Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический университет имени П. О. Сухого»

Кафедра «Информационные технологии»

И. А. Мурашко

ОПТИМИЗАЦИЯ ПРОЕКТНЫХ РЕШЕНИЙ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
по одноименной дисциплине для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» дневной формы обучения

Гомель 2011

УДК 519.85(075.8)
ББК 22.18+22.193я73
М91

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 8 от 28.03.2011 г.*)

Рецензент: канд. техн. наук, доц. каф. «Информатика» ГГТУ им. П. О. Сухого
В. И. Мисюткин

Мурашко, И. А.

М91 Оптимизация проектных решений : лаборатор. практикум по одноим. дисциплине для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» днев. формы обучения / И. А. Мурашко. – Гомель : ГГТУ им. П. О. Сухого, 2011. – 130 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://lib.gstu.local. – Загл. с титул. экрана.

Содержит краткие теоретические сведения и алгоритмы поиска оптимального решения широкого класса задач, возникающих в процессе проектирования изделий и технологических процессов в машиностроении и приборостроении. Приведены задания с вариантами к лабораторным работам.

Для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» специализации 1-40 01 02-01 «Информационные системы и технологии (в проектировании и производстве)» дневной формы обучения.

УДК 519.85(075.8)
ББК 22.18+22.193я73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Методы оптимизации [1–4] находят широкое применение во многих технических и экономических приложениях, а именно там, где возникают задачи принятия оптимальных решений. Это, прежде всего задачи, связанные с проектированием изделий и технологических процессов их изготовления [1,6].

В общем виде задача оптимизации проектных решений может быть сформулирована следующим образом [4]: найти значения переменных, которые позволяют минимизировать (максимизировать) целевую функцию и удовлетворяют заданным требованиям и ограничениям.

В лабораторном практикуме рассмотрен ряд методов, используемых для улучшения (оптимизации) проектных решений в системах автоматизированного проектирования изделий и технологических процессов [5, 8, 9, 11]. Основной упор сделан на алгоритмическую реализацию методов оптимизации, что позволяет расширить область их применения.

Выполнение лабораторных работ предусматривает программную реализацию конкретных алгоритмов оптимизации, что позволяет оценить эффективность их применения для задач различной размерности. Практически все работы содержат конкретный пример решения, что позволяет легко правильность работы программы. В остальных случаях (работа №2 и №3) проверка может быть выполнена в каком-либо математическом (например, в MathCAD или Maple [7]) или в офисном (Microsoft Exel) пакете.

Лабораторная работа № 1
«Простейший пример оптимизации»
(4 часа)

Цель работы: изучение различных методов решения простейших задач оптимизации.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Постановка задачи оптимизации

При изготовлении некоторого изделия используются пять рабочих мест в различных местах завода. Последовательность обработки не имеет значения. Время обработки на всех рабочих местах одинаково и равно 1 мин. Время перемещения изделия от одного рабочего места к другому различно и определяется таблицей 1. Стоит задача минимизации времени изготовления изделия. В качестве ограничения стоит требование забирать изделие из того рабочего места, на которое оно поступило первоначально. Таким образом:

$T_{\text{изготовления}} = T_{\text{обработки}} + T_{\text{транспортировки}}$.

Учитывая, что $T_{\text{обработки}} = const$, то задача сводится к минимизации времени транспортировки.

Таблица 1 – Время транспортировки

	P_1	P_2	P_3	P_4	P_5
P_1	0	И(10)	в(3)	а(1)	н(15)
P_2	10	0	о(16)	в(3)	П(17)
P_3	3	16	0	е(6)	т(20)
P_4	1	3	6	0	р(18)
P_5	15	17	20	18	0

В общем случае для определения минимального времени транспортировки необходимо выполнить полный перебор всех возможных вариантов. Однако для больших значений числа рабочих мест данный метод неприемлем. Поэтому используют другие подходы, которые имеют меньшую вычислительную сложность. Одним из таких подходов является «жадный» алгоритм. Суть алгоритма заключается в следующем. В первой строке находим минимальный элемент (в примере он равен 1) и переходим с строке, которая имеет номер столбца, в котором расположен минимальный элемент (в примере – 4). Затем в

четвертой строке ищем минимальный элемент, при этом игнорируем столбцы с номерами строк, на которых мы уже побывали, то есть

min (3, 6, 18)=3.

Данные действия повторяются до тех пор, пока не будут пройдены все строки. В примере:

$P_1 - (1) - P_4 - (3) - P_2 - (16) - P_3 - (20) - P_5 - (15) - P_1$.

$1 + 3 + 16 + 20 + 15 = 55$.

Таким образом, используя «жадный» алгоритм, получили:

$T_{\text{транспортировки}}$=55 мин.

Для примера рассмотрим еще одну последовательность транспортировки

$P_1 - (10) - P_2 - (16) - P_3 - (6) - P_4 - (18) - P_5 - (15) - P_1$.

$10 + 16 + 6 + 18 + 15 = 65$.

Таким образом, применение «жадного» алгоритма позволяет найти некоторое оптимальное (квазиоптимальное) решение данной задачи. Однако следует помнить, что данный алгоритм не гарантирует нахождение минимума.

2 ЗАДАНИЕ

1. Построить таблицу 1 (5x5). В качестве исходных данных записать свою фамилию.

2. Найти минимальное время транспортировки, используя «жадный» алгоритм. Написать программу, реализующую «жадный» алгоритм для произвольной матрицы (10x10).

3. Разработать программу для нахождения минимального и максимального времени транспортировки методом полного перебора.

4. Оценить, к какому значению ближе решение, найденное в п.2.

5. Оценить вычислительную сложность алгоритмов.

3 ТРЕБОВАНИЕ К ОТЧЕТУ

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Краткие теоретические сведения.
3. Пример решения задания для своего варианта (только «жадный» алгоритм).
4. Схемы алгоритмов.
5. Листинги основных частей программы.
6. Тесты.
7. Результат выполнения программы.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем заключается суть «жадных» алгоритмов?
2. В чем заключается суть алгоритмов полного перебора.
3. Область применения «жадных» алгоритмов.
4. Область применения алгоритмов полного перебора.
5. Оценить вычислительную сложность «жадного» алгоритма.
6. Оценить вычислительную сложность алгоритма полного перебора.
7. Какой из рассмотренных алгоритмов гарантирует нахождение абсолютного минимума?

Лабораторная работа №2
«Безусловная одномерная оптимизация»
(6 часов)

Цель работы: изучение различных методов одномерной оптимизации.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Постановка задачи одномерной безусловной оптимизации

Составной частью многих методов многомерной оптимизации является поиск экстремума функции одной переменной. Будем рассматривать задачу оптимизации на примере задачи минимизации, которая формулируется в виде:

$$\min f(x), \quad x \in X, \tag{1.1}$$

где X – множество допустимых точек, среди которых ищется точка x^*, доставляющая минимум целевой функции $f(x)$.

В общем случае, оптимизация предполагает поиск экстремального (минимального или максимального) значения функции. Однако задача (1.1) эквивалента задаче

$$\max(-f(x)), \quad x \in X, \tag{1.2}$$

что позволяет переносить решения, полученные для задачи минимизации на задачу максимизации, и наоборот.

Если X есть вещественная ось чисел и целевая функция $f(x)$ имеет только один аргумент, то имеет место задача одномерной безусловной минимизации.

В задачах проектирования не существует бесконечных величин. Поэтому решение ищется на некотором отрезке $X = [a,b]$, содержащем искомое решение x^*. Такой отрезок называется отрезком локализации. Относительно целевой функции $f(x)$ предполагается, что она унимодальная.

Определение 1.1: Функция $f(x)$ называется унимодальной на $X = [a,b]$, если существует такая точка $x^* \in X$, что

$f(x_1) > f(x_2)$, если $x_1 < x_2 < x^*$, $x_1, x_2 \in X$,

$f(x_1) < f(x_2)$, если $x^* < x_1 < x_2$, $x_1, x_2 \in X$.

В ряде методов относительно целевой функции $f(x)$ предполагается, что она выпуклая на X.

Определение 1.2: Функция $f(x)$ называется выпуклой на $X=[a,b]$, если $f(\alpha\cdot x_1+(1-\alpha)x_2)\le \alpha\cdot f(x_1)+(1-\alpha)\cdot f(x_2)$ при любых $x_1,x_2\in X$ и любых α, $0\le\alpha\le 1$.

Если при любых $x_1,x_2\in X$ неравенство будет строгим, то функция $f(x)$ называется строго выпуклой. Непрерывная строго выпуклая функция является унимодальной. Однако не всякая унимодальная функция является выпуклой или непрерывной.

1.1.2 Алгоритмы пассивного и активного поиска минимума

Пусть $[a,b]$ – исходный отрезок локализации и пусть N – количество точек, в которых необходимо провести вычисления целевой функции $f(x)$. Точки, в которых необходимо провести вычисления, определяются следующим образом:

1) Если N – нечетное ($N=2k-1$), то

$$x_i=a+\frac{b-a}{N+1}\cdot i,\ i=1,2,3,...N.$$

2) Если N –четное ($N=2k$), то

$$x_{2i}=a+\frac{b-a}{k+1}\cdot i,\ x_{2i-1}=x_{2i}-\delta,\ \delta>0,\ i=1,2,3,...k.$$

Среди множества вычисленных значений $f(x_i)$ ищется точка x_j, в которой достигается минимум:

$$f(x_j)=\min\{f(x_i)\},\ i=\overline{1.N}.$$

Найденная точка принимается за приближенное решение задачи $\tilde{x}=x_j$.

Исходный отрезок локализации сужается до отрезка $[x_{j-1},x_{j+1}]$, длина которого равна

$$L=x_{j+1}-x_{j-1},$$

причем

$$L_N=2\frac{b-a}{N+1},\ \text{если}\ N=2k-1,$$

$$L_N=2\frac{b-a}{k+1}+\delta,\ \text{если}\ N=2k.$$

Точность найденного решения $\tilde{x}$ равна половине отрезка локализации, т.е.

$$| x^* - \tilde{x} | \le \varepsilon,$$

где $\varepsilon = \frac{1}{2} L_N$, x^* – точное решение.

В алгоритмах активного поиска очередная точка, в которой происходит вычисление функции, выбирается с учетом информации, полученной на предыдущих этапах. Рассмотрение этих алгоритмов начнем с методов блочного поиска, которые сочетают в себе пассивные и последовательные стратегии поиска. При этом вычисления в точках объединены в блоки, в каждом из которых проводится одновременно n_i вычислений. Тогда общее число вычислений

$$N = \sum_{i=1}^{m} n_i .$$

Таким образом, блок – это совокупность из нескольких вычислений, которые проводятся одновременно (пассивный поиск).

Результаты, полученные в $(i - 1)$–м блоке, становятся известны перед началом вычислений в *i*–м блоке (x_{ij}, где $j = \overline{1, n_i}$). Это определяет последовательный поиск. Если размеры блоков равны единице, т.е. $n_i = 1$, то имеем обычный последовательный алгоритм поиска.

Алгоритм равномерного блочного поиска

*Шаг*_1. Задаются исходный отрезок локализации $[a, b]$, ε – точность приближенного решения $\tilde{x}$, число вычислений в блоке – n (нечетное, $n = 2k - 1$). Проводим вычисление функции в середине отрезка $[a, b]$, т.е. вычисляем

$y_k = f(x_k)$, где $x_k = (a + b)/2$.

*Шаг*_2. Вычисляем значение функции в остальных точках:

$y_i = f(x_i)$,

где $x_i = a + i\dfrac{(b - a)}{(n+1)}$, $j = \overline{1, n_i}$, $i \ne k$.

Находим точку x_j, в которой достигается минимум среди вычисленных значений:

$$f(x_j) = \min\{f(x_i)\},$$

следовательно, точное значение минимума x^* содержится на отрезке $[x_{j-1}, x_{j+1}]$.

*Шаг*_3. Полагаем:

$a = x_{j-1},\ b = x_{j+1},\ x_k = x_j,\ y_k = y_j.$

Если $b - a \le 2\varepsilon$, то $\tilde{x} = x_k,\ \tilde{y} = y_k$ и поиск заканчивается. Иначе перейти к шагу 2. Если заданная точность ε достигнута после *m* итераций, то длина отрезка локализации после всех *N* вычислений (где $N = n + (m-1)(n-1) = (n-1)m + 1$) будет:

$$L_N = \left(2\frac{b-a}{n+1}\right)^m,$$

и

$$|x^* - \tilde{x}| \le \frac{1}{2}L_N.$$

Алгоритм деления интервала пополам

Данный алгоритм является вариантом предыдущего алгоритма при $n = 3$.

*Шаг_*1. Задаются a, b, ε. Вычисляем значение функции в точке $x_2 = (a+b)/2$, т.е. находим $y_2 = f(x_2)$.

*Шаг_*2. Вычисляем значение функции в остальных точках блока:

$x_1 = (a + x_2)/2,\ y_1 = f(x_1),\ x_3 = (x_2 + b)/2,\ y_3 = f(x_3).$

Находим значение x_j из условия $f(x_j) = \min\{f(x_i)\},\ i = 1,2,3.$

Тогда точное решение x^* содержится на отрезке $[x_{j-1}, x_{j+1}]$. Предполагается $x_0 = a,\ x_4 = b$.

*Шаг_*3. Полагаем:

$a = x_{j-1},\ b = x_{j+1},\ x_2 = x_j,\ y_2 = y_j.$

Если $b - a \le 2\varepsilon$, то $\tilde{x} = x_2,\ \tilde{y} = y_2$ и поиск заканчивается. Иначе перейти к шагу 2.

После *k* итераций общее число вычислений значения функции равно $N = 2k + 1$, а длина получившегося отрезка локализации будет:

$$L_N = \frac{b-a}{2^k} = \frac{b-a}{2^{[N/2]}},$$

где $[z]$– целая часть числа *z*.

Следовательно, достигнутая точность будет:

$|x^* - \tilde{x}| \le \varepsilon,\ \varepsilon = 1/2L_N.$

1.1.3 Метод дихотомии

Это алгоритм блочного поиска для $n = 2$, т.е. когда в блоке два вычисления значения функции. Так как пассивная составляющая алгоритма, т.е. блок, содержит четное число точек, то оптимальный выбор точек x_{ij}, в которых необходимо вычислить значение функции, будет неравномерным, в отличие от предыдущих алгоритмов, где число точек в блоке было нечетным и, соответственно, расположение точек равномерным. Если блок содержит две точки, то они должны быть расположены как можно ближе к середине отрезка. Такое расположение точек позволяет получить наименьший отрезок неопределенностей.

Схема алгоритма.

*Шаг_*1. Задаются a, b, ε и δ – малое положительное число, значительно меньшее чем ε.

*Шаг_*2. Определяется середина отрезка:

$x = (a + b)/2$.

Вычисляем значение функции в двух точках, близких к середине:

$y_1 = f(x - \delta)$, $y_2 = f(x + \delta)$.

*Шаг_*3. Определяется следующий отрезок локализации, т.е. определяется какой из отрезков $[a, x + \delta]$ или $[x - \delta, b]$ содержит точное решение x^*. Если $y_1 \le y_2$, то это отрезок $[a, x + \delta]$ и $b = x + \delta$, иначе это отрезок $[x - \delta, b]$ и $a = x - \delta$, т.е. выбранный отрезок локализации мы снова обозначили как $[a, b]$.

Шаг_4. Если $b - a \le 2\varepsilon$, то

$\tilde{x} = (a + b)/2$,

$\tilde{y} = f(\tilde{x})$

и поиск заканчивается. Иначе перейти к шагу 2.

После k итераций число вычислений функции – $N = 2k$, а длина отрезка локализации:

$$L_N < \frac{b - a}{2^{N/2}} + \delta.$$

Следовательно,

$$| x^* - \tilde{x} | < \frac{1}{2} L_N.$$

1.1.4 Метод золотого сечения

Для того чтобы уменьшить отрезок локализации $[a,b]$, необходимо вычислить значение целевой функции $f(x)$, по крайней мере, в двух точках на отрезке $[a,b]$ (рисунок 1.1). В результате этого отрезок локализации сузится до отрезка $[a,x_2]$ или $[x_1,b]$. Так как у нас нет никаких оснований предпочесть один из этих вариантов, то точки x_1 и x_2 должны быть симметричны относительно середины отрезка $[a,b]$. В этом случае длины отрезков $[a,x_2]$ и $[x_1,b]$ будут равны. Таким образом, остаётся вопрос: как выбрать точку x_1.

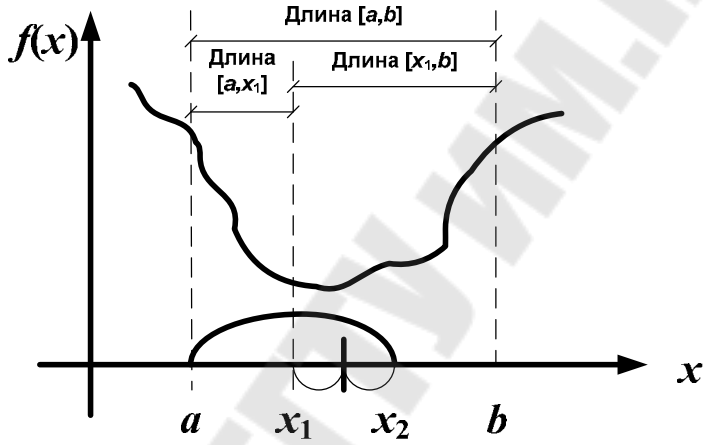

Рисунок 1.1 – Метод золотого сечения

В методе золотого сечения точка x_1 выбирается из соотношения:

$$\frac{длина\ [a,b]}{длина\ [x_1,b]} = \frac{длина\ [x_1,b]}{длина\ [a,x_1]} = \lambda = 1{,}618033989...,$$

т.е. точка x_1 делит отрезок $[a,b]$ по правилу «золотого сечения», где λ – есть «золотое отношение». Точка x_2 определяется как точка симметричная к x_1 относительно середины отрезка.

В результате вычисления значение целевой функции получается отрезок локализации $[a,x_2]$, содержащий точку x_1, или отрезок локализации $[x_1,b]$, содержащий точку x_2. Оказывается, что остающаяся точка на суженном отрезке локализации делит его вновь по правилу «золотого сечения». Следовательно, чтобы, в свою очередь, уменьшить новый отрезок локализации, требуется вычисления целевой

функции в точке, симметричной к оставшейся точке относительно середины этого нового отрезка.

В приводимой ниже схеме алгоритма остающиеся отрезки локализации переименовываются каждый раз как $[a,b]$, а точки, в которых проводятся вычисления целевой функции, обозначается через x_1 и x_2, причём $x_1 < x_2$. Кроме того, $y_1 = f(x_1)$ и $y_2 = f(x_2)$.

Схема алгоритма

*Шаг_*1. Задаются a,b,ε и $\lambda = 1.618....$ Вычисляют:

$$x_1 = b - \frac{b - a}{\lambda},\ x_2 = a + \frac{b - a}{\lambda},\ y_1 = f(x_1),\ y_2 = f(x_2).$$

*Шаг_*2.

а) Если $y_1 \leq y_2$, то полагают $b = x_2,\ x_2 = x_1,\ y_2 = y_1$ и вычисляют $x_1 = a + b - x_2,\ y_1 = f\{x_1\}$.

б) Если $y_1 > y_2$, то полагают $a = x_1,\ x_1 = x_2,\ y_1 = y_2$ и вычисляют $x_2 = a + b - x_1,\ y_2 = f(x_2)$.

*Шаг_*3. Если $b - a > \varepsilon$, то переходят к шагу 2. Иначе если $y_1 < y_2$, то полагают $\tilde{x} = x_1$ и $\tilde{y} = y_1$, если $y_1 \geq y_2$, то полагают $\tilde{x} = x_2$ и $\tilde{y} = y_2$.

После каждой итерации длина отрезка локализации уменьшается в λ раз. Так как первая итерация начинается после вычисления функции в двух точках, то после N вычислений длина отрезка локализации будет:

$$L_N = \frac{b - a}{\lambda^{N-1}}.$$

1.1.5 Метод чисел Фибоначчи

Этот метод применяется, когда число вычислений функции N заранее задано. Метод чисел Фибоначчи, также как и метод золотого сечения относится к симметричным методам, т.е. точки расположены симметрично относительно середины отрезка. Вот только выбор точки x_1 происходит на основе других соотношений. Для этого используются числа Фибоначчи: $F_0, F_1, F_2, F_3, \ldots$, где $F_i = F_{i-2} + F_{i-1}$ $(i = 2,3,\ldots)$ и $F_0 = F_1 = 1$.

Точка x_1 определяется из соотношения:

$$\frac{длина\ [a,x_1]}{длина\ [a,b]} = \frac{F_{N-2}}{F_N},$$

то есть

$$x_1 = a + (b-a)\frac{F_{N-2}}{F_N}.$$

Точка x_1 делит $[a,b]$ на две неравные части. Отношение малого отрезка к большему равно F_{N-2}/F_{N-1}. Точка x_2 определяется как точка, симметричная к x_1 относительно середины отрезка $[a,b]$:

$$x_2 = b - (b-a)\frac{F_{N-2}}{F_N} = a + (b-a)\frac{F_{N-1}}{F_N},$$

при этом будет выполняться условие $x_1 < x_2$.

В результате вычисления значение целевой функции в точках x_1 и x_2 получится отрезок локализации $[a,x_2]$, содержащий точку x_1, или отрезок локализации $[x_1,b]$, содержащий точку x_2. Остающаяся точка делит новый отрезок локализации на две неравные части в отношении F_{N-3}/F_{N-2}. То есть в методе Фибоначчи остающаяся точка делит отрезок на две неравные части в пропорциях, определяемых числами Фибоначчи. На k-ом шаге это отношение равно: $\frac{F_{N-K-2}}{F_{N-K}}$, а длины отрезков равны: $\frac{F_{N-K-1}}{F_N}(b-a)$ и $\frac{F_{N-K}}{F_N}(b-a)$, соответственно. Пример деления отрезка для случая, когда $a=0$ и $b=1$, представлен на рисунке 1.2. Если выполнить замену переменной

$$x' = a + x(b-a),$$

то полученные результаты будут перенесены на отрезок $[a,b]$.

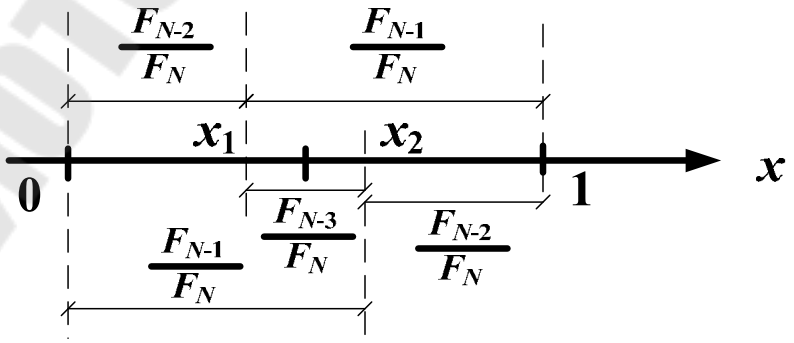

Рисунок 1.2 – Метод чисел Фибоначчи

Схема алгоритма

*Шаг_*1. Задаются a, b, N. Вычисляются числа Фибоначчи F_0, F_1, ..., F_N. Определяется:

$x_1 = a + (b - a)F_{N-2}/F_N$,

$x_2 = a + (b - a)F_{N-1}/F_N$,

$y_1 = f(x_1)$, $y_2 = f(x_2)$.

*Шаг_*2.

а) Если $y_1 \leq y_2$, то полагают $b = x_2$, $x_2 = x_1$, $y_2 = y_1$ и вычисляют $x_1 = a + b - x_2$, $y_1 = f(x_1)$.

б) Если $y_1 > y_2$, то полагают $a = x_1, x_1 = x_2, y_1 = y_2$ и вычисляют $x_2 = a + b - x_1, y_2 = f(x_2)$.

Повторить *Шаг_*2 $N - 2$ раза.

*Шаг_*3. Если $y_1 < y_2$, то полагают $\tilde{x} = x_1$ и $\tilde{y} = y_1$. Если $y_1 \geq y_2$, то полагают $\tilde{x} = x_2$ и $\tilde{y} = y_2$.

Длина отрезка локализации в методе Фибоначчи:

$L_N \approx (b - a)/F_N$.

1.2 Линейный поиск с использованием производных

1.2.1 Метод касательных

Пусть функция $f(x)$ выпукла и дифференцируема на $[a,b]$. Идея метода состоит в следующем. Пусть $[a,b]$ – отрезок локализации и $f(a)$, $f'(a)$, $f(b)$, $f'(b)$ – результаты вычислений в точках a и b. По этой информации строится аппроксимирующая функция, представляющую из себя кусочно-линейную функцию, состоящую из касательной

$L_a(x) = f(a) + f'(a)(x - a)$

к $f(x)$ в точке a и касательной

$L_b(x) = f(b) + f'(b)(x - b)$

к $f(x)$ в точке b.

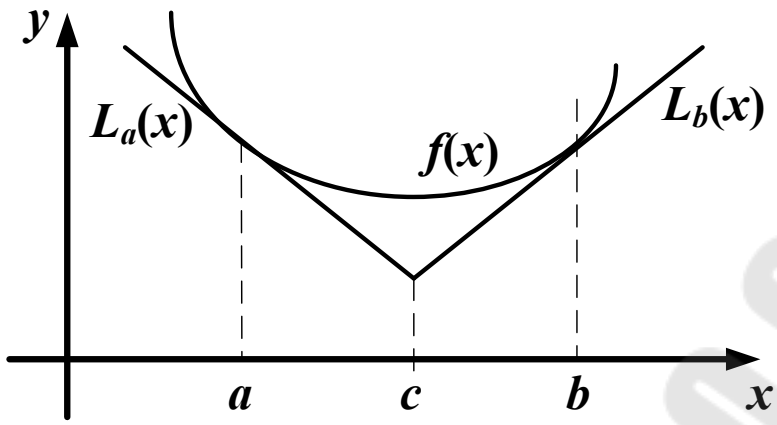

Рисунок 1.3 – Метод касательных

Полученная аппроксимирующая функция есть ломанная состоящая из прямой $L_a(x)$ на $[a,c]$ и $L_b(x)$ на $[c,b]$, где c – точка пересечения касательных. Легко заметить, что при $f(a)<0$ и $f(b)>0$ минимум аппроксимирующей функции достигается в точке c. Значение точки пересечения c можно вычислить по формуле

$$c=\frac{(bf'(b)-af'(a))-(f(b)-f(a))}{f'(b)-f'(a)}.$$

В точке c производятся вычисления $f(c)$ и $f'(c)$. Если $f'(c)=0$, то решением задачи будет $x^*=c$. Если же $f'(c)>0$, то в качестве следующего отрезка локализации будет $[a,c]$. Если же $f'(c)<0$, то – отрезок $[c,b]$. Процесс повторяется до тех пор, пока $f'(c)=0$ или отрезок локализации не достигнет заданной точности.

