1. Двойственный метод решения задач линейного программирования с двухсторонними ограничениями.

Рассмотрим задачу линейного программирования следующего вида: $c'x \to max,\ Ax = b,\ d_* \le x \le d^*.$ (30)

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

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

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

Итерация метода состоит из следующих шагов.

1. Найдем m-вектор $y' := c'_Б B$
и оценки $\Delta_j := y' A_j - c_j$, $j \in J$;
сформируем множества

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

2. Построим вектор $\aleph = (\aleph_j, j \in J)$ по следующему правилу:
$\aleph_j = d_{*j}$, $j \in J_H^+$; $\aleph_j = d^*_j$, $j \in J_H^-$;

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

3. Проверим критерий оптимальности: если выполняются соотношения $d_{*j} \le \aleph_j \le d^*_j,\ j \in J_Б,$ (31)
то вектор $x^0 := \aleph = (\aleph_j, j \in J)$ – оптимальный план задачи (30). Решение задачи (30) прекращается.
В противном случае (т.е. если соотношения (31) не выполняются) идем на шаг 4.

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

5. Положим $\mu_{j_k} = 1$, если $\aleph_{j_k} < d_{*j_k}$, $\mu_{j_k} = -1$, если $\aleph_{j_k} > d^*_{j_k}$.
Подсчитаем m-вектор $\Delta y' = \mu_{j_k} e'_k B$
и числа $\mu_j = \Delta y' A_j$, $j \in J_H^+ \cup J_H^-$.

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

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

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

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

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

$\bar{\Delta}_j = \Delta_j + \sigma_0 \mu_j$, $j \in J_H \cup j_k$; $\bar{\Delta}_j = 0$, $j \in J_Б \setminus j_k$.

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

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

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

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

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

$\bar{J}_H^+ = (J_H^+ \cup j_k)$, если $\mu_{j_k} = 1$, $j_* \notin J_H^+$;

$\bar{J}_H^+ = J_H^+$, если $\mu_{j_k} = -1$, $j_* \notin J_H^+$;

$\bar{J}_H^- = \bar{J}_H \setminus \bar{J}_H^+$.

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

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

2. Постановки задач целочисленного программирования. Связь и отличия от задач линейного программирования.

задача ЦЛП имеет вид $c'x \to max$, (10)

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

xj – целое, $j \in I$, (12)

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

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

Однако справедливо обратное.

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

Но «метод округления» нельзя считать универсальным.

Еще более неприятная особенность, свойственная задачам целочисленного программирования, состоит в том, что нет простого способа, позволяющего определить, является ли данный допустимый план оптимальным. В этом одно из главных отличий задач ЦЛП от задач ЛП.

3. Общие сведения о методах решения задач целочисленного программирования

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

К необходимости рассматривать целочисленные задачи мы приходим по двум причинам:

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

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

Рассмотрим ряд примеров задач второго типа. (подробно в билете 4)

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

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

3. Задача с дихотомией. В некоторой ситуации требуется выполнение k ограничений из общего числа m ограничений. При этом заранее не известно, какие именно ограничения должны выполняться.

4. Примеры задач, которые сводятся к задачам целочисленного программирования.

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

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

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

Естественно стремление минимизировать общие затраты по всем типам продукции: $$Z=\sum_{j=1}^{N} C_j(x_j) \to min.$$ (1)

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

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

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

Последнее условие можно переписать в виде $x_j \le M y_j$,(3)

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

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

при дополнительных ограничениях $0 \le x_j \le M y_j$, $y_j = 0 \vee 1$, $j=\overline{1,N}$. (5)

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

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

2. Задача планирования производственной линии. Предположим, что имеется один станок, на котором надо произвести n различных операций. На порядок выполнения некоторых операций наложены ограничения технологического характера и, кроме того, есть

ограничения, согласно которым каждая операция должна быть выполнена к заданному сроку.

Таким образом, в задаче имеются ограничения трех типов:

1) ограничения на последовательность выполнения операций;

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

3) наличие временных сроков на осуществление операций.

Рассмотрим первый тип ограничений. Пусть xj – момент начала выполнения j-й операции, aj – время, требуемое на выполнение j-й операции. Если операция i должна предшествовать операции j и должна начаться не ранее, чем через c_{ij}, где $c_{ij} > a_i$, после начала выполнения операции i, то должно выполняться неравенство

$$x_i + c_{ij} \le x_j, \quad (i,j) \in U_* . \qquad (6)$$

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

$x_i - x_j \ge a_j$, если в оптимальном решении j предшествует i, $(i,j) \notin U_*$,

или

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

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

$$y_{ij} = \begin{cases} 0, & \text{если } j \text{ предшествует } i, \\ 1, & \text{если } i \text{ предшествует } j. \end{cases} \qquad (i,j) \notin U_*$$

Пусть $M > 0$ достаточно велико, тогда ограничение вида «или–или» эквивалентно следующей системе неравенств для $(i,j) \notin U_*$:

$$My_{ij} + (x_i - x_j) \ge a_j, \qquad (7)$$

$$M(1 - y_{ij}) + (x_j - x_i) \ge a_i, \; y_{ij} = 0 \vee 1 . \qquad (8)$$

Действительно, если на оптимальном решении $y_{ij} = 0$, то ограничение (8) становится избыточным, а ограничение (7) – активным: $(x_i - x_j) \ge a_j$, $M + (x_j - x_i) \ge a_i$.

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

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

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

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

$x_j + a_j \le d_j, \quad j \in J = \{1,2,...,n\}$. (9)

Если обозначить через t суммарное время, требуемое для выполнения всех n операций, то рассматриваемая задача принимает вид $Z = t \to min$ при ограничениях (6) – (9) и

$x_j + a_j \le t, \quad j = \overline{1,n}; \quad y_{ij} = 0 \vee 1, \quad i = \overline{1,n}; \; j = \overline{1,n}, \quad (i,j) \notin U_*.$

3. Задача с дихотомией. В некоторой ситуации требуется выполнение k ограничений из общего числа m ограничений. При этом заранее не известно, какие именно ограничения должны выполняться.

Запишем ограничения в виде $g_i(x_1,...,x_n) \le b_i, \quad i = 1,2,...,m.$
Определим новую переменную

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

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

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

Аналогичная ситуация возникает, когда правая часть некоторого ограничения может принимать одно из нескольких значений: $g(x_1,x_2,...,x_n) \le b, \; b \in \{b_1,b_2,...,b_k\}$.
Данное ограничение можно записать в виде

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

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

5. Метод ветвей и границ (общая схема).

Этот метод применим как к полностью целочисленным задачам, так и частично целочисленным задачам.

Рассмотрим задачу

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

$Ax \le b$, (18)

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

xj – целое, $j \in I$, (20)

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

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

Идея метода ветвей и границ основана на следующем элементарном факторе. Рассмотрим любую переменную x_j , $j \in I$. Пусть l – некоторое целое число, такое что $d_{*j} \le l \le d_j^* - 1$.

Тогда оптимальное значение x_j^o будет удовлетворять либо неравенству $l+1 \le x_j \le d_j^*$,либо неравенству $d_{*j} \le x_j \le l$.

(как решать в билете 6)

6. Обоснование использования двойственного симплекс-метода при реализации метода ветвей и границ.

(общий вид в билете 5)

рассмотрим две задачи:

задачу $c'x \to \max$, (21)

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

$d_{*1} \le x_1 \le 1$, (23)

и задачу (21), (22) с условием $2 \le x_1 \le d_1^*$. (24)

Предположим теперь, что каждая из полученных задач имеет оптимальные целочисленные решения соответственно x01 и x02. Тогда очевидно, что решением исходной задачи (17) – (20) является тот из векторов x01 или x02, на котором значение $c'x$ больше.

2. АЛГОРИТМ. РАССМОТРИМ ОБЩУЮ ИТЕРАЦИЮ МЕТОДА. ПУСТЬ МЫ ОСУЩЕСТВЛЯЕМ Т-Ю ИТЕРАЦИЮ.

В начале t-й итерации имеем:

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

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

3) n-вектор μ, число μ_0.

Замечание. На первой итерации, т.е. при t=1, список задач ЛП состоит только из одной задачи (17) – (19). Для определения r_0^1 ,μ, μ_0 можно поступить следующим образом. Если известен какой-либо целочисленный план $\bar{x}$ задачи (17) – (20), то полагаем $r_0^1 = c'x$, $\mu = \bar{x}, \ \mu_0 = 1.$ Если такого плана нет, то полагаем $\mu_0 = 0$, в качестве μ берем любой n-вектор, а для построения оценки r_0^1 , удовлетворяющей неравенству $r_0^1 \le c'x^0$, где x0 – решение задачи (17) – (20), можно привлечь любую дополнительную информацию. Например, если $c \ge 0$ и $d_* \ge 0$, то очевидно, что $c'x^0 \ge 0$,

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

На произвольной итерации t выполняем следующие шаги.

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

Шаг 2. Решаем выбранную задачу ЛП. Если эта задача не имеет решения либо она имеет решение x^* , для которого $c'x^* \le r_0^t$,

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

Если выбранная задача ЛП имеет решение x^* и $c'x^* > r_0^t$,

то идем на шаг 3.

Шаг 3. Если на решении x^* задачи ЛП выполняется условие целочисленности (20), то фиксируем это решение, т.е. полагаем $\mu = x^*, \mu_0 = 1$. Изменяем оценку $r_0^{t+1} = c'x^*$,

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

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

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

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

В первой новой задаче ЛП эти ограничения имеют вид

$\bar{d}_{*j_0} \le x_{j0} \le [l_{j_0}]$,во второй задаче эти ограничения имеют вид

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

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

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

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

7. Отсекающая плоскость в задаче целочисленного программирования. Ее свойства.

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

1) отсекать имеющееся в наличии нецелочисленное решение задачи ЛП;

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

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

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

1) ограничения задачи (32), (33) совместны;

2) целевая функция ограничена сверху.

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

Рассмотрим любое линейное уравнение, которое можно получить из уравнений Ax = b путём алгебраических операций над этими уравнениями. В общем случае такое уравнение можно представить в виде y' Ax = y' b, (36)

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

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

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

По построению любой вектор x, удовлетворяющий Ax = b, должен удовлетворять и (37). Так как все $x_j \geq 0$, $j \in J$, и $[a_j] \leq a_j$, $j \in J$, то из (37) следует более слабое условие $\sum_{j=1}^{n} [a_j] x_j \leq \beta$. (38)

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

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

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

Условие (39) можно переписать в эквивалентной форме

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

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

В алгоритме используется не условие (40), а разность между (40) и (37).Определим значения f_j и f тождественными

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

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

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

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

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

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

8. Метод отсечений Гомори для полностью целочисленных задач (общая схема).

Общую схему алгоритма Гомори можно представить в виде последовательности следующих шагов.

Шаг 1. Найти оптимальное решение задачи линейного программирования.

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

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

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

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

Опишем перечисленные шаги алгоритма подробнее.

