МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ОДЕССКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ И. И. МЕЧНИКОВА
ИНСТИТУТ МАТЕМАТИКИ, ЭКОНОМИКИ И МЕХАНИКИ
Кафедра оптимального управления и экономической кибернетики

Г. А. Ефимова, Е. М. Страхов

ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ВАРИАНТЫ ЗАДАНИЙ
ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ
НАПРАВЛЕНИЙ ПОДГОТОВКИ 6.040201 «МАТЕМАТИКА»,
6.040301 «ПРИКЛАДНАЯ МАТЕМАТИКА»

Одесса
2014

УДК 519.854.2, 519.854.3

ББК 22.18

Методические указания и варианты заданий для самостоятельной работы студентов направлений подготовки 6.040201 «математика», 6.040301 «прикладная математика».

Составители: **Ефимова Г. А.,** кандидат физико-математических наук,
доцент кафедры оптимального управления
и экономической кибернетики;
Страхов Е. М., кандидат физико-математических наук,
старший преподаватель кафедры оптимального управления
и экономической кибернетики.

Рецензенты: **Молчанюк И. В.,** кандидат физико-математических наук,
доцент кафедры прикладной, вычислительной математики и САПР
Одесской государственной академии строительства и
архитектуры.
Кичмаренко О. Д., кандидат физико-математических наук,
доцент кафедры оптимального управления
и экономической кибернетики
Одесского национального университета имени И. И. Мечникова.

Рекомендовано к печати Ученым советом ИМЭМ ОНУ имени И. И. Мечникова
(протокол № 4 от 19.03.2014).

ВВЕДЕНИЕ

Существует довольно широкий класс экономических задач, математические модели которых содержат переменные, принимающие только целочисленные значения. Например, некоторые виды ресурсов производства могут измеряться только целыми числами: люди, машины, станки, животные и т. д. Если требование целочисленности накладывается лишь на часть переменных, то задачу называют **частично целочисленной**. Целочисленные задачи являются разделом более широкого класса **дискретных** задач, в которых переменные принимают значения из некоторого дискретного множества. К задачам дискретного программирования относятся, например, задача планирования перевозки груза в контейнерах, объемы которых принимают дискретные значения, задачи планирования производства при условиях, что используемые агрегаты имеют определенные дискретные мощности (например, мощности электродвигателей). Задачи целочисленного программирования включают также и задачи булевого программирования, в которых переменные принимают только два значения: 0 или 1 (являются булевыми переменными).

На первый взгляд казалось, что решение задач целочисленного программирования можно найти, округляя решение задачи, полученной из исходной, отказавшись от требования целочисленности переменных. Однако можно привести примеры, когда построенное таким образом решение будет далеким от оптимального (особенно часто это происходит в задачах булевого программирования) или даже недопустимым.

Для нахождения оптимальных планов задач целочисленного программирования используют три основных группы методов:

- методы отсечений;
- комбинаторные методы;
- приближенные методы.

Методы отсечений, основным из которых является метод Гомори, реализуются путем построения на этапах алгоритма дополнительных ограничений-неравенств, которые и определяют отсекающие гиперплоскости. Дополнительные ограничения должны удовлетворять двум необходимым условиям.

Условие правильности заключается в том, что дополнительному ограничению должны удовлетворять все планы исходной целочисленной задачи.

Условие отсечения заключается в том, что этому ограничению не должен удовлетворять оптимальный план соответствующей нецелочисленной задачи.

Разные алгоритмы методов отсечений отличаются способами построения дополнительных ограничений.

Комбинаторные методы базируются на переборе всех допустимых целочисленных планов, реализуя процедуру целенаправленного поиска оптимума на дискретном множестве планов задачи. На каждом шаге этой процедуры из рассмотрения исключается определенное количество неоптимальных планов.

Наиболее распространенным из этой группы методов является метод ветвей и границ. Его суть заключается в том, что множество планов разбивается на ряд подмножеств. Для каждого из подмножеств находят оценку целевой функции по оптимуму, что представляет собой самостоятельную, более простую, чем исходная, задачу. На основе этих оценок выбирают «перспективные» подмножества и повторяют процедуру их деления на новые подмножества.

Комбинаторные методы отличаются способом деления совокупности планов на подмножества и способом построения оценок целевой функции. Среди них можно отметить метод последовательного анализа вариантов, метод вектора спуска и метод Беллмана, который используется в динамическом программировании.

Приближенные методы дискретного программирования делятся на два вида: методы, которые используют случайный поиск (итерационные методы Ю. П. Зайченко [4], А. А. Корбут, Ю. Ю. Финкельштейна [5], И. В. Сергиенко [9] и др.) и детерминированные методы, использующие эвристические приемы.

МЕТОДЫ ОТСЕЧЕНИЙ

Первый алгоритм Гомори

Первый алгоритм Гомори используется для решения полностью целочисленных линейных задач [4, 5].

Рассмотрим задачу целочисленного линейного программирования (ЗЦЛП):

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

$$\sum_{j=1}^{n} a_{ij} x_j = b_i,\ i = \overline{1,m},\quad x_j \ge 0,\ j = \overline{1,n}, \tag{2}$$

где x_j – целые числа, $j = \overline{1,n}$.

В соответствии с общей идеей методов отсечения, сначала решаем задачу (1)-(2) без учета целочисленности переменных. Если она неразрешима, то неразрешима и задача целочисленного линейного программирования.

Пусть $x = (x_1, x_2, ..., x_n)$ – оптимальный план задачи (1)-(2). Если компоненты вектора x являются целыми числами, то этот план является и решением ЗЦЛП. В противном случае необходимо построить первое отсечение, которое удовлетворяет условиям правильности и отсечения (см. Введение). Для этого напомним некоторые понятия.

Определение. **Целой частью** $[a]$ числа a называется ближайшее целое число, не превосходящее данное число a. **Дробной частью** $\{a\}$ числа a называется разность между самим числом и его целой частью,

$$\{a\} = a - [a].$$

Для любого числа a дробная часть $\{a\} \in [0, 1)$, для целого числа $\{a\} = 0$.

Например, для числа a = 7/4 целая часть [7/4] = 1, дробная часть {7/4} = 3/4, для числа a = -7/4 целая часть [-7/4] = -2, дробная часть {-7/4} = 1/4.

Без ограничения общности будем обозначать элементы оптимальной симплекс-таблицы ЗЛП также через a_{ij}, а элементы столбца b – b_i.

Пусть в оптимальном плане задачи (1)-(2) значение переменной x_i не является целым. Построим отсечение для этой переменной. Условия правильности и отсечения для нее выражаются в виде неравенства

$$\sum_{j=1}^{n} \{a_{ij}\} x_j \ge \{b_i\},$$

которое записывается в виде равносильного ограничения-равенства

$$-\sum_{j=1}^{n}\{a_{ij}\}x_j + s_1 = -\{b_i\} \qquad (3)$$

и включается в систему ограничений задачи (1)-(2). Здесь s_1 – неотрицательная базисная дополнительная переменная.

Полученную расширенную задачу решают двойственным симплекс-методом. При необходимости строят новое отсечение, вводя переменную s_2, и т. д.

Пример. $z = x_1 + 2x_2 \to \max$

при ограничениях

$$2x_1 + 4x_2 \le 7,$$
$$4x_1 - 5x_2 \le 10,$$

$$x_1,\ x_2 \ge 0,\ \ x_1,\ x_2 - \textit{целые числа}\ .$$

Сведем задачу к каноническому виду, вводя дополнительные переменные $x_3,\ x_4$:

$$z = x_1 + 2x_2 \to \max$$

$$2x_1 + 4x_2 + x_3 \qquad = 7,$$
$$4x_1 - 5x_2 \qquad + x_4 = 10,$$

$$x_1,\ x_2,\ x_3,\ x_4 \ge 0\,.$$

Оптимальная симплекс-таблица для этой ЗЛП будет иметь вид:

Таблица 1

$c_б$	$x_б$	c	1	2	0	0
		b	x_1	x_2	x_3	x_4
2	x_2	$\frac{7}{4}$	$\frac{1}{2}$	1	$\frac{1}{4}$	0
0	x_4	$\frac{75}{4}$	$\frac{13}{2}$	0	$\frac{5}{4}$	1
	Δ	$3\frac{1}{2}$	0	0	$\frac{1}{2}$	0