Схема алгоритма

*Шаг_*1. Заданы a,b,ε. Вычислить:

$y_1=f(a),\ y_2=f(b),\ z_1=f'(a),\ z_2=f'(b)$.

*Шаг_*2. Если $b-a\le 2\varepsilon$, то полагаем:

$\tilde{x}=(a+b)/2,\ \tilde{y}=f(\tilde{x})$.

Поиск окончен.

Если $b-a>2\varepsilon$, то вычислить:

$$c=\frac{(bz_2-az_1)-(y_2-y_1)}{z_2-z_1},\ y=f(c),\ z=f'(c).$$

Если $z=0$, то полагаем $\tilde{x}=c,\ \tilde{y}=y$ и поиск окончен.

Если $z<0$, то $a=c,\ y_1=y,\ z_1=z$. Если $z>0$, то $b=c,\ y_2=y,\ z_2=z$. Повторить *Шаг_2*.

1.2.2 Метод кубической интерполяции

Суть алгоритма заключается в том, что на каждой итерации функция $f(x)$ аппроксимируется кубическим полиномом $f(x)$, точка минимума $\tilde{x}$ которого берется в качестве текущего приближения к искомому минимуму x^*. Алгоритм предполагает вычисления в каждой очередной точке значений функции $f(x)$ и ее производной $f'(x)$.

Вначале необходимо задать параметры метода: ε – погрешность вычисления минимума, x_1 – начальную точку поиска, h_1 – направление поиска и установить начальный интервал $[a,b]$. Если $f'(x_1)>0$, то изменить направление поиска: $h_1=-h_1$. Вычислять $f'(x_k)$ в точках $x_{k+1}=x_k+h_k$ при $h_k=2h_{k-1},\ k=\overline{2,m-1}$ до тех пор, пока не продвинемся в точку x_m, такую, что $f'(x_m)f'(x_{m-1})<0$. Если $x_{m-1}<x_m$, то положить $a=x_{m-1},\ b=x_m$, иначе $a=x_m,\ b=x_{m-1}$.

Схема алгоритма

Шаг_1. Вычислить параметр:

$$\gamma=\frac{z+w-f'(a)}{f'(b)-f'(a)+2w},$$

где $z=f'(a)+f'(b)+3\dfrac{f(b)-f(f)}{b-a}$, $w=\sqrt{z^2-f'(a)f'(b)}$ и принять аппроксимирующий минимум:

–если $0\le\gamma\le1$, то $\tilde{x}=a+\gamma(b-a)$;

–если $\gamma<0$, то $\tilde{x}=a$;

–если $\gamma>1$, то $\tilde{x}=b$.

Шаг_2. Проверить критерий окончания поиска: $\left|f'(x^*)\right|<\varepsilon$, или $\tilde{x}=a$, или $\tilde{x}=b$. В противном случае вернуться к шагу 1, используя новый интервал:

–если $f'(x^*)>0$, то $[a,x^*]$;

–если $f'(x^*)<0$, то $[x^*,b]$.

2 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

2.1. Изучить предлагаемые методы одномерной безусловной оптимизации.

2.2. Разработать программу, реализующую методы поиска (в соответствии с заданным вариантом), и найти точку минимума функции $f(x)$ на отрезке $[a,b]$.

Таблица 2.1 – Варианты заданий

№	Метод	Целевая функция	№	Метод	Целевая функция
1	*б,в,г,д,з*	$x^2 + 6 \cdot e^{0,15x}$	16	*а,в,г,е,ж*	$1,4x + e^{\lvert x-2 \rvert}$
2	*а,в,г,е,ж*	$x^2 + 4 \cdot e^{-0,25x}$	17	*а,б,д,е,з*	$x^2 + e^x$
3	*а,б,д,е,з*	$x^4 + 0,4 \cdot arctg5x$	18	*а,б,г,д,ж*	$\lvert x \rvert + e^{10x}$
4	*а,б,г,д,ж*	$x^4 - 1,5arctgx$	19	*а,в,г,е,з*	$10\cos x + e^x$
5	*а,в,г,е,з*	$x^2 + 8 \cdot e^{0,55x}$	20	*б,в,д,е,ж*	$-4x^2 + x^4$
6	*б,в,д,е,ж*	$-4x + e^{\lvert x-0,2 \rvert}$	21	*а,б,д,е,з*	$x^2 + e^{\lvert x-2 \rvert}$
7	*б,в,г,д,з*	$1,4x + e^{\lvert x-2 \rvert}$	22	*а,б,г,д,ж*	$x^4 + e^{x-2}$
8	*а,в,г,е,ж*	$x^2 + e^x$	23	*а,в,г,е,з*	$x^2 + 6 \cdot e^{0,15x}$
9	*а,б,д,е,з*	$\lvert x \rvert + e^{10x}$	24	*б,в,д,е,ж*	$x^2 + 4 \cdot e^{-0,25x}$
10	*а,б,г,д,ж*	$10\cos x + e^x$	25	*б,в,г,д,з*	$x^4 + 0,4 \cdot arctg5x$
11	*а,в,г,е,з*	$-4x^2 + x^4$	26	*а,в,г,е,ж*	$x^4 - 1,5arctgx$
12	*б,в,д,е,ж*	$x^2 + e^{\lvert x-2 \rvert}$	27	*а,б,д,е,з*	$x^2 + 8 \cdot e^{0,55x}$
13	*б,в,г,д,з*	$x^4 + e^{x-2}$	28	*а,б,г,д,ж*	$x^2 + e^{\lvert x-2 \rvert}$
14	*а,в,г,е,ж*	$\lvert x^3 \rvert + e^x$	29	*а,в,г,е,з*	$x^4 + e^{x-2}$
15	*а,б,д,е,з*	$7\cos x + e^{x+3}$	30	*б,в,д,е,ж*	$\lvert x^3 \rvert + e^x$

Методы одномерной безусловной оптимизации:

а) пассивный оптимальный алгоритм;

б) алгоритм блочного равномерного поиска;

в) алгоритм деления интервала пополам;

г) метод дихотомии;

д) метод золотого сечения;

е) метод Фибоначчи;

ж) метод касательных;

з) метод кубической интерполяции.

3 ТРЕБОВАНИЕ К ОТЧЕТУ

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Краткие теоретические сведения.
3. График функции в *MathCAD*.
4. Схемы алгоритмов.
5. Листинги основных частей программы.
6. Таблица результатов сравнения рассмотренных методов.
7. Заключение по результатам сравнения методов.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое экстремум функции.
2. Как аналитически найти экстремум функции.
3. Пассивный оптимальный алгоритм поиска минимума.
4. Блочный равномерный поиск.
5. Алгоритм деления интервала пополам.
6. Метод дихотомии.
7. Метод золотого сечения.
8. Метод Фибоначчи.
9. Метод касательных.
10. Метод кубической интерполяции.

Лабораторная работа №3
«Многомерная безусловная оптимизация» (7 часов)

Лабораторная работа №3.1 «Методы прямого поиска» (4 часа)
Лабораторная работа №3.2 «Методы многомерного поиска с использованием производных» (3 часа)

Цель работы: знакомство с методами многомерной безусловной оптимизации и сравнение эффективности применения этих методов для конкретных целевых функций.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Общая постановка задачи

В общем виде задача многомерной безусловной оптимизации формулируется как: $\min f(x)$, при $x \in X$, где $x = \{x^{(1)}, x^{(2)}, ..., x^{(n)}\}$ – точка в *n*-мерном пространстве $X = IR^n$.

Таким образом, целевая функция $f(x) = f(x^{(1)}, x^{(2)}, ..., x^{(n)})$ является функцией *n* аргументов. По аналогии с предыдущим анализом, будем решать задачу минимизации. Основная идея численных методов поиска минимума состоит в определении последовательности точек $\{x_k\}$, удовлетворяющих условию $f(x_0) > f(x_1) > ... > f(x_n)$. Методы построения таких последовательностей называются методами спуска. В этих методах точки последовательности $\{x_k\}$ вычисляются по формуле $x_{k+1} = x_k + \alpha_k p_k$, где p_k – направление спуска, α_k – длина шага в этом направлении, $k = 0,1,2,...,n$. Различные методы спуска отличаются друг от друга способами выбора направления спуска p_k и длины шага α_k вдоль этого направления.

Алгоритмы безусловной минимизации принято делить на классы, в зависимости от максимального порядка производных минимизируемой функции, вычисление которых предполагается. Так методы, использующие только значения целевой функции, относят к методам нулевого порядка или методами прямого поиска. Если в процессе поиска требуется вычисление первых производных целевой функции, то эти методы являются методами первого порядка. При использовании

для поиска минимума (максимума) вторых производных получаем методы второго порядка и т. д.

Основное достоинство методов нулевого порядка состоит в том, что они не требуют непрерывности целевой функции и существования производных. Однако прибегая к этим методам, надо быть уверенным в том, что метод другого типа применить нельзя, иначе можно дорого заплатить потерями машинного времени. Кроме того, общим их недостатком является плохая сходимость.

Простейшим методом поиска является метод покоординатного спуска. Рассмотрим функцию двух переменных (рисунок 1.1), минимум которой расположен точке $x = (x^{(1)*}, x^{(2)*})$. Из точки x_0 производим поиск минимума вдоль направления оси $x^{(1)}$ и, таким образом, находим точку x_1, в которой касательная к линии постоянного уровня параллельна оси. Затем, производя поиск из точки x_1 в направлении оси $x^{(2)}$, получаем точку x_2, производя поиск параллельно оси $x^{(1)}$, получаем точку x_3, затем x_4 и т. д. Таким образом, через некоторое число итераций находим точку минимума. Данная идея может быть применена и для функций *n* переменных.

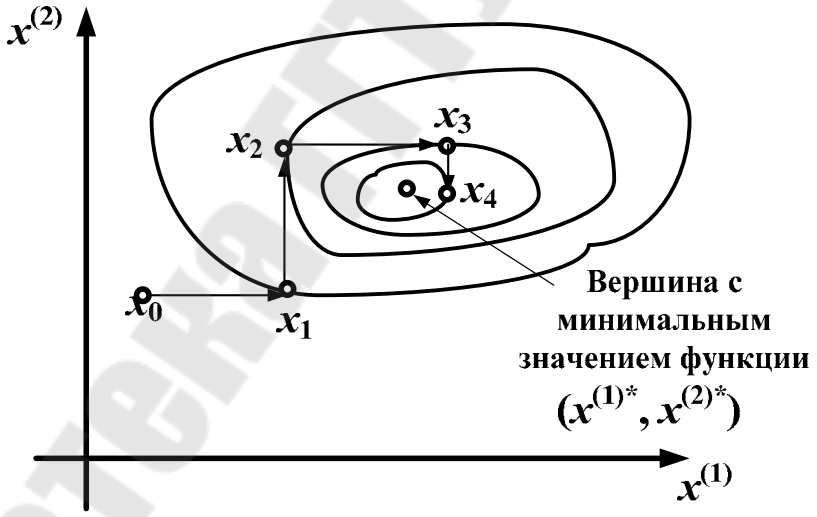

Рисунок 1.1 – Иллюстрация метода покоординатного спуска

Однако на практике данный метод оказался слишком медленным. Поэтому были разработаны более сложные методы, использующие больше информации на основании уже полученных значений функции. Среди методов прямого поиска рассмотрим поиск по симплексу (по многограннику) с модификацией Нелдера–Мида и метод Хука–Дживса.

1.1.2 Метод поиска по симплексу

Под симплексом понимается *n*-мерный выпуклый многогранник *n*-мерного пространства, имеющий $n+1$ вершину. Для $n=2$ это треугольник, а при $n=3$ это тетраэдр.

Идея метода состоит в сравнении значений функции в $n+1$ вершинах симплекса и перемещении симплекса в направлении лучшей точки. В рассматриваемом методе симплекс перемещается с помощью операций отражения. При анализе будем использовать следующие обозначения: $x_0(k)$, $x_1(k)$, ..., $x_n(k)$ – вершины симплекса (вектора в *n*-мерном пространстве), *к* – номер итерации.

Наименьшее число точек содержит регулярный симплекс (равноотстоящие вершины). В случае двух переменных (плоскость) это равносторонний треугольник, в трехмерном пространстве – тетраэдр. Метод наиболее эффективен при $n \le 6$.

Алгоритм начинается с построения регулярного симплекса в пространстве *n* переменных. При этом определяется вершина, которой соответствует наибольшее значение функции (рисунок 1.2). Эта вершина проецируется через центр тяжести остальных вершин в новую точку, которая будет вершиной нового симплекса. Итерации выполняются до тех пор, пока не будет накрыта точка минимума, либо не начнется циклическое движение (движение по двум или более симплексам).

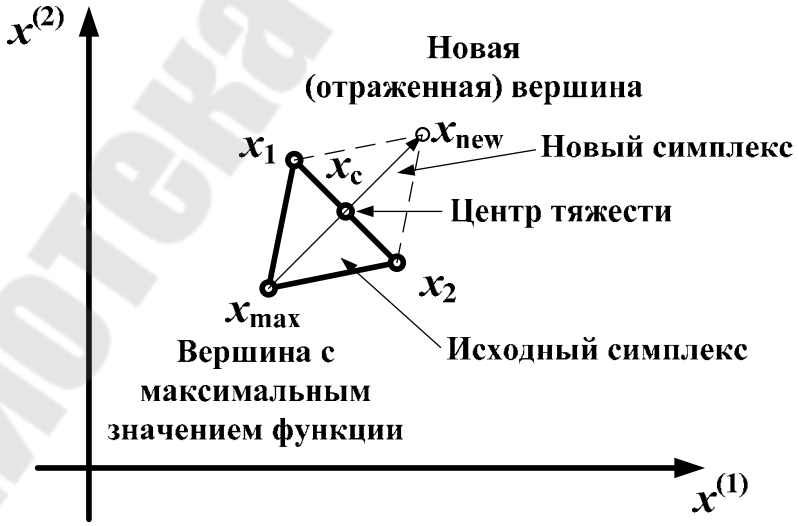

Рисунок 1.2 – Метод поиска по симплексу

Схема алгоритма

*Шаг_*1. Построение начального симплекса.

Задается начальная точка $x_0(0)$ и длина ребра симплекса *a* (если $\alpha = 1$, то ребра единичной длины). Формируются остальные вершины симплекса: $x_i(k) = x_0(k) + \alpha e_i$, где e_i – единичные векторы, $k = 0,1,2,...,n$.

*Шаг_*2. Определение направления улучшения решения.

Вычисляются значения целевой функции в каждой точке симплекса (на *k*-й итерации), среди которых находится максимальное значение. Пусть $f(x_i(k)) \geq f(x_{\max}(k))$, для $i = 0,1,2,...,n$. Определяется центр тяжести всех точек, исключая точку $x_{\max}(k)$:

$$c_k = \frac{\sum_i x_i(k)}{n}. \qquad (1.1)$$

Тогда направление улучшения решения определяется вектором $c_k - x_{\max}(k)$.

*Шаг_*3. Построение отраженной точки.

Точки прямой, проходящей через c_k и $x_{\max}(k)$, задаются формулой: $x = x_{\max}(k) + \alpha(c_k - x_{\max}(k))$.

При $\alpha = 0$ получим исходную точку $x_{\max}(k)$, при $\alpha = 1$ получаем точку c_k, при $\alpha = 2$ получаем новую вершину. Таким образом:

$x_{new}(k) = 2c_k - x_{\max}(k)$.

*Шаг_*4. Построение нового симплекса.

Вычисляем значение функции в новой точке. При этом возможны два случая: $f(x_{new}(k)) < f(x_{\max}(k))$ или $f(x_{new}(k)) \geq f(x_{\max}(k))$. В первом случае вершину $x_{\max}(k)$ заменяем на $x_{new}(k)$ и переходим на следующую итерацию.

Во втором случае производится пропорциональное уменьшение симплекса. Для этого находят вершину с минимальным значением целевой функции, делают ее базовой и строят новый симплекс с меньшими (например, в 2 раза) размерами.

*Шаг_*5. Проверка сходимости.

Если условие:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(x_i(k+1)) - f(x_0(k+1)))^2} \leq \varepsilon$$

выполняется, то поиск заканчивается и полагается $\tilde{x} = x_0(k+1),\ \tilde{y} = f(x_0(k+1))$.

В противном случае $k = k + 1$ и переход к шагу 2.

1.1.3 Метод Нелдера-Мида

Метод Нелдера-Мида или метод деформируемого симплекса обладает большей общностью и позволяет учитывать локальные свойства поверхности целевой функции. Симплексы вытягиваются в направлении наклона поверхности, их оси поворачиваются при встрече с оврагом на поверхности целевой функции, вблизи минимума они сжимаются. В данном методе симплекс перемещается с помощью трех основных операций над симплексом: нормальное отражение (рис.1.3,а), растяжение (рис.1.3,б) и сжатие (рис.1.3,в,г).

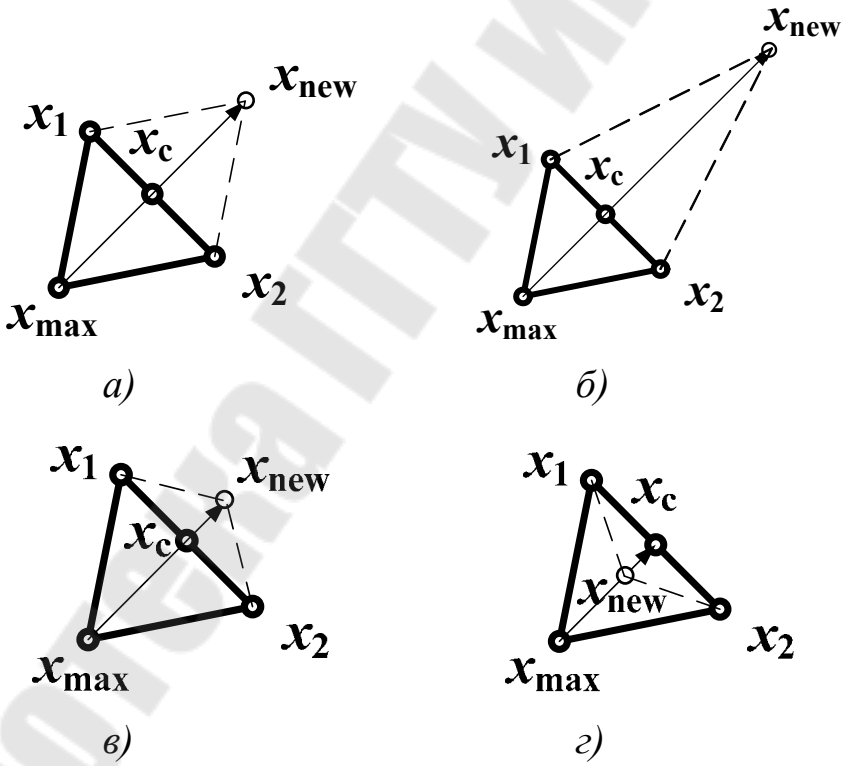

Рисунок 1.3 – Основные операции над симплексом

Схема алгоритма.

*Шаг*_1. Построение начального симплекса.

Задается начальная точка $x_0(0)$ и длина ребра симплекса $\alpha = 1$. Формируются остальные вершины симплекса: $x_i^{\,0} = x_0^{\,(0)} + \alpha e_i$, где e_i – единичные векторы, $i = 1,2,...,n$.

Шаг_2. Определение направления улучшения решения.

Вычисляются значения целевой функции в каждой точке симплекса (на *k*-й итерации), среди которых находятся максимальное и минимальное значение. Пусть $f(x_{\min}(k)) \leq f(x_i(k)) \leq f(x_{\max}(k))$. Определяется центр тяжести всех точек (1.1), исключая точку $x_{\max}(k)$. Тогда направление улучшения решения определяется вектором $c_k - x_{\max}(k)$.

Шаг_3. Построение отраженной точки.

Отражаем вершину $x_{\max}(k)$ с максимальным значением целевой функции через центр тяжести остальных точек и находим новую точку $x_{new}(k)$, координаты которой $x_{new}(k) = c_k - \alpha(c_k - x_{\max}(k))$, где α – коэффициент отражения.

Шаг_4. Построение нового симплекса.

Вычисляем значение функции в новой точке. При этом возможны три случая:

– $f(x_{new}(k)) < f(x_{\min}(k))$,

– $f(x_{new}(k)) > f(x_{\max}(k))$,

– $f(x_{\min}(k)) \leq f(x_{new}(k)) \leq f(x_{\max}(k))$.

В первом случае отражённая точка является точкой с наилучшим значением целевой функции. Поэтому направление отражение является перспективным и можно попытаться растянуть симплекс в этом направлении. Для этого строиться точка:

$x'_{new}(k) = c_k + \beta(x'_{new}(k) - c_k)$,

где $\beta > 1$ – коэффициент расширения.

Если $f(x'_{new}(k)) < f(x_{new}(k))$, то вершину $x_{\max}(k)$ заменяем на $x'_{new}(k)$, в противном случае $x_{\max}(k)$ заменяем на $x_{new}(k)$, и переходим на следующую итерацию.

Во втором случае отражённая точка является точкой с наихудшим значением целевой функции. Поэтому в этом случае производится сжатие симплекса. Для этого строится точка $x'_{new}(k)$, координаты которой определяются следующим образом:

$x'_{new}(k) = c_k + \gamma(x_{\max}(k) - c_k)$, если $f(x_{\max}(k)) \leq f(x_{new}(k))$,

или

$x'_{new}(k) = c_k + \gamma(x_{new}(k) - c_k)$, если $f(x_{\max}(k)) > f(x_{new}(k))$,

где $0 < \gamma < 1$ – коэффициент сжатия.

Если $f(x'_{new}(k)) < \{f(x_{\max}(k)), f(x_{new}(k))\}$ вершину $x_{\max}(k)$ заменяем на $x'_{new}(k)$, в противном случае вершинам $x_i(k+1)$ $(i = 0,1,2,...,n)$ присваиваются средние значения $\tilde{x} = \frac{x_i(k) + x_{\min}(k)}{2}$, и переходят на следующую итерацию.

В третьем случае $x_{\max}(k)$ заменяем на $x_{new}(k)$, и переходим на следующую итерацию.

*Шаг_*5. Проверка сходимости.

Если условие $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(x_i(k+1) - f(x_0(k+1)))^2} \le \varepsilon$ выполняется, то поиск минимума заканчивается и полагается $\tilde{x} = x_0(k+1), \tilde{y} = f(x_0(k+1))$. В противном случае $k = k+1$ и происходит переход к шагу 2.

В качестве параметров алгоритма (коэффициент отражения, растяжения и сжатия) целесообразно выбирать следующие значения: $\alpha = 1$, $\beta = 2$, $\gamma = 0{,}5$.

1.1.4 Метод Хука-Дживса

Метод Хука-Дживса (метод конфигураций) был разработан в 1961 году, но до сих пор является одним из самых эффективных методов многомерной безусловной оптимизации. Основная идея заключается в использовании исследующего поиска вокруг базисной точки, за которым в случае успеха следует поиск по образцу. Таким образом, алгоритм включает в себя два основных этапа поиска:

–исследуется окрестность выбранной базисной точки, в результате находится приемлемое направление спуска;

–в выбранном направлении находится точка с наименьшим значением целевой функции, то есть новая базисная точка.

Процедура продолжается пока в окрестностях базисных точек удается находить приемлемые направления спуска.

Схема алгоритма

*Шаг_*1. Задается начальное приближение (первая базисная точка) $x_0 = \{x_0^{(1)}, x_0^{(2)}, ..., x_0^{(n)}\}$, начальный *Шаг_h* для поиска направления спуска, точность решения δ (предельное значение для шага *h*), номер итерации $k = 0$.

*Шаг_*2. Исследующий поиск.

Определяется направление минимизации целевой функции $f(x) = f(x^{(1)}, x^{(2)}, ..., x^{(n)})$ в базисной точке $x_0 = \{x_0^{(1)}, x_0^{(2)}, ..., x_0^{(n)}\}$. Для этого последовательно дают приращение переменным $x^{(j)}$ в точке x_k.

Присвоим $z = x_k$. Для всех j ($j = 1,2,...,n$) циклически даем приращение переменным $x^{(j)}$ и формируем новое значение *z*:

– $z^{(j)} = x_k^{(j)} + h$, если $f(z) < f(x_k)$,

– $z^{(j)} = x_k^{(j)} - h$, если $f(z) > f(x_k)$,

– $z^{(j)} = x_k^{(j)}$, если $f(z) = f(x_k)$.

*Шаг_*3. Оценка результатов исследующего поиска.

Если не определилось подходящее направление (то есть $z = x_k$), то обследование окрестности базисной точки x_k повторяется с меньшим шагом *h* (например, $h = h/2$).

Проверяется текущее значение шага. Если $h > \delta$, то перейти к шагу 2, то есть повторить обследование точки x_k. Если $h \le \delta$, то поиск заканчивается, так как достигнуто предельное значение для шага *h*, и найти приемлемое направление спуска не удалось. В этом случае полагается $\tilde{x} = x_k$, $\tilde{y} = f(x_k)$.

*Шаг_*4. Поиск по образцу.