Рассмотрим шаг 1. Пусть в текущей задаче ЛП есть переменные $x_j \geq 0$, $j \in J \cup J_{иск}$,

и есть $\overline{m} \geq m$ основных ограничений. По построению $\overline{m} = m + |J_{иск}|$ и каждой искусственной переменной x_{j_*} поставлено в соответствие некоторое основное ограничение текущей задачи ЛП. Отметим, что это ограничение не входит в число основных ограничений исходной задачи ЛП или ЦЛП, т.е. является дополнительным отсекающим ограничением. Решим текущую задачу ЛП и обозначим через x_j^0, $j \in J \cup J_{иск}$, $J_Б^0 \subset J \cup J_{иск}$, $|J_Б^0| = \overline{m}$

ее оптимальный базисный план.

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

Шаг 3 описан подробно ранее.

Рассмотрим шаг 4. Предположим, что выбран некоторый индекс i_0, такой, что $i_0 \in J_Б^0 \cap J$, $x_{i_0}^0 > 0$ – нецелое.

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

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

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

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

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

то равенство (42) можно представить в виде

$x_{i_0} + \sum_{j \in J_н} a_j x_j = \beta$, (43)

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

$\sum_{j \in J_н} -f_j x_j + x_{j_*} = -f$, $x_{j_*} \geq 0$ – целое, (44)

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

Рассмотрим шаг 5. Ограничение (44) добавляем к основным ограничениям имеющейся задачи ЛП, индекс j_* и соответствующую ему переменную x_{j_*} добавляем к переменным текущей задачи ЛП. При этом произойдут следующие замены:

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

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

Отметим, что если оказалось, что $f_j = 0, j \in J_н$, то прекращаем решение исходной задачи, т.к. у нее нет допустимых (целочисленных) планов

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

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

9. Двойственный симплекс-метод как средство решения последовательности задач линейного программирования в методе Гомори.

Рассмотрим задачу:

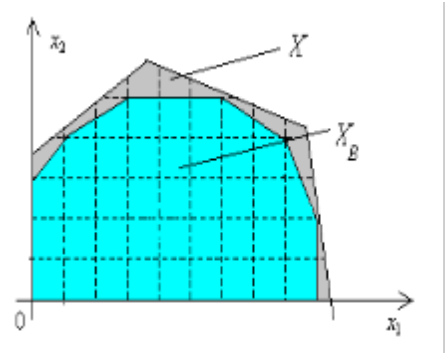
$c'x \to$ max, (32)

$Ax = b,\ x \geq 0,\ x_j$ – целое, $j \in J$, (33)

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

Рассмотрим теперь задачу ЦЛП (32), (33) с m основными ограничениями. В общем случае оптимальный план задачи (32), (33) будет содержать более чем m компонент, отличных от нуля. То есть задача должна содержать более чем m основных ограничений, т.е. мы должны к задаче (32) добавить еще некоторые основные ограничения.

Рассмотрим множество допустимых планов X задачи (32) (рис.1.3). Понятно, что множеством допустимых планов $X_ч$ задачи (32), (33) будут все «целые» точки, принадлежащие X. Рассмотрим ещё одно множество $X_в$, называемое выпуклой оболочкой точек $X_ч$. Согласно определению, $X_в$ – это наименьшее выпуклое множество, содержащее все точки $X_ч$ (см. рис. 1.3). Справедливы включения $X_ч \subset X_в \subset$ X. (34)



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

$c'x \to$ max, $x \xi X_в$ (35)

обязательно есть целочисленное решение и это решение будет решением задачи (32), (33). Мы видим, что множество $X_в$ получилось из X путём «отсечения» лишних кусков, не содержащих целых точек. При этом важно, что мы не отсекли ни одной целой точки! На рисунке для R^2 построить множество $X_в$ просто. При большом n сделать это заранее перед решением задачи невозможно.

Кроме того, нам не надо всё «чистое» множество X_B . Нам важно «высечь» только часть этого множества в окрестности оптимальной точки.

10. Основные принципы динамического программирования.

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

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

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

+ 11 билет

11. Общая схема применения динамического программирования при решении оптимизационных задач.

(что такое ДП из 10 билета)

Метод динамического программирования основан на трех главных этапах.

1. Инвариантное погружение исходной задачи в семейство аналогичных задач (исходная задача должна принадлежать этому семейству). На первом этапе, когда настроено инвариантное погружение, вводится функция Беллмана – оптимальное значение целевой функции произвольной задачи из рассматриваемого семейства.

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

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

12. Решение методом динамического программирования задачи распределения ресурсов.

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

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

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

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

Решим задачу (1) методом динамического программирования.

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

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

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

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

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

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

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

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

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

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

Таким образом, если запас сырья равен y, то при выделении k-му процессу z единиц ресурса от всех k процессов получаем прибыль $f_k(z) + B_{k-1}(y-z)$. (4)

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

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

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

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

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

Уравнение (6) называется уравнением Беллмана.

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

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

Таким образом, начальное условие для уравнения Беллмана (6) имеет вид $B_1(y) = f_1(y)$. (7)

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

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

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

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

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

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

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

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

$x^o = (x_1^o, x_2^o, \ldots, x_n^o)$ исходной задачи (1) определена: $x_n^0 = x_n^0(c)$.

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

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

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

13. Постановка задачи сетевого планирования. Расчет минимального времени выполнения комплекса работ.

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

Составим сетевую модель задачи. Факт (явление) начала (или окончания) какого-либо множества работ из заданной совокупности работ проекта назовем событием и поставим ему в соответствие узел i∈I. Работу, которая может начаться после события i∈I и которая предшествует событию j, обозначим дугой (i, j) ∈ U. Ни одна работа (i, j) ∈ U не может начаться, если не завершены все работы (k, i) ∈ U, k ∈ I_i^- = {k ∈ I: ⋔ (k, i) ∈ U}, т.е. ни одна работа (i, j) ∈ U не может начаться до наступления события i; момент наступления события i определяется моментом завершения всех работ (k, i), k ∈ I_i^- .

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

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

$$x_i + c_{ij} \le x_j,\ i \in I_j^-,\ j \in I, \qquad (9)$$

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

Из (9) следует, что в сети S нет контуров. Действительно, предположим, что в S существует контур и обозначим через $U_{\text{конт}}$ ∈ U дуги этого контура. Просуммируем неравенства (9) по дугам (i, j) ∈ $U_{\text{конт}}$ и получим $\sum_{(i,j)\in U_{\text{конт}}} c_{ij}$ ≥ 0.

Однако это противоречит предположению cij > 0, (i, j) ∈ U.

Далее будем предполагать, что

I_i^+ ≠ ∅, i ∈ I \ t; I_i^- ≠ ∅, i ∈ I \ s,

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

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

$$x_i^0 \ge 0, i \in I \setminus t;\ x_s^0 = 0 \qquad (10)$$

удовлетворяет неравенствам (9).

Поскольку для окончания проекта необходимо, чтобы все работы были завершены, то длина каждого пути из s в t, равная сумме ∑ cij , вычисленной вдоль дуг пути, не меньше чем x_t^0 . Чтобы убедиться в этом, достаточно сложить неравенства (9) вдоль этого пути. С другой

1

стороны, очевидно, что найдется такая последовательность работ, составляющая путь из s в t, общая продолжительность которых равна x_t^o.

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

Заметим, что сформулированная задача:

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

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

(+ весь 15 вопрос)

14. Связь задачи о расчете минимального времени выполнения комплекса работ и задачи о нахождении критического (максимальной длины) пути.

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

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

$$x_i^0 \geq 0, i \in I \setminus t; x_s^0 = 0 \qquad (10)$$

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

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

Заметим, что сформулированная задача:

найти $x_t - x_s \rightarrow$ min при условиях (9), (10) (11)

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

15. Построение критического пути методом динамического программирования (алгоритм).

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

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

Общая задача семейства состоит в построении максимального пути из s в любой узел j I. Длина Bj этого пути называется функцией Беллмана.

Осуществим второй этап – составим уравнение Беллмана, которому удовлетворяет функция Беллмана Bj. Для этого поступим следующим образом. Будем исследовать пути из s в j. Вначале предположим, что последней дугой пути из s в j является дуга (i, j) U, где i I_j^- , и что в узел i из s мы попали вдоль пути максимальной длины, т. е. вдоль пути длины Bi. Тогда длина рассматри-ваемого пути из s в j через i равнаBi + cij . (12)

Переберем все узлы i I_j^- и найдем максимальное среди чисел (12): $\max_{i \in I_j^-} (B_i + c_{ij})$. (13)

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

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

Уравнение (14) – это уравнение Беллмана.

Значение функции Беллмана для узла s известно:

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

Равенство (15) – краевое (начальное) условие для уравнения (14).

В отличие от рассмотренного ранее примера уравнение Беллмана (14) не является явно рекуррентным.

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

Алгоритм метода пометок. Обозначим через I* множество узлов i I, для которых известна функция Беллмана Bi. Множество I* не пусто, так как,

согласно (7), s I*. Если t I*, то задача решена, Bt – длина максимального пути

из s в t. Для восстановления этого пути надо осуществить «обратный ход» решения уравнения Беллмана. Эту процедуру рассмотрим ниже. Для того чтобы легче осуществить «обратный ход», будем для узлов i I* кроме функции Беллмана Bi задавать еще функцию f(i) I. На первом шаге имеем

I* = {s}, Bs = 0, f(s) = 0.

Пусть t I*. В сети S построим множество узлов w(I*) = {j I: (i, j) U, j I*, i I*}, соседних с I*. В силу того, что в сети S нет контуров, среди узлов j w(I*) обязательно найдется узел j*, для которого $I_{j^*}^-$ I*. (16)

Поскольку для узлов i I* значения Bi функции Беллмана уже известны, то из уравнения (14) можно найти

Bj* = $\max_{i \in I^*}$ (cij* + Bi) = ci*j* + Bi*. (17)

Полагаем I* = I* ∪ j* , f(j*) = i* и переходим к следующей итерации.

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

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

Осуществим «обратный ход» для нахождения максимального пути: {t, i1, i2, …, ik, s} по правилу:

i1 = f(t), i2 = f(i1), …, ik = f(ik-1), s = f(ik). (18)

16. Недостатки и достоинства метода динамического программирования (на примере задачи распределения ресурсов).

Достоинства метода:

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

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

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

Недостатки метода

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

17. Прикладные задачи, математические модели которых имеют сетевую структуру. Причины выделения сетевых задач в отдельный класс.

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

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

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

Виды задач:

1. классическая транспортная задача (в матричной форме)
2. сетевая модель транспортной задачи (задача о потоке минимальной стоимости)
3. Многопродуктовая транспортная задача)
4. Задача о построении дерева кратчайших путей
5. Задача о расчете минимального выполнения комплекса работ (задача о критическом пути)
6. З о назначениях
7. З коммивояжера
8. З о макс потоке

18. Задача о потоке минимальной стоимости. Постановка задачи и ее связь с общей задачей линейного программирования.

Большое значение имеет обобщение классической транспортной задачи путём включения в неё случаев, когда некоторые пункты являются транзитными.