Таким образом, для исходной ЗЛП без учета целочисленности переменных оптимальный план имеет вид: $x_1^* = 0,\ x_2^* = 1\frac{3}{4}$. Однако этот план не удовлетворяет условию целочисленности переменных x_1, x_2, поэтому решение задачи нужно продолжить, расширяя ее ограничения дополнительным ограничением, соответствующим отсечению. По первой строке симплекс-таблицы, соответствующей нецелочисленной переменной x_2, восстановим уравнение, соответствующее этой переменной:

$$\frac{1}{2}x_1 + x_2 + \frac{1}{4}x_3 = 1\frac{3}{4}.$$

Уравнение (3) в данном случае будет иметь вид:

$$-\frac{1}{2}x_1 - \frac{1}{4}x_3 + s_1 = -\frac{3}{4}.$$

Для продолжения решения задачи в симплекс-таблицу дописываем столбец и строку с базисной переменной s_1 и делаем итерации двойственным симплекс-методом, т. к. в столбце b в строке, соответствующей s_1, стоит отрицательный элемент:

Таблица 2

$c_б$	$x_б$	с	1	2	0	0	0
		в	x_1	x_2	x_3	x_4	S_1
2	x_2	$\frac{7}{4}$	$\frac{1}{2}$	1	$\frac{1}{4}$	0	0
0	x_4	$\frac{7}{4}$	$\frac{13}{2}$	0	$\frac{5}{4}$	1	0
0	S_1	$-\frac{3}{4}$	$-\frac{1}{2}$	0	$-\frac{1}{4}$	0	1
	Δ	$3\frac{1}{2}$	0	0	$\frac{1}{2}$	0	0

Последняя симплекс-таблица расширенной задачи имеет вид:

Таблица 3

$c_б$	$x_б$	с	1	2	0	0	0
		в	x_1	x_2	x_3	x_4	S_1
2	x_2	1	0	1	0	0	1
0	x_4	9	0	0	−2	1	13
1	x_1	$1\frac{1}{2}$	1	0	$\frac{1}{2}$	0	-2
	Δ	$3\frac{1}{2}$	0	0	$\frac{1}{2}$	0	0

Оптимальный план расширенной задачи: $x_1^* = 1\frac{1}{2}$, $x_2^* = 1$. Теперь нарушено условие целочисленности переменной x_1, поэтому строим отсечение по третьей строке таблицы 3, соответствующей этой переменной:

$$-\frac{1}{2}x_3 + s_2 = -\frac{1}{2}.$$

Расширенная симплекс-таблица:

Таблица 4

$c_б$	$x_б$	c	1	2	0	0	0	0
		в	x_1	x_2	x_3	x_4	S_1	S_2
2	x_2	1	0	1	0	0	1	0
0	x_4	9	0	0	−2	1	13	0
1	x_1	$1\frac{1}{2}$	1	0	$\frac{1}{2}$	0	-2	0
0	S_2	$-\frac{1}{2}$	0	0	$-\frac{1}{2}$	0	0	1
	Δ	$3\frac{1}{2}$	0	0	$\frac{1}{2}$	0	0	0

Решая задачу двойственным симплекс-методом, получим таблицу 5:

Таблица 5

$c_б$	$x_б$	c	1	2	0	0	0	0
		в	x_1	x_2	x_3	x_4	S_1	S_2
2	x_2	1	0	1	0	0	1	0
0	x_4	11	0	0	0	1	13	4
1	x_1	1	1	0	0	0	-2	1
0	x_3	1	0	0	1	0	0	-2
	Δ	3	0	0	0	0	0	1

Таким образом, $x_1^* = 1, x_2^* = 1$. Условие целочисленности выполняется, и поэтому решением исходной ЗЦЛП является вектор $x^* = (1, 1)$ при максимальном значении целевой функции $z^* = 3$.

Замечание 1. Первый алгоритм Гомори называют **дробным алгоритмом.** Это связано с тем, что все ненулевые коэффициенты нового ограничения типа (3) являются правильными дробями.

Замечание 2. Если в оптимальном плане ЗЛП несколько нецелочисленных переменных, то отсечение обычно строится для той переменной, у которой дробная часть наибольшая.

Второй алгоритм Гомори

Второй алгоритм Гомори используется для решения **частично** целочисленных линейных задач [4, 5] и отличается от первого лишь способом определения коэффициентов дополнительного ограничения.

Пусть x_k – **целочисленная** переменная в ЗЛП, а остальные переменные вещественные; в оптимальной симплекс-таблице для ЗЛП без учета целочисленности переменной x_k эта переменная является базисной, нецелочисленной и все элементы таблицы – a_{ij}, b_i. Введем обозначения: $J = \{1,...,n\}$, J_H – совокупность индексов **небазисных** переменных в плане x, $J_H^+ = \{ j \in J_H : a_{kj} \geq 0 \}$, $J_H^- = \{ j \in J_H : a_{kj} < 0 \}$. Тогда разложение базисной переменной x_k по небазисным будет иметь вид:

$$x_k = b_k - \sum_{j \in J_H} a_{kj} x_j ,$$

где b_k – значение базисной переменной x_k, а дополнительное ограничение будет следующим:

$$-\left(\sum_{j \in J_H^+} a_{kj} x_j + \frac{\{b_k\}}{\{b_k\}-1} \sum_{j \in J_H^-} a_{kj} x_j \right) + s_1 = -\{b_k\},$$

где s_1 – неотрицательная базисная дополнительная переменная.

Последнее уравнение есть необходимое условие целочисленности переменной x_k (но не достаточное!). Дальнейшие преобразования симплекс-таблицы проводятся с помощью двойственного симплекс-метода.

Пример. $z = x_1 + 2x_2 \to \max$

при ограничениях

$$2x_1 + 4x_2 \leq 7,$$
$$4x_1 - 5x_2 \leq 9,$$

$$x_1,\ x_2 \ge 0,\ \ x_2 - \textit{целое число}.$$

Сведем задачу к каноническому виду, вводя дополнительные переменные $x_3,\ x_4$:

$$z = x_1 + 2x_2 \to \max$$

$$\begin{aligned} 2x_1 + 4x_2 + x_3 \quad &= 7, \\ 4x_1 - 5x_2 \quad + x_4 &= 9, \end{aligned}$$

$$x_1,\ x_2,\ x_3,\ x_4 \ge 0.$$

Оптимальная симплекс-таблица для этой ЗЛП будет иметь вид:

Таблица 6

$c_б$	$x_б$	с	1	2	0	0
		в	x_1	x_2	x_3	x_4
2	x_2	$\frac{7}{2}$	1	1	$\frac{1}{2}$	0
0	x_4	$\frac{53}{2}$	9	0	$\frac{5}{2}$	1
	Δ	7	1	0	1	0

Таким образом, оптимальный план имеет вид: $x_1^* = 0,\ x_2^* = 3\frac{1}{2}$. Однако этот план не удовлетворяет условию целочисленности переменной x_2, поэтому решение задачи нужно продолжить, расширяя ее ограничения дополнительным ограничением, соответствующим отсечению. Здесь $J_H = \{1,3\}$, $J_H^+ = \{1,3\}$, $J_H^- = \varnothing$. Поэтому дополнительное ограничение имеет вид:

$$s_1 - x_1 - \frac{1}{2}x_3 = -\frac{1}{2}.$$

В таблице 6 увеличивается количество строк и столбцов:

Таблица 7

$c_б$	$x_б$	с	1	2	0	0	0
		в	x_1	x_2	x_3	x_4	S_1
2	x_2	$\frac{7}{2}$	1	1	$\frac{1}{2}$	0	0
0	x_4	$\frac{53}{2}$	9	0	$\frac{5}{2}$	1	0
0	S_1	-1/2	-1	0	-1/2	0	1
	Δ	7	1	0	1	0	0

Решая задачу двойственным симплекс-методом, получим таблицу 8.

Таблица 8

$c_б$	$x_б$	c	1	2	0	0	0
		в	x_1	x_2	x_3	x_4	S_1
2	x_2	3	0	1	0	0	1
0	x_4	21	0	0	−2	1	9
1	x_1	1/2	1	0	1/2	0	-1
	Δ	$6\frac{1}{2}$	0	0	1/2	0	1

Теперь $x_1^* = \frac{1}{2}, x_2^* = 3$, что удовлетворяет всем условиям задачи, максимальное значение целевой функции $z^* = 6\frac{1}{2}$.