Если в результате исследующего поиска найдена точка с меньшим значением целевой функции ($z \neq x_k$), то требуется найти новую базисную точку в направлении вектора $(z - x_k)$: $x_{k+1} = x_k + \lambda(z - x_k)$, где λ – коэффициент «ускорения поиска». Определяется такое значение $\lambda = \lambda_k$, при котором достигается наименьшее значение целевой функции в выбранном направлении, то есть функции $\varphi(\lambda) = f(x_k + \lambda(z - x_k))$. В зависимости от способа выбора λ_k возможны следующие варианты метода:

– $\lambda_k = const$ – постоянная для всех итераций;

–задается начальное $\lambda_0 = \lambda$, а далее $\lambda_k = \lambda_{k+1}$, если $f(x_{k+1}) < f(x_k)$, иначе дробим λ_k, пока не выполнится это условие;

$-\lambda_k$ определяется решением задачи одномерной минимизации функции $\varphi(\lambda)$.

В результате этого шага определяется новая базисная точка $x_{k+1} = x_k + \lambda(z - x_k)$, и поиск оптимального решения повторяется с шага 2, при этом $k = k + 1$.

1.2 Многомерный поиск с использованием производных

1.2.1 Градиентные методы

Как известно, градиент функции в некоторой точке x_k направлен в сторону наискорейшего локального возрастания функции и перпендикулярен линии уровня (поверхность постоянного значения функции $f(x)$, проходящей через точку x_k). Вектор, противоположный градиенту $f'(x_k)$, называется антиградиентом, который направлен в сторону наискорейшего убывания функции $f(x)$. Выбирая в качестве направления спуска p_k антиградиент $-f'(x_k)$ в точке x_k, приходим к итерационному процессу вида: $x_{k+1} = x_k - \alpha_k f'(x_k)$, где $\alpha_k > 0$, $k = 0,1,2,...$. В координатной форме этот процесс записывается следующим образом: $x_{k+1}^{(i)} = x_k^{(i)} - \alpha_k \frac{\partial f}{\partial x^i}(x_k)$, где $i = 1,2,...,n$.

Все итерационные процессы, в которых направление движения на каждом шаге совпадает с антиградиентом функции, называются градиентными методами. Они отличаются друг от друга только способом выбора шага α_k. Существует много различных способов выбора α_k, но наиболее распространены: метод с постоянным шагом, метод с дроблением шага и метод наискорейшего спуска.

Градиентный метод с постоянным шагом

Основная проблема в градиентных методах – это выбор шага α_k. Достаточно малый шаг α_k обеспечивает убывание функции, то есть выполнение неравенства $f(x_k - \alpha_k f'(x_k)) < f(x_k)$, но может привести к неприемлемо большому количеству итераций, необходимых для достижения точки минимума. С другой стороны, слишком большой шаг может вызвать неожиданный рост функции (невыполнение условия убывания) либо привести к колебаниям около точки

минимума. Однако проверка условия убывания на каждой итерации является довольно трудоемкой, поэтому в методе градиентного спуска с постоянным шагом задают $\alpha = \alpha_k$ постоянным и достаточно малым, чтобы можно было использовать этот шаг на любой итерации. При этом приходится мириться с возможно большим количеством итераций. Положительным фактором является то, что трудоемкость каждой итерации, минимальна (вычисляется только градиент $f'(x_k)$).

Схема алгоритма

*Шаг_*1. Задаются начальное приближение x_0, постоянный шаг α, условия останова алгоритма ε. Вычисляется значение градиента $f'(x_k)$ – направление поиска. Присваивается $k = 0$.

*Шаг_*2. Определяется точка очередного эксперимента $x_{k+1} = x_k - \alpha f'(x_k)$, или, в координатной форме: $x_{k+1}^{(i)} = x_k^{(i)} - \alpha \dfrac{\partial f}{\partial x^i}(x_k^{(i)},...,x_k^{(n)})$, где $i = 1,2,...,n$.

*Шаг_*3. Вычисляется значение градиента $f'(x_{k+1})$, или, в координатной форме: $\{\dfrac{\partial f}{\partial x^{k+1}}(x_{k+1}^{(1)},...,x_{k+1}^{(n)}),...,\dfrac{\partial f}{\partial x^n}(x_{k+1}^{(1)},...,x_{k+1}^{(n)})\}$.

*Шаг_*4. Если $\left\|f'(x_{k+1})\right\| \le \varepsilon$, то поиск заканчивается, при этом $\tilde{x} = x_{k+1}, \tilde{y} = f(x_{k+1})$, иначе $k = k + 1$ и переход к шагу 2.

Градиентный метод с дроблением шага

В методе градиентного спуска с дроблением шага величина шага α_k выбирается так, чтобы выполнялось неравенство $f(x_k - \alpha_k f'(x_k)) - f(x_k) \le -\delta\alpha_k \left\|f'(x_{k+1})\right\|^2$, где $0 \le \delta \le 1$ – произвольно выбранная постоянная (одна и та же для всех итераций). Это требование на выбор шага α_k более жесткое, чем условие убывания, но имеет тот же смысл: функция должна убывать от итерации к итерации. Однако при выполнении неравенства функция будет уменьшаться на гарантированную величину, определяемую правой частью неравенства.

Процесс выбора шага протекает следующим образом. Выбираем число $\alpha>0$, одно и то же для всех итераций. На *k*-й итерации проверяем выполнение неравенства при $\alpha_k = \alpha$. Если оно выполнено, полагаем $\alpha_k = \alpha$ и переходим к следующей итерации. Если нет, то шаг α_k

дробим, например, уменьшаем каждый раз в два раза, до тех пор, пока оно не выполнится.

Схема алгоритма

*Шаг*_1. Задаются x_0, ε, δ и начальное значение шага α, $k = 0$. Вычисляется значение градиента $f'(x_k)$ – направление поиска.

*Шаг*_2. Проверяется условие $f(x_k - \alpha f'(x_k)) \le -\delta\alpha_k \left\| f'(x_k) \right\|^2$. Если оно выполняется, то переходим к шагу 3, иначе дробим значение α ($\alpha = \alpha / 2$) и повторяем *Шаг*_2.

*Шаг*_3. Определяется точка очередного эксперимента: $x_{k+1} = x_k - \alpha f'(x_k)$.

*Шаг*_4. Вычисляется значение градиента в точке x_{k+1}: $f'(x_{k+1})$.

*Шаг*_5. Проверка сходимости.

Если $\left\| f'(x_{k+1}) \right\| \le \varepsilon$, то поиск заканчивается, при этом $\tilde{x} = x_{k+1}, \tilde{y} = f(x_{k+1})$. Иначе $k = k + 1$ и переход к шагу 2.

Метод наискорейшего спуска.

В градиентном методе с постоянным шагом величина шага, обеспечивающая убывание функции $f(x)$ от итерации к итерации, оказывается очень малой, что приводит к необходимости проводить большое количество итерации для достижения точки минимума. Поэтому методы спуска с переменным шагом являются более экономными. Алгоритм, на каждой итерации которого шаг α_k выбирается из условия минимума функции $f(x)$ в направлении движения, то есть

$$f(x_k - \alpha_k f'(x_k)) = \min_{\alpha \ge 0} f(x_k - \alpha f'(x_k)),$$

называется методом наискорейшего спуска.

Реализация метода наискорейшего спуска предполагает решение на каждой итерации довольно трудоемкой вспомогательной задачи одномерной минимизации. Как правило, метод наискорейшего спуска дает выигрыш в числе машинных операций, поскольку обеспечивает движение с самым выгодным шагом. Решение задачи одномерной минимизации связано с дополнительными вычислениями только самой функции $f(x)$, тогда как основное машинное время тратится на вычисление ее градиента $f'(x_k)$. Следует иметь в виду, что одномерную минимизацию можно производить любым методом одномерной оптимизации, что порождает различные варианты метода наискорейшего спуска.

Схема алгоритма

*Шаг_*1. Задаются x_0, ε, $k = 0$. Вычисляется градиент $f'(x_0)$ – направление поиска.

*Шаг_*2. Определяется точка очередного эксперимента: $x_{k+1} = x_k - \alpha f'(x_k)$, где α_k – минимум задачи одномерной минимизации: $\alpha_k = \arg\min_{\alpha \geq 0} f(x_k - \alpha f'(x_k))$.

*Шаг_*3. Вычисляется значение градиента в точке x_{k+1}: $f'(x_{k+1})$.

*Шаг_*4. Если $\|f'(x_{k+1})\| \leq \varepsilon$, то поиск заканчивается, при этом $\tilde{x} = x_{k+1}, \tilde{y} = f(x_{k+1})$. Иначе $k = k + 1$ и переход к шагу 2.

1.2.2 Метод покоординатного спуска

Желание уменьшить объем вычислительной работы, требуемой для осуществления одной итерации метода наискорейшего спуска, привело к созданию методов покоординатного спуска. Пусть $x_0 = \{x_0^{(1)}, x_0^{(2)}, ..., x_0^{(n)}\}^T$ – начальное приближение. Вычислим частную производную по первой координате и примем: $x_1 = x_0 - \alpha_0 \frac{\partial f}{dx^1}(x_0) e_1$, где $e_1 = \{1,0,...,0\}^{\mathrm{T}}$ – единичный вектор оси $x^{(1)}$. Следующая итерация состоит в вычислении точки x_2 по формуле: $x_2 = x_0 - \alpha_0 \frac{\partial f}{dx^2}(x_0) e_2$, где $e_2 = \{0,1,...,0\}^{\mathrm{T}}$ – единичный вектор оси $x^{(2)}$ и т. д. Таким образом, в методах покоординатного спуска поиск происходит по ломаной линии, состоящей из отрезков прямых, параллельных координатным осям (см. рис.1.1).

Спуск по всем координатам составляет одну «внешнюю» итерацию. Пусть k – номер очередной внешней итерации, а j – номер той координаты, по которой производится спуск. Тогда формула, определяющая следующее приближение к точке минимума, имеет вид: $x_{kn+j} = x_{kn+j-1} - \alpha_{kn+j-1} \frac{\partial f}{\partial x^j}(x_{kn+j-1}) e_j$, где $k = 0,1,2,...,\ j = 1,2,...,n$.

В координатной форме формула имеет вид: $x_{kn+j}^{(i)} = x_{kn+j-1}^{(i)}$, если $i = j$, и $x_{kn+j}^{(i)} = x_{kn+j-1}^{(i)} - \alpha_{kn+j-1} \frac{\partial f}{\partial x^j}(x_{kn+j-1})$, если $i \neq j$.

После $j = n$ итераций счетчик внешних итераций k увеличивается на единицу, а j принимает значение равное единице.

Величина шага α_k выбирается на каждой итерации аналогично тому, как это делается в градиентных методах. Например, если $\alpha_k = \alpha$, то имеем покоординатный спуск с постоянным шагом.

Схема алгоритма покоординатного спуска с постоянным шагом

*Шаг*_1. При $k = 0$ вводятся исходные данные x_0, ε, α.

*Шаг*_2. Осуществляется циклический покоординатный спуск по j ($j = 1,2,...,n$) из точки x_{kn} по формуле:

$$x_{kn+j} = x_{kn+j-1} - \alpha \frac{\partial f}{\partial x^j}(x_{kn+j-1}) e_j \ .$$

*Шаг*_3. Если $\left\|(x_{(k+1)n} - x_{kn})\right\| \le \varepsilon$, то поиск минимума заканчивается – $\tilde{x} = x_{(k+1)n}$, $\tilde{y} = f(x_{(k+1)n})$. Иначе $k = k + 1$ и переход к шагу 2.

Если же шаг α_k выбирается из условия минимума функции $\varphi(\alpha) = f(x_{kn+j-1} - \alpha \frac{\partial f}{\partial x^j}(x_{kn+j-1}) e_j)$, то мы получаем аналог метода наискорейшего спуска, называемый методом Гаусса – Зейделя.

Схема алгоритма Гаусса – Зейделя

*Шаг*_1. При $k = 0$ вводятся исходные данные x_0, ε.

*Шаг*_2. Осуществляется циклический покоординатный спуск по j ($j = 1,2,...,n$) из точки x_{kn} по формуле

$$x_{kn+j} = x_{kn+j-1} - \alpha_{kn+j-1} \frac{\partial f}{\partial x^j}(x_{kn+j-1}) e_j \ ,$$

где α_{kn+j-1} является решением задачи одномерной минимизации функции $\varphi(\alpha) = f(x_{kn+j-1} - \alpha \frac{\partial f}{\partial x^j}(x_{kn+j-1}) e_j)$.

*Шаг*_3. Если $\left\|(x_{(k+1)n} - x_{kn})\right\| \le \varepsilon$, то поиск минимума заканчивается, иначе $k = k + 1$ и переход к шагу 2.

1.2.3 Методы оврагов

Градиентные методы медленно сходятся в тех случаях, когда поверхности уровня целевой функции $f(x)$ сильно вытянуты. Этот

факт известен в литературе как «эффект оврагов». Суть эффекта в том, что небольшие изменения одних переменных приводят к резкому изменению значений функции – эта группа переменных характеризует «склон оврага», а по остальным переменным, задающим направление «дно оврага», функция меняется незначительно. На рисунке 1.4 изображены линии уровня «овражной» функции траектория градиентного метода характеризуется довольно быстрым спуском на «дно оврага», и затем медленным зигзагообразным движением в точку минимума.

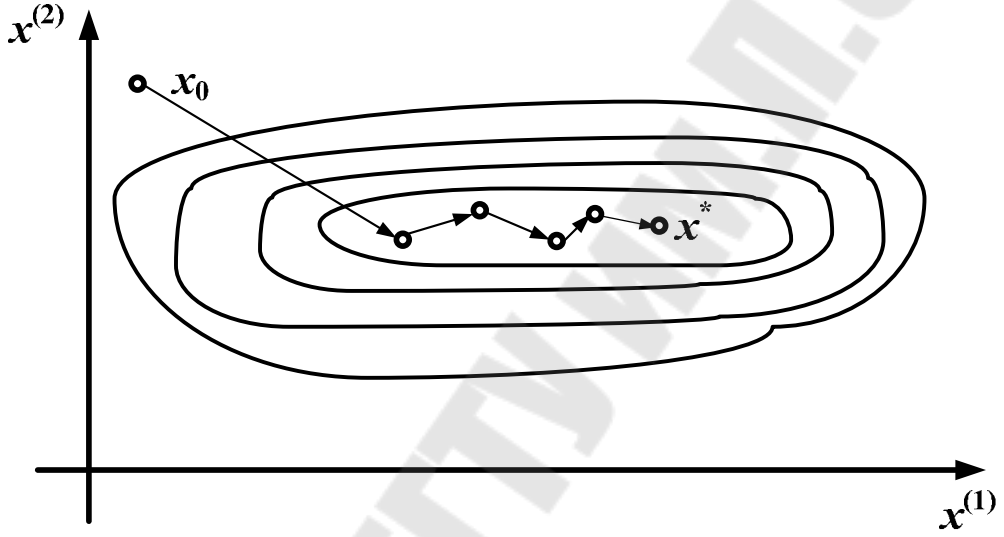

Рисунок 1.4 – Пример «эффекта оврагов»

Существуют различные подходы для определения точки минимума функции $f(x)$ в овражной ситуации. Большинство из этих методов основаны на эвристических подходах. Их можно применять в ситуациях, когда применение более совершенных методов невозможно или нецелесообразно, например, значение целевой функции вычисляется со значительными погрешностями, информация о ее свойствах недостаточна, и т. д. Эти методы просты в реализации и довольно часто применяются на практике, позволяя в ряде случаев получить удовлетворительное решение задачи.

Рассмотрим эвристическую схему, которая содержит два основных этапа. Оба этапа представляют собой аналоги градиентного спуска с постоянным шагом. Только вместо градиента $f'(x_k)$ используется вектор $g(x)$, формируемый из координат $f'(x_k)$, но на каждом из этапов по разным правилам.

На первом этапе задается малое число $\delta_1 << 1$ и используется градиентный спуск, где вместо градиента $f'(x_k)$ берется вектор $g(x) = \{g^{(1)}(x),...,g^{(n)}(x)\}$, который определяется следующим образом:

$$g^{(i)}(x) = \frac{\partial f}{\partial x^i}(x), \text{ если } |\frac{\partial f}{\partial x^i(x)}| > \delta_1,$$

$$g^{(i)}(x) = 0, \text{ если } |\frac{\partial f}{\partial x^i(x)}| \le \delta_1, \qquad (1.2)$$

где $i = 1,2,...n$.

Таким образом, спуск производится лишь по тем переменным, в направлении которых производная целевой функции достаточно велика. Это позволяет быстро спуститься на «дно оврага». Спуск продолжается до тех пор, пока метод не зациклится, то есть до тех пор, пока каждая следующая итерация позволяет найти точку, в которой значение функции меньше, чем значение, найденное в предыдущей итерации. После этого переходим к следующему этапу.

На втором этапе задается некоторое большое число $\delta_2 >> 1$ и используется процедура спуска, где вместо градиента $f'(x_k)$ берется вектор $g(x) = \{g^{(1)}(x),...,g^{(n)}(x)\}$, который определяется следующим образом:

$$g^{(i)}(x) = \frac{\partial f}{\partial x^i}(x), \text{ если } |\frac{\partial f}{\partial x^i(x)}| < \delta_2,$$

$$g^{(i)}(x) = 0, \text{ если } |\frac{\partial f}{\partial x^i(x)}| \ge \delta_2, \qquad (1.3)$$

где $i = 1,2,...n$.

В этом случае перемещение происходит по «берегу» оврага вдоль его «дна». Как и на первом этапе, спуск продолжается до тех пор, пока метод не зациклится.

После выполнения первого и второго этапов принимается решение о завершении работы или продолжении. Для этого сравнивается норма разности предыдущей точки, то есть точки, которую мы имели до применения первого и второго этапов, с текущей точкой, то есть полученной после применения с точностью решения задачи ε_1. Если эта норма меньше ε_1 и норма градиента в текущей точке меньше ε_3, то поиск заканчивается и последняя вычисленная точка принимается за приближенное решение задачи. Иначе для текущей точки вновь повторяем первый и второй этапы и т. д.

Схема алгоритма

*Шаг_*1. Задаются x_0, ε_1, ε_3, δ_1, δ_2, α_1 – постоянный шаг первого этапа и α_2 – постоянный шаг второго этапа ($\alpha_1 < \alpha_2$), $k = 0$.

*Шаг_*2. Первый этап.

Из точки $x_к$ осуществляется спуск на «дно оврага» с постоянным шагом α_1. При спуске очередная точка вычисляется по формуле: $x_{j+1} = x_j - \alpha_1 g(x_j)$, где $g(x) = \{g^{(1)}(x),...,g^{(n)}(x)\}$, $g^{(i)}(x)$ вычисляются по формуле (3.2). Предположим, что процесс остановится в точке x_l. Тогда переходим ко второму этапу.

*Шаг_*3. Второй этап.

Из точки x_l осуществляется спуск вдоль «дна оврага» с постоянным шагом α_2. При спуске очередная точка вычисляется по формуле $x_{j+1} = x_j - \alpha_2 g(x_j)$, где $g(x) = \{g^{(1)}(x),...,g^{(n)}(x)\}$, а $g^{(i)}(x)$ вычисляются по формуле (3.3). Предположим, что процесс остановится в точке x_m.

*Шаг_*4. Если $\|x_k - x_m\| \le \varepsilon_1$ и $\|f'(x_k)\| \le \varepsilon_3$, то полагаем $\tilde{x} = x_m, \tilde{y} = f(x_m)$ и поиск минимума заканчивается. Иначе $k = m$ и переходим к шагу 2.

Метод Гельфанда

Следующая эвристическая схема, предложенная И.М. Гельфандом, состоит в следующем. Пусть x_0 и $\tilde{x}_0$ – две произвольные близкие точки. Из x_0 совершают обычный градиентный спуск с постоянным шагом и после нескольких итераций с малым шагом α попадают в точку u_0. То же самое делаем для точки $\tilde{u}_0$, получая точку $\tilde{x}_1$. Две точки u, $\tilde{x}_0$ лежат в окрестности «дна оврага». Соединяя их прямой, делаем «большой шаг» λ в полученном направлении, перемещаясь «вдоль дна оврага» (шаг λ называют овражным шагом). В результате получаем точку x_1. В ее окрестности выбираем точку $\tilde{u}_0$ и повторяем процедуру.

Схема алгоритма

*Шаг_*1. Вводятся начальное приближение x_0, точность решения ε_1 и ε_3, шаг α для градиентного спуска, начальное значение λ для овражного шага, $k = 0$. Из точки x_0 осуществляется градиентный

спуск с постоянным шагом α на дно оврага. В результате получается точка u_0.

Шаг_2. В окрестности x_k берется точка $\tilde{x}_k$ и из нее осуществляется градиентный спуск. В результате получается точка $\tilde{u}_k$.

Шаг_3. Новая точка x_{k+1} определяется следующим образом. По формуле $x'_{k+1} = u_k + \lambda \dfrac{(\tilde{u}_k - u_k)}{\|\tilde{u}_k - u_k\|}$ при $f(\tilde{u}_k) < f(u_k)$, или $x'_{k+1} = \tilde{u}_k + \lambda \dfrac{(u_k - \tilde{u}_k)}{\|u_k - \tilde{u}_k\|}$ при $f(\tilde{u}_k) > f(u_k)$. Из нее осуществляем градиентный спуск и получаем точку u'_{k+1}. Если $f(u'_{k+1}) > f(u_k)$, то полагаем $x_{k+1} = x'_{k+1}$ и $u_{k+1} = u'_{k+1}$. Иначе уменьшаем овражный шаг λ (например, в два раза – $\lambda = \lambda / 2$) и повторяем *Шаг_3*.

Шаг_4. Если $\|u_{k+1} - u_k\| \le \varepsilon_1$ и $\|f'(u_{k+1})\| \le \varepsilon_3$, то полагаем $\tilde{x} = u_{k+1}$, $\tilde{y} = f(u_{k+1})$ и поиск минимума заканчивается. Иначе $k = k + 1$ и переходим к шагу 2.

2 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Изучить изложенные методы многомерной безусловной оптимизации:

ЛР 3.1 – Методы прямого поиска.

ЛР 3.2 – Методы многомерного поиска с использованием производных.

2. Составить программы реализующие методы многомерной безусловной минимизации и найти точку минимума целевой функции $f(x) = f(x^{(1)}, x^{(2)})$ с заданной точностью ε указанными методами. Начальное приближение x_0 и точность ε приведены в таблице 2.1. Сравнить результаты, полученные разными методами для одной и той же целевой функции (в частности, сравнить число вычислении целевой функции, понадобившихся для получения заданной точности). Для каждого применяемого метода построить траекторию промежуточных точек, получаемых на очередных шагах метода и сходящихся к точке минимума.

3. Оформить отчет о выполнении задания с приведением условия задачи, алгоритмов и программ, указанных в задании методов минимизации, графиков траекторий промежуточных приближений

(файл – *MathCAD*), таблицы результатов сравнения рассмотренных методов, заключения по результатам сравнения методов.

Целевая функция зависит от двух аргументов:

$$f(x) = (x_1 - a)^2 + (x_2 - b)^2 + e^{c \cdot x_1^2 + d \cdot x_2}$$

Таблица 2.1 – Варианты задания

№	Целевая функция				Начальное приближение	Точность решения
	a	*b*	*c*	*d*		
1	1	-1,4	0,01	0,11	(1;0)	0,0001
2	2	-1,3	0,04	0,12	(0;1)	0,00005
3	10	-0,5	0,94	0,2	(0;0)	0,0001
4	15	0	1,96	0,25	1,96	0,025
5	3	-1,2	0,02	1,3	(0;-1)	0,00005
6	11	-0,4	1	0,21	(-1;0)	0,0001
7	10	-1	1	2	(1;0)	0,0003
8	15	-0,5	2,25	2,5	(0;0)	0,0002
9	20	0,4	0,3	0,3	(0;-1)	0,0001
10	25	0,9	0,35	0,35	(1;0)	0,0004
11	15	-1	2	2,5	(0;0)	0,0002
12	7	0,4	0,1	0,2	(0;-1)	0,0001
13	5	0,3	0,35	0,35	(1;0)	0,0001
14	20	3	1,96	0,25	1,96	0,005
15	3	-1,2	0,02	1,3	(0;-1)	0,00005
16	11	-0,4	1	0,21	(-1;0)	0,0001
17	10	-1	1	2	(1;0)	0,0003
18	15	-0,5	2,25	2,5	(0;0)	0,0002
19	20	0,4	0,3	0,3	(0;-1)	0,0001
20	3	-1,2	0,02	1,3	(0;-1)	0,00005
21	25	0,9	0,35	0,35	(1;0)	0,0004
22	15	-1	2	2,5	(0;0)	0,0002
23	7	0,4	0,1	0,2	(0;-1)	0,0001
24	5	0,3	0,35	0,35	(1;0)	0,0001
25	20	3	1,96	0,25	1,96	0,005
26	3	-1,2	0,02	1,3	(0;-1)	0,00005
27	10	-0,5	0,94	0,2	(0;-1)	0,0001
28	15	0	1,96	0,25	(1;0)	0,0001
29	3	-1,2	0,02	1,3	1,96	0,005
30	15	-1,2	0,01	1,3	(0;-1)	0,005

3 ТРЕБОВАНИЕ К ОТЧЕТУ

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Краткие теоретические сведения.
3. График функции в MathCAD.
4. Схемы алгоритмов.
5. Листинги основных частей программы.
6. Таблица результатов сравнения рассмотренных методов.
7. Заключение по результатам сравнения методов.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое экстремум функции многих переменных?
2. Как аналитически найти экстремум функции многих переменных?
3. Что такое симплекс?
4. Что такое отраженная точка?
5. Какие условия применяются для останова алгоритмов?
6. Метод поиска по симплексу.
7. Метод Хука-Дживса.
8. Метод деформируемого симплекса.
9. Достоинства и недостатки методов прямого поиска.
10. Градиентные методы.
11. Методы оврагов.
12. Метод покоординатного спуска.
13. Метод Гельфанда.
14. Что такое «эффект оврагов»?
15. Что такое «овражный шаг»?