Имеется n пунктов i ∈ I = {1,2,…,n}, которые связаны между собой системой дорог, которую удобно представлять в виде набора пар $(i,j) \in U \subset \{(i,j), i=\overline{1,n}, j=\overline{1,n}\}$.

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

Все пункты i ∈ I делятся на непересекающиеся три группы:

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

где i ∈ Iпр – пункты производства; обозначим через ai > 0 объёмы производства в этих пунктах;

i ∈ Iпотр – пункты потребления; обозначим через ai < 0 объёмы потребления в этих пунктах;

i ∈ Iтр – транзитные пункты, для них ai = 0.

Числа a_i, i ∈ I, называются интенсивностями узлов i∈I.

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

для каждого пункта соблюдалось условие баланса:

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

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

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

- суммарная стоимость всех перевозок была минимальна.

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

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

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

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

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

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

Задача (4), (5), (6') называется задачей о потоке минимальной стоимости с ограничениями на пропускные способности дуг. Методы решения задачи (4) – (6) рассмотрены в билете 21

19. Свойства дерева сети. Базисный поток. Критерий оптимальности базисного потока (доказательство).

$$\sum_{|i,j|\in U} c_{ij}x_{ij} \to \min, \qquad \sum_{j\in I_i^+} x_{ij} - \sum_{j\in I_i^-} x_{ji} = a_i, \quad i\in I, \tag{1}$$

$$x_{ij} \ge 0, \ (i,j)\in U,$$

где $I_i^+ = \{ j\in I : (i,j)\in U \}; \quad I_i^- = \{ j\in I : (j,i)\in U \};$

$$a_s = n-1; \ a_i = -1, \ i\in I \setminus s: \quad n = |I|. \tag{2}$$

было доказано, что если в задаче (1) существует оптимальный поток, то для нее найдется и оптимальный базисный поток, который определяется следующими свойствами: пусть $x^0 = (x_{ij}^0, (i,j)\in U)$ – оптимальный базисный поток, тогда существует такое базисное множество (являющееся деревом) $U_Б^0 \subset U$, что

$$x_{ij}^0 \ge 0, \ (i,j)\in U_в^0, \quad x_{ij}^0 = 0, \ (i,j)\in U \setminus U_в^0. \tag{3}$$

В силу специального подбора чисел $a_i, i\in I$, можно утверждать, что $x_{ij}^0 > 0$ (и даже $x_{ij}^0 \ge 1$), $(i,j)\in U_в^0$.

Покажем, что базисное множество $U_Б^0$ и будет деревом кратчайших путей из s в $j_* \in I \setminus S$. Действительно, выше отмечалось, что множество $U_Б^0$ является деревом. Из свойств дерева следует, что для любого узла $j_* \in I \setminus S$ существует единственная цепь, принадлежащая $U_Б^0$ и соединяющая s и j_*. В силу (3) ненулевые потоки могут быть только на дугах дерева $U_Б^0$. У нас только один узел-источник (с $a_i > 0$) – это узел s. Значит, единичный поток, который попадает в узел j_* с $a_{j_*} = -1 < 0$, приходит в узел j_* из узла s. Следовательно, все дуги упомянутой цепи должны быть прямыми. Значит, эта цепь является путем из s в j_*. Обозначим дуги этого пути через $U_п^{j_*} \subset U_в^0$. Так как из s в j_* вдоль пути $U_п^{j_*}$ идет единица потока, то по построению $x_{ij}^0 \ge 1, (i,j)\in U_п^{j_*}$. Предположим, что существует другой путь из s в j_* меньшей длины, т.е. существует такой путь $\overline{U}_п^{j_*}$, что

$$\sum_{(i,j)\in \overline{U}_п^{j_*}} c_{ij} < \sum_{(i,j)\in U_п^{j_*}} c_{ij}. \tag{4}$$

Построим новый поток $x^* = (x_{ij}^*, (i,j)\in U)$ в сети S:

$x_{ij}^* = x_{ij}^0 + 1$, если $(i,j)\in \overline{U}_п^{j_*}$, $(i,j)\notin U_п^{j_*}$,

$x_{ij}^* = x_{ij}^0 - 1$, если $(i,j)\notin \overline{U}_п^{j_*}$, $(i,j)\in U_п^{j_*}$,

$x_{ij}^* = x_{ij}^0$ для остальных дуг $(i,j)\in U$.

Нетрудно проверить, что x^* – допустимый поток в сети S, причем

$$\sum_{(i,j)\in U} c_{ij}(x_{ij}^* - x_{ij}^0) = \sum_{(i,j)\in \overline{U}_п^{j_*}} c_{ij} - \sum_{(i,j)\in U_п^{j_*}} c_{ij} < 0.$$

Однако последнее неравенство противоречит оптимальности потока x^0 в сети S.

Таким образом, мы показали, что, решив задачу (1), (2), мы находим дерево $U_{в}^{o}$ (оптимальный базис, соответствующий потоку минимальной стоимости), которое является деревом кратчайших путей из s в узлы $j \in I \setminus S$

20. Достаточное условие неограниченности снизу целевой функции в задаче о потоке минимальной стоимости (доказательство).

Предположим,что для базисного плана x с базисным множеством $J_{\text{б}}$ достаточное условие оптимальности (1.17) не выполняется, т.е. $\exists j_0 \in J_{\text{н}}$:

$$\Delta_{j_0} < 0.$$

Рассмотрим случай, когда все компоненты вектора $z = A_{\text{б}}^{-1} A_{j_0}$ неположительны, т.е.

$$(z_i, i = \overline{1,m})' = A_{\text{б}}^{-1} A_{j_0} \leq 0.$$

В этом случае легко видеть, что компоненты вектора $\overline{x} = (\overline{x}_{\text{б}}, \overline{x}_{\text{н}})$, построенного по правилам

$$\overline{x}_{\text{н}} = \Delta x_{\text{н}}, \quad \Delta x_{j_0} = \theta \geq 0, \Delta x_j = 0, j \in J_{\text{н}} \backslash j_0,$$
$$\overline{x}_{\text{б}} = x_{\text{б}} - \theta A_{\text{б}}^{-1} A_{j_0} = x_{\text{б}} - \theta z,$$

положительны при $\forall \theta \geq 0$. Следовательно, вектор $\overline{x}$ — допустимый план при $\forall \theta \geq 0$ и на нем приращение целевой функции равно

$$c'\overline{x} - c'x = -\Delta_{j_0}\theta > 0.$$

При $\theta \to \infty$ целевая функция неограниченно возрастает. Таким образом, мы доказали следующую теорему.

Теорема 2 *Если среди оценок $\Delta_{\text{н}}$ базисного плана x существует отрицательная (Δ_{j_0}) и ей соответствует вектор $A_{\text{б}}^{-1} A_{j_0} = z$ с неположительными компонентами, то целевая функция исходной задачи линейного программирования неограничена сверху на множестве допустимых планов.*

21. Решение задачи о потоке минимальной стоимости методом потенциалов.

Пример. Рассмотрим сеть $S=\{I,U\}$, изображенную на Рис.1, где на каждой дуге (i,j) красными цифрами указаны «стоимости» c_{ij}. Для этой сети задан начальный допустимый базисный поток $\{x,U_A\},\ x=(x_{ij},(i,j)\in U),\ \ U_A\subset U.$ На Рис. 5.1 значения дуговых потоков x_{ij} указаны черными цифрами на дугах, дуги множества U_A выделены красным цветом.

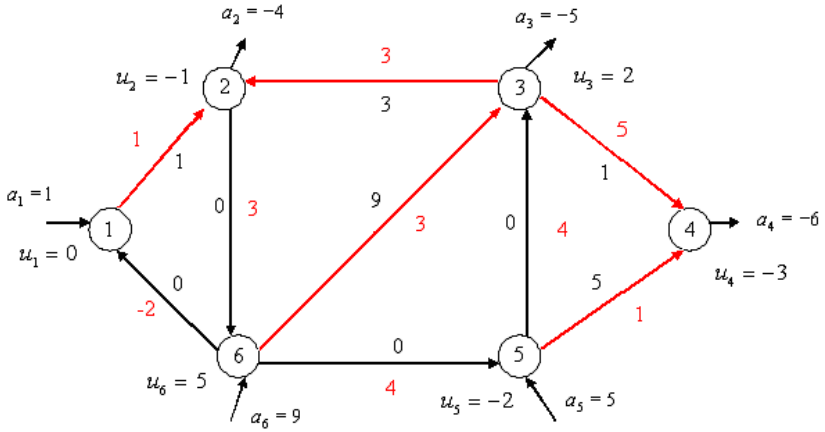


Рис. 5.1.

Итерация 1.

Используя базисное множество дуг U_A, подсчитаем потенциалы $u_i, i\in I,$ узлов по правилу $u_i-u_j=c_{ij},\ (i,j)\in U_A,\ u_1=0.$

Найденные потенциалы приведены на Рис. 5.1. Зная потенциалы, подсчитаем оценки $\Delta_{ij}=u_i-u_j-c_{ij},\ (i,j)\in U_H=U\setminus U_A,$ и проверим выполнение условий $\Delta_{ij}\le 0,\ (i,j)\in U_H.$ (5.1)

На данной итерации условия (5.1) не выполняются, поскольку для дуги (6,1) $\in U_H$ имеем $\Delta_{61}=5-0-(-2)=7>0.$

Полагаем $(i_0,j_0)=(6,1)$. Во множестве дуг $U_A\cup(i_0,j_0)$ найдем цикл и выделим в этом цикле множество прямых $U^+_{\text{цикл}}$ и обратных $U^-_{\text{цикл}}$. Направление движения по циклу задается дугой $(i_0,j_0)=(6,1)$. В результате получим

$$U^+_{\text{цикл}}=\{(6,1),(1,2)\},\ \ U^-_{\text{цикл}}=\{(6,3),(3,2)\}.$$

Найдем $\theta:=\min\limits_{(i,j)\in U^-_{\text{цикл}}} x_{ij}=x_{32}=3.$ Положим $(i_*,j_*)=(3,2).$

Построим новый поток $\overline{X} = (\overline{x}_{ij}, (i,j) \in U)$ по правилу

$$\overline{x}_{ij} = x_{ij},\ (i,j) \in U \setminus U^{+}_{цепи};\ \overline{x}_{ij} = x_{ij} + \theta,\ (i,j) \in U^{+}_{цепи};\ \ \overline{x}_{ij} = x_{ij} - \theta,\ (i,j) \in U^{-}_{цепи}.$$

Для нового потока построим новое базисное множество дуг по правилу $\overline{U}_A = (U_A \setminus (i_*, j_*)) \cup (i_0, j_0).$

На Рис. 5.2 приведены новые дуговые потоки $\overline{x}_{ij}, (i,j) \in U,$ и новое множество базисный дуг $\overline{U}_A$.

Переходим к новой итерации, используя новый базисный поток $\{\overline{x}, \overline{U}_A\}$ в качестве исходного базисного потока $\{x, U_A\}$.