Замечание. Если среди переменных целочисленными должны быть несколько, например, первые $n_1, n_1 \le n$, то ограничение (3) для переменной x_i примет вид

$$-\sum_{j \in J_H} \beta_{ij} x_j + s_1 = -\{b_i\},\ s_1 \ge 0.$$

Коэффициенты β_{ij} вычисляются по формулам

$$\beta_{ij} = \begin{cases} a_{ij}, & \text{если } j > n_1,\ a_{ij} \ge 0, \\ -\dfrac{\{b_i\}}{1-\{b_i\}}, & \text{если } j > n_1,\ a_{ij} < 0, \\ \{a_{ij}\}, & \text{если } j \le n_1,\ \{a_{ij}\} \le \{b_i\}, \\ \dfrac{\{b_i\}}{1-\{b_i\}}\left(1-\{a_{ij}\}\right), & \text{если } j \le n_1,\ \{a_{ij}\} > \{b_i\}. \end{cases}$$

МЕТОД ВЕТВЕЙ И ГРАНИЦ

Впервые метод ветвей и границ (МВГр) был предложен А. Лэндом и Дж. Дойгом в 1960 г. Он является наиболее широко применяемым комбинаторным методом [1, 2, 3, 7, 8]. Его можно использовать для решения как полностью, так и частично целочисленных задач. Рассмотрим полностью целочисленную задачу. Согласно общей идее метода сначала, так же, как и в методе Гомори, решают задачу линейного программирования с ослабленными ограничениями, без учета требования целочисленности. Пусть x^* – ее оптимальный план, в котором $x_k^* = b_k$ – не целое число. Так как интервал $\left(\left[x_k^*\right], \left[x_k^*\right]+1\right)$ не содержит целых значений x_k, то любое допустимое целое значение этой переменной удовлетворяет одному из неравенств:

$$\text{а)}\ x_k \le \left[x_k^*\right] \text{ или б)}\ x_k \ge \left[x_k^*\right]+1.$$

Введение этих ограничений в ЗЛП порождает две новые задачи: ЛП1 и ЛП2, допустимые множества которых не пересекаются (говорят, что исходная задача разветвляется на две новые подзадачи, а саму процедуру ее замены совокупностью двух задач, эквивалентных ей в смысле оптимального решения, называют **ветвлением,** переменную x_k – **переменной разветвления**). Оптимальный план целочисленной задачи находится в допустимом множестве либо ЛП1, либо ЛП2. Проводимый в процессе ветвления учет требования целочисленности позволяет исключить из рассмотрения часть множества допустимых решений.

Если один из найденных оптимальных планов подзадач удовлетворяет требованию целочисленности, это решение фиксируют как наилучшее. Для дальнейшего разветвления выбирают подзадачу с наибольшим значением целевой функции (при ее максимизации) и производят новое ветвление. Как только получают оптимальное решение ЗЛП с ослабленными ограничениями, удовлетворяющее требованию целочисленности, его сопоставляют с уже имеющимися (если таковые есть) и фиксируют наилучшее из них (в смысле оптимального значения целевой функции). Процесс ветвления продолжают до тех пор, пока каждая порожденная подзадача не приведет к оптимальному решению, удовлетворяющему требованию целочисленности, или пока не будет установлена невозможность улучшения уже зафиксированного наилучшего решения.

Замечание. Если некоторая переменная является непрерывной, то ее не выбирают в качестве переменной разветвления.

Для повышения эффективности рассмотренной процедуры вводят понятие границы, на основе которого можно судить о необходимости дальнейшего ветвления каждой из подзадач, порожденных исходной ЗЦЛП. Сначала задаем нижнюю границу оптимального значения целевой функции исходной задачи равной $-\infty$ (в случае минимизации она равна $+\infty$).

Подзадача определяет новую границу для значения целевой функции, если значения дискретных переменных являются целочисленными и значение целевой функции улучшено по сравнению с текущей границей.

Рассмотрим использование метода ветвей и границ на примере.

Пример. Максимизировать

$$z = 350x_1 + 150x_2$$

при ограничениях

$$25x_1 + 10x_2 \le 100,$$
$$40x_1 + 20x_2 \le 190,$$
$$x_1, x_2 \ge 0,\ x_1, x_2 - \textit{целые числа}.$$

Отбросив условие целочисленности, решим ЗЛП симплекс-методом. Получим решение $x_1^* = 1,\ x_2^* = 7\frac{1}{2},\ z^* = 1475$. Так как переменная x_2^* нецелочисленная, то выберем ее в качестве переменной ветвления. Допустимое целое значение переменной x_2 должно удовлетворять одному из неравенств

$$x_2 \le \left[7\frac{1}{2}\right] = 7 \text{ или } x_2 \ge \left[7\frac{1}{2}\right] + 1 = 8.$$

Далее добавляем к последней симплекс-таблице одно из этих ограничений и получаем задачи ЛП1 (с ограничением $x_2 \le 7$) и ЛП2 (с ограничением $x_2 \ge 8$). Для задачи ЛП1 оптимальным будет решение $x_1 = 1.2,\ x_2 = 7,\ z = 1470$, для задачи ЛП2: $x_1 = 0.75,\ x_2 = 8,\ z = 1462.5$.

Так как целочисленный план не найден, то процесс необходимо продолжить, взяв для следующего разветвления задачу ЛП1, оптимальный план которой дает большее значение целевой функции. Переменной разветвления будет x_1. Далее решаем задачи ЛП3 и ЛП4, добавляя ограничения

$$x_1 \le 1, \quad x_1 \ge 2$$

соответственно.

Задача ЛП3 решения не имеет, т. к. ее допустимое множество пусто. Решение задачи ЛП4 имеет вид: $x_1 = 2,\ x_2 = 5,\ z = 1450$. Это значение целевой функции меньше, чем значение целевой функции при решении задачи ЛП2, поэтому возвращаемся к ветви, соответствующей ЛП2, и проводим ее ветвление по переменной x_1, добавляя ограничения

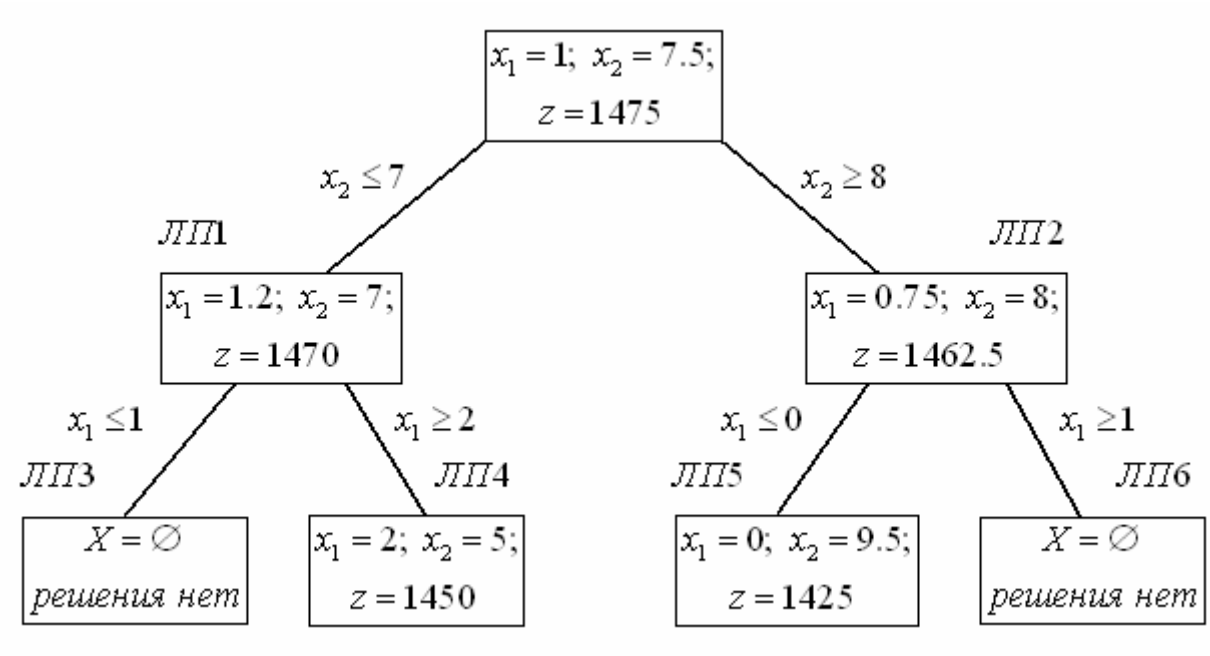
$$x_1 \le 0 \text{ и } x_1 \ge 1.$$

Первое неравенство с учетом требования неотрицательности переменных дает ограничение $x_1 = 0$, поэтому в ЛП5 дополнительное ограничение имеет вид $x_1 = 0$. В ЛП6 дополнительное ограничение будет в виде $x_1 \ge 1$. Допустимое множество задачи ЛП6 пусто. А решением задачи ЛП5 будет вектор с компонентами $x_1 = 0,\ x_2 = \frac{19}{2},\ z = 1425$.