Лабораторная работа №4
«Симплекс-метод решения задач линейного программирования» (6 часов)

Цель работы: изучение симплекс-метода решения задач линейного программирования

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Постановка задачи линейного программирования

Линейное программирование (ЛП) – это раздел математического программирования, в котором рассматриваются методы решения экстремальных задач с линейным функционалом и линейными ограничениями, которым должны удовлетворять искомые переменные.

В общем случае задача ЛП формулируется следующим образом: найти значение переменных $x_1, x_2, ..., x_n$, доставляющие минимум (максимум) линейной целевой функции

$$f(x) = c_1 x_1 + c_2 x_2 + ... + c_n x_n$$

и удовлетворяющие ограничениям вида:

$$a_{i1} x_1 + a_{i2} x_2 + ... + a_{in} x_n \le b_i,$$

$$a_{i1} x_1 + a_{i2} x_2 + ... + a_{in} x_n \ge b_i,$$

$$a_{i1} x_1 + a_{i2} x_2 + ... + a_{in} x_n = b_i,$$

при этом, как правило, все или часть переменных должны быть неотрицательны:

$$x_j \ge 0.$$

В компактной форме общая задача ЛП имеет вид:

$$Z = \sum_{j=1}^{n} c_j x_j \to \min(\max)$$

при ограничениях

$$\sum_{j=1}^{n} a_{ij} x_j \le b_j \text{ , при } i = \overline{1,k},$$

$$\sum_{j=1}^{n} a_{ij} x_j \ge b_j \text{ при } i = \overline{k+1,l},$$

$\sum_{j=1}^{n} a_{ij} x_j = b_j$ при $i = \overline{l+1, m}$

$x_j \geq 0$ при $j = \overline{1, s}, s \leq n, k \leq m, l \leq m$,

где a_{ij}, b_i, c_j – заданные постоянные величины.

Различаются также две основные формы задачи ЛП в зависимости от наличия ограничений разного типа.

Стандартная форма задачи ЛП:

$$Z = \sum_{j=1}^{n} c_j x_j \to \min(\max),$$

при ограничениях

$\sum_{j=1}^{n} a_{ij} x_j \leq b_j$, при $i = \overline{1, m}$,

$x_j \geq 0$, $j = \overline{1, n}$.

Каноническая форма задачи линейного программирования:

$$Z = \sum_{j=1}^{n} c_j x_j \to \min(\max)$$

при ограничениях

$\sum_{j=1}^{n} a_{ij} x_j = b_j$, $i = \overline{1, m}$,

$x_j \geq 0$, $j = \overline{1, n}$.

Стандартная форма интересна тем, что большое число прикладных задач естественным образом сводится к этому виду моделей. Каноническая форма важна ввиду того, что основные вычислительные методы решения разработаны именно для этой формы.

Указанные выше три формы задачи линейного программирования эквивалентны в том смысле, что каждая из них с помощью несложных преобразований может быть приведена к любой из двух остальных. Следовательно, любую задачу линейного программирования можно привести к каноническому виду. Поэтому умение решать задачу в канонической форме позволяет решать задачу и в любой другой форме.

Рассмотрим основные приемы преобразования задач линейного программирования из одной формы в другую.

Переход от задачи минимизации к задаче максимизации.

Для этого нужно изменить знак целевой функции, т.е. задача

$$Z = \sum_{j=1}^{n} c_j x_j \to \min$$

эквивалентна задаче на поиск максимума

$$Z = \left(- \sum_{j=1}^{n} c_j x_j \right) \to \max .$$

Переход к эквивалентной системе неравенств.

Меняя знаки свободного члена и коэффициенты в ограничениях-неравенствах, можно поменять знак этого неравенства на обратный. Например, ограничение

$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq b_i$

можно заменить следующим:

$(-a_{i1})x_1 + (-a_{i2})x_2 + \ldots + (-a_{in})x_n \geq -b_i$.

Переход от ограничения-неравенства к равенству.

Любое ограничение в форме неравенства можно преобразовать в ограничение-равенство введением дополнительной неотрицательной переменной. Так, например, условие

$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq b_i$

эквивалентно двум ограничениям:

$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n + x_{n+1} = b_i$ и $x_{n+1} \geq 0$,

где x_{n+1} – фиктивная (или дополнительная) переменная.

Представление ограничения – равенства парой неравенств.

Ограничение

$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i$

можно представить парой ограничений

$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq b_i$,

$(-a_{i1})x_1 + (-a_{i2})x_2 + \ldots + (-a_{in})x_n \leq -b_i$.

Переход к неотрицательным переменным.

Если на знак переменной x_j не наложено ограничений, можно заменить ее разностью двух неотрицательных переменных:

$x_j = x_j' - x_j''$, $x_j' \geq 0$, $x_j'' \geq 0$.

Переход от переменных, ограниченных снизу к неотрицательным переменным.

Пусть переменная ограничена снизу: $x_j \geq c$. Заменив ее на переменную $x_j = x_j^{'} + c$, перейдем к задаче, в которой фигурирует неотрицательная переменная $x_j^{'} \geq 0$.

1.2 Решение задач линейного программирования симплексным методом

Симплексный метод (симплекс-метод) является одним из универсальных методов решения задач линейного программирования, называемый также методом последовательного улучшения плана. Симплекс-метод позволяет вести расчеты вручную и на ЭВМ.

Дана задача линейного программирования в канонической форме: *Каноническая форма задачи линейного программирования:*

$$Z = \sum_{j=1}^{n} c_j x_j \to \max \qquad (1.1)$$

при ограничениях

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \ \ (i = \overline{1,m}), \qquad (1.2)$$

$$x_j \geq 0, \ j = \overline{1,n}. \qquad (1.3)$$

Предположим, что в системе (1.2) $m < n$ и все m уравнений линейно независимы (ранг системы $r = m$). В этом случае система имеет бесчисленное множество решений. Ее можно разрешить относительно m переменных $x_1, x_2, ..., x_m$, если векторы-коэффициенты при этих переменных линейно независимы:

$$x_i = b_i - \sum_{j=m+1}^{n} b_{ij} x_j, \quad (i = \overline{1, m}). \qquad (1.4)$$

В этом случае $x_1, ..., x_m$ – *базисные переменные*, а $x_{m+1}, ..., x_n$ – *свободные переменные*. Тогда и целевую функцию можно выразить через свободные переменные:

$$f = b_0 - \sum_{j=m+1}^{n} b_j x_j \ . \qquad (1.5)$$

Если все $b_{i0} > 0$, то план $x^0 = (x_1 = b_{10}, x_2 = b_{20}, ..., x_m = b_{m0}, x_{m+1} = 0, ..., x_n = 0\}$ будет являться опорным и $f(x^0) = b_0$. Если

при этом опорном плане значение целевой функции максимально, то опорный план является *оптимальным.*

Решение задачи симплексным методом носит итеративный характер и состоит в построении и последовательном преобразовании симплексной таблицы, в результате которого от начального плана можно за конечное число шагов получить оптимальный план, либо установить, что задача не имеет решения. Таким образом, решение задачи симплексным методом проводится в два этапа. Сначала находится начальное базисное решение (начальный опорный план), а затем проводится направленный перебор базисных решений для получения оптимального плана.

Задачу, разрешенную относительно базисных переменных, удобно представить в виде симплексной таблицы (таблица 1.1).

Таблица 1.1 – Вид симплексной таблицы

Базисные переменные (БП)	План	Свободные переменные (СП)			
		x_{m+1}	x_{m+2}	…	x_n
x_1	b_{10}	b_{11}	b_{12}	…	$b_{1,m-n}$
x_2	b_{20}	b_{21}	b_{22}	…	$b_{2,m-n}$
…	…	…	…	…	…
x_m	b_{m0}	b_{m1}	b_{m2}	…	$b_{m,m-n}$
f	$-b_0$	$-b_m$	$-b_{m+1}$	…	$-b_n$

Симплексное преобразование

Преобразование системы (1.2) к новому базису называется *симплексным преобразованием*. Правило выбора переменных при направленном преобразовании (при переходе от одного опорного плана к другому, более близкому к оптимальному) одного базиса в другой: *в базис вводят переменную x_{m+j}, соответствующую отрицательному элементу f – строки с наибольшей абсолютной величиной.*

Столбец коэффициентов при переменной, включенной в базис, называется *разрешающим*. Для определения переменной, подлежащей исключению из базиса, составляют отношения свободных членов к положительным элементам разрешающего столбца (такие отношения называются *симплексными*) и находят среди них наименьшее, которое и определяет переменную, исключаемую из базиса. Строка, которой соответствует переменная, исключаемая из базиса, называется *разрешающей*. Элемент симплекс-таблицы, стоящий на пересечении разрешающей строки и разрешающего столбца, называется *разрешаю-*

щим. Предполагая, что разрешающий элемент выбран, можно сформулировать следующее правило перерасчета элементов симплекс-таблицы при переходе к новому опорному плану:

– разрешающий элемент заменяется обратной величиной;

– остальные элементы разрешающей строки делятся на разрешающий элемент;

– остальные элементы разрешающего столбца делятся на разрешающий элемент и меняют свои знаки;

– прочие элементы вычисляют по формуле :

$$b'_{ij} = (b_{ij}b_{ks} - b_{is}b_{kj}) / b_{ks}, \qquad (1.6)$$

где $i = 0, ..., m, i \neq k,\ j = 0, ..., n - m,\ j \neq s$, b_{ks} –разрешающий элемент.

При вычислении элементов по формуле (1.6) удобно пользоваться правилом прямоугольника. Элементы, входящие в эту формулу, расположены в вершинах воображаемого «прямоугольника».

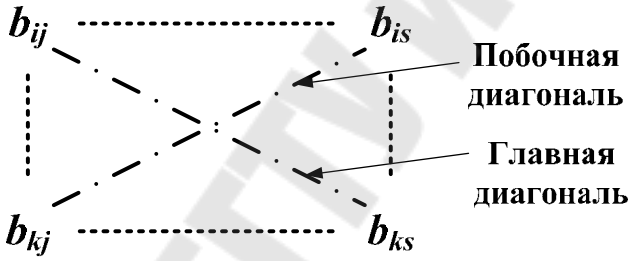

Рисунок 1.1 – Правило прямоугольника

Диагональ этого прямоугольника, на которой расположены разрешающий b_{ks} и преобразуемый b_{ij} элементы, называется главной, а другая – побочной. Преобразованный элемент b'_{ij} равен разности произведений элементов главной и побочной диагоналей, деленной на разрешающий элемент.

Сформированного правила придерживаются независимо от того, в какой вершине прямоугольника расположен разрешающий элемент.

Нахождение начального опорного плана.

Симплексное преобразование можно применить для нахождения опорного плана, т.е. для преобразования системы (1.2) к виду (1.4).

Система (1.2) записывается в виде:

$$0 = a_{10} + a_{11} \cdot (- x_1) + ... + a_{1n} \cdot (- x_n),$$
$$...$$
$$0 = a_{m0} + a_{m1} \cdot (-x_1) + ... + a_{mn} \cdot (-x_n) \quad (1.7)$$

а целевая функция f:

$$f = 0 + (- c_1) \cdot (- x_1) ... + (- c_n) \cdot (- x_n) \quad (1.8)$$

Равенства (1.7) и (1.8) вносят в симплекс-таблицу, при этом некоторые уравнения системы (1.7) умножают на –1, чтобы элементы столбца свободных членов были положительными. В результате получается таблица 1.2.

Таблица 1.2 – Построение симплексной таблицы

	СЧ	$- x_1$	…	$- x_n$
0=	a_{10}	a_{11}	…	a_{1n}
…	…	...	…	…
0=	a_{m0}	a_{m1}	…	a_{mn}
f =	0	$- c_1$	…	$- c_n$

Данная таблица подвергается симплексным преобразованиям. На каждом шаге один из x и 0 меняются местами. Через r шагов (r – ранг матрицы коэффициентов системы (1.7)) r переменных x поменяются с нулями первого столбца. Пусть это будут переменные $x_1, x_2, ..., x_r$. Получится таблица следующего вида (таблица 1.3).

Таблица 1.3 – Результат преобразования симплексной таблицы

	СЧ	0	…	0	$- x_{r+1}$	…	$- x_n$
x_1	b_{10}	b_{11}	…	b_{1r}	$b_{1,r+1}$	…	b_{1n}
…	…	…	…	…	…	…	…
x_r	b_{r0}	b_{r1}	…	b_{rr}	$b_{r,r+1}$	…	b_{rn}
0=	$b_{r+1,0}$	$b_{r+1,1}$	…	$b_{r+1,r}$	0	…	0
…	…	…	…	…	…	…	…
0=	b_{m0}	b_{m1}	…	b_{mr}	0	…	0
f =	b_{00}	b_{01}	…	b_{0r}	$b_{0,r+1}$	…	b_{0n}

Система (1.2) будет совместной, если в таблице 1.3 свободные члены $b_{r+1,0},...,b_{m0}$ равны нулю. Если хотя бы один из них отличен от нуля, то система будет несовместна.

При расчетах опускают разрешающие столбцы и строки, целиком состоящие из нулей. Если в ходе преобразований встретится 0-строка, в которой все элементы не положительны (свободный член положителен), то соответствующая система не имеет опорных решений. Если система (1.2) совместна в области неотрицательных решений, то через *r* шагов перейдем к таблице аналогичной табл. 1.1 и содержащей начальный опорный план $x^0 = (b_{10}, b_{20},..., b_{r0}, 0,...,0)$, а также $f(x^0) = b_{00}$. Опорный план соответствует базису $b^0 = (x_1, x_2,..., x_r)$, относительно которого оказалась разрешима система.

Рассмотрим пример определения начального опорного плана следующей задачи:

$$f = 2x_1 + 3x_2 - x_3 + 4x_4 - \max$$

$$\left.\begin{aligned} 3x_1 - 3x_2 - x_3 + x_4 = 8 \\ x_1 - x_2 + 2x_3 + x_4 = 4 \end{aligned}\right\}$$

$$x_j \geq 0, \quad j = \overline{1,4}$$

Систему уравнений запишем в виде:

$0 = 8 - 3x_1 + 3x_2 + x_3 - x_4,$

$0 = 4 - x_1 + x_2 - 2x_3 - x_4,$

и занесем в симплексную таблицу (таблица 1.4).

Таблица 1.4 – Пример определения начального опорного плана

	СЧ	Переменные			
		$-x_1$	$-x_2$	$-x_3$	$-x_4$
0=	8	3	–3	–1	1
0=	4	1	–1	2	1 =
f =	0	–2	–3	1	–4

В табл. 1.4 за разрешающий столбец можно взять любой столбец за исключением второго, так как он не содержит ни одного положительного элемента. Возьмем, например, четвертый столбец и вычислим симплексные отношения: $\min(8/1, 4/1) = 4$ (вторая строка разре-

шающая). Выполним симплексное преобразование и получим таблицу 1.5. Разрешающим столбцом в табл. 1.5 можно взять только первый, и разрешающей строкой будет только первая, т.к. $\min(4/2,4/1)=2$. После симплексного преобразования с разрешающим элементом 2 получим таблицу 1.6 с начальным опорным планом $x^0=(2,0,0,2)$, при котором $f(x^0)=12$.

Таблица 1.5 – *Шаг_1*

	СЧ	СП		
		$-x_1$	$-x_2$	$-x_3$
0=	4	2	–2	–3
x_4	4	1	–1	2
$f=$	16	2	–7	9

Таблица 1.6 – *Шаг_2*

БП	СЧ	СП	
		$-x_2$	$-x_3$
$-x_1$	2	–1	–3/2
x_4	2	0	7
$f=$	12	–5	12

Нахождение оптимального плана.

Если в симплексной таблице, содержащей опорный план, все элементы *f*-строки (не считая свободного члена) неотрицательны, то данный опорный план является оптимальным. Полученный оптимальный план будет единственным, если все элементы *f*-строки положительны. Если среди неотрицательных элементов встречается хотя бы один нулевой, то задача имеет бесконечное множество оптимальных планов.

Если хотя один элемент *f*-строки отрицательный, то оптимальный опорный план находят по алгоритму:

– выбирают разрешающий столбец по отрицательному элементу *f*-строки (если в *f*-строке отрицательных элементов несколько, то наибольший по абсолютной величине отрицательный элемент укажет на разрешающий элемент);

– разрешающая строка находится по минимальному симплексному отношению;

– делают симплексное преобразование с выбранным разрешающим элементом и получают новый опорный план, который опять проверяют на оптимальность.

Решение проводится до тех пор, пока не будет получен оптимальный план, либо установлена неразрешимость задачи. Если в *f*-строке симплексной таблицы, содержащей опорный план, есть хотя бы один отрицательный элемент, а в соответствующем этому элемен-

ту столбце нет ни одного положительного, то целевая функция не ограничена в области допустимых решений, т.е. $f - \propto$.

2 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Изучить симплекс-метод решения задач линейного программирования.

2. Разработать программу решения задачи линейного программирования симплекс-методом. Решить задачу ЛП, рассматривая в качестве начального базисного плана план $x^{(0)}$, приведенный в задании.

3. Оформить отчет о выполнении задания с приведением условия задачи, описания алгоритма реализации и фрагментов программ, результатов работы программы и выводов.

Варианты заданий.

Вариант 1

$x_1 - 3x_2 - 5x_3 - x_4 - \max,$

$x_1 + 4x_2 + 4x_3 + x_4 = 5,$

$x_1 + 7x_2 + 8x_3 + 2x_4 = 9,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 2

$x_1 + x_2 + x_3 + x_4 - \max,$

$x_1 + 3x_2 + x_3 + 2x_4 = 5,$

$2x_1 - x_3 + x_4 = 1,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (0,1,0,1);$

Вариант 3

$-2x_1 + x_2 + x_3 - x_4 - \max,$

$3x_1 + x_2 + 2x_3 + 6x_4 = 5,$

$x_1 + x_2 - 2x_3 + 2x_4 = 2,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,0,1).$

Вариант 4

$x_1 - x_2 - 2x_3 - x_4 - \max,$

$x_1 + 4x_2 + 3x_3 + 2x_4 = 4,$

$x_1 + 7x_2 + 6x_3 + x_4 = 7,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 5

$x_1 - 3x_2 + 3x_3 - 2x_4 - \max,$

$x_1 + 3x_2 + x_3 + x_4 = 5,$

$x_1 + 7x_2 + 8x_3 + x_4 = 4,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 6

$x_1 + 3x_2 + 2x_3 + x_4 - \max,$

$x_1 + 2x_2 + 2x_3 + x_4 = 5,$

$2x_1 - 2x_3 + x_4 = 1,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (0,1,0,1);$

Вариант 7

$-2x_1 + x_2 + x_3 - x_4 - \max,$

$3x_1 + x_2 + 2x_3 + 6x_4 + = 15,$

$x_1 + 2x_2 - x_3 + 2x_4 + = 5,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,0,0).$

Вариант 8

$x_1 - 2x_2 - 2x_3 - x_4 - \max,$

$x_1 + x_2 + x_3 + 2x_4 = 4,$

$x_1 + 4x_2 + 5x_3 + x_4 = 7,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 9

$x_1 - x_2 - x_3 - x_4 - \max,$

$x_1 + x_2 + 2x_3 + x_4 = 5,$

$x_1 + 7x_2 + x_3 + 2x_4 = 9,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 10

$x_1 + 2x_2 - x_3 + x_4 - \max,$

$x_1 + x_2 + 2x_3 + 2x_4 = 5,$

$2x_1 - x_3 + 2x_4 = 5,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (0,1,0,1);$

Вариант 11

$x_1 - 2x_2 - x_3 - x_4 - \max,$

$x_1 + 3x_2 + x_3 + 2x_4 = 3,$

$x_1 + x_2 + 2x_3 + x_4 = 3,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 12

$x_1 - x_2 - x_3 - 2x_4 - \max,$

$x_1 + x_2 + x_3 + 2x_4 = 4,$

$x_1 + x_2 + 6x_3 + x_4 = 5,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 13

$x_1 - 3x_2 + x_3 - 2x_4 - \max,$

$x_1 + 3x_2 + x_3 + x_4 = 4,$

$x_1 + 7x_2 + 2x_3 + x_4 = 1,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 14

$x_1 + x_2 + 2x_3 + x_4 - \max,$

$x_1 + 2x_2 + 2x_3 + x_4 = 3,$

$2x_1 - x_3 + x_4 = 3,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (0,1,0,1);$

Вариант 15

$x_1 - 3x_2 + x_3 - 2x_4 - \max,$

$x_1 + 3x_2 + x_3 + 2x_4 = 4,$

$x_1 + 7x_2 + x_3 + x_4 = 5,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 16

$2x_1 + x_2 + x_3 + x_4 - \max,$

$x_1 + 2x_2 + 2x_3 + x_4 = 2,$

$x_1 - x_3 + x_4 = 1,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (0,1,0,1);$

Вариант 17

$x_1 + x_2 + x_3 - x_4 - \max,$

$3x_1 + 3x_2 + 2x_3 + 6x_4 + = 15,$

$x_1 + 2x_2 - x_3 + x_4 + = 5,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0).$

Вариант 18

$3x_1 - 2x_2 - 2x_3 - x_4 - \max,$

$x_1 + x_2 + x_3 + 2x_4 = 4,$

$x_1 + 2x_2 + 3x_3 + x_4 = 7,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,1,1,0);$

Вариант 19

$x_1 - x_2 - 2x_3 - x_4 - \max,$

$x_1 + x_2 + 2x_3 + x_4 = 5,$

$x_1 + 7x_2 + 3x_3 + 2x_4 = 9,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,1);$

Вариант 20

$x_1 + x_2 - x_3 + x_4 - \max,$

$x_1 + 2x_2 + 2x_3 + 2x_4 = 5,$

$2x_1 - x_3 + 2x_4 = 5,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,1,0,1);$

Вариант 21

$x_1 - 2x_2 - x_3 - 2x_4 - \max,$

$x_1 + 3x_2 + 2x_3 + 2x_4 = 3,$

$x_1 + x_2 + 2x_3 + x_4 = 3,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,1,1,0);$

Вариант 22

$x_1 - x_2 - x_3 - 2x_4 - \max,$

$x_1 + 2x_2 + x_3 + 2x_4 = 4,$

$x_1 + x_2 + 3x_3 + x_4 = 5,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 23

$x_1 - 3x_2 + 2x_3 - 2x_4 - \max,$

$x_1 + 3x_2 + x_3 + x_4 = 4,$

$x_1 + x_2 + 2x_3 + 3x_4 = 1,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,1,1,0);$

Вариант 24

$x_1 + x_2 - x_3 + x_4 - \max,$

$x_1 + x_2 + 2x_3 + x_4 = 3,$

$2x_1 - x_3 + x_4 = 3,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (0,0,0,1);$

Вариант 25

$x_1 - 3x_2 - x_3 - x_4 - \max,$

$x_1 + x_2 - x_3 + x_4 = 5,$

$x_1 + 7x_2 + 8x_3 + 2x_4 = 9,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,0);$

Вариант 26

$x_1 + x_2 + x_3 + x_4 - \max,$

$x_1 + x_2 + x_3 + 4x_4 = 5,$

$2x_1 - x_3 + 2x_4 = 1,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (0,1,1,1);$

Вариант 27

$-2x_1 + x_2 + x_3 - x_4 - \max,$

$3x_1 + x_2 + 2x_3 + 2x_4 = 5,$

$x_1 + x_2 - x_3 + 2x_4 = 2,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,1,0,1).$

Вариант 28

$x_1 - x_2 - 2x_3 - x_4 - \max,$

$x_1 + x_2 + 3x_3 + 2x_4 = 9,$

$x_1 + 7x_2 + 6x_3 + x_4 = 3,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,0,1,1);$

Вариант 29

$x_1 - x_2 + 3x_3 - 2x_4 - \max,$

$x_1 + 3x_2 + x_3 + x_4 = 5,$

$x_1 + 7x_2 + x_3 - x_4 = 4,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (1,1,1,0);$

Вариант 30

$x_1 + 3x_2 + 2x_3 + x_4 - \max,$

$x_1 + x_2 - x_3 + x_4 = 5,$

$2x_1 - x_3 + x_4 = 1,$

$x_i \geq 0, \quad i = \overline{1,4},$

$x^{(0)} = (0,0,0,1);$

3 ТРЕБОВАНИЕ К ОТЧЕТУ

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Краткие теоретические сведения.
3. Схема алгоритма.
4. Листинги основных частей программы.
5. Результаты работы программы.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие бывают основные формы задач линейного программирования?
2. Что такое базисный план?
3. Каноническая форма ЗЛП.
4. Стандартная форма ЗЛП.
5. Что такое свободная переменная?
6. Что такое базисная переменная?
7. Как проверить план на оптимальность?

Лабораторная работа №5
«Транспортная задача» (6 часов)

Цель работы: изучить методы решения транспортной задачи

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Постановка задачи

Важный частный случай задач линейного программирования – транспортные задачи. Это математические модели разнообразных прикладных задач по оптимизации перевозок. Рассмотрим постановку транспортной задачи, методы отыскания исходной крайней точки, решение задачи методом потенциалов, двойственную к транспортной задачу, обоснование метода потенциалов, задачу о назначении, примеры. Транспортной задачей по критерию стоимости называется следующая задача о минимизации стоимости перевозок.

Пусть в пунктах отправления $A_1,...,A_m$ сосредоточено соответственно $a_1,...,a_m$ единиц некоторого однородного груза. Этот груз следует перевезти в n пунктов назначения $B_1,...,B_n$, причем в каждый из них надлежит завезти соответственно $b_1,...,b_n$ единиц груза. Стоимость перевозки единицы груза из пункта A_i в пункт B_j равна c_{ij}.

Обозначим через x_{ij} количество единиц груза, предназначенного к отправке из пункта A_i в пункт B_j, получим задачу нахождения плана перевозок, при котором общая стоимость окажется минимальной:

$$f = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \to \min, \tag{1.1}$$

$$x_{ij} \geq 0, \tag{1.2}$$

$$\sum_{j=1}^{n} x_{ij} = a_i \quad (i = 1,..., m), \tag{1.3}$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad (j = 1, ..., n). \tag{1.4}$$

План перевозок и стоимость перевозок могут быть представлены в виде матриц (таблиц) $X = \{x_{ij}\}$ и $C = \{c_{ij}\}$, соответственно, где $i = 1,..., m$, $j = 1,..., n$. Матрица $C = \{c_{ij}\}$ называется матрицей тарифов

(издержек или транспортных расходов). Планом транспортной задачи называется матрица $X = \{x_{ij}\}$, где каждое число x_{ij} обозначает количество единиц груза, которое надо доставить из *i*-го пункта отправления в *j*-й пункт назначения. В компактном виде план транспортной задачи и тарифы перевозок можно представить в виде таблицы 1.1.