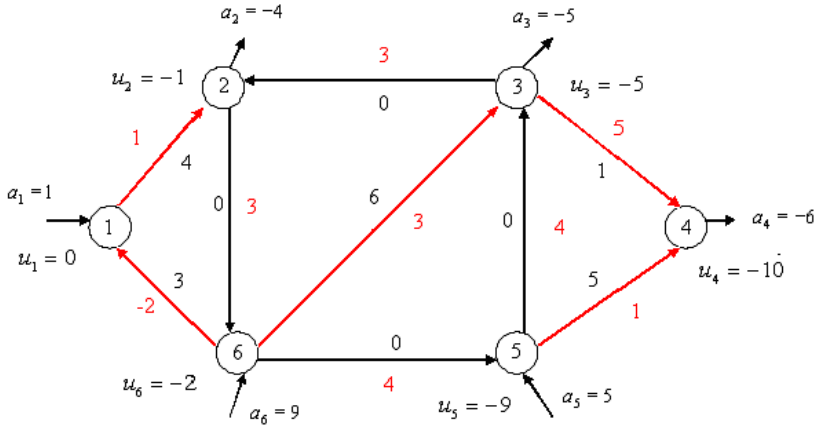


Рис. 5.2.

Итерация 2.

Используя базисное множество дуг U_A, подсчитаем потенциалы $u_i, i \in I,$ узлов по правилу $u_i - u_j = c_{ij},\ (i,j) \in U_A,\ u_1 = 0.$ Найденные потенциалы приведены на Рис. 5.2. Зная потенциалы, подсчитаем оценки $\Delta_{ij} = u_i - u_j - c_{ij}, (i,j) \in U_H = U \setminus U_A,$ и проверим выполнение условий (5.1). На данной итерации эти условия не выполняются, поскольку для дуги (6,5) $\in U_H$ имеем $\Delta_{65} = -2 - (-9) - 4 = 3 > 0.$ Полагаем $(i_0, j_0) = (6,5)$.

Во множестве дуг $U_A \cup (i_0, j_0)$ найдем цикл и выделим в этом цикле множество прямых $U^{+}_{цепи}$ и обратных $U^{-}_{цепи}$. Направление движения по циклу задается дугой $(i_0, j_0) = (6,5)$. В результате получим

$$U^{+}_{цепи} = \{(6,5),(5,4)\},\ \ U^{-}_{цепи} = \{(6,3),(3,4)\}.$$

$$\theta := \min_{(i,j)\in U^-_{\text{цепи}}} x_{ij} = x_{34} = 1.$$

Найдем $(i_*, j_*) = (3,4).$ Положим

Построим новый поток $\overline{X} = (\overline{x}_{ij}, (i,j)\in U)$ по правилу

$\overline{x}_{ij} = x_{ij}, (i,j)\in U \setminus U_{\text{цепи}}; \overline{x}_{ij} = x_{ij} + \theta, (i,j)\in U^+_{\text{цепи}}; \ \overline{x}_{ij} = x_{ij} - \theta, (i,j)\in U^-_{\text{цепи}}.$

Для нового потока построим новое базисное множество дуг по правилу

$\overline{U}_A = (U_A \setminus (i_*, j_*)) \cup (i_0, j_0).$

На Рис. 5.3 приведены новые дуговые потоки $\overline{x}_{ij}, (i,j)\in U,$ и новое множество базисный дуг $\overline{U}_A$.

Переходим к новой итерации, используя новый базисный поток $\{\overline{x}, \overline{U}_A\}$ в качестве исходного базисного потока $\{x, U_A\}$.

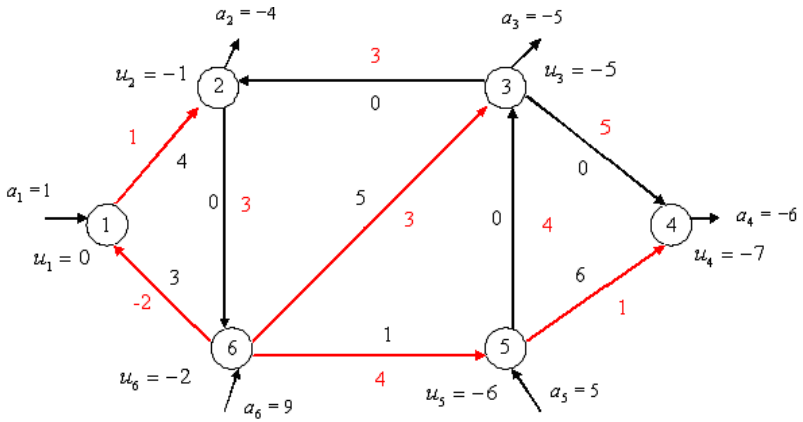


Рис.5.3.

Итерация 3.

Используя текущее базисное множество дуг U_A, подсчитаем потенциалы $u_i, i\in I,$ узлов по правилу $u_i - u_j = c_{ij}, \ (i,j)\in U_A, \ u_1 = 0.$

Найденные потенциалы приведены на Рис. 5.3. Зная потенциалы, подсчитаем оценки $\Delta_{ij} = u_i - u_j - c_{ij}, (i,j)\in U_H = U \setminus U_A,$ и проверим выполнение условий (5.1). На данной итерации эти условия выполняются. Следовательно, текущий поток (приведенный на Рис. 5.3) является оптимальным потоком. Задача решена.

Ответ: оптимальный поток

$x_{12}=4, x_{61}=3, x_{63}=5, x_{65}=1, x_{54}=6, x_{26}=0, x_{23}=0, x_{34}=0, x_{53}=0.$

22. Первая фаза метода потенциалов (построение начального базисного потока). Анализ решения задачи первой фазы.

Приведем этот упрощенный вариант метода потенциалов, предназначенный для решения задачи (1), (2).

Опишем вначале алгоритм построения начального базиса $U_Б$, который является деревом путей (необязательно кратчайших) из s в произвольный узел $j \in I$. Это – аналог алгоритма первой фазы метода потенциалов.

Положим $I_* = \{s\}, B_s = 0, f(s) = s.$ Пусть на некотором шаге известны множество $I_*,$ числа $B_j, f(j), j \in I_*.$ Если $I_* = I,$ то начальный базис-дерево $U_Б$ построен. Его легко восстановить по вторым меткам $f(i), i \in I,$ согласно правилам (14), (15).

Пусть $I_* \neq I.$ Если существует узел $j \in I \setminus I_*,$ для которого найдется дуга $(i,j) \in U \; c \; i \in I_*,$ то полагаем $B_j = B_i + c_{ij}, \; f(j) = i, \quad I_* := I_* \cup j.$

Повторяем описанные выше операции с новым множеством $I_*.$

Если $I_* \neq I$ и не существует узла $j \in I \setminus I_*,$ для которого найдется дуга $(i,j) \in U \; c \; i \in I_*,$ то в заданной сети $S = \{I, U\}$ нельзя построить дерево кратчайших путей, поскольку не во все узлы есть пути из s.

Пусть начальный базис-дерево $U_Б$ построен. При этом будут построены и числа $B_j, j \in I,$ соответствующие $U_Б.$ Очевидно, что B_j – длина пути из s в j вдоль дуг дерева $U_Б.$

23. Постановка задачи о кратчайшем пути. Связь данной задачи с задачей о потоке минимальной стоимости.

Покажем, что задача построения кратчайших путей из заданного узла s во все другие узлы $i \in I$ \s сети $S=\{I,U\}$ является частным случаем задачи о потоке минимальной стоимости [2, 3, 7], которая, в свою очередь, является частным случаем задачи линейного программирования.

Рассмотрим задачу построения кратчайших путей из узла s в узлы $i \in I$ \s. Покажем, что эта задача эквивалентна следующей задаче о потоке минимальной стоимости на сети $S=\{I,U\}$:

$$\sum_{|i,j|\in U} c_{ij}x_{ij} \to \min, \qquad \sum_{j\in I_i^+} x_{ij} - \sum_{j\in I_i^-} x_{ji} = a_i, \quad i \in I, \tag{1}$$

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

где $I_i^+ = \{ j\in I:(i,j)\in U \}; \quad I_i^- = \{ j\in I:(j,i)\in U \};$

$$a_s = n-1; \ a_i = -1, \ i \in I \setminus s; \quad n = |I|. \tag{2}$$

С учетом того, что все числа $a_i, i\in I,$ – целые, можно показать, что задача (1) имеет оптимальный целочисленный поток. Ранее в курсе методов оптимизации [9] было доказано, что если в задаче (1) существует оптимальный поток, то для нее найдется и оптимальный базисный поток, который определяется следующими свойствами: пусть $x^0 = (x_{ij}^0, (i,j)\in U)$ – оптимальный базисный поток, тогда существует такое базисное множество (являющееся деревом) $U_Б^0 \subset U,$ что $x_{ij}^0 \geq 0, \ (i,j)\in U_в^0, \ x_{ij}^0 = 0, \ (i,j)\in U \setminus U_в^0.$ (3)

В силу специального подбора чисел $a_i, i\in I$, можно утверждать, что $x_{ij}^0 > 0$ (и даже $x_{ij}^0 \geq 1$), $(i,j)\in U_в^0$.

Покажем, что базисное множество $U_Б^0$ и будет деревом кратчайших путей из s в $j \in I \setminus S$. Действительно, выше отмечалось, что множество $U_Б^0$ является деревом. Из свойств дерева следует, что для любого узла $j_* \in I \setminus S$ существует единственная цепь, принадлежащая $U_Б^0$ и соединяющая s и j_*. В силу (3) ненулевые потоки могут быть только на дугах дерева $U_Б^0$. У нас только один узел-источник (с $a_i > 0$) – это узел s. Значит, единичный поток, который попадает в узел j_* с $a_{j_*} = -1 < 0$, приходит в узел j_* из узла s. Следовательно, все дуги упомянутой цепи должны быть прямыми. Значит, эта цепь является путем из s в j_*. Обозначим дуги этого пути через $U_п^{j_*} \subset U_в^0$. Так как из s в j_* вдоль пути $U_п^{j_*}$ идет единица потока, то по построению $x_{ij}^0 \geq 1, (i,j)\in U_п^{j_*}$. Предположим, что существует другой путь из s в j_* меньшей длины, т.е. существует такой путь $\overline{U}_п^{j_*}$, что $\sum_{(i,j)\in \overline{U}_п^{j_*}} c_{ij} < \sum_{(i,j)\in U_п^{j_*}} c_{ij}.$ (4)

Построим новый поток $x^* = (x_{ij}^*, (i,j)\in U)$ в сети S:

$x^*_{ij} = x^0_{ij} + 1$, если $(i,j) \in \overline{U}^{\,0}_{n}$, $(i,j) \notin U^{\,0}_{n}$,

$x^*_{ij} = x^0_{ij} - 1$, если $(i,j) \notin \overline{U}^{\,0}_{n}$, $(i,j) \in U^{\,0}_{n}$,

$x^*_{ij} = x^0_{ij}$ для остальных дуг $(i,j) \in U$.

Нетрудно проверить, что x^* – допустимый поток в сети S, причем $\sum_{(i,j)\in U} c_{ij}\,(x^*_{ij} - x^0_{ij}) = \sum_{(i,j)\in \overline{U}^{\,0}_{n}} c_{ij} - \sum_{(i,j)\in U^{\,0}_{n}} c_{ij} < 0.$

Однако последнее неравенство противоречит оптимальности потока x^0 в сети S.