Значение целевой функции для задачи ЛП5 меньше, чем в задаче ЛП4, поэтому эта ветка неперспективна (при ее ветвлении значения целевых функций не превысят значения 1425). Следовательно, оптимальным решением исходной задачи будет решение задачи ЛП4:

$$x_1^* = 2,\ x_2^* = 5,\ z^* = 1450.$$

Решение задачи проиллюстрируем следующей схемой:



ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Найдите оптимальный план задачи целочисленного линейного программирования (N – порядковый номер студента в списке группы), используя

- первый алгоритм Гомори;
- второй алгоритм Гомори (x_1 – произвольное, x_2 – целое);
- метод ветвей и границ (решение проиллюстрируйте схемой).

Вариант 1

$$z = 3x_1 + 4x_2 \to \min$$

при ограничениях

$$5x_1 + 2x_2 \geq 5 + N,$$
$$2x_1 + 5x_2 \geq 7 + N,$$
$$x_1, x_2 \geq 0, \ \ x_1, x_2 - \textit{целые.}$$

Вариант 2

$$z = 3x_1 + 2x_2 \to \max$$

при ограничениях

$$2x_1 + 5x_2 \leq 9 + N,$$
$$5x_1 + 2x_2 \leq 11 + N,$$
$$x_1, x_2 \geq 0, \ \ x_1, x_2 - \textit{целые.}$$

ЗАДАЧА О НАЗНАЧЕНИЯХ. ВЕНГЕРСКИЙ МЕТОД [1, 3, 8, 10]

Пусть имеются n сотрудников, которые могут выполнять n работ, причем использование i-го сотрудника на j-й работе требует c_{ij} ден.ед. затрат $\left(i, j = \overline{1,n}\right)$. Требуется поручить каждому сотруднику выполнение одной определенной работы так, чтобы минимизировать суммарные затраты.

Введем переменные

$$x_{ij} = \begin{cases} 1, \text{ если } i-\text{й сотрудник выполняет } j-\text{ю работу}, \\ 0, \text{ в противном случае.} \end{cases}$$

С учетом того, что каждый сотрудник выполняет одну работу и каждую работу кто-то выполняет, получаем такую задачу:

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

при ограничениях

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

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

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

Замечание 1. Если число работ не совпадает с числом сотрудников, то так же, как и в транспортной задаче, вводят фиктивные работы или сотрудников. При этом считается, что назначение сотрудников на фиктивные работы не связано с затратами и соответствующие элементы матрицы издержек полагаются равными нулю. Если вводятся фиктивные исполнители, то соответствующие элементы матрицы C полагаются очень большими (M).

Замечание 2. Если ставится задача максимизации дохода, то ее сводят к задаче на минимум. Пусть задана матрица доходов $C = \left(c_{ij}\right)$. В каждом столбце находят максимальный элемент l_j и строится матрица $C' = \left(c_{ij}'\right)$, $c_{ij}' = l_j - c_{ij}$. Тогда задача

$$\max_{x \in X} Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij},$$

где X – допустимое множество задачи, эквивалентна задаче

$$\min_{x \in X} Z' = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}' x_{ij} .$$

Замечание 3. Если выполнение какой-либо работы каким-либо исполнителем невозможно или запрещено, то в матрицу потерь вводят достаточно большую стоимость *M*.

Венгерский метод решения задачи о назначениях впервые предложил венгерский математик К. Эгервари в 1931 г., но его работа осталась незамеченной. В 1953 г. математик Г. Кун перевел эту работу на английский язык, развил идеи К. Эгервари и усовершенствовал метод, который в честь первого автора был назван «венгерским».

Венгерский метод

Метод базируется на следующих двух утверждениях.

Определение. Две матрицы C и D размерности $m \times n$ называются эквивалентными, если

$$d_{ij} = c_{ij} - u_i - \upsilon_j,\ i = \overline{1,m},\ \overline{j = 1,n},\ u_i, \upsilon_j \text{ – некоторые числа.}$$

Утверждение 1. Оптимальные решения задач с эквивалентными матрицами совпадают.

Утверждение 2. Если все $c_{ij} \geq 0,\ i, j = \overline{1,n}$ и найдена матрица n-го порядка $X^* = \left(x_{ij}^*\right)$ такая, что $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}^* = 0$, то X^* – оптимальное решение исходной задачи.

Алгоритм венгерского метода

Он состоит из подготовительного этапа и не более, чем *n – 2* итераций.

Подготовительный этап.

1. *Приведение матрицы C по строкам*. В каждой строке матрицы C находим минимальный элемент и вычитаем его из элементов соответствующей строки. В результате получим матрицу C_1, в каждой строке которой есть хотя бы один ноль.
2. *Приведение матрицы C_1 по столбцам*. В каждом столбце матрицы C_1 находим минимальный элемент и вычитаем его из элементов соответствующего столбца. В результате получим матрицу *D*, в каждом столбце которой есть хотя бы один ноль.

Основной этап.

1. Отмечаем знаком *(*)* нули в столбцах матрицы **D** так, чтобы в каждой строке был только один такой знак. Если количество помеченных *(*)* нулей равно *n*, то найдено оптимальное решение: расположение помеченных нулей соответствует $x_{ij}^{*}=1$, все остальные $x_{ij}^{*}=0$.

   Если количество помеченных *(*)* нулей меньше *n*, то помечаем знаком *(+)* столбцы матрицы, в которых есть *0**, и считаем эти столбцы *занятыми*. В процессе решения будут появляться занятые строки. Элементы матрицы, которые стоят на пересечении незанятых столбцов и строчек, назовем *незанятыми,* остальные элементы матрицы – *занятыми*. Переходим к п. 2.
2. Просматриваем строки матрицы: если незанятых нулей нет, то переходим на п. 4 алгоритма.

   Если незанятый нуль есть, то отмечаем его знаком *(^)*. Если в соответствующей строке нет *0**, то переходим к п. 3. Если в этой строке есть *0**, то снимаем знак *(+)* со столбца с *0** и помечаем знаком *(+)* строку с *0^*, после чего возвращаемся на начало п. 2, пока не просмотрим все строки матрицы.
3. Строим цепочку: от *0^* по столбцу до *0**, от него по строке до *0^* и т. д., пока это возможно. Цепочка может состоять даже из одного *0^*. Снимаем знаки *(*)* и заменяем *(^)* на *(*)* у нулей из цепочки. Таким образом, количество *0** возрастет хотя бы на единицу. Возвращаемся к п. 1, сняв все отметки с матрицы и оставив только *0**.
4. Находим минимальный элемент среди незанятых. Отнимаем этот элемент от элементов всех незанятых строчек и добавляем ко всем элементам занятых столбцов. Отметки нулей и всех занятых столбцов, и строк оставляем без изменений. Благодаря этому получим новые незанятые нули и возвращаемся к п. 2.

Пример. Эстафета на Олимпийских играх состоит из четырех дистанций в 100, 200, 500 и 800 метров. В команде четыре спортсмена, результаты которых на каждой дистанции приведены в таблице 9.

Таблица 9

Спортсмены	Результаты спортсменов, с			
	100 м	200 м	500 м	800 м
Иванов	10	21	54	86
Петров	10	20	55	88
Коваленко	9	22	55	87
Шевченко	9	21	56	89

Один спортсмен может бежать только одну дистанцию и на каждой дистанции может выступать только один спортсмен из команды. Как распределить спортсменов по дистанциям, чтобы общий результат эстафеты был наилучшим?

Построим математическую модель задачи. Переменные x_{ij} означают результат назначения *i*-го спортсмена на *j*-ю дистанцию. Время, за которое команда пробежит эстафету, будет определяться как

$$t(x) = 10x_{11} + 21x_{12} + 54x_{13} + 86x_{14} + 10x_{21} + 20x_{22} + 55x_{23} + 88x_{24} + \\ + 9x_{31} + 22x_{32} + 55x_{33} + 87x_{34} + 9x_{41} + 21x_{42} + 56x_{43} + 89x_{44} \to \min$$

То, что каждый спортсмен назначается только на одну дистанцию, выражается в виде равенства

$$x_{i1} + x_{i2} + x_{i3} + x_{i4} = 1,\ i = \overline{1,4},$$

а то, что на каждой дистанции выступает только один спортсмен из команды, отражается ограничением

$$x_{1j} + x_{2j} + x_{3j} + x_{4j} = 1,\ j = \overline{1,4}.$$

Кроме того, x_{ij} могут принимать одно из значений 0 или 1.

Для решения этой задачи применим венгерский метод.

Подготовительный этап.