Таблица 1.1 – План перевозок

Поставщик	Потребитель				Запас груза
	B_1	B_2	…	B_n	
A_1	c_{11} x_{11}	c_{12} x_{12}	…	c_{1n} x_{1n}	a_1
A_2	c_{21} x_{21}	c_{22} x_{22}	…	c_{2n} x_{2n}	a_2
…	…	…	…	…	…
A_m	c_{m1} x_{m1}	c_{m2} x_{m2}	…	c_{mn} x_{mn}	a_m
Потребность в грузе	b_1	b_2	…	b_n	

В таблице заданы стоимости перевозки единицы груза от каждого *i*-ого пункта отправления до каждого *j*-го пункта потребления. Требуется определить, какое количество груза $x_{ij} \geq 0$ необходимо перевезти от каждого пункта отправления до каждого пункта потребления, чтобы:

-ввезти грузы всех поставщиков;

-удовлетворить всех потребителей;

-достигнуть минимума целевой функции (затрат на перевозку).

Уравнение (1.3) означают, что из пункта отправления A_i весь груз вывезен в пункты назначения (потребления). Уравнения (1.4) означают, что количество груза, завезенного в пункт B_j, со всех пунктов отправления, соответствует требуемому значению. Считаем, что общий запас груза на всех пунктах отправления равен суммарной потребности всех пунктов назначения, т.е.

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = M , \qquad (1.5)$$

где $i = 1,..., m$, $j = 1, ..., n$.

В этом случае говорят, что имеется замкнутая модель транспортной задачи. Если суммарные запасы отправителей больше суммарной потребности пунктов назначения, т.е.

$$\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j , \qquad (1.6)$$

то равенства (1.3) заменяются неравенствами

$$\sum_{j=1}^{n} x_{ij} \le a_i , \; i = 1,..., m \qquad (1.7)$$

а условие (1.4) остается без изменений. В этом случае вводится фиктивный пункт назначения B_{n+1} с потребностями

$$b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j , \qquad (1.8)$$

и нулевыми стоимостями перевозок в этот пункт. Добавляя новые неотрицательные переменные $x_{i,n+1}$ $(i = 1,..., m)$, приходим к замкнутой модели транспортной задачи с ограничениями (1.2) – (1.4).

Если суммарные запасы отправителей меньше суммарных запросов пунктов назначения, т.е.

$$\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j , \qquad (1.9)$$

то равенства (1.4) заменяются неравенствами

$$\sum_{i=1}^{m} x_{ij} \le b_j , \; j = 1,..., n , \qquad (1.10)$$

а условие (1.3) остается без изменений. В этом случае вводится фиктивный пункт отправления A_{m+1} с запасами

$$a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i , \qquad (1.11)$$

и нулевыми стоимостями перевозок в этот пункт. Добавляя новые неотрицательные переменные $x_{m+1,j}$ $(j = 1,..., n)$, приходим к замкнутой модели транспортной задачи с ограничениями (1.2) – (1.4).

Таким образом, математически транспортная задача ставится следующим образом. Дана система ограничений (1.2) – (1.4) и целевая функция (1.1). Требуется среди множества решений системы найти такое неотрицательное решение (или план перевозок), которое минимизирует целевую функцию (1.1).

Рассмотрим основные особенности задачи. Транспортная задача является задачей линейного программирования и может быть решена симплекс-методом, который значительно упрощается в виду простого строения системы ограничений (1.2) – (1.4).

1.2 Определение исходного опорного плана

Существуют различные способы построения исходного опорного плана. Рассмотрим замкнутую модель транспортной задачи.

Правило «северо-западного угла». Будем заполнять таблицу, начиная с левого верхнего («северо-западного») угла. Назначим максимально возможную перевозку из пункта отправления A_1 в пункт назначения B_1. Занесем в эту клетку меньшее из чисел a_1 и b_1, т.е. $x_{11} = \min(a_1; b_1)$. Таким образом, либо пункт отправления A_1, либо пункт назначения B_1, либо оба эти пункта окажутся полностью обслуженными.

Если пункт отправления A_1 оказался полностью обслуженным, то в дальнейшем при нахождении исходного плана перевозок выводим из рассмотрения первую строку матрицы X и рассматриваем только оставшуюся часть матрицы. Если пункт назначения B_1 оказался полностью обслуженным, то аналогично выводим первый столбец из дальнейшего рассмотрения. Если же и пункт отправления, и пункт назначения оказались полностью обслуженными (так может случиться только в вырожденной задаче), то вывести из рассмотрения следует или первый столбец, или первую строку матрицы X. Условимся для определенности выводить из рассмотрения первый столбец матрицы. В этом случае в число базисных элементов на следующем этапе введем элемент с нулевым значением перевозки, стоящий в северо-западном углу оставшейся части матрицы X, т.е. в базис войдет второй элемент в строке, считая вычеркиваемый элемент первым. Эту процедуру продолжаем до тех пор, пока все пункты отправления и пункты назначения не будут обслужены. Последняя заполненная клетка окажется лежащей в последнем *n*-м столбце и в последней *m*-ой строке.

На каждом шаге обслуживается один из пунктов отправления или назначения, а на последнем шаге обслуживаются и последний пункт отправления и последний пункт назначения. Поэтому число базисных элементов будет ровно $m + n - 1$. Найденный план будет до-

пустимым планом перевозок, содержащим не более $m + n - 1$ положительных координат и являющийся крайней точкой множества допустимых решений.

Рассмотрим пример построения исходного опорного плана по правилу «северо-западного угла» следующей транспортной задачи (таблица 1.2).

Таблица 1.2 – Пример транспортной задачи

Запас a_i	Потребность, b_j			
	40	25	20	50
60	5	4	1	2
40	4	2	6	3
35	7	3	5	4

В клетку (1;1) помещаем $x_{11} = \min(60;40) = 40$. Спрос первого потребителя удовлетворен полностью, первый столбец из расчета исключается. Остаток груза от первого поставщика $(60 - 40 = 20)$ с учетом потребности второго потребителя помещаем в клетку (1;2), т.е. $x_{12} = \min(20;25) = 20$. В этом случае запас первого поставщика исчерпан, переходим к распределению груза второго поставщика. В клетку (2;2) помещаем необходимое количество груза, т.е. $x_{22} = \min(40;5) = 5$. В этом случае спрос второго потребителя удовлетворен полностью. Второй столбец в расчет не принимается, остаток груза от второго поставщика с учетом спроса третьего потребителя помещаем в клетку (2;3), т.е. $x_{23} = \min(35;20) = 20$. Спрос третьего потребителя удовлетворен полностью. Остаток груза второго поставщика с учетом спроса четвертого потребителя помещаем в клетку (2;4), т.е. $x_{24} = \min(15;50) = 15$. В этом случае запас второго поставщика исчерпан. Переходим к распределению запаса у третьего поставщика. В клетку (3;4) помещаем необходимое количество груза $x_{34} = \min(35;35) = 35$. Получили полное распределение запасов груза. При этом число занятых клеток удовлетворяет условию $m + n - 1 = 3 + 4 - 1 = 6$, поэтому план перевозок является невырожденным.

$$X_1 = \begin{pmatrix} 40 & 20 & 0 & 0 \\ 0 & 5 & 20 & 15 \\ 0 & 0 & 0 & 35 \end{pmatrix}. \qquad (1.12)$$

Данному плану соответствует значение целевой функции
$f_1 = 5 \cdot 40 + 4 \cdot 20 + 2 \cdot 5 + 6 \cdot 20 + 3 \cdot 15 + 4 \cdot 35 = 595$.

Отметим, что данный метод нахождения исходного опорного плана не учитывает стоимости перевозок. Поэтому начальный план может оказаться далеко не оптимальным. Рассмотрим другие методы нахождения начальной точки, учитывающие стоимости перевозок.

Правило «минимального элемента». Суть состоит в следующем. Распределение груза у поставщиков начинается с клетки, которой соответствует наименьший тариф c_{ij} из всей матрицы тарифов. В клетку с наименьшим тарифом помещают наименьшее из чисел a_i или b_j, т.е. $x_{ij} = \min(a_i; b_j)$. Затем из рассмотрения исключают строку, соответствующую поставщику, запасы которого полностью израсходованы, или столбец, соответствующий потребителю, спрос которого полностью удовлетворен. Может оказаться, что следует исключить строку и столбец одновременно, если полностью израсходованы запасы поставщика и полностью удовлетворен спрос потребителя. Далее из оставшихся строк (столбцов) таблицы снова выбирают наименьший тариф и процесс распределения запасов продолжают до тех пор, пока все они не будут распределены, а спрос удовлетворен.

Рассмотрим пример построения исходного опорного плана по правилу *«минимального элемента»* транспортной задачи, представленной в табл. 1.2.

Наименьший тариф имеем для клетки (1;3) – $c_{13} = 1$, поэтому в данную клетку помещаем максимально возможное количество груза – $x_{13} = \min(60;20) = 20$ и исключаем из дальнейшего рассмотрения третий столбец. Снова находим в таблице наименьший тариф $c_{14} = c_{22} = 2$. Начнем с клетки (1;4). Помещаем в нее максимально возможное количество груза – $x_{14} = \min(60 - 20;50) = 40$ и исключаем из дальнейшего рассмотрения первую строку. Далее в клетку (2;2) помещаем количество груза $x_{22} = \min(40;25) = 25$ и исключаем из дальнейшего рассмотрения второй столбец. В оставшейся матрице тарифов минимальным будет тариф для клеток (2;4) и (3;2) –

$c_{24} = c_{32} = 3$. Однако второй столбец уже исключен из рассмотрения, поэтому загружаем клетку (2;4) – $x_{24} = \min(40 - 25;50 - 40) = 10$, и исключаем из дальнейшего рассмотрения четвертый столбец. Новый наименьший тариф для оставшихся тарифов будет $c_{21} = 4$. В клетку (2;1) помещаем количество груза $x_{21} = \min(15 - 10;40) = 5$. Исключаем из дальнейшего рассмотрения вторую клетку.

На последнем этапе выберем единственный не вычеркнутый первый столбец и единственную третью строку, для которых имеем общую свободную клетку (3;1) с тарифом $c_{31} = 7$. В нее помещаем остаток груза $x_{31} = \min(35;40 - 5) = 35$. Получили полное распределение запасов груза:

$$X_2 = \begin{pmatrix} 0 & 0 & 20 & 40 \\ 5 & 25 & 0 & 10 \\ 35 & 0 & 0 & 0 \end{pmatrix}, \qquad (1.13)$$

при этом число занятых клеток удовлетворяет условию $m + n - 1 = 3 + 4 - 1 = 6$, поэтому план перевозок является невырожденным. Для данного плана значение целевой функции равно:

$f_2 = 1 \cdot 20 + 2 \cdot 40 + 4 \cdot 5 + 2 \cdot 25 + 3 \cdot 10 + 7 \cdot 35 = 445$.

Сравнивая значения целевых функций для опорных планов, полученных по правилам «северо-западного угла» и «минимального элемента», замечаем, что транспортные издержки уменьшились на 150 единиц.

Правило «минимального элемента» имеет различные модификации, в частности «м*инимум по строке»* и *«минимум по столбцу»*.

Минимум по строке. Выберем в первой строке матрицы тарифов минимальный по величине элемент. Предположим $c_{1x} = \min(c_{1j})$. Назначим максимально возможную перевозку из пункта отправления A_1 в пункт назначения B_x. Если минимум достигается на нескольких элементах, то выбираем любой из них. Тем самым пункт отправления A_1 или пункт назначения B_x (или оба пункта одновременно) будет обслужен. А в матрице тарифов первая строка или соответствующий столбец выводятся из дальнейшего рассмотрения. В первой строке оставшейся части матрицы вновь ищется минимальный элемент и процедура повторяется до тех пор пока опорный план перевозок не будет получен. Первой строкой может оказаться вновь первая строка исходной матрицы без какого-то элемента или вторая строка исход-

ной матрицы без какого-то элемента. В результате получаем исходный опорный план.

Минимум по столбцу. Выберем в первом столбце матрицы тарифов минимальный по величине элемент. Предположим $c_{x1} = \min(c_{i1})$. Назначим максимально возможную перевозку из пункта отправления A_x в пункт назначения B_1. Если минимум достигается на нескольких элементах, то выбираем любой из них. Тем самым пункт отправления A_x или пункт назначения B_1 (или оба пункта одновременно) будет обслужен. В матрице тарифов выводятся из рассмотрения обслуженная строка или первый столбец. Продолжаем данную процедуру, пока не получим исходный опорный план.

Способ аппроксимации Фогеля. Данный способ получил широкое распространение, так как в большинстве случаев он дает опорный план, наиболее близкий к оптимальному плану.

В основе способа аппроксимации Фогеля лежит концепция штрафов, взимаемых за выбор не самого оптимального с точки зрения транспортных издержек маршрута. Штраф по каждой строке и по каждому столбцу определяется из анализа маршрутов с различными показателями издержек (как разность двух различных уровней транспортных издержек). Первой заполняется клетка матрицы (таблицы), в которой фиксируется самый крупный штраф. После заполнения клетки штрафы пересчитываются и так до тех пор, пока все ресурсы не будут распределены.

Алгоритм включает следующие основные этапы.

1) Вычисление разностей в каждой строке и в каждом столбце матрицы между наименьшей стоимостью и ближайшей к ней величине. Разности по строкам записываются справа в столбце разностей, разности по столбцам – внизу в строке разностей.

2) Поиск из всех разностей как по строкам, так и по столбцам, максимальной. Обведем максимальную разность рамкой.

3) Размещение в клетку, где находится наименьшая стоимость, максимально возможного количества ресурсов.

4) Вычисление разностей по столбцам и строкам, не принимая во внимание стоимости в клетках, имеющих ресурсы, и клетках с точкой (исключенную строку или столбец), и определение максимальной разности в строке или столбце.

5) Поиск минимального элемента в строке или столбце с максимальной разностью и размещение в данную клетку максимального количества ресурса, возвращение к этапу 4 и т.д.

1.3 Метод потенциалов

Для решения транспортной задачи широко используется метод потенциалов ([3]). Чтобы найти оптимальный план перевозок, необходимо:

–получить опорный план перевозок по одному из изложенных выше правил;

–вычислить потенциалы поставщиков u_i и потребителей v_j, соответственно;

–вычислить сумму потенциалов (косвенные тарифы) для свободных клеток $u_i + v_j = c'_{ij}$;

–проверить разность $s_{ij} = c_{ij} - c'_{ij} = c_{ij} - (u_i + v_j)$.

Если все разности свободных клеток $s_{ij} \geq 0$, то полученный план оптимальный. Если хотя бы одна разность $s_{ij} < 0$, то в число занятых клеток вводят переменную x_{ik}, для которой эта разность минимальна, и получают новый план перевозок. Процесс продолжают до тех пор, пока не будет определен план, для которого все разности $s_{ij} \geq 0$.

В качестве примера рассмотрим решение транспортной задачи, представленной в табл. 1.2, методом потенциалов.

Пусть, например, исходное опорное решение получено по правилу «минимального элемента» (1.14). Занесем полученные значения в матрицу тарифов (табл. 1.2). Причем в правом верхнем углу клетки запишем тариф, а в левом нижнем – план перевозок (табл. 1.3). Полученный план невырожденный, так как число занятых клеток удовлетворяет условию $r - 3 + 3 - 1 = 6$.

Таблица 1.3 – Исходный опорный план P_0

	40	25	20	50	u_i
60	5	4	1 20	2 40	0
40	\+ 4 5	– 2 25	6	3 10	1
35	– 7 35	\+ 3	5	4	4
v_j	3	1	1	2	

Для определения потенциалов поставщиков и потребителей будем использовать шесть уравнений:

$u_1 + v_3 = 1;$

$u_1 + v_4 = 2;$

$u_2 + v_1 = 4;$

$u_2 + v_2 = 2;$

$u_2 + v_4 = 3;$

$u_3 + v_1 = 7.$

Поскольку число уравнений на единицу меньше числа неизвестных, то одно из неизвестных оказывается свободным и может принимать любое значение. Положим, например, $u_1 = 0$. Тогда все остальные потенциалы для данного опорного решения определяются однозначно: $u_1 = 0;\ u_2 = 1;\ u_3 = 4;\ v_1 = 3;\ v_2 = 1;\ v_3 = 1;\ v_4 = 2.$

Найденные потенциалы поставщиков и потребителей указаны справа и внизу (табл. 1.3). Расчет их можно производить непосредственно в таблице, не решая систему уравнений. Очевидно, неизвестный потенциал поставщика (потребителя) определяется разностью между тарифом занятой клетки и известным потенциалом поставщика (потребителя) – строки (столбца), на пересечении которых расположена занятая клетка.

Определим разности для свободных клеток s_{ij} по формуле $s_{ij} = c_{ij} - (u_i + v_j)$. Получим:

$s_{11} = 5 - (0 + 3) = 2,$

$s_{12} = 4 - (0 + 1) = 3,$

$s_{23} = 6 - (1 + 1) = 4,$

$s_{32} = 3 - (4 + 1) = -2,$

$s_{33} = 5 - (4 + 1) = 0,$

$s_{34} = 4 - (4 + 2) = -2.$

Полученный план не является оптимальным, так как имеются отрицательные разности: $s_{32} = s_{34} = -2$. Получены две перспективные (потенциальные) клетки с одинаковой оценкой. Загрузим, например, клетку (3;2). В таком случае для нее строится замкнутый цикл, который включает клетки: (3;2), (3;1), (2;1), (2;2). В свободной клетке указанного цикла условно запишем знак плюс, в следующей клетке (по часовой стрелке или против часовой стрелки) – минус, а в следующей плюс и т.д. В отрицательных вершинах цикла выбираем наименьшее

количество груза $\min(25;35) = 25$, т.е. $x_{32} = 25$. Чтобы общий баланс цикла не изменился, необходимо в клетки цикла со знаком плюс прибавить 25, а из клеток со знаком минус вычесть столько же. Новый опорный план P_1 приведен в таблице 1.4. Он также невырожденный.

Таблица 1.4 – Новый опорный план P_1

	40	25	20	50	u_i
60	5	4	1 20	2 40	0
40	+ 4 30	2	6	– 3 10	1
35	– 7 10	3 25	5	+ 4	4
v_j	3	1	1	2	

Найдем разности для свободных клеток s_{ij}:

$s_{11} = 5 - (0 + 3) = 2$,

$s_{12} = 4 - (0 + 1) = 3$,

$s_{22} = 2 - (1 + 1) = 0$,

$s_{23} = 6 - (1 + 1) = 4$,

$s_{33} = 5 - (4 + 1) = 0$,

$s_{34} = 4 - (4 + 2) = -2$.

План P_1 еще не оптимальный, так как разность $s_{34} = -2$. За счет загрузки клетки (3;4), план можно улучшить. Замкнутый цикл для клетки (3;4) виден из табл. 1.4 (выделенные клетки). Отнимаем 10 из клеток с минусом и прибавляем это значение в клетки с плюсом. Получаем новый опорный план (таблица 1.5). Он вырожденный, так как число занятых клеток не удовлетворяет условию $r - 3 + 3 - 1 = 6$ (число занятых клеток равно 5). Поместим нуль, например в клетку (2;2) и будем считать эту клетку занятой. Очевидно, загрузка данной клетки не приводит к образованию замкнутого цикла из занятых клеток.

Таблица 1.5 – Новый опорный план P_2

	40	25	20	50	u_i
60	5	4	1 20	2 40	0
40	+ 4 40	2 0	6	– 3	1
35	– 7	3 25	5	+ 4 10	2
v_j	3	1	1	2	

Потенциалы поставщиков и потребителей для плана, представленного в табл. 2.5, определены в этой же таблице.

Найдем разности для свободных клеток s_{ij}:

$s_{11} = 5 - (0 + 3) = 2,$

$s_{12} = 4 - (0 + 1) = 3,$

$s_{23} = 6 - (1 + 1) = 4,$

$s_{24} = 3 - (2 + 1) = 0,$

$s_{31} = 7 - (3 + 2) = 2,$

$s_{33} = 5 - (2 + 1) = 2.$

Полученный план является оптимальным, так как он не содержит ни одной отрицательной разности s_{ij}. Таким образом, оптимальный план имеет вид:

$$X = \begin{pmatrix} 0 & 0 & 20 & 40 \\ 40 & 0 & 0 & 0 \\ 0 & 25 & 0 & 10 \end{pmatrix}.$$

Для данного плана значение целевой функции равно:

$f_3 = 1 \cdot 20 + 2 \cdot 40 + 4 \cdot 5 + 2 \cdot 25 + 3 \cdot 10 + 7 \cdot 35 = 445.$

Следует отметить, что полученный оптимальный план не единственный. Например, если поместить разность, равную нулю, в клетку (2;4), то структура плана будет иная, однако значение целевой функции не изменится.

2 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Изучить методы решения транспортной задачи.

2. Разработать программу решения транспортной задачи. Предусмотреть возможность использования произвольной матрицы тарифов.

3. Оформить отчет о выполнении задания с приведением условия задачи, описания алгоритма реализации и фрагментов программ, результатов работы программы и выводов.

3 ТРЕБОВАНИЕ К ОТЧЕТУ

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Краткие теоретические сведения.
3. Схема алгоритма.
4. Листинги основных частей программы.
5. Результаты работы программы.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Математическая постановка транспортной задачи.
2. Этапы решения транспортной задачи.
3. Способы нахождения исходного опорного плана
4. Правило «северо-западного» угла.
5. Правило минимального элемента.
6. Разновидности правила минимального элемента.
7. Способ аппроксимации Фогеля.
8. Что такое матрица тарифов?
9. Что такое опорный план?
10. Как проверить план на оптимальность?

Вариант 1

Запас a_i	Потребность, b_j				
	40	45	25	20	50
60	5	1	4	1	2
45	4	4	2	6	3
35	7	7	3	5	4
40	3	4	5	7	6

Вариант 2

Запас a_i	Потребность, b_j				
	25	40	45	40	45
70	7	3	5	2	2
35	5	1	4	1	2
45	7	2	3	5	1
45	5	2	4	1	5

Вариант 3

Запас a_i	Потребность, b_j				
	20	45	25	40	45
50	5	1	4	2	2
55	5	1	4	1	3
25	7	7	3	5	4
45	5	2	4	1	6

Вариант 4

Запас a_i	Потребность, b_j				
	35	45	55	40	30
60	1	1	4	1	2
55	1	4	2	1	3
70	7	7	3	7	4
20	2	4	5	2	6

Вариант 5

Запас a_i	Потребность, b_j				
	45	45	25	45	50
70	1	1	4	1	2
45	1	4	2	1	3
75	7	7	3	7	4
20	2	4	5	2	6

Вариант 6

Запас a_i	Потребность, b_j				
	30	45	75	10	20
30	5	7	3	7	4
35	4	2	2	6	3
35	7	2	4	2	5
80	3	4	5	4	4

Вариант 7

Запас a_i	Потребность, b_j				
	50	45	25	40	50
30	5	1	3	1	2
35	4	2	2	6	3
35	7	2	3	1	5
110	3	4	5	5	7

Вариант 8

Запас a_i	Потребность, b_j				
	10	25	35	40	50
20	5	1	4	1	2
45	4	4	2	6	3
45	7	7	3	5	4
50	3	4	5	7	6

Вариант 9

Запас a_i	Потребность, b_j				
	50	45	25	40	50
30	5	1	4	1	2
45	4	5	1	4	3
45	7	7	3	5	4
90	3	4	5	1	4

Вариант 10

Запас a_i	Потребность, b_j				
	50	40	30	20	10
20	2	4	6	1	2
30	4	4	2	6	3
40	3	4	5	5	4
60	3	4	5	5	5

Вариант 11

Запас a_i	Потребность, b_j				
	30	35	35	30	40
60	5	1	4	1	2
50	4	4	2	6	3
40	4	2	6	3	4
20	3	4	2	6	3

Вариант 12

Запас a_i	Потребность, b_j				
	30	45	35	20	30
10	5	1	4	1	2
45	4	4	2	6	3
45	7	7	3	5	4
60	3	4	5	7	6

Вариант 13

Запас a_i	Потребность, b_j				
	40	45	45	40	30
55	5	1	4	1	2
45	4	4	2	6	3
35	4	5	6	5	4
65	3	4	5	9	3

Вариант 14

Запас a_i	Потребность, b_j				
	20	25	25	20	20
30	5	2	3	8	2
25	4	4	8	6	8
25	9	3	3	5	4
30	3	4	5	7	9

Вариант 15

Запас a_i	Потребность, b_j				
	30	45	25	50	40
40	5	1	3	2	5
45	4	4	3	4	7
55	7	3	4	5	4
50	3	4	5	7	6

Вариант 16

Запас a_i	Потребность, b_j				
	40	40	30	30	20
50	5	1	4	2	2
50	4	4	2	4	2
10	4	2	3	5	4
50	3	4	5	4	2

Вариант 17

Запас a_i	Потребность, b_j				
	40	45	25	20	50
60	5	1	4	1	2
45	4	4	2	6	3
35	7	7	3	5	4
40	3	4	5	7	6

Вариант 18

Запас a_i	Потребность, b_j				
	40	45	45	40	40
50	5	7	4	7	2
55	4	4	2	6	3
55	7	7	6	5	6
50	3	3	8	7	6

Вариант 19

Запас a_i	Потребность, b_j				
	20	25	35	20	30
30	3	4	5	5	2
25	4	4	2	6	3
35	5	3	4	5	4
40	3	4	5	7	5

Вариант 20

Запас a_i	Потребность, b_j				
	50	45	55	50	50
60	6	6	4	3	2
65	4	6	7	6	3
65	7	7	3	9	4
60	3	9	5	9	6

Вариант 21

Запас a_i	Потребность, b_j				
	50	50	55	50	55
70	6	7	6	1	2
65	7	6	7	6	3
65	7	7	3	9	4
60	3	4	5	7	6

Вариант 22

Запас a_i	Потребность, b_j				
	20	45	75	40	40
50	4	2	9	3	2
85	4	4	2	6	3
35	7	7	3	4	4
60	3	4	9	7	6

Вариант 23

Запас a_i	Потребность, b_j				
	40	45	35	30	50
60	4	2	4	2	2
45	7	6	7	6	3
55	7	4	2	4	2
40	3	4	5	7	6

Вариант 24

Запас a_i	Потребность, b_j				
	30	35	35	20	30
40	5	2	3	3	5
35	4	4	8	6	8
35	3	3	5	5	4
40	3	4	5	7	9

Вариант 25

Запас a_i	Потребность, b_j				
	30	35	35	50	40
40	6	1	3	2	5
35	4	4	6	4	6
65	4	3	4	5	4
50	3	4	4	7	4

Вариант 26

Запас a_i	Потребность, b_j				
	40	40	40	30	30
50	5	2	3	5	2
35	4	4	2	3	5
35	4	2	3	5	4
60	3	4	5	4	2

Вариант 27

Запас a_i	Потребность, b_j				
	50	55	55	50	50
60	9	4	5	4	2
85	4	4	2	6	3
85	3	4	9	4	4
30	3	3	4	5	4

Вариант 28

Запас a_i	Потребность, b_j				
	40	45	45	40	40
50	5	7	4	7	2
55	4	4	2	6	3
55	7	7	4	7	6
50	3	3	7	4	7

Вариант 29

Запас a_i	Потребность, b_j				
	40	45	45	20	30
70	3	4	5	5	2
25	4	4	2	6	3
35	5	3	4	5	4
30	3	4	5	7	5

Вариант 30

Запас a_i	Потребность, b_j				
	50	45	55	50	50
60	6	6	4	9	7
65	4	2	7	6	3
65	7	7	3	1	4
60	3	9	4	6	7

Лабораторная работа №6
«Задача об оптимальных назначениях»
(6 часов)

Цель работы: изучить методы решения задачи об оптимальных назначениях

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Постановка задачи

Пусть некоторая фирма нанимает m служащих на n вакантных мест. Известно, что при назначении i-го служащего на j-е место фирма получит c_{ij} единиц прибыли. Стоит задача определить, на какие должности кого надо назначить, чтобы общая прибыль фирмы после назначений была наибольшей? В такой постановке задача о назначении является частным случаем транспортной задачи при условии, что $a_i = b_j = 1,\ i = 1,..., m,\ j = 1, ..., n$, а целевую функцию требуется максимизировать. Соответственно, данная задача может быть решена изученным ранее методом потенциалов.