Таким образом, мы показали, что, решив задачу (1), (2), мы находим дерево $U^{\,0}_{n}$ (оптимальный базис, соответствующий потоку минимальной стоимости), которое является деревом кратчайших путей из s в узлы $j \in I \setminus S$.

Следовательно, для построения дерева кратчайших путей можно использовать метод потенциалов [2, 3], рассмотренный ранее в курсе «Методы оптимизации» [9].

Заметим, что задача (1), (2) является частным случаем общей задачи о потоке минимальной стоимости. Специфика задачи (1), (2) состоит в специальной структуре интенсивностей $a_i, i \in I.$ Кроме того, мы получим дополнительную специфику, если, например, предположим, что $c_{ij} \geq 0,\ (i,j) \in U.$ Учет этой специфики позволяет разработать специальные методы, учитывающие особенности данной задачи.

24. Дерево кратчайших путей. Критерий существования решения.

Рассмотрим сеть (ориентированную) $S=\{I,U\}$ со множеством узлов I и дуг U. Каждой ориентированной дуге $(i,j)\in U$ поставим в соответствие пару точек $\{i,j\}$, которую назовем ребром с граничными узлами i и j.

Последовательность различных ребер

$\{i_1,i_2\}, \{i_2,i_3\}, \ldots, \{i_{k-1},i_k\}$

называется (простой) цепью, соединяющей узлы i_1 и i_k. Пусть данная цепь рассматривается в направлении от узла i_1 к узлу i_k. Если это направление совпадает с направлением $i \to j$ дуги (i,j) в этой цепи, то дуга (i,j) называется прямой. Дуга с противоположным направлением называется обратной.

Путем из узла $s\in I$ в узел $t\in I$ называется простая цепь, соединяющая s и t, при этом все дуги цепи являются прямыми при движении из s в t.

Цепь (путь) $\{i_1,i_2\}, \{i_2,i_3\}, \ldots, \{i_{k-1},i_k\}$ с совпадающими узлами i_1 и i_k: $i_1 = i_k$ называется циклом (контуром).

Каждой дуге $(i,j)\in U$ припишем некоторый параметр c_{ij} («стоимость»), который определяет длину дуги (i,j). В этом случае под длиной пути будем понимать сумму длин дуг, образующих данный путь.

В §1 данной главы изучается задача о нахождении кратчайшего (минимального) пути из фиксированного узла s в другой заданный узел t или во все другие узлы сети. В последнем случае говорят о задаче построения дерева кратчайших путей. Следует различать три случая:

1. Все дуги сети имеют неотрицательную длину.
2. Некоторые дуги имеют отрицательную длину, но в сети не существует контуров с суммарной отрицательной длиной.
3. В сети существует один или несколько контуров с отрицательной суммарной длиной.

В третьем случае задача о построении кратчайшего пути из s в t может не иметь решения. (Задача будет иметь решение только в том случае, если нет ни одного пути из s в узел i, принадлежащий контуру с отрицательной длиной.) В таких ситуациях используются алгоритмы, позволяющие обнаружить и указать контуры с отрицательной длиной.

Заметим, что отсутствие в сети $S=\{I,U\}$ контуров отрицательной длины и существование пути из i в j для любых i, $j^{\in I}$ являются необходимыми и достаточными условиями существования кратчайшего пути из любого узла $i\in I$ в любой узел $j\in I$.

25. Использование идей динамического программирования для построения кратчайших путей.

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

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

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

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

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

а) пути, не содержащие узел $j+1$;

б) пути, содержащие узел $j+1$.

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

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

Следовательно, согласно определению функции Беллмана, имеем $d_{ik}^{j+1}=min\{d_{ik}^{j},d_{ij+1}^{j}+d_{j+1k}^{j}\},\ \forall i,\ k\in I$. (18)

Уравнение (18) – это уравнение Беллмана. Из него, в частности, следует, что $d_{j+1k}^{j+1}=d_{j+1k}^{j},\ \ d_{ij+1}^{j+1}=d_{ij+1}^{j},\ \forall i,k\in I$.

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

Зададим начальные условия для уравнения Беллмана:

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

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

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

пути.

26. Метод пометок для построения дерева кратчайших путей.

2. Перед первой итерацией алгоритма полагаем

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

Пусть перед началом текущей итерации известно множество узлов I_* сети, для которых найдены значения функции Беллмана $B_i, i \in I_*$, а также некоторой функции $f(i) \in I_*, i \in I_*$.

Обозначим через $\omega(I_*) = \{j \in I : (i, j) \in U(I_*)\}$ множество узлов, соседних со множеством I_*. Если $\omega(I_*) =$ ∅, то в сети S нет путей из s в t.

Пусть $\omega(I_*) \neq$ ∅. Подсчитаем числа (временные метки):

$$B'_j = \min_{i \in I_* \cap I_j^-} (B_i + c_{ij}), \quad j \in \omega(I_*), \tag{12}$$

и найдем минимальное среди них

$$B'_{j_*} = \min B'_j, \quad j \in \omega(I_*). \tag{13}$$

Узлу j_*, на котором реализовался минимум в (13), приписываем постоянную метку

$B_{j_*} = B'_{j_*}$,

полагаем $f(j_*) = i_*$, где $i_* \in I_*$ – такой узел, что существует дуга $(i_*, j_*) \in U$ и $B_{j_*} = B_{i_*} + c_{i_* j_*}$. Узел j_* добавляем к узлам I_*. Переходим к следующей итерации.

На каждой итерации алгоритма количество узлов, имеющих постоянные метки, увеличивается на единицу. Следовательно, через конечное число шагов придем к одной из следующих ситуаций:

а) $t \in I_*$; б) $\omega(I_*) =$ ∅.

Случай «б» означает, что в сети $S = \{I, U\}$ нет путей из s в t. Случай «а» означает, что B_t – длина кратчайшего пути из s в t. Для построения самого кратчайшего пути из s в t осуществим «обратный ход», используя функцию $f(i), i \in I_*$. С этой целью определим узлы по правилу

$$i_0 := f(t),\ i_1 := f(i_0),\ i_2 := f(i_1),\ \ldots,$$
$$i_k := f(i_{k-1}),\ \ldots,\ i_m := f(i_{m-1}) = s \tag{14}$$

.

Тогда путь из s в t имеет вид

$$s \to i_{m-1} \to \ldots \to i_k \to i_{k-1} \to \ldots \to i_1 \to i_0 \to t. \tag{15}$$

27. Алгоритмы Дейкстры.

Описанный метод пометок можно сделать еще более детальным. Приводимая ниже модификация называется «жадным» алгоритмом, или алгоритмом Дейкстры. Опишем этот алгоритм.

ПЕРЕД НАЧАЛОМ РАБОТЫ АЛГОРИТМА ПОЛАГАЕМ

$I_* = \{s\}, B_s = 0, f(s) = s$, B'_j =□, $f'(j)=0, j \in I \setminus s, i_* = s.$

Пусть заданы множество и числа $I_*, B_j, f(i), i \in I_*, B'_j, f'(j), j \in I \setminus I_*$, а также индекс $i_* \in I_*$. Опишем алгоритм Дейкстры по шагам.

Шаг 1. Рассмотрим узлы

$$\tilde{I}^{+}_{i_*} = \{ j \in I \setminus I_* : \exists (i_*, j) \in U \} = I^{+}_{i_*} \cap (I \setminus I_*).$$

Для любого $j \in \tilde{I}^{+}_{i_*}$ при $B'_j > B_{i_*} + c_{i_* j}$ положим

$B'_j := B_{i_*} + c_{i_* j}, f'(j) := i_*,$

в противном случае (т.е. при $B'_j \le B_{i_*} + c_{i_* j}$) временные метки узла j не меняем $: B'_j := B'_j, f'(j) = f'(j)$. Переходим к шагу 2.

Шаг 2. Среди всех узлов $j \in I \setminus I_*$ (т.е. узлов с временными метками) находим узел j_* с минимальной временной меткой:

$B'_{j_*} = \min B'_j, j \in I \setminus I_*.$

Если $B'_{j_*} =$ □, то STOP – алгоритм заканчивает свою работу, так как в сети нет путей из s в t. Если $B'_{j_*} <$ □, то переходим к шагу 3.

Шаг 3. Полагаем $B_{j_*} := B'_{j_*}, \quad f(j_*) = f'(j_*)$, т.е. узлу j_* приписываем постоянные метки и относим узел j_* ко множеству I_*, заменяя I_* на $I_* := = I_* \cup j_*$. Полагаем $i_* := j_*$ и переходим к шагу 1, если $t \notin I_*$. Случай $t \in I_*$ означает, что кратчайший путь из s в t найден. Восстанавливаем этот путь по вторым постоянным меткам $f(i), i \in I_*,$ согласно правилу (14), (15). Останавливаем работу алгоритма.

Заметим, что описанные алгоритмы обладают достоинством, о котором уже говорилось при изучении метода динамического программирования: мы можем изменить условия задачи и быстро найти решение новой задачи. Например, мы можем легко найти кратчайший путь из s в другой узел $j_0 \in I$, отличный от t. Действительно, если узел j_0 уже принадлежит множеству помеченных узлов I_*, то кратчайший путь из s в j_0 можно сразу восстановить по вторым постоянным меткам $f(i), i \in I_*$. Если же данный узел $j_0 \notin I_*$, т.е. не имеет еще постоянной метки, то работу алгоритма надо продолжить до тех пор, пока ни реализуется одна из ситуаций:

$B'_{j_*} = \min_{j \in I \setminus I_*} B'_j = \infty$

$j_0 \in I_*$ либо

28. Построение кратчайших путей в сети с отрицательными длинами дуг.

Предположим теперь, что в сети $S=\{I,U\}$ могут быть дуги с отрицательной длиной, но нет контуров с отрицательной суммарной длиной. В этом случае описанные выше алгоритмы, основанные на идеях динамического программирования, не применимы. В данной ситуации разумно использовать метод потенциалов [2] для решения задачи (1), (2). Отметим, что в силу специфики данной задачи (см. условие (2)) метод потенциалов [2] можно упростить. Приведем этот упрощенный вариант метода потенциалов, предназначенный для решения задачи (1), (2).

Опишем вначале алгоритм построения начального базиса $U_Б$, который является деревом путей (необязательно кратчайших) из s в произвольный узел $j \in I$. Это – аналог алгоритма первой фазы метода потенциалов.

Положим $I_* = \{s\}, B_s = 0, f(s) = s.$ Пусть на некотором шаге известны множество I_*, числа $B_j, f(j), j \in I_*$. Если $I_* = I$, то начальный базис-дерево $U_Б$ построен. Его легко восстановить по вторым меткам $f(i), i \in I$, согласно правилам (14), (15).

Пусть $I_* \neq I$. Если существует узел $j \in I \setminus I_*$, для которого найдется дуга $(i,j) \in U \ c \ i \in I_*$, то полагаем $B_j = B_i + c_{ij}$, $f(j) = i$, $I_* := I_* \cup j$.