Сделаем приведение матрицы по строкам и столбцам.

Таблица 10

Спортсмены	Результаты спортсменов, с				Минимальный элемент в строке
	100 м	200 м	500 м	800 м	
Иванов	10	21	54	86	10
Петров	10	20	55	88	10
Коваленко	9	22	55	87	9
Шевченко	9	21	56	89	9

Таблица 11

Спортсмены	Результаты спортсменов, с			
	100 м	200 м	500 м	800 м
Иванов	0	11	44	76
Петров	0	10	45	78
Коваленко	0	13	46	78
Шевченко	0	12	47	80
Минимальный элемент в столбце	0	10	44	76

Таблица 12

Спортсмены	Результаты спортсменов, с				
	100 м	200 м	500 м	800 м	
Иванов	0	1	0*	0^	+
Петров	0	0*	1	2	
Коваленко	0	3	2	2	
Шевченко	0*	2	3	4	
	+	+	+		

Основной этап.

Отметим знаком *(+)* занятые столбцы. Переходим к п. 2. В первой строке таблицы 12 есть один незанятый ноль ($c_{14}=0$), который отметим *(^)*, и *0** ($c_{13}=0$), поэтому снимаем знак *(+)* с третьего столбца (знаки, которые снимаем, выделим другим цветом) и обозначим первую строку знаком *(+)*. Больше незанятых нулей в таблице 12 нет, поэтому переходим к п. 4 алгоритма.

Среди незанятых элементов таблицы находим минимальный:

$$\min\{c_{23}, c_{24}, c_{33}, c_{34}, c_{43}, c_{44}\} = \min\{1,2,2,2,3,4\} = 1.$$

Отнимаем это число от всех незанятых строк (таблица 13) и прибавляем ко всем занятым столбцам (таблица 14).

Таблица 13

Спортсмены	Результаты спортсменов, с				
	100 м	200 м	500 м	800 м	
Иванов	0	1	0*	0^	+
Петров	-1	-1*	0	1	
Коваленко	-1	2	1	1	
Шевченко	-1*	1	2	3	
	+	+			

Таблица 14

Спортсмены	Результаты спортсменов, с				
	100 м	200 м	500 м	800 м	
Иванов	1	2	0*	0^	+
Петров	0	0*	0^	1	+
Коваленко	0	3	1	1	
Шевченко	0*	2	2	3	
	+	+			

Возвращаемся к п. 2 алгоритма. Незанятый ноль отвечает элементу c_{23}. Пометим его знаком *(^)*, вторую строку делаем занятой и снимаем пометку *(+)* со второго столбца. Незанятых нулей в таблице 14 нет. Переходим к п. 4 алгоритма и заполняем таблицу 15:

$$\min\{c_{32}, c_{33}, c_{34}, c_{42}, c_{43}, c_{44}\} = \min\{3,1,1,2,2,3\} = 1.$$

Таблица 15

Спортсмены	Результаты спортсменов, с				
	100 м	200 м	500 м	800 м	
Иванов	2	2	0	0*	+
Петров	1	0*	0^	1	+
Коваленко	0	2	0*	0	
Шевченко	0*	1	1	2	
	+				

Возвращаемся к п. 2. В третьей строке есть два незанятых нуля (c_{33}, c_{34}). Выберем, например, c_{33} и пометим его знаком *(^)*. Так как в третьей строке нет *0**, то переходим к п. 3.

Строим цикл $c_{33} \to c_{13} \to c_{14}$. Снимем *(*)* с c_{13} и заменим *(^)* на *(*)* у c_{14} и c_{33}. Таким образом, новый набор нулей с *(*)* на один больше, чем предыдущий. Возвращаемся к п. 1 алгоритма и делаем назначения. Количество *(*)* равно четырем, поэтому можно определить оптимальный вариант назначений (таблица 16).

Таблица 16

Спортсмены	Результаты спортсменов, с			
	100 м	200 м	500 м	800 м
Иванов				*
Петров		*		
Коваленко			*	
Шевченко	*			

Таким образом, Шевченко должен бежать на дистанции 100 м, Петров – 200 м, Коваленко – 500 м и Иванов – 800 м. Минимальное время, за которое команда может пробежать эстафету, составит 170 с.

Описанный алгоритм венгерского метода можно использовать для его программной реализации. Если задачу о назначениях решать вручную, то этапы алгоритма можно выполнять проще. При этом подготовительный этап не меняется.

Основной этап.

1. Находим строку матрицы *D*, содержащую только один нулевой элемент и пометим этот элемент знаком *(*)*. Если таких строк нет, а есть строки с большим количеством нулей, то выбираем любой из них и также помечаем его *(*)*. Переходим к п. 2.
2. Если в соответствующих столбце и строке есть другие нулевые элементы, то их зачеркиваем.
   Пункты 1 и 2 повторяем до тех пор, пока это возможно. Если количество помеченных нулевых элементов равно *n*, то найдем оптимальный план задачи. Если количество таких элементов меньше *n*, то переходим к п. 3.
3. Проводим минимальное количество горизонтальных и вертикальных линий через строки и столбцы матрицы так, чтобы все нулевые элементы были зачеркнуты.
4. Находим минимальный элемент среди невычеркнутых. Вычитаем его из всех невычеркнутых элементов матрицы и прибавляем ко всем элементам матрицы, находящимся на пересечении прямых.

Все элементы, через которые проходит только одна прямая, остаются без изменения. В результате такой процедуры в матрице появится, по крайней мере, один новый ноль.

Возвращаемся в п. 1 и повторяем действия до тех пор, пока не будет получен оптимальный план.

Решим тот же пример вторым способом.

$$D=\begin{pmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 2\\ 0 & 3 & 2 & 2\\ 0 & 2 & 3 & 4\end{pmatrix} \quad \rightarrow \quad D=\begin{pmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 2\\ 0 & 3 & 2 & 2\\ 0^{*} & 2 & 3 & 4\end{pmatrix}$$

Переходим к основному этапу. Выберем строку, содержащую только один ноль ($c_{41}=0$). Пометим этот ноль символом *(*)*. Вычеркиваем элемент c_{31}. На данном этапе можно сделать только три назначения: $x_{41}=1$, $x_{22}=1$, $x_{14}=1$ (или $x_{13}=1$). Так как требуется четыре назначения, то переходим к п. 3. Проводим наименьшее число вертикальных и горизонтальных прямых, проходящих через все нулевые элементы матрицы:

$$D=\begin{pmatrix}\cancel{0} & \cancel{1} & \cancel{0} & \cancel{0}\\ 0 & 0 & 1 & 2\\ 0 & 3 & 2 & 2\\ 0 & 2 & 3 & 4\end{pmatrix}$$

Наименьшим элементом, через который не проходит ни одна прямая, есть $c_{23}=1$. Скорректируем матрицу в соответствии с п. 5:

$$D_1=\begin{pmatrix}1 & 2 & 0 & 0^{*}\\ 0 & 0^{*} & 0 & 1\\ 0 & 3 & 1 & 1\\ 0^{*} & 2 & 2 & 3\end{pmatrix}$$

$$D_1=\begin{pmatrix}\cancel{1} & \cancel{2} & \cancel{0} & \cancel{0}\\ \cancel{0} & \cancel{0} & \cancel{0} & \cancel{1}\\ 0 & 3 & 1 & 1\\ 0 & 2 & 2 & 3\end{pmatrix}$$

$$D_2 = \begin{pmatrix} 2 & 2 & 0 & 0^* \\ 1 & 0^* & 0 & 1 \\ 0 & 2 & 0^* & 0 \\ 0^* & 1 & 1 & 2 \end{pmatrix}$$

Получаем то же решение, что и венгерским методом.

