U_{ij}x_i\ge 1,j=1..n\\ x_i\ge 0,i=1..m\end{cases} \tag{20}$$

Полученную задачу линейного программирования (20) можно решать симплекс-методом с применением метода искусственных переменных. Вычислив оптимальное решение $x^*=(x_1^*,x_2^*,...,x_m^*)$, из условия (19) находится значение цены игры в смешанных стратегиях, а из условия (18) – оптимальные смешанные стратегии первого игрока.

Для второго игрока, повторяя сделанные ранее рассуждения, приходим к двойственной по отношению к (20) задаче линейного программирования:

$$\begin{cases} \sum_{j=1}^{n} y_i \to \max \\ \sum_{j=1}^{n} U_{ij} y_j \le 1, i = 1..m \\ y_j \ge 0, j = 1..n \end{cases} \quad (21)$$

Таким образом, получив двойственную пару задач линейного программирования и используя основные теоремы двойственности, мы можем решать ту из задач пары, которая имеет меньшее число неизвестных либо решение которой получить легче из каких-либо других соображений.

На примере игр двух игроков с нулевой суммой мы показали возможность применение аппарата линейного программирования для решения задач теории игр.

К Разделу 8.2.

Рекомендуемая литература

1. Таха Х. А. Введение в исследование операций, 7-е издание. Пер. с англ. – М.: Издательский дом «Вильямс», 2005 – 912с.
2. Конюховский П. В. Математические методы исследования операций в экономике – СПб.:Питер, 200 – 208с.
3. Волков И.К., Загоруйко Е.А. Исследование операций: Учеб. для вузов / Под ред. В.С. Зарубина, А.П. Крищенко. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2000. - 436 с.
4. Акулич И.Л. Математическое программирование в примерах и задачах: Учеб. пособие для студентов. – Минск: Высшая школа, 1986. - 319 с

Примеры индивидуальных заданий

1. При помощи графического метода решить задачу линейного программирования с двумя неизвестными.
2. При помощи симплекс-метода решить задачу линейного программирования.

Вопросы для самоконтроля

1. Какая задача называется задачей линейного программирования в общей форме?
2. Сформулируйте задачу линейного программирования в стандартной форме.
3. Можно ли свести задачу линейного программирования в общей форме к задаче в стандартной форме?
4. Каковы этапы решения задачи линейного программирования графическим методом?
5. В чем состоят экстремальные свойства угловых точек области допустимых решений?
6. Всегда ли базисное решение отвечает некоторой угловой точке?
7. Сформулируйте задачу линейного программирования в канонической форме.
8. Каковы основные этапы алгоритма Жордана-Гаусса?
9. Укажите основные этапы алгоритма симплекс-метода.
10. В чем заключается метод искусственных переменных?
11. Каковы правила построения двойственных задач?
12. Сформулируйте основные теоремы теории двойственности.

К разделу 8.3.

Рекомендуемая литература

1. Таха Х. А. Введение в исследование операций, 7-е издание. Пер. с англ. – М.: Издательский дом «Вильямс», 2005 – 912с.

2. Конюховский П. В. Математические методы исследования операций в экономике – СПб.:Питер, 200 – 208с.
3. Волков И.К., Загоруйко Е.А. Исследование операций: Учеб. для вузов / Под ред. В.С. Зарубина, А.П. Крищенко. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2000. - 436 с.
4. Костюкова О.И. Исследование операций: Учеб. пособие для студ. – Минск: БГУИР, 2003 – 94 с.

Примеры индивидуальных заданий

1. Согласно существующим соглашениям компании необходимо развести груз по дилерской сети, которая охватывает 6 городов. Известно, что стоимость перевозки необходимого количества груза с i-го города в j-й равна c_{ij}. Отделу логистики компании необходимо организовать доставку груза одним автомобилем исходя из минимальной стоимости перевозки и условия, что в каждый город необходимо въехать и выехать из него только один раз.

Вопросы для самоконтроля

1. Какая задача называется задачей целочисленного линейного программирования?
2. Сформулируйте общий алгоритм метода ветвей и границ.
3. Сформулируйте задачу коммивояжера.
4. Почему удобно использовать специальные методы решения задачи коммивояжера?
5. Что такое константы приведения?
6. Дайте определение цикла и его стоимости.
7. В чем состоит критерий оценки стоимости цикла?
8. Сформулируйте критерий оценки нулей.
9. Каким образом выбирается первая пара городов при решении задачи коммивояжера?
10. Как можно проверить, что найденный маршрут оптимален?
11. Как подчитать стоимость оптимального маршрута?
12. В каких задачах метод прямого перебора решений оправдан?