Повторяем описанные выше операции с новым множеством I_*.

Если $I_* \neq I$ и не существует узла $j \in I \setminus I_*$, для которого найдется дуга $(i,j) \in U \ c \ i \in I_*$, то в заданной сети $S=\{I,U\}$ нельзя построить дерево кратчайших путей, поскольку не во все узлы есть пути из s.

Пусть начальный базис-дерево $U_Б$ построен. При этом будут построены и числа $B_j, j \in I$, соответствующие $U_Б$. Очевидно, что B_j – длина пути из s в j вдоль дуг дерева $U_Б$. Общий вид множества дуг $U_Б$ приведен на рис. 3.2.

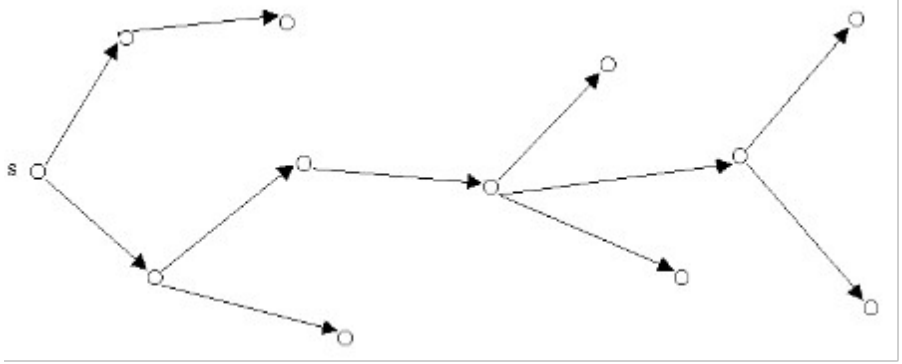


Рис. 3.2

Опишем шаги алгоритма построения дерева кратчайших путей

Шаг 1. Для небазисных дуг $U_H = U \setminus U_Б$ проверяем условия оптимальности $B_i + c_{ij} \geq B_j, (i,j) \in U_H$. (16)

Если соотношения (16) выполняются, то текущее дерево $U_Б$ является деревом кратчайших путей. Алгоритм прекращает работу. Если найдется дуга $(i_0, j_0) \in U_H$, для которой

$B_{i_0} + c_{i_0 j_0} < B_{j_0}$, то зафиксируем эту дугу и перейдем к шагу 2.

Шаг 2. Рассмотрим множество дуг $U_Б \cup (i_0, j_0)$. В [2] показано, что добавление дуги $(i_0, j_0) \in U_H$ к дереву $U_Б$ приводит к образованию единственного цикла $U_{цикл} \subset U_Б \cup (i_0, j_0)$. В нашем случае этот цикл обладает спецификой: цикл состоит как бы из двух частей. В одну часть входят дуги, направление которых совпадает с направлением дуги (i_0, j_0), другую часть цикла образуют дуги, имеющие обратное направление (рис. 3.3). Отметим, что «прямые» дуги идут подряд, затем подряд идут «обратные» дуги, т.е. нет чередования прямых и обратных дуг (что могло иметь место в общей задаче о потоке минимальной стоимости).

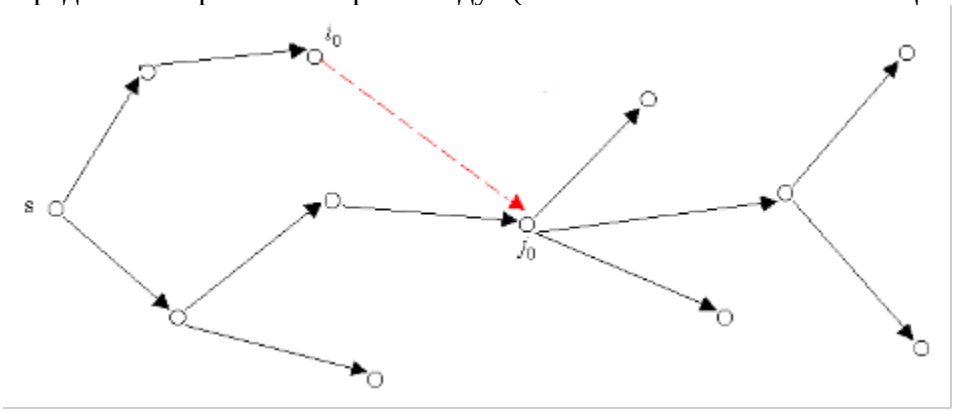


Рис. 3.3

Шаг 3. Далее, согласно методу потенциалов, надо вычислить шаг $\theta_0 = \min x_{ij}, (i,j) \in U^-_{цикл}$, где $U^-_{цикл}$ – обратные дуги цикла. В общей задаче шаг θ_0 мог реализоваться на любой дуге $(i,j) \in U^-_{цикл}$. В силу специфики (2) видно, что в нашем случае шаг θ_0 обязательно

реализуется на обратной дуге $(i_*, j_0) \in U_{\text{цикл}}$, ведущей в узел j_0. Удалим дугу (i_*, j_0) из множества $U_{\text{Б}}$, получим две компоненты связности $\{I_1, U_1\}, \{I_2, U_2\}$, одна из которых содержит узел s, другая – узел j_0, $s \in I_1, j_0 \in I_2, U_1 \cup U_2 = U_{\text{Б}} \setminus (i_*, j_0)$ (рис. 3.4).

Вычислим новые потенциалы $B_j, j \in I,$ по правилу

$$\bar{B}_j = B_j - (B_{j_0} - c_{i_0 j_0} - B_{i_0}), j \in I_2;$$

$$\bar{B}_j = B_j, j \in I_1. \qquad (17)$$

Шаг 4. Построим новое базисное дерево (рис. 3.5)

$$\bar{U}_{\text{Б}} = (U_{\text{Б}} \setminus (i_*, j_0)) \cup (i_0, j_0).$$

Перейдем к шагу 1 с новым множеством $\bar{U}_{\text{Б}}$ и соответствующими ему потенциалами (17).
Приведенный алгоритм решает задачу за конечное число итераций.

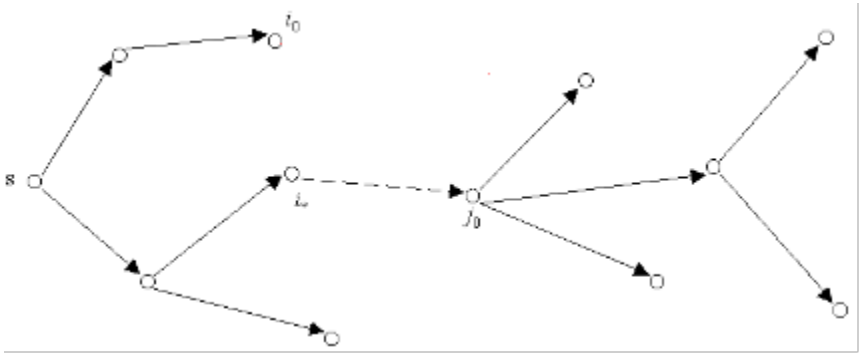


Рис. 3.4

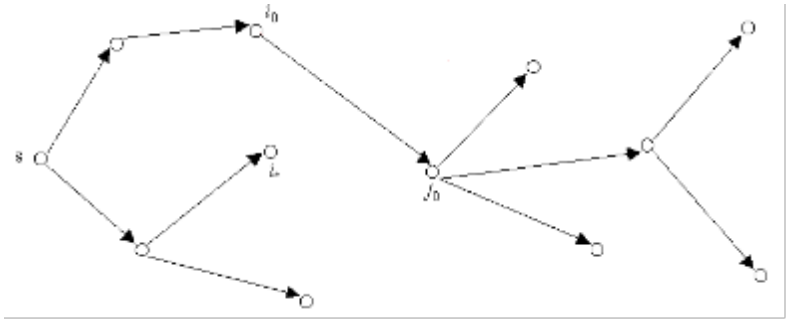


Рис. 3.5

Описанный выше алгоритм, базирующийся на методе потенциалов, можно использовать и для выявления отрицательных контуров в сети. Для этого на шаге 2 надо рассмотреть цикл $U_{цикл} \subset U_A \cup (i_0, j_0)$. Если в этом цикле не все дуги имеют одно направление, то действуем как и раньше. Если в цикле $U_{цикл}$ все дуги имеют одно направление, то STOP – в сети есть отрицательные контуры. В этом случае задача построения дерева кратчайших путей не имеет решения.

29. Кратчайшие пути между всеми парами вершин. Алгоритм Флойда .

На каждой итерации алгоритма строятся две $(n\times n)$-матрицы: *D* и *R*. Матрица *D* называется *матрицей длин кратчайших путей* и содержит текущие оценки длин кратчайших путей, т.е. на *j*-й итерации имеем $D^j=(d_{ik}^j, k=\overline{1,n}, i=\overline{1,n})$.

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

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

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

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

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

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

Шаг 2. Для каждого элемента (i,k), $i\neq j, k\neq j$, матрицы расстояний (начиная с первого и не принадлежащего ни базисной строке, ни базисному столбцу) сравним числа d_{ik}^{j-1} и сумму соответствующих элементов базового столбца и базовой строки $d_{ij}^{j-1}+d_{jk}^{j-1}$. Если $d_{ij}^{j-1}+d_{jk}^{j-1}\geq d_{ik}^{j-1}$, то полагаем $d_{ik}^j=d_{ik}^{j-1}$ и выбираем новые значения для переменных i и *k*. Если $d_{ij}^{j-1}+d_{jk}^{j-1}<d_{ik}^{j-1}$, то полагаем $d_{ik}^j=d_{ij}^{j-1}+d_{jk}^{j-1}$

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

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

т.е. они не меняются.

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

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

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

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

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

Опишем процедуру восстановления кратчайшего пути.

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

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

2. Найдём элемент $r_{i_1 j_0}^n$. Пусть $r_{i_1 j_0}^n = i_2$, следовательно, i_2 – первый промежуточный узел кратчайшего пути из i_1 в j_0; i_2 – второй промежуточный узел кратчайшего пути из i_0 в j_0 и т.д., пока для некоторого i_s ни получим $r_{i_s j_0}^n = j_0$. Таким образом, минимальный путь из i_0 в j_0 последовательно проходит через узлы $i_0, i_1, i_2, \ldots, i_s, j_0$.

30. Задача о максимальном потоке. Теорема Форда-Фалкерсона

общая интерпретация (заканчивается формулой 12)

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

Математическая модель данной задачи имеет вид

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

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

$$0 \leq x_{ij} \leq d_{ij}, \quad (i,j) \in U. \qquad (12)$$

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

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

Алгоритм составим из следующих этапов.

Этап 1. Определим начальный поток следующим образом:

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

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

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

31. Связь задачи о максимальном потоке с задачей о потоке минимальной стоимости.

Общее из 30 вопроса +

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

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

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

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

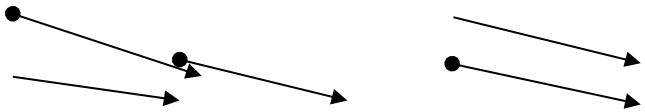
$$0 \le x_{ij} \le d_{ij}, \quad (i, j) \in \overline{U}.$$

(13)

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

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

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



32. Задача о максимальном потоке. Двойственная интерпретация.