Замечание. Данный оптимальный план не единственный. В матрице D_2 знак *(*)* может быть расставлен следующим образом:

$$D_2 = \begin{pmatrix} 2 & 2 & 0^* & 0 \\ 1 & 0^* & 0 & 1 \\ 0 & 2 & 0 & 0^* \\ 0^* & 1 & 1 & 2 \end{pmatrix},$$

и тогда Иванов должен бежать 500 м, а Коваленко – 800 м. При этом время, затраченное на эстафету, останется равным 170 с.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Вариант 1

В определенный день компания по перевозке грузов должна забрать пять грузов в точках *A, B, C, D, E* и доставить их в пункты *a, b, c, d, e* соответственно. Расстояния (в милях) между точками загрузки и пунктами назначений грузов приведены ниже (N – порядковый номер студента в группе):

A – a	**B – b**	**C – c**	**D – d**	**E – e**
30 + N	30	70 – N	50	40

Фирма располагает пятью грузовиками двух типов *X* и *Y* в точках *S, T, U, V, W*; типы грузовиков: *X* в *S*, *Y* в *T*, *X* в *U*, *X* в *V* и *Y* в *W*. Грузовики типа *X* новее и экономичнее грузовиков типа *Y*, и стоимости перевозки на них ниже. Стоимости пробега одной мили (в центах) для грузовиков обоих типов (включающие горючее, страховку, поддержку оборудования и т. д.) приведены ниже:

Тип	**Пустой**	**Загруженный**
X	20	40
Y	30	60

Расстояния от мест стоянки грузовиков до пунктов загрузки приведены в таблице:

Места стоянки	**Пункты загрузки**				
	A	**B**	**C**	**D**	**E**
S	30	20	40	10	20
T	30	10	30	20	30
U	40	10	10	40	10
V	20	20	40	20	30
W	30	20	10	30	40

Определите, в какие пункты следует отправить грузовики с каждой стоянки, чтобы общая стоимость перевозок была минимальной. Возможность загрузки одной машины в нескольких пунктах исключается. Также предполагается, что все грузы имеют приблизительно одинаковый размер и для них требуется одинаковый объем работ по упаковке, размещению и т. п.

Замечание. Учесть, что после доставки груза к месту назначения грузовик должен вернуться на стоянку.

Вариант 2

Группе, исследующей рынок, требуются данные из пяти различных городов. Группа располагает тремя рабочими днями и намеревается провести по полдня в каждом городе. Точки, предназначенные для опроса, выбраны заранее. Пользуясь имеющимся опытом, группа оценивает вероятности успешных контактов в каждом городе. Необходимые данные приведены в таблице (N – порядковый номер студента в группе):

Время	**Города**				
	A	**B**	**C**	**D**	**E**
Среда утром	0.67	0.62	0.52	0.40	0.63
Среда после обеда	0.90	0.70	0.65	0.87	0.83
Четверг утром	0.57	0.25	0.60	0.60	0.53
Четверг после обеда	0.40	0.52	0.45	0.43	0.50
Пятница утром	0.63	0.60	0.40	0.36	0.67
Количество назначенных опросов	5*N	20+N	40	20+N	5*N

Как следует группе распределить время по пяти городам, чтобы максимизировать ожидаемое количество успешных опросов?

ЗАДАЧА КОММИВОЯЖЕРА [1, 6, 8]

Постановка задачи. Пусть имеется n городов и известно расстояние между любой парой городов c_{ij}, $i, j = \overline{1,n}$. Расстояние между городами удобно записывать в виде матрицы n-го порядка C. Коммивояжер, выезжая из какого-либо города, должен посетить все n городов, побывав в каждом из них только один раз, и вернуться в исходный пункт, при этом длина маршрута должна быть *минимальной*.

Замечание 1. Если прямого сообщения между городами i и j нет, то полагают $c_{ij} = \infty$.

Замечание 2. Если городам поставить в соответствие вершины некоторого графа, а соединяющим их дорогам – дуги, то в терминах теории графов задача заключается в определении гамильтонова контура минимальной длины, в котором под длиной контура понимается не количество дуг, входящих в контур, а их суммарная длина.

Замечание 3. Так как решается задача минимизации длины маршрута, то естественно задать элементы матрицы C, стоящие на ее главной диагонали, равными ∞.

Формализуем задачу. Введем неизвестные:

$$x_{ij} = \begin{cases} 1, \textit{ если коммивояжер переезжает из } i-\textit{го города непосредственно в } j-\textit{й}, \\ 0, \textit{ в противном случае.} \end{cases}$$

Тогда целевая функция Z примет вид

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

На матрицу неизвестных X накладываются ограничения:

$\sum_{i=1}^{n} x_{ij} = 1,\ j = \overline{1,n}$ (коммивояжер въезжает в каждый город только один раз);

$\sum_{j=1}^{n} x_{ij} = 1,\ i = \overline{1,n}$ (коммивояжер выезжает из каждого города только один раз);

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

Однако этих ограничений недостаточно, т. к. они не учитывают требование полного цикла, включающего все города. Устранение подциклов достигается добавлением ограничений

$$u_i - u_j + n x_{ij} \le n - 1,\ i, j = \overline{1, n},$$

где u_i могут принимать произвольные значения, в том числе и неотрицательные, и целые.

Задача коммивояжера в математической постановке эквивалентна задаче упорядочения конечного числа работ на машине с учетом потерь от переналадки. Длительность переналадки машины перед выполнением некоторой работы обычно зависит от характера предшествовавшей работы. Примером такой задачи о переналадке может быть задача о красках.

Пример. Пусть выпускаются четыре вида красок. Длительность очистки оборудования при перемене цвета представлена в таблице 17:

Таблица 17

Предшествовавший цвет	Последующий цвет			
	1. Белый	2. Желтый	3. Красный	4. Голубой
1. Белый	0	1	2	3
2. Желтый	6	0	1	2
3. Красный	8	6	0	1
4. Голубой	10	8	6	0

Требуется найти такую последовательность производства красок, при которой суммарные потери от переналадок были бы минимальными.

Метод ветвей и границ решения задачи коммивояжера

Существуют различные версии метода ветвей и границ решения задачи коммивояжера. Рассмотрим стандартный метод Дж. Литтла и др. (1963 г.).

Основная идея метода состоит в том, что вначале строят нижнюю границу длин маршрутов для всего множества гамильтоновых контуров K. Затем множество K разбивают на два подмножества таким образом, чтобы первое подмножество K_{ij}^{1} состояло из гамильтоновых контуров, содержащих некоторую дугу *(i,j)*, а другое подмножество $K_{\bar{ij}}^{1}$ не содержало этой дуги.

Для каждого из подмножеств определяют нижние границы по тому же правилу, что и для первоначального множества. Полученные нижние границы подмножеств K_{ij}^1 и $K_{\overline{ij}}^1$ будут не меньше нижней границы для всего множества гамильтоновых контуров. Сравнивая нижние границы подмножеств K_{ij}^1 и $K_{\overline{ij}}^1$, можно выделить среди них то, которое с большей вероятностью содержит оптимальный план («перспективное» подмножество).

Это «перспективное» подмножество по аналогичному правилу разбивается на два новые K_{lp}^2 и $K_{\overline{lp}}^2$. Для них снова отыскивают нижние границы и т. д. Процесс ветвления продолжается до тех пор, пока не отыщется единственный полный гамильтонов контур. Получив его, просматривают отброшенные подмножества. Если их нижние границы больше длины полученного контура, то задача решена. Если есть такие, для которых нижние границы меньше этой длины, то найденное подмножество подвергается дальнейшему ветвлению, пока не убедимся, что оно не содержит лучшего гамильтонова контура. Если же такой найдется, то анализ оборванных ветвей продолжается относительно нового значения длины контура. Процесс решения будет закончен тогда, когда проанализируем все подмножества.

Таким образом, для практической реализации идеи метода ветвей и границ применительно к задаче коммивояжера нужно найти метод определения нижних границ подмножеств и разбиения множества гамильтоновых контуров на подмножества (ветвление).

Определение нижних границ базируется на утверждении 1, которое было приведено для задачи о назначениях. Если ко всем элементам строки или столбца прибавить (или отнять) некоторое число, то задача останется эквивалентной прежней, т. е. оптимальный маршрут коммивояжера не изменится, а длина любого гамильтонова контура изменится на данную величину.

Так же, как и в задаче о назначениях, приведем матрицу C по строкам и по столбцам, обозначая минимальные элементы в строках через u_i и минимальные элементы в столбцах через υ_j. Величину $s = \sum_i u_i + \sum_j \upsilon_j$ называют *константой приведения*. Для длины любого контура справедливо условие $Z_1 = Z - s$. Отсюда $Z = Z_1 + s$. Так как в приведенной матрице все элементы неотрицательные, то $Z_1 \geq 0$, поэтому $Z \geq s$, т. е. s можно брать в качестве нижней границы длины гамильтонова контура.

Разбиение множества гамильтоновых контуров на подмножества осуществляется дугой *(i,j)*. Первое подмножество составляют гамильтоновы контуры, содержащие дугу, второе – не содержащие эту дугу. Включение дуги *(i,j)* в контур автоматически приводит к сокращению матрицы *C* на строку *i* и столбец *j*. Так как гамильтонов контур не может одновременно содержать дуги *(i,j)* и *(j,i)*, то, включив *(i,j)* в контур, следует *(j,i)* исключить, положив $c_{ji} = \infty$.

Для выявления претендентов на включение во множество дуг, по которым производится ветвление, найдем степени нулевых элементов приведенной матрицы.