Задача о назначении может быть интерпретирована в различных вариантах. Например, под видами работ, можно понимать различные изделия, строительные площадки и т.п. Работы могут выполняться людьми, станками, механизмами, приспособлениями и т.п.

Рассмотрим простейший вариант задачи о назначениях. Пусть имеется n видов работ $A_1,..., A_n$ и m механизмов $B_1,..., B_m$, производительность которых равна c_{ij}. Требуется определить оптимальное назначение механизмов на работы, при котором будет достигнута максимальная суммарная производительность.

При такой интерпретации задачи важно, чтобы одновременно на каждый вид работы был назначен только один механизм. Рассмотрим случай, когда количество работ равно числу механизмов, занятых их выполнением, т.е. $n = m$. Положим, $x_{ij} = 1$, если i-й механизм назначен на j-ю работу и $x_{ij} = 0$ в противном случае. Так как каждый механизм можно назначать только на один вид работы, то

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad (1.1)$$

где $i = 1, ..., n$.

Так как каждая работа предназначена только для одного механизма, то

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad (1.2)$$

где $j = 1, ..., n$.

В качестве целевой функции используем суммарную производительность:

$$f = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} . \qquad (1.3)$$

Таким образом, требуется найти переменные $x_{ij} \in \{0;1\}$, удовлетворяющие (1) и (2) и обращающие целевую функцию (3) в максимум. Решать задачу о назначениях можно теми же методами, какими решается транспортная задача.

1.2 Венгерский метод

Прежде чем приступить к алгоритму, необходимо привести подготовительный этап. Заменяем задачу поиска максимума (3) задачей поиска минимума:

$$f' = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}' x_{ij} , \qquad (1.4)$$

где $c'_{ij} = l_j - c_{ij}$, $l_j = \max\{c_{ij}\}$, $i = 1, ..., n$.

Основной принцип венгерского метода – оптимальность решения задачи о назначении не нарушается при уменьшении (увеличении) строки (столбца) на одну и ту же величину d_i (d_j). Решение считается оптимальным, если все измененные искусственные затраты $c_{ij}' \geq 0$ ($i, j = 1,...,n$) и можно отыскать такой набор x_{ij}, что

$$F' = \sum_{i=1}^{n}\sum_{j=1}^{n} c'_{ij} x_{ij} \to \min . \qquad (1.5)$$

Венгерский метод включает следующие основные этапы.

1) *Получение нулей* в каждой строке, для чего необходимо найти наименьший элемент в каждой строке d_i и вычесть его из всех ее

элементов. Получаем новую таблицу. По аналогии выполняем эту операцию для каждого столбца и получаем новую таблицу.

2) *Поиск оптимального решения*. Для поиска оптимального решения необходимо рассмотреть сначала одну из строк таблицы, имеющую меньше нулей. Отметить точкой один из нулей этой строки и зачеркнуть все остальные нули этой строки и того столбца, в которых находится этот нуль. Аналогичные операции последовательно проводят для всех строк. Если назначения, которые получены при всех нулях, отмеченных точкой, являются полными (т.е. число нулей, отмеченных точкой, равно n), то решение является оптимальным. В противном случае следует переходить к следующему этапу.

3) Поиск минимального набора строк и столбцов, содержащих нули. Для этого необходимо отметить точкой:

а) все строки, в которых нет отмеченного точкой нуля;

б) все столбцы, содержащие перечеркнутый нуль, хотя бы в одной из отмеченных точкой строк;

в) все строки, содержащие отмеченные точкой нули, хотя бы в одном из отмеченных точкой столбцов.

Действия б) и в) повторяются поочередно до тех пор, пока есть что отмечать. После этого необходимо зачеркнуть каждую непомеченную строку и каждый помеченный столбец. Цель этого этапа – провести минимальное число горизонтальных и вертикальных прямых, пересекающих по крайне мере, один раз все нули.

4) Перестановка некоторых нулей. Взять наименьшее число из тех клеток, через которые не проведены прямые. Вычесть его из каждого числа не вычеркнутых столбцов и прибавить к каждому числу вычеркнутых строк. Получим новую таблицу. Эта операция не изменяет оптимального решения, после чего весь цикл начинается с этапа 2 и продолжается до получения оптимального решения.

Рассмотрим работу венгерского метода на примере. Пусть для монтажа четырех объектов ($n = 4$) требуется четыре крана. Из отчетных данных известно, какое время необходимо каждому крану A_i для монтажа объекта B_j. Нужно так распределить краны по объектам, чтобы суммарное время на монтаж этих объектов было минимально. В этом случае c_{ij} – это матрица затрат времени i-го крана при монтаже j-го объекта. Исходные данные представлены в таблице 1.1.

Таблица 1.1 – Исходные данные задачи о назначении

Кран	Объект, B_j					
A_i	B_1	B_2	B_3	B_4	a_i	d_i
A_1	3	7	5	8	1	3
A_2	2	4	4	5	1	2
A_3	4	7	2	8	1	2
A_4	9	7	3	8	1	3
b_j	1	1	1	1		

1) Вначале получим нули в каждой строке. Для этого найдем минимальный элемент в каждой строке (столбец d_i табл. 1.1). Получим новую таблицу 1.2.

Таблица 1.2 – Таблица после приведения по строкам

Кран	Объект, B_j				
A_i	B_1	B_2	B_3	B_4	a_i
A_1	0	4	2	5	1
A_2	0	2	2	3	1
A_3	2	5	0	6	1
A_4	6	4	0	5	1
b_j	1	1	1	1	
d_i	0	2	0	3	

По аналогии получаем нули в каждом столбце (таблица 1.3).

Таблица 1.3 – Таблица после приведения по столбцам

Кран	Объект, B_j				
A_i	B_1	B_2	B_3	B_4	a_i
A_1	0	2	2	2	1
A_2	0	0	2	0	1
A_3	2	3	0	3	1
A_4	6	2	0	2	1
b_j	1	1	1	1	

2) *Поиск оптимального решения*. Рассмотрим первую строку таблицы, имеющую минимальное число нулей. Отмечаем ноль в ячейке (1;1)точкой и зачеркиваем остальные пули, расположенные в первой строке и первом столбце (в данном случае зачеркивается ноль в ячейке (2;1)). Аналогичные операции последовательно проводим для всех строк (таблица 1.4). Получили, что число отмеченных точкой нулей не равно $n = 4$, поэтому переходим к следующему этапу.

Таблица 1.4 – Пометка нулей

Кран	Объект, B_j				
A_i	B_1	B_2	B_3	B_4	a_i
A_1	0•	2	2	2	1
A_2	∅	0•	2	∅	1
A_3	2	3	0•	3	1
A_4	6	2	∅	2	1
b_j	1	1	1	1	

3) Поиск минимального набора строк и столбцов, содержащих нули.

а) отмечаем строки, в которых нет ни одного отмеченного точкой нуля – строка 4;

б) отмечаем столбцы, содержащие перечеркнутый нуль, хотя бы в одном из отмеченных точкой строк – столбец 3;

в) отмечаем строки, содержащие отмеченные точкой нули, хотя бы в одном из отмеченных точкой столбцов – строка 3;

Действия б) и в) повторяются поочередно до тех пор, пока есть что отмечать.

4) Перестановка некоторых нулей. Возьмем наименьшее число из не закрашенных клеток (число 2). Вычтем его из каждого числа не закрашенных столбцов и прибавим к каждому числу закрашенных строк (табл. 1.5).

Получили, что число отмеченных точкой нулей равно $n = 4$, поэтому назначение является полным, а план – оптимальным. Значение целевой функции равно: $f = c_{11} + c_{22} + c_{33} + c_{44} = 3 + 4 + 2 + 8 = 17$.

Таблица 1.5 – Таблица после перестановки

Кран	Объект, B_j				
A_i	B_1	B_2	B_3	B_4	a_i
A_1	0•	2	4	2	1
A_2	∅	0•	4	∅	1
A_3	∅	1	0•	1	1
A_4	4	∅	∅	0•	1
b_j	1	1	1	1	

2 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Изучить методы решения задачи об оптимальных назначениях.

2. Разработать программу решения задачи об оптимальных назначениях. Предусмотреть возможность использования произвольной матрицы тарифов.

3. Оформить отчет о выполнении задания с приведением условия задачи, описания алгоритма реализации и фрагментов программ, результатов работы программы и выводов.

3 ТРЕБОВАНИЕ К ОТЧЕТУ

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Краткие теоретические сведения.
3. Схема алгоритма.
4. Листинги основных частей программы.
5. Результаты работы программы.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Математическая постановка задачи об оптимальных назначениях.
2. Этапы решения задачи об оптимальных назначениях.
3. Как получить нули в каждой строке?

4. Как получить нули в каждом столбце?

5. Как пометить нули?

6. Как выполнить поиск минимального набора строк и столбцов, содержащих нули?

7. Как выполнить перестановке нулей?

8. Что такое матрица тарифов?

9. Что такое опорный план?

10. Как проверить план на оптимальность?

Вариант 1

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	3	7	5	8	7
A_2	8	4	4	5	1
A_3	4	7	4	4	4
A_4	9	7	7	7	2
A_5	10	12	7	7	3

Вариант 2

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	3	7	9	7	7
A_2	8	9	10	12	7
A_3	4	10	12	7	4
A_4	9	7	9	7	7
A_5	10	12	10	12	7

Вариант 3

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	3	7	9	7	7
A_2	8	9	10	12	7
A_3	4	10	12	7	4
A_4	9	7	9	7	7
A_5	10	12	10	12	7

Вариант 4

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	9	7	7
A_2	10	12	15	12	11
A_3	4	10	12	7	4
A_4	9	7	3	7	7
A_5	10	12	10	5	7

Вариант 5

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	3	7	5	8	7
A_2	8	9	7	7	11
A_3	4	10	12	7	4
A_4	9	7	5	8	2
A_5	10	9	7	7	3

Вариант 6

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	7	11	7
A_2	9	7	7	9	7
A_3	10	12	7	10	12
A_4	9	9	7	8	2
A_5	10	10	12	7	3

Вариант 7

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	7	5	8	8	7
A_2	9	7	7	5	13
A_3	4	7	7	5	8
A_4	9	7	9	7	7
A_5	10	12	7	7	3

Вариант 8

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	5	8	8	8	9
A_2	7	7	5	13	11
A_3	10	2	7	4	4
A_4	9	9	5	8	8
A_5	10	10	7	7	5

Вариант 9

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	5	8	8	8	9
A_2	7	3	7	5	8
A_3	3	8	9	7	7
A_4	8	4	10	12	7
A_5	4	10	12	7	5

Вариант 10

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	3	7	5	8	7
A_2	8	9	7	7	7
A_3	4	10	3	7	5
A_4	9	9	8	9	7
A_5	10	10	4	10	12

Вариант 11

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	5	8	8	8	9
A_2	7	3	7	5	8
A_3	10	8	3	7	5
A_4	9	4	8	9	7
A_5	10	10	4	10	12

Вариант 12

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	7	11	7
A_2	9	7	7	9	7
A_3	3	7	5	8	12
A_4	8	3	7	5	8
A_5	4	8	9	7	7

Вариант 13

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	5	8	3	7	5
A_2	3	7	8	8	7
A_3	8	9	4	7	12
A_4	4	10	12	7	8
A_5	10	10	7	7	5

Вариант 14

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	3	7	5	8	7
A_2	8	9	7	7	7
A_3	4	3	7	5	8
A_4	9	8	9	7	7
A_5	10	4	10	12	7

Вариант 15

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	5	8	8	8	9
A_2	5	13	5	13	11
A_3	7	4	5	13	4
A_4	5	8	7	4	8
A_5	10	10	5	8	5

Вариант 16

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	7	5	13
A_2	5	13	7	7	4
A_3	7	4	5	5	8
A_4	5	8	7	4	2
A_5	10	10	5	8	3

Вариант 17

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	5	13	7	8	7
A_2	7	7	5	8	7
A_3	13	9	7	7	7
A_4	13	10	3	7	5
A_5	10	9	8	9	7

Вариант 18

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	7	5	8	8	7
A_2	9	7	10	3	7
A_3	10	3	9	8	5
A_4	9	8	10	14	7
A_5	17	10	4	15	12

Вариант 19

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	7	11	7
A_2	9	7	7	9	7
A_3	5	13	5	13	7
A_4	7	7	7	7	5
A_5	13	7	13	7	7

Вариант 20

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	7	11	7
A_2	9	7	7	9	7
A_3	3	7	5	8	12
A_4	8	3	7	5	8
A_5	4	8	9	7	7

Вариант 21

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	3	7	5	8	7
A_2	8	9	5	13	7
A_3	4	3	7	7	5
A_4	9	8	13	7	7
A_5	10	4	13	7	7

Вариант 22

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	5	13	7	8	7
A_2	7	7	5	7	7
A_3	13	7	5	13	7
A_4	9	8	7	7	5
A_5	10	4	13	7	7

Вариант 23

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	7	5	13
A_2	5	13	7	7	4
A_3	7	4	7	7	8
A_4	5	8	13	7	2
A_5	10	10	5	8	3

Вариант 24

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	7	5	13
A_2	5	13	7	7	4
A_3	7	7	5	5	8
A_4	13	7	7	7	2
A_5	10	10	13	7	3

Вариант 25

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	5	13	7	8	7
A_2	7	7	5	8	7
A_3	13	9	9	7	7
A_4	13	8	3	7	5
A_5	8	9	8	9	7

Вариант 26

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	5	6	7	8	7
A_2	7	7	5	8	7
A_3	13	9	7	7	7
A_4	13	3	3	7	5
A_5	10	9	8	3	7

Вариант 27

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	7	11	7
A_2	9	8	7	9	7
A_3	5	2	5	13	7
A_4	7	7	7	4	5
A_5	13	7	13	7	7

Вариант 28

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	9	7	7	3	7
A_2	9	7	7	9	7
A_3	5	13	5	11	7
A_4	7	7	7	7	5
A_5	11	7	12	7	7

Вариант 29

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	3	7	5	8	7
A_2	8	4	5	7	7
A_3	4	3	7	7	5
A_4	9	8	11	7	7
A_5	10	4	12	7	7

Вариант 30

	B_j				
A_i	B_1	B_2	B_3	B_4	B_5
A_1	3	7	5	8	7
A_2	8	9	5	13	8
A_3	4	3	7	6	5
A_4	9	8	11	7	7
A_5	10	4	17	7	7

Лабораторная работа №7.
«Дискретное программирование. Задача о коммивояжере»
(8 часов)

Цель работы: знакомство с методами решения задач дискретного программирования.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Постановка задачи

Широкое распространение для решения задач целочисленного (частично целочисленного) программирования нашел «метод ветвей и границ». Термин «метод ветвей и границ» является собирательным и включает в себя целое семейство методов, объединяемых общими принципами. Конкретная реализация метода ветвей и границ связана с правилами разбиения на подмножества (правила ветвления) и построению оценок значений целевых функций на данных подмножествах (определение границ).

Пусть задана конечная область, в которой требуется найти набор переменных $X \in \Omega$, доставляющих максимум целевой функции $f(x)$. Во многих случаях можно определить верхнюю границу $f(x)$ на множестве планов ς. Обозначим ее через $\varphi(\Omega)$, то есть $f(X) \leq \varphi(\Omega)$. Разобьем ς на несколько непересекающихся подмножеств $\varsigma_1, \varsigma_2, ..., \varsigma_k$, то есть разветвим множество ς. На каждом из подмножеств оценим сверху функцию $f(x)$, т.е. определим числа $\alpha(\varsigma_1), \alpha(\varsigma_2), ..., \alpha(\varsigma_k)$. Если удастся найти некоторый набор переменных $X^o \in \Omega_p$, для которого $f(X^0) = \varphi(\Omega_p) \geq \varphi(\Omega_i), \quad i = \overline{1,k}$, то X^o оптимальное решение исходной задачи. Если такой план не обнаружен, то продолжим процесс разбиения, начиная с подмножества с самой высокой оценкой и т.д. Так как множество всех допустимых планов предполагается конечным, то после конечного числа шагов будет найден оптимальный план. В случае задачи линейного целочисленного программирования верхнюю границу находят решением соответствующей задачи с отброшенным условием целочисленности. Ветвление производится последовательным включением в условия дополнительных ограничений.

Рассмотрим работу метода на примере задачи о коммивояжере. Коммивояжер должен посетить каждый из n городов только один раз и вернуться в исходный пункт. Требуется найти маршрут минимальной длины.

Для некоторых пар (i, j) непосредственный переход от i к j может быть запрещен. В этом случае элемент матрицы C полагается равным бесконечности. В других случаях требуют, чтобы определенная дуга обязательно входила в маршрут.

На каждом шаге описываемого алгоритма задача включает n городов, причем из n шагов маршрута k шагов могут быть уже установлены и нужно выбрать оптимальным образом из оставшихся $n - k$ шагов. Формализуем задачу. Введем неизвестные величины:

$$x_{ij} = \begin{cases} 1, & \textit{коммивояжер из города } i \textit{ переезжает в город } j; \\ 0 & \textit{в противном случае.} \end{cases} \qquad (1.1)$$

Пусть $C = \|c_{ij}\|$ – матрица расстояний между городами. Тогда

$$F = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1.2)$$

есть длина маршрута для выбранного плана переездов. На матрицу $X = \|x_{ij}\|$ неизвестных должны быть наложены ограничения:

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = \overline{1, n}, \qquad (1.3)$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = \overline{1, n}, \qquad (1.4)$$

Система ограничений (1.3) обеспечивает построение маршрута, при котором коммивояжер въезжает в каждый город только раз, а система ограничений (1.4) – маршрута, когда он выезжает из каждого города только раз. Эти ограничения недостаточны, так как среди допускаемых ими решений имеются маршруты, не образующие полный цикл, включающий все города.

Устранение подциклов достигается при добавлении системы ограничений:

$$u_i - u_j + n x_{ij} \le n - 1, \quad i, j = \overline{1, n}, \quad i \ne j, \qquad (1.5)$$

где переменные u могут принимать произвольные вещественные значения.

Задача о коммивояжере в математической постановке эквивалентна задаче упорядочения конечного числа работ на машине при учете потерь от переналадок.

Для задачи упорядочения каждая работа соответствует городу, а длительность настройки – расстоянию между городами. Матрица C может быть и несимметричной, так как длительность настройки при переходе от работы i к работе j, как правило, отлична от длительности настройки при переходе от j к i.

1.2 Алгоритм Литтла

Алгоритм Литтла для решения задачи о коммивояжере можно сформулировать в виде следующих шагов.

*Шаг_*1. Находим в каждой строке матрицы $C = \|c_{ij}\|$ минимальный элемент $u_i = \min\limits_j c_{ij}$ и вычитаем его из всех элементов соответствующей строки. Получим матрицу, приведенную по строкам, с элементами $c_{ij}' = c_{ij} - \min\limits_j c_{ij}$.

*Шаг_*2. Если в матрице C', приведенной по строкам, окажутся столбцы, не содержащие нуля, то ее приводим по столбцам. Для этого в каждом столбце матрицы C' выбираем минимальный элемент v_j, $j = \overline{1,n}$ и вычитаем его из всех элементов соответствующего столбца. Получим матрицу $C'' = \left\| c_{ij} - \min\limits_j c_{ij} - \min\limits_i c_{ij}' \right\|$, каждая строка и каждый столбец которой содержит хотя бы один ноль. Такая матрица называется приведенной по строкам и столбцам.

*Шаг_*3. Суммируем элементы u_i и v_j, получим константу приведения $\gamma = \sum\limits_{i=1}^{n} u_i + \sum\limits_{j=1}^{n} v_j$, которая будет нижней границей множества всех допустимых гамильтоновых контуров, т.е. $\gamma = \varphi(\Omega^0) \le f(X)$.

Примечание. Контур – это конечный путь, у которого начальная вершина совпадает с конечной. При этом контур называется элементарным (гамильтоновым), если все его вершины различны, за исключением начальной и конечной, которые совпадают).

*Шаг*_4. Находим степени нулей для приведенной по строкам и столбцам матрицы. Для этого мысленно нули в матрице заменяем на $\propto$ и находим сумму минимальных элементов строки и столбца, соответствующих этому нулю. Записываем ее в правом верхнем углу клетки:

$$\theta_{ij} = \min_{j' \neq j} c_{ij'} + \min_{i' \neq i} c_{i'j} .$$

*Шаг*_5. Выбираем дугу (i_0, j_0), для которой степень нулевого элемента достигает максимального значения $\theta_{i_0 j_0} = \max_{c_{ij}''} \theta_{ij}$.

*Шаг*_6. Разбиваем множество всех гамильтоновых контуров Ω^0 на два подмножества $\Omega^1_{i_0 j_0}$ и $\Omega^1_{\overline{i_0 j_0}}$. Подмножество $\Omega^1_{i_0 j_0}$ гамильтоновых контуров содержит дугу (i_0, j_0), а $\Omega^1_{\overline{i_0 j_0}}$ – не содержит. Для получения матрицы контуров $\Omega^1_{i_0 j_0}$, включающих дугу (i_0, j_0), вычеркиваем в матрице C'' строку i_0 и столбец j_0. Чтобы не допустить образования негамильтонова контура, заменяем симметричный элемент (i_0, j_0) на $\propto$.

*Шаг*_7. Приводим матрицу гамильтоновых контуров $\Omega^1_{i_0 j_0}$. Пусть $h^1_{i_0 j_0}$ – константа ее приведения. Тогда нижняя граница множества $\Omega^1_{i_0 j_0}$ определяется как $\omega(\Omega^1_{i_0 j_0}) = \gamma + h^1_{i_0 j_0}$.

*Шаг*_8. Находим множество гамильтоновых контуров $\Omega^1_{\overline{i_0 j_0}}$, не включающих дугу (i_0, j_0). Исключение дуги (i_0, j_0) достигается заменой элемента $c''_{i_0 j_0}$ в матрице C'' на $\propto$.

*Шаг*_9. Делаем приведение матрицы гамильтоновых контуров $\Omega^1_{\overline{i_0 j_0}}$. Пусть $h^1_{\overline{i_0 j_0}}$ – константа ее приведения. Нижняя граница множества $\Omega^1_{\overline{i_0 j_0}}$ установится так: $\omega(\Omega^1_{\overline{i_0 j_0}}) = \gamma + h^1_{\overline{i_0 j_0}}\omega$.

*Шаг*_10. Сравниваем нижние границы подмножества гамильтоновых контуров $\Omega^1_{i_0 j_0}$ и $\Omega^1_{\overline{i_0 j_0}}$. Если $\omega(\Omega^1_{i_0 j_0}) < \omega(\Omega^1_{\overline{i_0 j_0}})$, то дальнейшему ветвлению в первую очередь подлежит подмножество $\Omega^1_{i_0 j_0}$. Если $\omega(\Omega^1_{i_0 j_0}) > \omega(\Omega^1_{\overline{i_0 j_0}})$, то ветвлению подлежит подмножество $\Omega^1_{\overline{i_0 j_0}}$ и т.д.

Процесс разбиения множеств на подмножества сопровождается построением дерева ветвления.

*Шаг*_11. Если в результате ветвления получаем размерность сокращенной матрицы 2×2, то определяем полученный ветвлением гамильтонов контур и его длину (рекорд).