Общее из 30 вопроса +

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

$$\sum_{(i,j)\in U} d_{ij}\delta_{ij} \to \min_{u,\delta},$$

$$u_i - u_j + \delta_{ij} \geq 0, (i,j)\in U,$$

$$u_z - u_s \geq 1,\ u_s = 0,\ \delta_{ij} \geq 0, (i,j)\in U.$$

(14)

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

33. Метод пометок для построения максимального потока.

Пометка узла используется для указания как величины потока, так и источника потока, вызывающего изменение текущей величины потока по дуге, т.е. указывается узел, с помощью которого помечается данный узел.

Это из 30 вопроса, но он не нужен, пометки обозначаются через g и p в 34 вопросе

34. Построение увеличивающих путей в сети заданным потоком.

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

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

I_c – счётчик итераций,

I_t – счётчик меток,

L – множество помеченных узлов,

$g(j)$ – метка узла j,

p_j – вторая метка узла j.

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

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

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

Шаг 5. Положим $I_c := I_c + 1$. Найдём помеченный узел j0 с меткой $p_{j_0} = I_c$.

Если такой узел найден, то полагаем i := j0 и возвращаемся к шагу 2. Если такого узла найти не удалось, то не существует увеличивающего пути из s в t. STOP.

Опишем алгоритм восстановления пути и увеличения потока.

По условию узел t помечен. Пусть q(t) = i1, следовательно, из узла i1 попадаем в узел t по прямой дуге (i1, t). Полагаем $\alpha_1 = d_{i_1 t} - x_{i_1 t}$.

Рассмотрим метку узла i1. Предположим, что q(i1) = i2. Полагаем

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

Если q(i1) = –i2 , то полагаем

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

(в последнем случае в увеличивающем пути дуга (i1, i2) является обратной). И так далее, пока ни получим на некотором шаге

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

Таким образом, увеличивающий путь проходит через узлы s, im, im-1, …, i2, i1, t.

Изменяем дуговые потоки на дугах этого пути: дуговые потоки на прямых дугах увеличиваем на α_m, дуговые потоки на обратных дугах уменьшаем на α_m, поток ^{V} увеличиваем на α_m. STOP.

35. Построение максимального потока с помощью метода потенциалов.

31 билет после 13 формулы будет абзац где говорится о том, что задаче 13 надо применить метод потенциалов из 21 вопроса и задача будет решена.

36. Задача о назначениях. Примеры приложений. Свойства редуцированной матрицы стоимости.

КРАТКО

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

Построим математическую модель данной задачи. Определим переменную

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

Тогда математическая модель рассматриваемой задачи имеет вид

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

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

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

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

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

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

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

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

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

$$x_{ij} = 0 \vee 1,\ i=\overline{1,n},\ j=\overline{1,n}. \qquad 23)$$

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

Здесь

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

Можно заменить условия

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

на условия

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

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

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

37. Венгерский метод решения задачи о назначениях (общая схема).

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

$$d_{ij} = c_{ij} + \gamma_i + \delta_j,\ i = \overline{1,n},\ j = \overline{1,n} \quad (26)$$

Из (23), (26) получаем

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

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

const

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

Общая схема метода следующая:

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

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

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

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

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

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

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

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

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

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

строится по правилу $x_{ij}^0=1,\quad$ åñëè $\ (i,j+n)\in U_0\ $ è $\ y_{ij+n}^0=1;\quad x_{ij}^0=0\ -\ $ в противном случае, (29)

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

Алгоритм прекращает свою работу.

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

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

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

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

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

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

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

Найдем число

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

(30)

Из свойств «а» – «в» и правил построения множества U_0 следует, что $\alpha > 0.$

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

$$\overline{c}_{ij} = c_{ij}, \quad \text{если} \quad i \in N^{(1)}, \ j \in N^{(2)} \text{ либо } i \notin N^{(1)}, \ j \notin N^{(2)};$$

$$\overline{c}_{ij} = c_{ij} - \alpha, \ \text{если} \ i \in N^{(1)}, \ j \notin N^{(2)}; \qquad (31)$$

$$\overline{c}_{ij} = c_{ij} + \alpha, \ \text{если} \ i \notin N^{(1)}, \ j \in N^{(2)}.$$

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

38. Обоснование правил пересчета матрицы стоимости в задаче о назначениях.

Общие методы в 37 вопросе

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

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

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

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

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

2) полученная матрица является редуцированной матрицей по отношению к исходной матрице стоимостей C, т.е. она может быть получена из C в результате преобразования $c_{ij} \to d_{ij} = c_{ij} + \gamma_i + \delta_j$, где γ_i, δ_j – некоторые действительные числа;

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

39. Доказательство конечности венгерского метода.
Сам метод в 37 вопросе

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

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

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

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

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

Из 3-го и 4-го свойств следует, что при каждом последующем использовании процедуры шага 2 либо увеличивается множество помеченных узлов I_* , либо величина максимального потока по дугам с нулевой стоимостью увеличивается хотя бы на 1. Поскольку $I_*\subset I,\ |I|=2n+2\ \ \text{и}\ \ v^0\leq n$, то очевидно, что через конечное число повторений шага 2 мы придем к ситуации, когда будет найден максимальный поток (28) с $v^0=n$. Согласно алгоритму, это означает, что исходная задача решена. Оптимальный план назначений строится по правилам (29).

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

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

40. Задача коммивояжера, первая схема метода ветвей и границ (алгоритм разрыва подциклов).

КРАТКО:

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

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

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

Обозначим

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

Математическая модель задачи коммивояжёра совпадает с задачей (7) – (10)

плюс ещё одно дополнительное требование[2]:

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

Дополнительное ограничение (11) является существенным. Решив задачи (7) – (9) (без дополнительного условия (11)), мы можем получить такой оптимальный план x0, для которого множество $U_*(x0)$ состоит из двух и более циклов, что недопустимо в исходной задаче о коммивояжёре. Отметим, что ограничение (11) существенно усложняет решение задачи о коммивояжёре. К настоящему времени разработано много методов решения задачи о коммивояжере. Некоторые из них будут описаны в § 4.

РАСШИРЕННО:

3. Математическая модель задачи коммивояжера

имеет вид

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

$$\sum_{j=1}^{n} x_{ij} = 1,\ i=\overline{1,n}$$

(отъезд из города i), (33)

$$\sum_{i=1}^{n} x_{ij} = 1,\ j=\overline{1,n}$$

(прибытие в город j), (34)

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

2

множество $U_* = \{(i,j): i=\overline{1,n},\ j=\overline{1,n}; x_{ij}=1$ } есть единственный цикл. (36)

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

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

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

САМ ОТВЕТ НА ВОПРОС:

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

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

На итерации t выполняются следующие шаги.

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

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

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

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

41. Задача коммивояжера, вторая схема метода ветвей и границ (алгоритм задания маршрутов)

краткая и расширенная версия в вопросе 40

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

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

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

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

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

На итерации t выполняются следующие шаги.

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

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

Шаг 3. Если зафиксированные переменные $x_{ij}=1$ в выбранной задаче образуют один цикл, то зафиксируем его; положим r_0^{t+1} равным длине полученного цикла; вернёмся к шагу 1. В противном случае (т.е. если дуги с $x_{ij}=1$ не образуют цикла) перейдём к шагу 4.

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

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

2) для которой текущее $c_{i_0j_0}<\infty$;

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

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

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

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

42. Сравнение двух схем метода ветвей и границ для решения задачи коммивояжера.

Они в 40 и 41 вопросе

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

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

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

43. Матричные игры. Постановка задачи. Чистые и смешанные стратегии.

Основное содержание теории игр состоит в изучении следующей проблемы: если n партнеров $P_1, P_2, \ldots, P_n$ играют в данную игру Г, то как должен вести партию i-й игрок для достижения наиболее благоприятного для себя исхода?

Здесь под термином игра понимается совокупность предварительно оговоренных правил и условий игры, а термин партия связан с частной возможной реализацией этих правил. В дальнейшем предполагается, что в конце каждой партии игры каждый игрок P_i получает сумму денег v_i, называемую выигрышем этого игрока, и стремится максимизировать сумму получаемых им денег.

В большинстве салонных игр общая сумма денег, теряемых проигравшими игроками, равна сумме денег, получаемых выигравшими партнерами. В этом случае для каждой партии

$v_1 + v_2 + \ldots + v_n = 0$.

Число v_i может быть любого знака, при этом:

$v_i > 0$ соответствует выигрышу,

$v_i < 0$ соответствует проигрышу,

$v_i = 0$ соответствует нейтральному исходу.

Игры, в которых алгебраическая сумма выигрышей равна нулю, называются играми с нулевой суммой.

Игры также классифицируются по числу игроков и числу возможных ходов. Далее игры можно подразделять на кооперативные и некооперативные. В кооперативных играх партнеры могут образовывать коалиции и играть как команды, тогда как в некооперативных играх каждого игрока интересует лишь его собственный результат. Игры двух партнеров являются, очевидно, некооперативными.

Мы будем рассматривать только игры двух партнеров с нулевой суммой и конечным числом возможных ходов. Такие игры называются прямоугольными или матричными, поскольку они задаются платежной матрицей (или матрицей выигрышей):

$$A = \begin{pmatrix} a_{ij}\ ,\ j = \overline{1,n} \\ i = \overline{1,m} \end{pmatrix}.$$

Первый игрок имеет возможность сделать m выборов (m чистых стратегий), а второй – n выборов (n чистых стратегий). Если первый игрок выбирает

i-ю чистую стратегию, а второй – j-ю, то выигрыш первого (проигрыш второго) равен a_{ij} , сумма выигрышей обоих игроков равна нулю.

Задача теории игр заключается в выборе принципов поведения игроков в каждой конкретной ситуации.

Решение игры – это выбор линии поведения игроков, обеспечивающих состояние равновесия, т.е. состояние, к которому стремился бы каждый разумный игрок, сознавая, что отступление от этой линии может только уменьшить его выигрыш.

Вектор $x=(x_1,x_2,...,x_m)$, каждая компонента которого указывает относительную частоту (вероятность), с которой соответствующая чистая стратегия используется в игре, называется смешанной стратегией первого игрока.

Вектор $y=(y_1,y_2,...,y_n)$ – смешанная стратегия второго игрока. Ясно, что

$x_i \geq 0,\ \ i=1,m$; $y_j \geq 0$, $j=1,n$; $\sum_{i=1}^{m} x_i = 1$, $\sum_{j=1}^{n} y_j = 1$.

Чистая стратегия может быть определена как смешанная стратегия, в которой все компоненты, кроме одной, равны нулю.

В дальнейшем будем обозначать чистые стратегии обоих игроков в виде единичных векторов

Оптимальная стратегия игрока – это стратегия, обеспечивающая ему максимально возможный гарантированный средний выигрыш. (При этом предупреждается, что игра ведется без обмана и подглядывания).