Определение. Степень нулевого элемента *d* равна сумме минимальных элементов в строке *i* и в столбце *j* при исключении перехода *(i,j)* (т. е. мысленно положив $c_{ij} = \infty$).

С наибольшей вероятностью искомому гамильтонову контуру принадлежат те дуги, длины которых в матрице равны нулю. Ноль с максимальной степенью определяет пару (i_0, j_0), которую включаем в множество $K^1_{i_0 j_0}$ и которая является запретной для множества $K^1_{\overline{i_0 j_0}}$.

Далее определим нижние границы этих множеств. Ветвление производим для подмножества с меньшей нижней границей.

Сравниваем длину построенного контура с нижними границами висячих ветвей. Если длина контура не превышает нижних границ этих ветвей, то задача решена. В противном случае продолжается ветвление подмножества, имеющего наименьшую нижнюю границу.

Применим описанный метод для решения примера о красках.

Матрица *C* имеет вид:

Таблица 18

j / i	1	2	3	4	u_i
1	∞	1	2	3	1
2	6	∞	1	2	1
3	8	6	∞	1	1
4	10	8	6	∞	6

Приведем ее по строкам:

Таблица 19

j / i	1	2	3	4
1	∞	0	1	2
2	5	∞	0	1
3	7	5	∞	0
4	4	2	0	∞
v_j	4	0	0	0

Приведем по столбцам:

Таблица 20

j / i	1	2	3	4
1	∞	0	1	2
2	1	∞	0	1
3	3	5	∞	0
4	0	2	0	∞

Вычислим константу приведения, которая будет нижней границей множества всех допустимых гамильтоновых контуров: *s=1+1+1+6+4=13*.

Найдем степени нулей приведенной матрицы:

$$d_{12} = 1 + 2 = 3,\ d_{23} = 1 + 0 = 1,$$

$$d_{34} = 1 + 3 = 4,\ d_{41} = 1 + 0 = 1,\ d_{43} = 0 + 0 = 0.$$

Максимальный из них соответствует дуге *(3,4)*, которая и является претендентом на включение в гамильтонов контур.

Множество всех гамильтоновых контуров разделим на два: K_{34}^{1} и $K_{\overline{34}}^{1}$. Матрицу, соответствующую K_{34}^{1}, с дугой *(3,4)* получаем из приведенной матрицы, вычеркивая строку 3 и столбец 4, а также меняя элемент *(4,3)* на ∞ (таблица 21).

Таблица 21

j / i	1	2	3
1	∞	0	1
2	1	∞	0
4	0	2	∞

Матрицу, соответствующую $K_{\overline{34}}^{1}$, получим из таблицы 20 путем замены элемента c_{34} на ∞ (таблица 22).

Таблица 22

j / i	1	2	3	4
1	∞	0	1	2
2	1	∞	0	1
3	3	5	∞	∞
4	0	2	0	∞

Матрица таблицы 21 приведена, поэтому нижняя граница множества K_{34}^{1}

$$s\left(K_{34}^{1}\right)=13+0=13.$$

Находим константу приведения для множества контуров $K_{\overline{34}}^{1}$ по таблице 22:

$$s\left(K_{\overline{34}}^{1}\right)=13+3+1=17.$$

Сравниваем нижние границы подмножеств K_{34}^{1} и $K_{\overline{34}}^{1}$. Так как $s\left(K_{34}^{1}\right)<s\left(K_{\overline{34}}^{1}\right)$, то дальнейшему ветвлению подвергаем множество K_{34}^{1}. Найдем степени нулей в таблице 21:

$$d_{12}=1+2=3,\ d_{23}=1+1=2,\ d_{41}=2+1=3.$$

Максимальную степень имеют два нуля. Выберем дугу *(1,2)* и множество K_{34}^{1} разобьем на два: K_{12}^{2} и $K_{\overline{12}}^{2}$. Первому подмножеству соответствует таблица 23, второму – таблица 24.

Таблица 23

j / i	1	3
2	∞	0
4	0	∞

Константа приведения для этой таблицы не меняется: $s\left(K_{12}^{2}\right)=13+0=13$.

Таблица 24

j / i	1	2	3
1	∞	∞	1
2	1	∞	0
4	0	2	∞

Здесь $s\left(K_{12}^{2}\right)=13+2+1=16$.

Так как $s\left(K_{12}^{2}\right)<s\left(K_{12}^{2}\right)$, то перспективной является ветвь, соответствующая множеству K_{12}^{2}. Однако матрица таблицы 23 является приведенной и содержит два нулевых элемента.

По таблице 23 можно заключить, что в оптимальный гамильтонов контур должны войти дуги *(2,3)* и *(4,1)*.

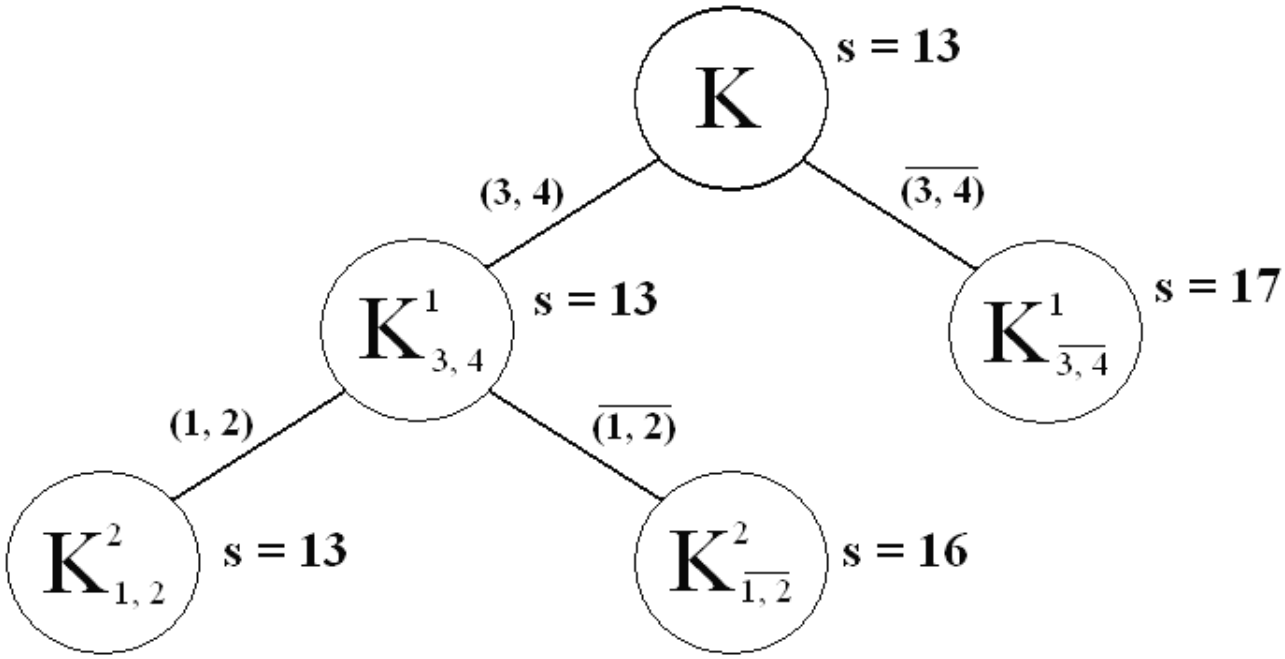
Таким образом, в оптимальный гамильтонов контур входят дуги *(3,4), (1,2), (2,3), (4,1)*. После упорядочения этих дуг, начиная, например, с дуги *(1,2)*, получим цикл

(1,2), (2,3), (3,4), (4,1),

т. е. последовательность изготавливаемых красок должна быть следующей: белая, желтая, красная и голубая. При этом затраты на переналадку составят 13 ед.

Замечание. Цикл можно начинать с любой дуги оптимального контура.