К разделу 8.4.

Рекомендуемая литература

1. Перельман М.А. Исследование операций в задачах автомобильного транспорта: Учеб. пособие. – Х.: ИСИО, ХГАДТУ, 1995. – 135с.
2. Таха Х. А. Введение в исследование операций, 7-е издание. Пер. с англ. – М.: Издательский дом «Вильямс», 2005 – 912с.
3. Конюховский П. В. Математические методы исследования операций в экономике – СПб.:Питер, 200 – 208с.
4. Волков И.К., Загоруйко Е.А. Исследование операций: Учеб. для вузов / Под ред. В.С. Зарубина, А.П. Крищенко. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2000. - 436 с.

Примеры индивидуальных заданий

1. На пять товарных баз $A_i, i = 1..5$ поступил однородный груз соответственно в количестве $a_i, i = 1..5$ тонн. Данный груз необходимо перевезти в 8 пунктов назначения $B_j, j = 1..8$ в количестве $b_j, j = 1..8$ тонн, соотв. Стоимость перевозки единицы груза от i-го поставщика к j-му покупателю заданы матрицей стоимостей $C = \{c_{ij}\}$. Необходимо составить математическую модель задачи, построить начальный опорный план как минимум тремя методами (северо-западного угла, минимума по столбцу, двойного предпочтения) и найти при помощи метода потенциалов оптимальный план перевозок, исходя из минимальной стоимости.

Вопросы для самоконтроля

1. Сформулируйте транспортную задачу.
2. Чем открытая транспортная задача отличается от закрытой? Дайте определения.

3. В чем состоит распределительный метод?
4. Дайте определения цикла.
5. Что такое ациклический набор клеток?
6. Сформулируйте теорему о нахождении оптимального решения.
7. Проведите аналогию между распределительным методом и симплекс-методом.
8. Дайте определение опорного плана.
9. Укажите свойства опорного плана.
10. В чем состоит признак достижения оптимального решения?
11. Какие методы нахождения начального опорного плана вам известны?
12. Сформулируйте основные этапы метода потенциалов.

К разделу 8.5

Рекомендуемая литература

1. Конюховский П. В. Математические методы исследования операций в экономике – СПб.:Питер, 200 – 208с.
2. Перельман М.А. Исследование операций в задачах автомобильного транспорта: Учеб. пособие. – Х.: ИСИО, ХГАДТУ, 1995. – 135с.
3. Костюкова О.И. Исследование операций: Учеб. пособие для студ. – Минск: БГУИР, 2003 – 94 с.

Примеры индивидуальных заданий

1. При помощи метода Минти найти длины и минимальный путь (набор ребер) от заданного узла к другим узлам транспортной сети.

Вопросы для самоконтроля

1. Дайте определение сети.
2. Что такое неориентированный граф?
3. Дайте определение цикла.
4. Что такое интенсивность вершины?
5. Дайте определение потока в сети.
6. Какой поток называется невырожденным?
7. Дайте определение линейной сетевой задачи.
8. Какие задачи могут быть сведены к линейным сетевым задачам?
9. Сформулируйте транспортную задачу в сетевой постановке.
10. В чем состоит метод потенциалов для транспортной задачи в сетевой постановке?
11. Сформулируйте задачу про кратчайший путь.
12. Приведите основные этапы метода Минти.

К разделу 8.6

Рекомендуемая литература

1. Мулен Э. Теория игр с примерами из математической экономики: Пер. с франц. – М.: Мир, 1985. – 200с.
2. Волков И.К., Загоруйко Е.А. Исследование операций: Учеб. для вузов / Под ред. В.С. Зарубина, А.П. Крищенко. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2000. - 436 с.
3. Костюкова О.И. Исследование операций: Учеб. пособие для студ. – Минск: БГУИР, 2003 – 94 с.

Примеры индивидуальных заданий

1. Решить игру двух лиц с нулевой суммой при помощи аналитического и (или) графического методов.

Вопросы для самоконтроля

1. Дайте определение игры.
2. Приведите примеры классификации игр.
3. Что такое игра в нормальной форме?
4. Дайте определение единой платежной матрицы.

5. В чем особенность игр двух лиц с нулевой суммой?
6. Дайте определение цене игры в чистых стратегиях.
7. Всегда ли существует седловая точка в чистых стратегиях?
8. Что такое смешанная стратегия?
9. Сформулируйте теорему о цене игры в смешанных стратегиях.
10. В чем состоит аналитический метод решения матричной игры 2x2?
11. Какие основные идеи графического метода для игр 2xn?
12. Каким образом задача двух лиц с нулевой суммой связана с задачами линейного программирования?