Рассмотрим матричную игру, определяемую матрицей выигрышей:

$$A=\begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{m1} & a_{m2} & ... & a_{mn} \end{pmatrix}.$$

Если первый партнер P_1 выбирает любую чистую стратегию i , то он уверен, что выиграет по крайней мере $\min_j a_{ij}$.

Поскольку игрок P_1 может выбирать любые i, то естественно предполагать, что он выберет ту стратегию i, при которой его гарантированный выигрыш максимальный. Следовательно, при использовании чистой стратегии игрок P_1 может выиграть не менее

$\max_i \min_j a_{ij} =: g_1$.

Число g_1 – гарантированный выигрыш игрока P_1 при использовании только чистых стратегий. В нашем примере

$g_1 = 1 = a_{22} = 1$.

Если второй игрок P_2 выбирает стратегию j, то наихудшее, что с ним может случиться, – это проигрыш в размере $\max_i a_{ij}$. Игрок P_2 может выбирать ту чистую стратегию j, при которой его проигрыш минимален. При выборе этой стратегии игрок P_2 гарантирует, что игрок P_1 не сможет выиграть больше чем

$g_2 := \min_j \max_i a_{ij}$.

Для нашей матрицы $g_2 := \min_j \max_i a_{ij} = 3$.

Отметим, что всегда

$$\min_{y \in Y} \max_{x \in X} f(x,y) \geq \max_{x \in X} \min_{y \in Y} f(x,y), \text{ следовательно, } g_2 \geq g_1.$$

Таким образом, в нашем примере при выборе только чистых стратегий игрок P_1 может гарантировать, что он выиграет не менее 1 (а хотел бы выиграть с гарантией больше!); игрок P_2 может гарантировать, что он не проиграет более 3 (а хотел бы проиграть с гарантией меньше!).

Если бы имело место равенство $\max_i \min_j a_{ij} = \min_j \max_i a_{ij} = v$, (1)

то P_1 может быть уверен, что выиграет не менее v, а игрок P_2 может добиться того, что гарантированно не проиграет больше чем v.

Матричная игра, для которой имеет место равенство (1), наилучшим образом разыгрывается партнерами P_1 и P_2, избирающими соответствующие чистые стратегии. Поскольку при любом отклонении игрока P_1 от этой стратегии его гарантированный выигрыш не увеличивается (а возможно, и уменьшится) и поскольку при каждом отклонении партнера P_2 от своей чистой стратегии он не уменьшит своего проигрыша (а возможно, увеличит его), указанные чистые стратегии естественно назвать оптимальными чистыми стратегиями.

Следующая матричная игра является одной из тех игр, которые имеют оптимальные чистые стратегии:

$$A=\begin{pmatrix}3 & 5 & 6\\ 1 & 2 & 3\\ 0 & 7 & 4\end{pmatrix}, \quad \max_i \min_j a_{ij} = \min_j \max_i a_{ij} = 3 .$$

Поскольку не все матричные игры могут оптимально разыгрываться при помощи чистых стратегий, необходимо ввести понятие оптимальной смешанной стратегии.

Если игрок P_1 при длительном процессе игры выбирает смешанную стратегию $x=(x_1, x_2, \ldots, x_m)$, т.е. с вероятностью x_i выбирает i-ю чистую стратегию, а игрок P_2 при длительном процессе игры выбирает смешанную стратегию $y=(y_1, y_2, \ldots, y_n)$, т.е. с вероятностью y_j выбирает j-ю стратегию, то математическое ожидание (средний выигрыш) выигрыша игрока P_1 равно

$$M(x,y)=\sum_{j=1}^{n}\sum_{i=1}^{m} a_{ij} x_i y_j = x' A y . \qquad (2)$$

Соответственно выигрыш игрока P_2 при этом равен $-M(x,y)$, т.е. его проигрыш равен $M(x,y)$.

Функция $M(x,y)$ называется функцией платежей.

Говорят, что игра имеет решение в смешанных стратегиях, если существуют такие стратегии x^*, y^* и число v, что при любых смешанных стратегиях x, y выполняется соотношение

$$M(x, y^*) \le v \le M(x^*, y) . \qquad (3)$$

Полагая в (3) $x=x^*, y=y^*$, получим

$$v = M(x^*, y^*) . \qquad (4)$$

Число v называется ценой игры.

Неравенство (3) означает, что если игрок P_1 будет использовать смешанную стратегию x^* (при длительном процессе игры), то его гарантированный выигрыш равен v, так как при любом y имеем $v \le M(x^*, y)$.

Для игрока P_2 соотношение (3) означает, что если игрок P_2 будет придерживаться стратегии y^*, то не проиграет больше чем v, так как при каждом выборе смешанной стратегии x игроком P_1 имеем $M(x, y^*) \le v$.

Стратегии x^*, y^*, удовлетворяющие (3), называются оптимальными стратегиями.

44. Матричные игры. Сведение к задаче линейного программирования.

Сформулируем основную теорему теории игр.

Теорема 1. Каждая матричная игра с нулевой суммой имеет решение в смешанных стратегиях, т.е. существуют такие смешанные стратегии x^0 и y^0 первого и второго игроков соответственно, что при любых смешанных стратегиях x, y имеют место неравенства и равенства:

$M(x,y^0) \le M(x^0,y^0) \le M(x^0,y)$,

$\max_{x\in X} \min_{y\in Y} M(x,y) = \min_{y\in Y} \max_{x\in X} M(x,y) = M(x^0,y^0)$,

где

$X = \left\{ x \in R^m : \sum_{i=1}^{m} x_i = 1, x_i \ge 0, i = \overline{1,m} \right\}$, $Y = \left\{ y \in R^n : \sum_{j=1}^{n} y_j = 1, y_j \ge 0, j = \overline{1,n} \right\}$,

множества смешанных стратегий игроков соответственно P_1 и P_2.

Пусть задана матрица $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$.

Согласно определению, задача игрока P_1 заключается в том, чтобы найти такое максимальное число v и вектор $x^0 = (x_1^0, x_2^0, \dots, x_m^0) \in X$, что

$M(x^0,y) \ge v$ для $\forall y \in Y$. (5)

Рассмотрим ограничения (5) подробнее:

$x^{0\prime} Ay \ge v, \forall y \in Y$. (6)

Покажем, что неравенства (6) эквивалентны условиям

$x^{0\prime} Ae_j \ge v, \; j = \overline{1,n}$. (7)

Отметим, что в (6) мы имеем континуум ограничений, так как неравенства должны выполняться для $\forall y \in Y$; в (7) мы имеем только n ограничений. Действительно, очевидно, что из (6) следует (7), так как $e_j \in Y, j = \overline{1,n}$. Покажем, что из (7) следует (6).

Пусть (7) имеет место. Рассмотрим $\forall y = (y_1, y_2, ..., y_n) \in Y$. Правую и левую части каждого j-го неравенства (7) умножим на $y_j \geq 0$ и просуммируем все неравенства $\sum_{j=1}^{n} x^{0\prime} A y_j e_j \geq \sum_{j=1}^{n} y_j v$. (8)

Учитывая, что $\sum_{j=1}^{n} y_j = 1$ и $y = \sum_{j=1}^{n} y_j e_j$, из (8) получаем (6). Эквивалентность (6) и (7) доказана.

Таким образом, мы пришли к тому, что задача игрока P_1 состоит в поиске таких чисел v и вектора x^0 , которые являются решением следующей задачи:

$$v \to \max_{x,v} \qquad \sum_{i=1}^{m} a_{ij} x_i \geq v, \ j = \overline{1,n};$$

$$\sum_{j=1}^{n} x_i = 1, \ x_i \geq 0, i = \overline{1,m}. \tag{9}$$

Задача (9) является задачей линейного программирования.

Аналогично, рассуждая за второго игрока P_2 , мы приходим к тому, что его задача состоит в нахождении такого минимального числа v и вектора $y^0 \in Y$, при которых

$M(x, y^0) \leq v, \forall x \in X$. (10)

Можно показать, что (10) эквивалентно условиям $e_j^{\prime} A y^0 \leq 0, i = \overline{1,m}$. Следовательно, задача игрока P_2 состоит в нахождении таких числа v и вектора y^0 , которые являются решением следующей задачи:

$$v \to \min_{v,y};$$

$$\sum_{j=1}^{n} a_{ij} y_j \leq v, i = \overline{1,m};$$

$$\sum_{j=1}^{n} y_j = 1, \ y_j \geq 0, \ j = \overline{1,n}. \tag{11}$$

Задача (11) является задачей линейного программирования.

Легко проверить, что задачи (9) и (11) составляют пару двойственных задач! Следовательно, не нужно решать каждую из этих задач отдельно. Из теории двойственности следует, что достаточно решить, например, симплекс-методом одну из этих задач и по оптимальному

плану решенной задачи легко восстановить оптимальный план второй задачи.

Таким образом, мы показали, что верна следующая теорема.

Теорема 2. Каждая матричная игра с платежной матрицей $A=\begin{pmatrix} a_{ij}\ ,\ j=\overline{1,n} \\ i=\overline{1,m} \end{pmatrix}$ эквивалентна паре двойственных задач (9) и (11).

Рассмотрим теперь обратную задачу. Попытаемся представить данную задачу линейного программирования в форме матричной игры. Каждой паре двойственных задач линейного программирования можно поставить в соответствие матричную игру, цена и оптимальные стратегии которой позволяют вычислить оптимальные прямой и двойственный планы (если они существуют!).

Подчеркнем, что в то время как матричные игры всегда имеют оптимальные стратегии (т.е. имеют решение), задачи линейного программирования могут и не иметь решений.

Рассмотрим пару двойственных задач:

$$c'x \to max,\quad Ax \le b, x \ge 0, \qquad (12)$$

$$b'y \to min,\quad A'y \ge c, y \ge 0. \qquad (13)$$

Здесь $A \in R^{m\times n}$. Построим игру с платежной матрицей

$$\Pi = \begin{pmatrix} 0 & A & -b \\ -A' & 0 & c \\ b' & -c' & 0 \end{pmatrix} \in R^{(n+m+1)\times(n+m+1)} . \qquad (14)$$

Отметим, что матрица П является кососимметричной, следовательно, если рассматривать матричную игру с платежной матрицей П, то стратегии (оптимальные) первого и второго игроков должны совпадать! Справедлива

Теорема 3. Пара двойственных задач (12) и (13) имеет решение тогда и только тогда, когда игра с платежной матрицей (14) имеет такую оптимальную стратегию

$u^* = (u_1^*, u_2^*, \ldots \mu_{n+m}^*, u_{n+m+1}^*)$,

что $u_{n+m+1}^* > 0$. При этом

$$y_i^0 = \frac{u_i^*}{u_{n+m+1}^*},\ i = \overline{1,m}; \qquad x_j^0 = \frac{u_{m+j}^*}{u_{n+m+1}^*},\ j = \overline{1,n}$$

.

Из теоремы 3 следует, что решение пары двойственных задач линейного программирования может быть сведено к вычислению оптимальных стратегий (которые совпадают!) симметричной игры с платежной матрицей П (14).