Ветвление, использованное при решении данной задачи, можно представить в следующем виде:



ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Вариант 1

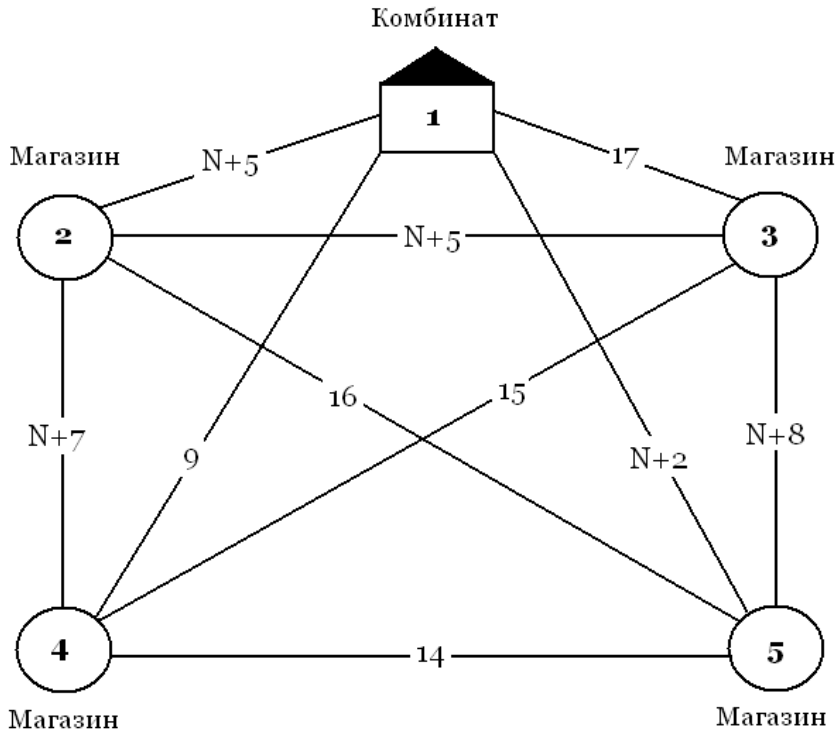
Продавец книг, проживающий в городе A, один раз в месяц должен посетить своих четырех клиентов, которые проживают в городах B, C, D, E соответственно. Приведенная ниже таблица содержит расстояния в километрах между различными городами (N – порядковый номер студента в группе).

	A	B	C	D	E
A	–	N+12	22	N+15	21
B	N+12	–	N+8	11	N+13
C	22	N+8	–	16	N+18
D	N+15	11	16	–	19
E	21	N+13	N+18	19	–

Необходимо составить такой маршрут движения продавца книг, чтобы суммарное пройденное им расстояние было минимальным.

Вариант 2

Молочный комбинат осуществляет доставку продукции в ряд торговых точек города автомобильным транспортом. Определите маршрут минимальной длины доставки продукции во все торговые точки и возвращения транспортного средства на комбинат для очередной загрузки. Расстояния между торговыми точками и комбинатом (в километрах) известны и представлены следующим графом (N – порядковый номер студента в группе**).**



РЕШЕНИЕ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ В СРЕДЕ MATLAB

В среде MATLAB задачи линейного программирования решаются с помощью функции `linprog()`.

Рассмотрим задачу линейного программирования

$$\begin{gathered} f^T \cdot x \to \min, \\ A \cdot x \le b, \\ Aeq \cdot x = beq, \\ lb \le x \le ub. \end{gathered} \tag{1}$$

Основными входными данными `linprog` являются:

- вектор коэффициентов целевой функции `f`;
- матрица ограничений-неравенств `A`;
- вектор правых частей ограничений-неравенств `b`;
- матрица ограничений-равенств `Aeq`;
- вектор правых частей ограничений-равенств `beq`;
- вектор `lb`, ограничивающий допустимый план `x` снизу;
- вектор `ub`, ограничивающий допустимый план `x` сверху.

На выходе функция `linprog` дает оптимальный план `x` задачи (1) и оптимальное значение целевой функции `fval`.

Пример. Решим в MATLAB задачу линейного программирования

$$\begin{gathered} f(x) = 3x_1 + x_2 + 2x_3 \to \min, \\ x_1 + x_2 + x_3 \ge 1, \\ 2x_1 + x_2 - x_3 \ge -1, \\ x_1 - x_2 + x_3 = 0, \\ 0 \le x_i \le 1,\ i = \overline{1,3}. \end{gathered}$$

Для приведения задачи к виду (1) необходимо умножить первое и второе ограничения на (-1). Соответствующая программа (m-файл) выглядит так:

```
clear all
close all
clc % удаляются все текущие переменные из памяти MATLAB,
% закрываются все графические окна, очищается экран консоли
```

```
f = [3 1 2]; % задается вектор длины три
A = [-1 -1 -1; -2 -1 1]; % строки матрицы разделяются точкой с
% запятой
b = [-1; 1]; % вектор-столбец правых частей ограничений-неравенств
% после приведения к виду (1)
Aeq = [1 -1 1];
beq = [0];
lb = zeros(3,1); % задается нулевой вектор-столбец длины три
ub = [1; 1; 1];
[x,fval] = linprog(f,A,b,Aeq,beq,lb,ub);
x
fval
```

Запустив программу, получим сообщение

```
Optimization terminated.
x =
   0
   0.5000
   0.5000
fval =
   1.5000
```

Дополнительно можно задать начальное приближение `x0`:

```
[x,fval] = linprog(f,A,b,Aeq,beq,lb,ub,x0)
```

Если какой-то из входных параметров отсутствует, на его место следует поставить квадратные скобки `[]`, за исключением случая, когда это последний параметр в списке. Например, если нужно решить задачу без ограничений-равенств, в которой не задано начальное приближение, то оператор вызова функции `linprog` будет выглядеть так:

```
[x,fval] = linprog(f,A,b,[],[],lb,ub)
```

С помощью входного параметра `options` устанавливаются некоторые дополнительные настройки, в частности, выбирается алгоритм решения. MATLAB решает задачи линейного программирования двумя способами: *алгоритмом внутренней точки* (Large-Scale Algorithm) и вариантом *симплекс-метода* (Medium-Scale Algorithm). По умолчанию используется алгоритм внутренней точки. Чтобы выбрать симплекс-метод, нужно написать

```
options = optimset('LargeScale','off','Simplex','on');
[x,fval] = linprog(f,A,b,Aeq,beq,lb,ub,[],options)
```

Разберёмся с выходными данными. MATLAB позволяет выводить информацию о том, как завершилось решение задачи. За это отвечает параметр `exitflag`. Если значение `exitflag` равно 1, то найдено решение задачи; если равно 0, то превышено допустимое число итераций; если равно −2 — множество планов задачи пусто; если равно −3 — целевая функция не ограничена снизу на множестве планов. Интерпретация других значений параметра `exitflag` приведена в MATLAB Help.

Для симплекс-метода допустимое число итераций (`MaxIter`) по умолчанию в 10 раз больше количества переменных. Значение `MaxIter` можно изменить. Чтобы установить допустимое число итераций равным, к примеру, 10, нужно написать

```
options =
optimset('LargeScale','off','Simplex','on','MaxIter',10);
[x,fval] = linprog(f,A,b,Aeq,beq,lb,ub,[],options)
```

Если после выполнения десятой итерации решение не будет найдено, параметр `exitflag` станет нулевым и на экране появится сообщение

```
Maximum number of iterations exceeded;
increase options.MaxIter.
```

Параметр `output` содержит информацию о процессе оптимизации, в частности, число итераций (`iterations`) и используемый алгоритм (`algorithm`). Другие поля параметра `output` описаны в MATLAB Help.

Запустим с данными из примера следующую программу:

```
options = optimset('LargeScale','off','Simplex','on');
[x,fval,exitflag,output] =
linprog(f,A,b,Aeq,beq,lb,ub,[],options);
exitflag
output.iterations
output.algorithm
```

На выходе получим:

```
Optimization terminated.
exitflag =
   1
ans =
   1
ans =
   medium scale: simplex
```

Это означает, что симплекс-метод успешно завершил работу, для нахождения решения потребовалась одна итерация.

ЛИТЕРАТУРА

1. Вагнер Г. Основы исследования операций. Т. 1–3. – М.: Мир, 1972.
2. Волков И. К., Загоруйко Е. А. Исследование операций. – М.: БГТУ им. Баумана, 2000.
3. Дякон В. М., Ковальов Л. Є. Математичне програмування. – К., 2007.
4. Зайченко Ю. П. Дослідження операцій. – К., 2001.
5. Корбут А. А., Финкельштейн Ю. Ю. Дискретное программирование. – М.: Наука, 1969.
6. Костевич Л. С., Лапко А. А. Исследование операций. Теория игр. – Минск: Выш. шк., 2008.
7. Кремер Н. Ш., Путко Б. А., Тришин И. М., Фридман М. М. Исследование операций в экономике. – М.: ЮНИТИ, 2003.
8. Сакович В. А. Исследование операций. – Минск: Выш. шк., 1985.
9. Сергиенко И. В. Математические модели и методы решения задач дискретной оптимизации. – К.: Наукова думка, 1985.
10. Таха Х. А. Введение в исследование операций. – М.: Вильямс, 2001.

СОДЕРЖАНИЕ