*Шаг*_12. Сравниваем длину гамильтонова контура с нижними границами оборванных ветвей. Если длина контура не превышает их нижних границ, то задача решена. В противном случае развиваем ветви подмножеств с нижней границей, меньшей полученного контура, до тех пор, пока не получим маршрут с меньшей длиной или не убедимся, что такого не существует.

1.3 Пример работы алгоритма Литтла

Решить задачу о коммивояжере для матрицы расстояний, представленной в таблице 1.1.

Таблица 1.1 – Матрица расстояний Ω^0

$i \setminus j$	1	2	3	4	5	u_i
1	∞	30	40	15	6	6
2	10	∞	18	7	9	7
3	20	30	∞	1	10	1
4	25	10	35	∞	5	5
5	9	8	7	6	∞	6

*Шаг*_1. Справа к табл. 1.1 присоединим столбец u_i, в котором записываем минимальные элементы строк. Вычитаем элементы u_i из соответствующих элементов матрицы *C*, получим матрицу приведенную по строкам (таблица 1.2).

Таблица 1.2 – Матрица приведенная по строкам

$i \setminus j$	1	2	3	4	5
1	∞	24	34	9	0
2	3	∞	11	0	2
3	19	29	∞	0	9
4	20	5	30	∞	0
5	3	2	1	0	∞
v_j	3	2	1	0	0

Шаг_2. Внизу матрицы табл. 1.2 Присоединяем строку v_j, в которой записываем минимальные элементы столбцов. Вычитаем элементы v_j из соответствующих столбцов матрицы C' (таблица 1.3).

Таблица 1.3 – Матрица, приведенная по строкам и столбцам $\widetilde{\Omega}^0$

$i \setminus j$	1	2	3	4	5
1	$\propto$	22	33	9	0^9
2	0^0	$\propto$	10	0^0	2
3	16	27	$\propto$	0^9	9
4	17	3	29	$\propto$	0^3
5	0^0	0^3	0^{10}	0^0	$\propto$

Шаг_3. Вычислим константу приведения $\gamma = \sum_{i=1}^{5} u_i + \sum_{j=1}^{5} v_j = 25 + 6 = 31$.

Нижней границей множества всех гамильтоновых контуров будет число $\gamma = 31 = \varphi(\Omega^0) \le f(X)$.

Шаг_4. Находим степени нулей полностью приведенной матрицы табл. 1.3. Для этого мысленно заменяем в ней нули на $\propto$ и устанавливаем сумму минимальных элементов соответствующей строки и столбца. Степени нулей записаны в правых верхних углах клеток, для которых $c_{ij} = 0$.

Шаг_5. Определяем максимальную степень нуля. Она равна 10 и соответствует клетке $(5,3)$. Таким образом, претендентом на включение в гамильтонов контур является дуга $(5,3)$.

Шаг_6. Разбиваем множество всех гамильтоновых контуров Ω^0 на два Ω^1_{53} и $\Omega^1_{\overline{53}}$. Матрицу Ω^1_{53} с дугой (5,3) получаем из табл. 1.3 путем вычеркивания строки 5 и столбца 3. Чтобы не допустить образования негамильтонова контура, заменяем элемент (5,3) на $\propto$ (таблица 1.4).

Таблица 1.4 – Матрица Ω^1_{53} с дугой (5,3)

$i \setminus j$	1	2	4	5	u_i
1	$\propto$	22	9	0	0
2	0	$\propto$	0	2	0
3	16	27	0	$\propto$	0
4	17	3	$\propto$	0	0
v_j	0	3	0	0	0

Шаг_7. Матрицу гамильтоновых контуров $\Omega^1_{\overline{53}}$ получим из табл. 1.3 путем замены элемента c''_{53} на $\propto$ (таблица 1.5).

Таблица 1.5 – Матрица гамильтоновых контуров $\Omega^1_{\overline{53}}$

$i \setminus j$	1	2	3	4	5
1	$\propto$	22	33	9	0
2	0	$\propto$	10	0	2
3	16	27	$\propto$	0^9	9
4	17	3	29	$\propto$	0
5	0	0	$\propto$	0	$\propto$
v_j			10		

Шаг_8. Делаем дополнительное приведение матрицы контуров Ω^1_{53}: $h^1_{53} = 3$ (таблица 1.6). Нижняя граница множества Ω^1_{53} равна $\varphi(\Omega^1_{53}) = 31 + 3 = 34$.

Таблица 1.6 – Матрица, приведенная по строкам и столбцам Ω^1_{53}

$i \setminus j$	1	2	4	5
1	$\propto$	19	9	0^9
2	0^{16}	$\propto$	0^0	2
3	16	24	0^{16}	$\propto$
4	17	0^{19}	$\propto$	0^0

Шаг_9. Находим константу приведения для множества контуров $\Omega^1_{\overline{53}}$: $h^1_{\overline{53}}$=10. Следовательно, нижняя граница $\varphi(\Omega^1_{\overline{53}}) = 31 + 10 = 41$.

Шаг_10. Сравниваем нижние границы подмножеств Ω^1_{53} и $\Omega^1_{\overline{53}}$. Так как $\varphi(\Omega^1_{53}) = 34 < \varphi(\Omega^1_{\overline{53}}) = 41$, то дальнейшему ветвлению подвергаем множество Ω^1_{53} (табл. 1.4). На рисунке 1.1 представлено ветвление по дуге $(5,3)$. Переходим к ветвлению подмножества Ω^1_{53} (приведенная матрица представлена в табл. 1.6). Определим степени нулей этой матрицы. Претендентом на включение в гамильтонов контур будет дуга $(4,2)$. Разбиваем множество Ω^1_{53} на два подмножества Ω^2_{42} и $\Omega^2_{\overline{42}}$ (таблицы 1.7 и 1.8).

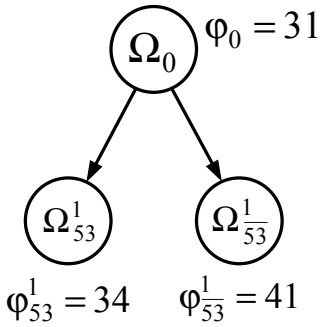

Рисунок 1.1 – Ветвление по дуге (5,3)

Таблица 1.7 – Матрица Ω^2_{42}

$i \setminus j$	1	4	5	u_i
1	∞	9	0^{11}	0
2	0^{18}	∞	2	0
3	16	0^{25}	∞	0
v_j	0	0	0	0

Таблица 1.8 – Матрица $\Omega^2_{\overline{42}}$

$i \setminus j$	1	2	4	5	u_i
1	∞	19	9	0	0
2	0	∞	0	2	0
3	16	24	0	∞	0
4	17	∞	∞	0	0
v_j	0	19	0	0	

Определяем константы приведения этих матриц: $h^2_{42} = 0$, $h^2_{\overline{42}} = 19$. Следовательно, $\varphi(\Omega^2_{42}) = 34$, $\varphi(\Omega^2_{\overline{42}}) = 53$. На рисунке 1.2 представлено ветвление с использованием дуги $(4,2)$. Так как $\varphi(\Omega^2_{42}) < \varphi(\Omega^2_{\overline{42}})$, то ветвлению подлежит подмножество Ω^2_{42} (табл. 1.7).

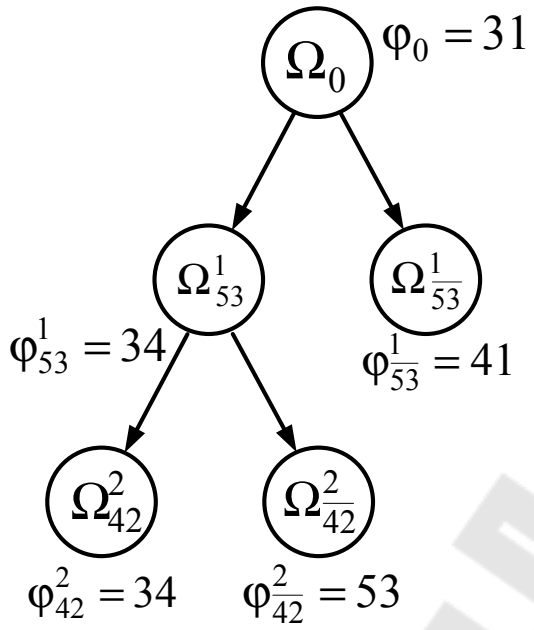

Рисунок 1.2 – Ветвление по дуге (4,2)

Вычислим степени нулей матрицы. Претендентом к включению в гамильтонов контур станет дуга $(3,4)$. Разбиваем множество Ω^2_{42} на подмножества Ω^3_{34} и $\Omega^3_{\overline{34}}$ (таблица 1.9 и 1.10). Очевидно, $\varphi(\Omega^3_{34}) = 34,\ \varphi(\Omega^3_{\overline{34}}) = 34 + 25 = 59$. Следовательно, ветвлению нужно подвергнуть подмножество Ω^3_{34}. Но его матрица имеет размерность $2x2$. Поэтому в гамильтонов контур следует включить дуги, соответствующие в матрице подмножества Ω^3_{34} нулевым элементам, т.е. дуги (1,5), (2,1).

Таблица 1.9 – Матрица Ω^3_{34}

$i \setminus j$	1	5	u_i
1	∞	0	0
2	0	2	0
v_j	0	0	

Таблица 1.10 – Матрица $\Omega^3_{\overline{34}}$

$i \setminus j$	1	4	5	u_i
1	∞	9	0	0
2	0	∞	2	0
4	17	∞	∞	16
v_j	0	9	0	

На рисунке 1.3 представлено дерево ветвлений. Определим полученный гамильтонов контур. В него вошли дуги $\{(5,3),(4,2),(3,4),(1,5),(2,1)\}$. Длина контура равна

$$c_{53} + c_{34} + c_{42} + c_{21} + c_{15} = 7 + 1 + 10 + 10 + 6 = 34.$$

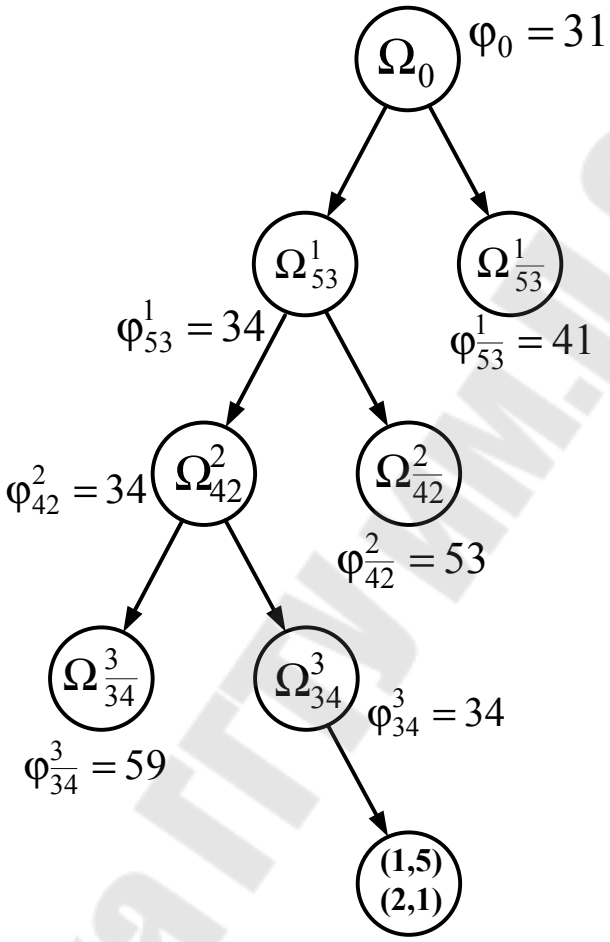

Рисунок 1.3 – Дерево ветвлений

Так как границы оборванных ветвей больше длины контура $5-3-4-2-1-5$, то этот контур имеет наименьшую длину. На рисунке 1.4 представлен замкнутый маршрут минимальной длины.

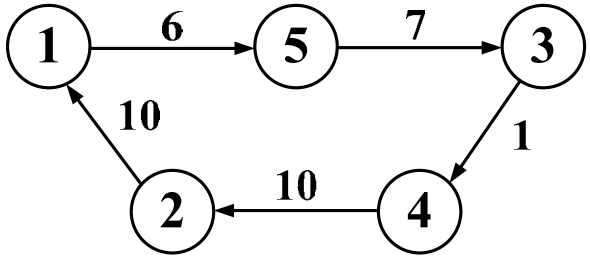

Рисунок 1.4 – Замкнутый маршрут минимальной длины

1.4 Решение задачи коммивояжера методом «ближайшего соседа»

Алгоритм «ближайшего соседа» состоит в последовательном добавлении к начальной вершине ближайшей к ней. Оптимальность решения зависит от выбора начальной точки. Поэтому алгоритм целесообразно применять, начиная с каждой вершины, и затем выбрать замкнутый контур, наименьшей длины. Отметим, что если ближайший сосед для некоторой вершины уже вошел в контур, то берется следующая по близости вершина и т.д. С увеличением размерности задачи n ошибка решения убывает.

При $n \leq 40$ целесообразно использовать точные методы, а при $n > 40$ – приближенные методы (например, «ближайшего соседа»).

Пример. Решить задачу о коммивояжере (табл.1.1) методом «ближайшего соседа».

Начнем с первой вершины: $\min c_{ij} = \min c_{15} = 6$. Вычеркиваем в таблице первую строку и пятый столбец. Среди оставшихся значений: $\min c_{ij} = \min c_{54} = 6$. Но аналогии:

$\min c_{ij} = \min c_{42} = 10$,

$\min c_{ij} = \min c_{23} = 18$.

Возвращаемся в город, с которого началось движение: $c_{31} = 20$. Получили следующий маршрут: $1 - 5 - 4 - 2 - 3 - 1$. Значение целевой функции равно:

$c_{15} + c_{54} + c_{42} + c_{23} + c_{31} = 6 + 6 + 10 + 18 + 20 = 60$.

2 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Изучить методы решения задач дискретного программирования – метод ветвей и границ и метод ближайшего соседа.

2. Разработать программы, решающие задачу о коммивояжере методом ветвей и границ и методом ближайшего соседа.

3. Решить задачу из задания 1 на бумаге сравнить полученные результаты.

4. Решить задачи из задания 1 и 2 при помощи разработанных программ.

5. Оформить отчет, содержащий:

–краткие теоретические сведения;

–решение задачи 1 на бумаге;

–решение задачи 1 и задачи 2 программой, реализующей метод ветвей и границ;

–решение задачи 1 и задачи 2 программой, реализующей метод ближайшего соседа;

–оценку эффективности решения задач различными методами.

3 ТРЕБОВАНИЕ К ОТЧЕТУ

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Краткие теоретические сведения.
3. Схема алгоритма.
4. Листинги основных частей программы.
5. Результаты работы программы.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Математическая постановка задачи дискретного программирования.
2. Методы решения задачи дискретного программирования.
3. Этапы решения задачи дискретного программирования.
4. Суть методов ветвей и границ.
5. Алгоритм Литтла.
6. Метод ближайшего соседа.
7. Когда целесообразно применять методов ветвей и границ?
8. Когда целесообразно применять метод ближайшего соседа?
9. Математическая постановка задачи о коммивояжере.

Варианты заданий

ЗАДАНИЕ 1

Вариант 1:

∞	11	25	21	3
1	∞	10	12	5
8	10	∞	21	11
14	13	24	∞	15
10	17	25	27	∞

Вариант 2:

∞	18	25	23	7
3	∞	10	13	9
7	9	∞	22	10
12	15	24	∞	15
11	9	25	23	∞

Вариант 3:

∞	11	25	21	3
5	∞	12	12	8
8	12	∞	21	18
14	13	22	∞	15
10	19	25	27	∞

Вариант 4:

∞	18	25	23	7
2	∞	18	13	3
7	9	∞	22	10
12	15	26	∞	5
11	9	25	23	∞

Вариант 5:

∞	11	25	23	3
4	∞	10	12	5
8	15	∞	21	14
4	17	24	∞	15
10	17	25	22	∞

Вариант 6:

∞	18	25	23	7
6	∞	4	13	9
7	9	∞	22	9
12	17	12	∞	15
11	9	29	23	∞

Вариант 7:

∞	11	25	1	12
5	∞	10	12	6
2	10	∞	21	11
14	13	24	∞	15
7	19	25	27	∞

Вариант 8:

∞	8	25	23	7
3	∞	10	23	9
7	9	∞	12	10
12	5	24	∞	5
11	14	25	13	∞

Вариант 9:

∞	17	27	21	3
1	∞	15	22	5
8	10	∞	21	11
14	23	24	∞	15
10	7	25	27	∞

Вариант 10:

∞	18	5	23	7
3	∞	10	13	9
7	9	∞	12	10
12	15	4	∞	5
21	11	5	23	∞

Вариант 11:

∞	19	25	21	8
1	∞	10	12	8
8	19	∞	21	19
14	15	4	∞	15
10	17	15	27	∞

Вариант 12:

∞	18	5	6	7
3	∞	10	23	9
7	9	∞	22	11
12	15	4	∞	15
11	9	25	3	∞

Вариант 13:

∞	17	27	21	3
1	∞	15	22	5
8	10	∞	21	11
14	23	24	∞	15
10	7	25	27	∞

Вариант 14:

∞	18	5	23	7
3	∞	10	13	9
7	9	∞	12	10
12	15	4	∞	5
21	11	5	23	∞

Вариант 15:

∞	19	25	21	18
1	∞	15	12	8
8	9	∞	2	19
14	15	4	∞	5
10	17	15	27	∞

Вариант 16:

∞	8	9	6	7
4	∞	10	23	9
17	9	∞	2	11
12	15	4	∞	5
11	9	25	23	∞

Вариант 17:

∞	17	7	21	3
1	∞	5	22	5
8	10	∞	21	11
14	23	24	∞	15
10	7	25	27	∞

Вариант 18:

∞	8	5	23	7
3	∞	10	13	9
7	9	∞	11	10
12	15	2	∞	5
21	3	5	23	∞

Вариант 19:

∞	11	25	29	3
1	∞	19	12	5
8	10	∞	21	11
14	13	24	∞	15
10	17	25	27	∞

Вариант 20:

∞	18	21	22	7
8	∞	12	13	9
7	9	∞	22	10
12	15	24	∞	5
11	19	25	23	∞

Вариант 21:

∞	11	29	21	9
19	∞	12	12	8
8	12	∞	21	18
14	13	22	∞	15
10	19	25	27	∞

Вариант 22:

∞	18	29	23	7
15	∞	18	13	3
17	9	∞	22	10
12	15	26	∞	5
11	9	25	23	∞

Вариант 23:

∞	11	25	23	3
14	∞	10	12	5
18	15	∞	21	14
14	17	24	∞	15
10	17	25	22	∞

Вариант 24:

∞	18	25	23	7
6	∞	14	13	9
7	19	∞	22	9
12	17	12	∞	15
11	9	29	23	∞

Вариант 25:

∞	17	9	21	8
19	∞	10	12	18
8	19	∞	21	19
14	15	14	∞	15
10	17	5	17	∞

Вариант 26:

∞	18	5	6	9
13	∞	10	23	9
17	19	∞	22	17
12	19	14	∞	15
11	9	5	13	∞

Вариант 27:

∞	17	27	21	13
19	∞	18	22	15
8	10	∞	21	11
14	23	27	∞	15
10	17	25	29	∞

Вариант 28:

∞	18	5	23	7
13	∞	17	13	9
7	9	∞	12	10
12	17	4	∞	5
21	19	15	29	∞

Вариант 29:

∞	19	25	21	18
1	∞	15	13	8
8	9	∞	2	11
14	15	4	∞	5
10	17	17	27	∞

Вариант 30:

∞	7	9	6	7
14	∞	10	27	9
17	9	∞	2	11
12	17	4	∞	5
11	9	25	23	∞

ЗАДАНИЕ 2

Вариант 1:

∞	8	12	7	5	0	11	5	13	9	18
10	∞	14	4	7	4	10	10	6	6	4
4	16	∞	13	3	2	5	5	5	7	11
3	7	11	∞	7	6	14	3	3	8	8
11	15	18	12	∞	19	12	13	11	16	1
8	7	16	19	1	∞	3	16	12	11	5
5	10	8	0	17	10	∞	6	13	1	5
6	7	6	5	1	5	17	∞	7	14	11
19	8	4	19	13	2	5	14	∞	12	15 6
11	8	8	3	4	3	4	11	2	∞	4
9	6	12	0	18	13	14	3	12	16	∞

Вариант 2:

∞	8	12	7	5	4	11	13	9	18	1
10	∞	14	4	7	4	10	10	6	4	3
4	16	∞	13	3	2	5	5	5	11	19
3	7	11	∞	7	6	14	3	3	8	18
11	15	18	12	∞	19	12	13	15	17	1
8	7	16	19	1	∞	16	12	11	7	5
5	10	8	6	17	10	∞	13	9	2	6
6	7	6	5	1	5	17	∞	14	11	5
19	8	4	19	19	2	5	14	∞	15	16
11	8	8	3	4	3	4	11	2	∞	15
9	6	12	3	18	13	14	3	12	16	∞

Вариант 3:

∞	8	12	7	5	4	1	5	13	9	18
10	∞	4	7	4	10	10	6	6	4	3
4	6	∞	3	2	5	9	5	7	11	9
3	7	11	∞	6	14	3	3	8	8	18
11	15	18	12	∞	12	13	15	7	1	12
8	7	14	19	1	∞	16	12	11	7	5
5	10	8	6	17	10	∞	13	1	2	6
6	7	6	5	1	5	17	∞	14	11	5
19	8	4	19	19	2	5	14	∞	15	6
11	8	8	3	2	3	4	11	2	∞	15
9	6	12	3	18	13	14	3	12	6	∞

Вариант 4:

∞	12	7	5	8	11	5	13	9	18	1
10	∞	4	7	4	10	10	6	6	4	3
4	16	∞	3	2	5	5	5	7	11	19
3	7	11	∞	6	14	3	3	8	8	18
11	15	18	12	∞	12	13	11	16	1	12
8	7	16	9	1	∞	16	12	11	4	5
5	10	8	7	17	10	∞	13	1	5	6
6	7	6	5	1	5	17	∞	14	11	5
19	8	4	19	13	2	5	14	∞	15	16
11	8	8	3	4	3	4	11	2	∞	15
9	6	12	6	18	13	14	3	12	16	∞

Вариант 5:

∞	8	7	5	4	11	5	13	9	18	1
10	∞	14	7	4	15	10	6	6	4	3
4	16	∞	13	3	5	5	5	7	11	9
3	7	11	∞	7	6	3	3	8	8	8
11	15	8	12	∞	19	12	15	17	1	12
8	7	6	19	1	∞	3	16	11	7	5
5	10	8	6	17	10	∞	6	13	2	6
6	7	6	5	1	5	7	∞	7	14	11
19	8	4	19	9	2	5	4	∞	12	15
11	8	8	3	4	3	4	11	2	∞	15
9	6	12	3	18	13	14	3	12	16	∞

Вариант 6:

∞	8	12	7	5	4	5	13	9	18	1
10	∞	4	4	7	10	16	6	6	4	3
4	6	∞	13	3	5	9	5	7	11	9
3	7	11	∞	7	6	14	3	8	8	8
11	15	8	2	∞	19	12	13	15	7	12
8	7	14	19	1	∞	16	12	11	7	5
5	10	8	6	17	10	∞	3	13	2	6
6	7	6	5	1	5	7	∞	14	11	5
19	8	4	19	19	2	5	14	∞	12	15
11	8	8	3	2	3	4	11	2	∞	5
9	6	12	3	18	13	14	3	12	6	∞

Вариант 7:

∞	8	12	7	5	8	11	5	13	9	18
10	∞	14	4	7	4	17	18	6	6	3
4	16	∞	13	3	5	5	5	7	11	19
3	7	11	∞	7	6	14	3	8	8	18
11	15	18	12	∞	9	12	13	11	16	12
8	7	16	9	1	∞	3	16	12	11	5
5	10	8	7	7	10	∞	6	13	5	6
6	7	6	5	5	17	7	∞	14	11	5
9	8	4	19	3	5	14	12	∞	15	6
3	8	8	3	4	3	4	11	2	∞	15
9	6	12	6	18	13	14	3	12	16	∞

Вариант 8:

∞	14	4	7	4	5	10	6	6	4	3
16	∞	13	3	2	5	5	5	7	11	9
7	11	∞	7	6	14	3	3	8	8	8
11	8	12	∞	19	12	13	15	17	1	2
7	6	19	1	∞	3	16	12	11	7	5
10	8	6	17	10	∞	6	13	1	2	6
7	6	5	1	5	7	∞	7	4	11	5
8	4	19	9	2	5	4	∞	12	15	6
11	8	8	4	3	4	11	2	∞	4	15
9	6	12	18	13	4	3	12	16	∞	4
8	10	8	18	17	6	19	7	0	12	∞

Вариант 9:

10	∞	4	4	7	5	10	16	6	6	4
6	∞	13	3	2	5	9	5	7	11	9
7	11	∞	7	6	4	3	3	8	8	8
11	15	8	∞	9	12	13	15	7	1	2
8	14	9	11	∞	3	16	12	11	7	5
10	8	6	7	10	∞	3	13	1	2	6
6	7	6	5	5	7	∞	7	14	11	5
19	9	4	19	19	5	14	∞	12	15	6
11	8	8	3	3	4	11	2	∞	4	5
9	6	12	18	13	14	3	12	6	∞	8
18	13	8	8	7	6	19	7	0	12	∞

Вариант 10:

∞	8	12	7	15	8	11	5	13	9	18
10	∞	14	4	7	4	7	18	6	6	4
4	16	∞	13	12	5	5	5	7	11	9
3	7	11	∞	7	6	4	3	8	8	8
13	5	8	12	∞	9	12	13	11	16	12
8	7	16	9	1	∞	16	2	11	4	5
5	10	8	7	7	10	∞	6	13	5	6
9	8	4	19	12	5	14	∞	12	15	6
3	8	8	13	4	4	11	2	∞	4	5
9	6	12	6	18	13	14	12	16	∞	4
8	10	8	18	17	16	19	7	11	12	∞

Вариант 11:

∞	8	12	7	5	11	5	13	9	8	11
10	∞	14	4	7	4	5	10	6	6	4
4	6	∞	13	3	5	5	5	7	11	9
3	7	11	∞	7	6	14	3	8	8	18
11	5	8	12	∞	19	12	13	15	17	12
8	7	6	19	13	∞	16	12	11	7	5
6	7	6	5	12	7	∞	7	4	11	5
9	8	14	19	9	5	4	∞	12	15	6
11	8	8	3	4	4	11	2	∞	4	15
9	6	12	3	18	13	4	12	16	∞	4
8	10	8	3	18	17	6	19	7	12	∞

Вариант 12:

∞	8	12	7	5	4	5	13	9	18	1
10	∞	4	4	7	5	10	16	6	6	4
4	6	∞	3	3	5	9	5	7	11	19
3	7	11	∞	7	6	4	5	8	8	8
11	15	8	2	∞	9	12	13	15	7	1
8	7	14	9	11	∞	16	12	11	6	15
5	10	8	6	7	10	∞	3	13	2	6
6	7	6	5	1	5	7	∞	7	14	11
11	8	8	3	2	3	4	11	∞	4	15
9	6	19	3	18	13	4	12	6	∞	8
18	13	8	8	7	6	19	7	0	12	∞

Вариант 13:

∞	8	12	7	15	8	11	13	9	18	9
14	∞	14	4	7	4	7	18	6	6	13
4	16	∞	13	13	12	5	5	7	11	9
13	7	11	∞	7	6	4	3	8	8	8
17	5	8	12	∞	9	12	13	11	16	19
18	7	16	9	14	∞	3	16	11	4	15
15	10	8	7	7	10	∞	6	13	5	6
6	7	6	5	17	5	17	∞	7	14	11
9	8	4	19	3	12	5	14	∞	12	15
3	8	8	13	4	3	4	11	2	∞	5
8	10	8	18	17	16	19	7	11	12	∞

Вариант 14:

∞	8	12	7	5	5	11	13	9	8	18
11	∞	14	4	7	4	5	10	6	6	4
14	6	∞	13	3	5	5	5	7	11	9
13	7	15	∞	7	6	14	3	8	8	18
11	5	8	2	∞	19	12	13	15	17	2
8	7	6	19	13	∞	16	12	11	7	5
15	11	18	6	7	10	∞	6	13	2	6
6	7	6	5	12	5	7	∞	7	11	11
11	8	8	4	3	4	11	2	∞	4	17
9	6	12	18	13	4	3	12	16	∞	4
8	10	8	18	17	6	19	7	0	12	∞

Вариант 15:

∞	8	2	7	5	4	14	5	13	9	18
10	∞	4	4	7	5	10	16	6	6	4
4	6	∞	3	3	5	9	5	7	11	9
3	7	11	∞	7	6	4	3	5	8	8
19	15	8	2	∞	9	12	13	15	7	2
8	7	4	9	15	∞	16	12	11	6	5
5	10	8	6	7	14	∞	3	13	2	6
16	7	6	5	1	5	7	∞	7	14	11
19	9	4	19	19	2	5	14	∞	12	6
11	8	8	3	2	3	4	11	2	∞	15
9	6	19	3	18	13	4	3	12	6	∞

Вариант 16:

∞	8	12	7	5	8	11	5	13	9	18	1
10	∞	14	4	7	4	10	10	6	6	4	3
4	16	∞	13	3	5	5	5	7	11	19	
3	7	11	∞	7	6	14	3	8	8	18	
11	15	18	12	∞	19	12	13	11	16	12	
8	7	16	9	1	∞	3	16	12	11	5	
5	10	8	7	17	10	∞	6	13	5	6	
6	7	6	5	1	5	17	∞	7	14	11	
19	8	4	19	13	5	14	12	∞	15	16	
11	8	8	3	4	4	11	2	4	∞	15	
9	6	12	6	18	13	14	3	12	16	∞	

Вариант 17:

∞	8	12	7	5	4	11	5	13	9	18
10	∞	14	4	7	4	15	10	6	6	4
4	16	∞	13	3	5	5	5	7	11	9
3	7	11	∞	7	6	14	3	8	8	8
11	15	8	12	∞	19	12	13	15	17	12
8	7	6	19	3	∞	16	12	11	7	5
5	10	8	6	17	10	∞	6	13	1	6
6	7	6	5	5	7	7	∞	14	11	5
19	8	4	19	9	5	4	12	∞	15	6
11	8	8	3	4	3	4	11	2	∞	15
9	6	12	3	18	13	14	3	12	16	∞

Вариант 18:

∞	4	4	7	4	10	16	6	6	4	3
6	∞	13	3	2	5	9	5	7	11	9
7	11	∞	7	6	14	3	3	8	8	8
11	15	8	∞	19	12	13	15	7	1	12
8	7	14	19	∞	3	16	12	11	7	5
10	8	6	17	10	∞	3	13	1	2	6
6	7	6	5	5	7	∞	7	14	11	5
19	8	4	19	19	5	14	∞	12	15	6
11	8	8	3	3	4	11	2	∞	4	5
9	6	12	18	13	14	3	12	6	∞	8
18	13	8	8	17	16	19	7	0	12	∞

Вариант 19:

∞	14	4	7	14	17	18	6	6	4	13
16	∞	13	3	12	5	5	5	7	11	19
7	11	∞	7	6	14	3	3	8	8	18
11	15	18	∞	12	9	12	13	11	16	12
8	7	16	9	∞	3	16	12	11	4	5
10	8	7	7	10	∞	6	13	1	5	6
6	7	6	5	5	17	∞	7	14	11	5
9	8	4	19	3	5	14	∞	12	15	6
3	8	8	3	4	4	11	2	∞	4	15
9	6	12	6	18	13	14	12	16	∞	4
8	10	8	18	17	16	19	7	11	12	∞

Вариант 20:

∞	14	4	7	4	5	10	6	6	4	9
16	∞	13	3	2	5	5	5	7	11	9
7	11	∞	7	6	14	3	3	8	8	8
11	5	8	∞	12	19	12	13	15	17	2
8	7	6	19	∞	3	16	12	11	7	5
5	10	8	6	17	∞	10	6	13	2	6
6	7	6	5	5	7	∞	7	4	11	5
9	8	4	19	9	5	4	∞	12	15	6
11	8	8	3	4	4	11	2	∞	4	15
9	6	12	3	18	13	4	12	16	∞	4
8	10	8	18	17	6	19	7	0	12	∞

Вариант 21:

∞	8	12	7	5	4	5	13	9	18	11
10	∞	4	4	7	5	10	16	6	6	4
4	6	∞	13	3	5	9	5	7	11	9
3	7	11	∞	7	6	4	3	8	8	8
11	15	8	2	∞	9	12	13	15	7	2
8	7	14	9	11	∞	16	12	11	7	5
5	10	8	6	7	10	∞	3	13	2	6
6	7	6	5	5	7	7	∞	14	11	5
19	9	4	19	19	5	14	12	∞	15	6
11	8	8	3	3	4	11	2	4	∞	5
9	6	12	3	18	13	14	3	12	6	∞

Вариант 22:

∞	8	12	7	15	8	11	5	13	9	18
10	∞	14	4	7	4	7	18	6	6	4
4	16	∞	13	3	12	5	5	5	7	11
3	7	11	∞	7	6	4	3	3	8	8
13	5	8	12	∞	9	12	13	11	16	12
8	7	16	9	1	∞	3	16	11	4	5
5	10	8	7	7	10	∞	6	13	5	6
6	7	6	5	5	17	7	∞	14	11	5
9	8	4	19	12	5	14	12	∞	15	6
3	8	8	13	4	3	4	11	2	∞	5
9	6	12	6	18	13	14	3	12	16	∞

Вариант 23:

∞	8	12	7	5	5	11	5	9	8	11
10	∞	14	4	7	4	5	10	6	6	4
4	6	∞	13	3	5	5	5	7	11	9
3	7	11	∞	7	6	14	3	8	8	18
11	5	8	12	∞	19	12	13	15	17	2
8	7	6	19	13	∞	16	12	11	7	5
15	10	18	6	7	10	∞	6	13	2	16
6	7	6	5	12	5	7	∞	7	4	5
9	8	14	19	9	2	5	4	∞	12	15
11	8	8	3	4	3	4	11	2	∞	4

Вариант 24:

∞	8	12	7	5	4	1	5	13	9	18
10	∞	4	4	7	5	10	16	6	6	4
4	6	∞	3	3	5	9	5	7	11	19
3	7	11	∞	7	6	4	3	5	8	8
11	15	8	2	∞	9	12	13	15	7	1
8	7	14	9	11	∞	16	12	11	6	15
5	10	8	6	7	10	∞	3	13	2	6
19	9	4	19	19	5	14	∞	12	5	6
11	8	8	3	3	4	11	2	∞	4	15
9	6	19	3	18	13	4	12	6	∞	8
18	13	8	3	8	7	6	19	7	12	∞

Вариант 25:

∞	8	12	7	15	8	11	5	13	9	18	9
14	∞	14	4	7	4	7	18	6	6	4	13
4	16	∞	13	13	12	5	5	5	7	11	9
13	7	11	∞	7	6	4	3	3	8	8	8
17	5	8	12	∞	9	12	13	11	16	1	19
18	7	16	9	14	∞	3	16	2	11	4	15
15	10	8	7	7	10	∞	6	13	1	5	6
6	7	6	5	17	5	17	∞	7	14	11	5
9	8	4	19	3	12	5	14	∞	12	15	6
3	8	8	13	4	3	4	11	2	∞	4	5
9	6	12	6	18	13	14	3	12	16	∞	14
8	10	8	3	18	17	16	19	7	11	12	∞

Вариант 26:

∞	14	4	7	4	5	10	6	6	4	3
14	∞	13	3	2	5	5	5	7	11	9
13	15	∞	7	6	14	3	3	8	8	18
11	5	8	∞	19	12	13	15	17	1	2
8	6	19	13	∞	3	16	12	11	7	5
15	11	6	7	10	∞	6	13	1	2	6
6	7	6	12	5	7	∞	7	4	11	5
9	8	4	9	2	5	4	∞	12	15	6
11	8	8	3	4	4	11	2	∞	4	17
9	6	12	3	18	4	3	12	16	∞	4
8	10	8	3	18	17	6	19	7	12	∞

Вариант 27:

∞	8	2	7	5	4	14	5	13	9	18
10	∞	4	4	7	5	10	16	6	6	4
4	6	∞	3	3	2	5	9	5	7	11
3	7	11	∞	7	6	4	3	5	8	8
19	15	8	2	∞	9	12	13	15	7	1
8	7	4	9	15	∞	3	16	12	11	6
5	10	8	6	7	14	∞	3	13	1	2
16	7	6	5	1	5	7	∞	7	14	11
19	9	4	19	19	2	5	14	∞	12	5
11	8	8	3	2	3	4	11	2	∞	4
9	6	19	3	18	13	4	3	12	6	∞

Вариант 28:

∞	8	12	7	15	8	11	5	13	9	18
10	∞	14	4	7	4	7	18	6	6	4
4	16	∞	13	7	12	5	5	5	7	11
3	7	11	∞	7	6	4	3	3	8	8
13	5	8	12	∞	9	12	13	11	16	12
8	7	16	9	7	∞	3	16	11	4	5
5	10	8	7	7	10	∞	6	13	5	6
6	7	6	5	5	17	5	∞	14	11	5
9	8	4	19	12	5	14	12	∞	15	6
3	8	8	13	4	9	4	11	2	∞	5
9	6	12	6	18	13	14	3	12	16	∞

Вариант 29:

∞	8	12	7	5	7	11	5	9	8	11
10	∞	14	4	7	8	5	10	6	9	4
4	6	∞	13	3	5	5	5	7	11	9
3	7	11	∞	3	6	14	3	8	8	18
11	5	8	12	∞	19	12	13	15	17	2
8	7	6	19	13	∞	16	12	11	7	5
15	10	18	6	7	10	∞	6	13	2	16
6	7	6	5	12	5	7	∞	7	4	5
9	8	14	9	9	2	5	4	∞	12	15
11	8	8	3	4	3	4	11	2	∞	4

Вариант 30:

∞	8	12	7	5	4	11	5	13	9	18
10	∞	4	14	7	5	10	16	16	6	4
4	6	∞	13	3	5	9	5	7	11	19
3	7	11	∞	7	6	4	3	5	8	8
11	15	8	12	∞	9	12	13	15	7	11
8	7	14	9	11	∞	16	12	11	6	15
5	10	8	6	7	10	∞	3	13	2	6
19	9	7	19	19	5	14	∞	12	5	6
11	8	8	3	3	4	11	2	∞	4	15
9	6	19	3	18	13	4	12	6	∞	8
18	13	8	3	8	7	6	19	7	12	∞

Проверка (11 городов).

	1	2	3	4	5	6	7	8	9	10	11
1)	∞	1	14	18	11	5	13	18	17	5	11
2)	4	∞	19	14	5	3	6	15	14	15	14
3)	12	6	∞	16	19	15	6	2	12	15	8
4)	14	4	18	∞	15	0	18	13	6	2	8
5)	19	15	19	14	∞	12	9	15	3	11	16
6)	10	6	11	4	15	∞	10	9	0	9	6
7)	16	0	10	17	18	6	∞	4	4	1	0
8)	7	17	17	6	7	12	10	∞	14	9	17
9)	19	5	7	6	16	4	6	17	∞	13	14
10)	2	11	11	16	12	7	14	12	15	∞	0
11)	1	14	10	0	10	3	1	0	5	6	∞

Оптимальный маршрут:

1 – 2 – 5 – 7 – 10 – 11 – 4 – 6 – 9 – 3 – 8 – 1.

Длина оптимального маршрута: 32.

Лабораторная работа №8.
«Поиск кратчайшего пути. Алгоритм Дейкстры» (4 часа)

Цель работы: знакомство с методами решения оптимизационных задач на графах.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Общая постановка задачи

Задачи, связанные с нахождением кратчайших путей, ассоциируются с решением проблем, требующих экономии затрат [8]. Например, имеется карта автомобильных дорог. Требуется минимизировать стоимость транспортировки груза из одного города в другой.

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

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

Алгоритм Дейкстры решает задачу о кратчайших путях из одной вершины для взвешенного ориентированного графа $G = (V, E)$, в котором веса всех рёбер неотрицательны, из исходной вершиной *s* в любую конечную.

Каждой вершине из V сопоставим метку – минимальное известное расстояние от этой вершины до *a*. Алгоритм работает пошагово – на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Инициализация. Метка самой вершины *a* полагается равной 0, метки остальных вершин – бесконечности. Это отражает то, что рас-

стояния от *a* до других вершин пока неизвестны. Все вершины графа помечаются как непосещенные.

Шаг алгоритма. Если все вершины посещены, алгоритм завершается. В противном случае из еще не посещенных вершин выбирается вершина *u*, имеющая минимальную метку. Рассматриваются всевозможные маршруты, в которых *u* является предпоследним пунктом. Вершины, соединенные с вершиной *u* ребрами, назовем соседями этой вершины. Для каждого соседа рассмотрим новую длину пути, равную сумме текущей метки *u* и длины ребра, соединяющего *u* с этим соседом. Если полученная длина меньше метки соседа, заменим метку этой длиной. Рассмотрев всех соседей, пометим вершину *u* как посещенную и повторим Шаг для следующей непосещенной вершины.

Рассмотрим работу алгоритма на примере графа, показанного на рисунке 1.1. Пусть требуется найти расстояния от первой вершины до всех остальных.

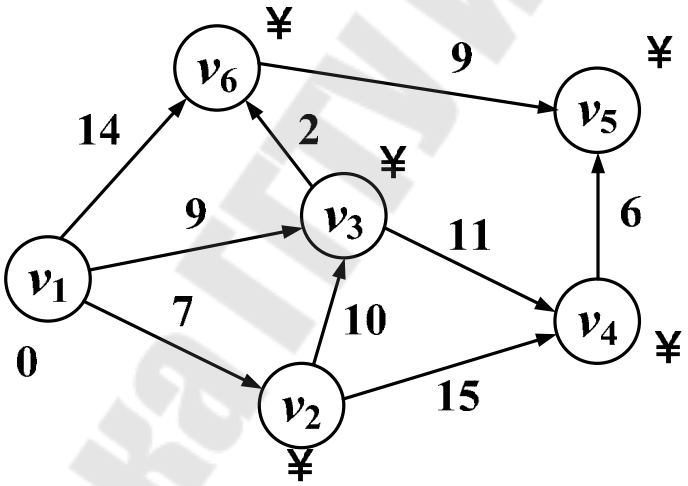

Рисунок 1.1 – Исходный граф

Номер вершины (v_1 - v_6) указан внутри вершины, над дугами указана их «цена» (длина пути, например). Рядом с каждой вершиной обозначена метка – длина кратчайшего пути из первой вершины в эту вершину.

Вначале все вершины графа помечаются как непосещенные (расстояние от первой вершины до текущей пока неизвестно), то есть метки вершин равны бесконечности, за исключением первой вершины, метка которой равна нулю (рис.1.1).

Первый шаг.

Минимальную метку имеет вершина v_1. Ее соседями являются вершины v_2, v_3 и v_6. Первый по очереди сосед вершины v_1 – вершина v_2, имеющая минимальную длину пути. Длина пути в нее через вершину v_1 равна сумме кратчайшего расстояния до вершины v_1 и длины ребра, идущего из v_1 в v_2, то есть $0 + 7 = 7$. Это меньше текущей метки вершины v_2, поэтому новая метка вершины v_2 равна 7. Аналогичная операция для v_3 и v_6. Все соседи вершины v_1 проверены. Текущее минимальное расстояние до вершины v_1 считается окончательным и пересмотру не подлежит (то, что это действительно так, впервые доказал Дейкстра). Вычеркнем её из графа, чтобы отметить, что эта вершина посещена (рисунок 1.2).

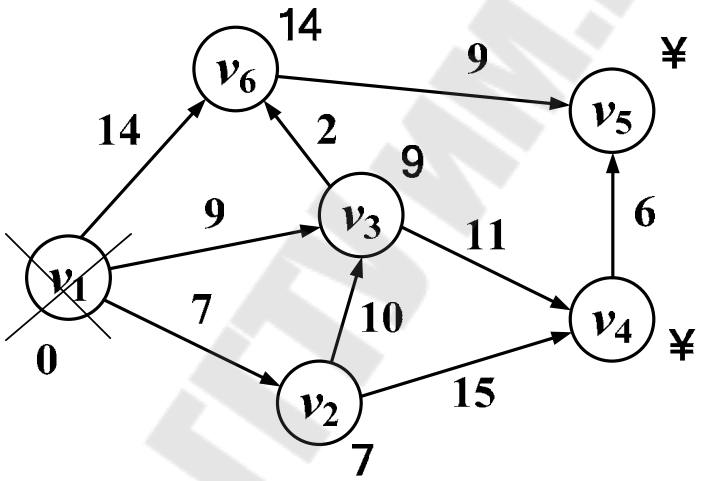

Рисунок 1.2 –Граф после первого шага

Второй шаг.

Повторяем шаг алгоритма для непосещенных вершин. Находим вершину с минимальной меткой (в данном случае это v_2) и пытаемся уменьшить метки соседей выбранной вершины. Соседями вершины v_2 являются v_1, v_3, v_4. Первый (по порядку) сосед – вершина v_1, но она уже посещена (вычеркнута). Следующий сосед – вершина v_4 (длина пути: $7 + 15 = 22$). Так как текущая метка v_4 равна бесконечности, то присваиваем этой вершине новое значение 22. Следующий сосед – вершина v_3 (длина пути: $7 + 10 = 17$). Но текущая метка v_3 равна $9 < 17$, поэтому метка не меняется.

Все соседи вершины v_2 просмотрены, замораживаем расстояние до неё и помечаем ее как посещенную.

Третий шаг.

Повторяем шаг алгоритма, выбрав вершину v_3. Ее соседями являются v_4 и v_6. Так как текущая метка v_4 равна 22, а путь в нее через v_3 равен $9 + 11 = 20 < 22$, то присваиваем этой вершине новое значение 20. По аналогии для v_6: $9 + 2 = 11 < 14$, поэтому v_6 присваиваем метку 11.

Повторяем шаги алгоритма для оставшихся вершин (рисунок 1.3). Алгоритм заканчивает работу, когда вычеркнуты все вершины. Результат его работы: кратчайший путь из v_1 в вершину v_2 составляет 7, в вершину v_3 составляет 9, в вершину v_4 составляет – 20, в вершину v_5 составляет 20 и в вершину v_6 составляет 11.

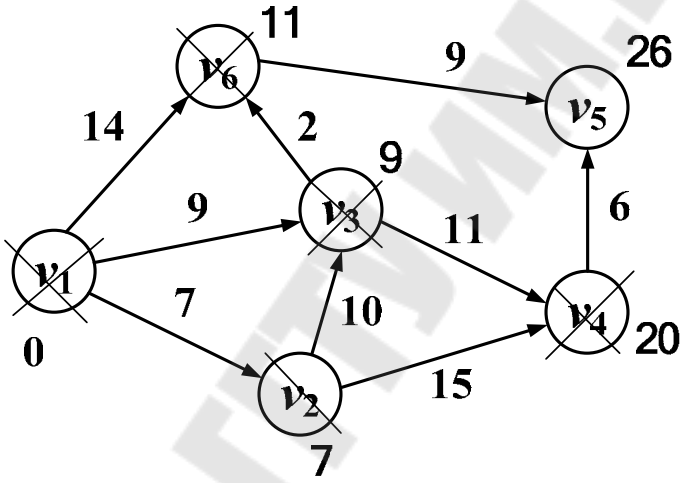

Рисунок 1.3 –Граф после последнего шага

Поскольку алгоритм Дейкстры всякий раз выбирает для обработки вершины с наименьшей оценкой кратчайшего пути, можно сказать, что он относится к жадным алгоритмам. Во многих практических приложениях требуется, чтобы кратчайший путь из *s* в *t* обладал некоторыми дополнительными свойствами. В этом случае целесообразно просто найти *K* кратчайших путей из *s* в *t* и выбрать из них тот, который обладает нужными свойствами.

Сложность алгоритма Дейкстры зависит от способа нахождения вершины *v*, способа хранения множества непосещенных вершин и способа обновления меток. Обозначим: *n* количество вершин, *m* – количество ребер в графе *G*. В общем случае просматривается все множество вершин, для хранения величин *d* используется массив, поэтому время работы алгоритма пропорционально $n^2 + m$.

2 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Изучить изложенный метод поиска кратчайшего пути (Алгоритм Дейкстры).

2. Составить программу для поиска кратчайшего пути на основе алгоритма Дейкстры.

3. Оформить отчет о выполнении задания с приведением условия задачи, алгоритма и программного кода, результаты работы программы и выводы по результатам.

3 ТРЕБОВАНИЕ К ОТЧЕТУ

В отчете должны быть отображены следующие пункты:

1. Задание.
2. Краткие теоретические сведения.
3. Схема алгоритма программы.
4. Листинг основных частей программы.
5. Результаты работы программы.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Математическая постановка задачи поиска кратчайшего пути на графе.

2. Суть алгоритма Дейкстры.

3. Основные шаги алгоритма Дейкстры.

4. Последовательность решения задачи поиска кратчайшего пути на графе.

5. Что такое *метка*?

6. Что такое *непосещенная вершина*?

7. Сложность алгоритма Дейкстры.

Варианты задания.

Вариант 1:

∞	8	∞	7	5	∞	11	5	∞	∞
0	∞	∞	4	7	4	∞	∞	6	∞
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	∞	3	3	∞
0	0	0	0	∞	19	∞	13	∞	∞
0	0	0	0	0	∞	3	16	∞	11
0	0	0	0	0	0	∞	6	13	∞
0	0	0	0	0	0	0	∞	∞	∞
0	0	0	0	0	0	0	0	∞	∞
0	0	0	0	0	0	0	0	0	∞

Вариант 2:

∞	13	3	2	5	5	5	7	11	∞
0	∞	7	6	∞	3	3	8	8	∞
0	0	∞	19	12	13	11	16	1	12
0	0	0	∞	3	16	12	11	∞	∞
0	0	0	0	∞	6	13	1	∞	∞
0	0	0	0	0	∞	7	∞	∞	∞
0	0	0	0	0	0	∞	12	∞	16
0	0	0	0	0	0	0	∞	4	∞
0	0	0	0	0	0	0	0	∞	4
0	0	0	0	0	0	0	0	0	∞

Вариант 3:

∞	8	12	7	5	4	11	5	13	∞
0	∞	∞	4	7	4	∞	∞	6	∞
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	∞	3	3	∞
0	0	0	0	∞	19	12	13	15	17
0	0	0	0	0	∞	3	16	12	11
0	0	0	0	0	0	∞	6	13	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	∞
0	0	0	0	0	0	0	0	0	∞

Вариант 4:

∞	13	3	2	5	5	5	7	11	∞
0	∞	7	6	∞	3	3	8	8	∞
0	0	∞	19	12	13	15	17	1	∞
0	0	0	∞	3	16	12	11	7	∞
0	0	0	0	∞	6	13	1	2	6
0	0	0	0	0	∞	7	∞	11	∞
0	0	0	0	0	0	∞	12	15	16
0	0	0	0	0	0	0	∞	4	∞
0	0	0	0	0	0	0	0	∞	∞
0	0	0	0	0	0	0	0	0	∞

Вариант 5:

∞	8	12	7	5	4	1	5	13	∞
0	∞	∞	4	7	4	∞	∞	∞	∞
0	0	∞	13	3	2	5	9	5	∞
0	0	0	∞	7	6	∞	3	3	8
0	0	0	0	∞	19	12	13	∞	∞
0	0	0	0	0	∞	3	16	12	11
0	0	0	0	0	0	∞	3	13	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 6:

∞	13	3	2	5	9	5	7	∞	∞
0	∞	7	6	∞	3	3	8	8	∞
0	0	∞	19	12	13	15	7	∞	∞
0	0	0	∞	3	16	12	11	∞	∞
0	0	0	0	∞	3	13	1	∞	6
0	0	0	0	0	∞	7	∞	11	5
0	0	0	0	0	0	∞	12	15	6
0	0	0	0	0	0	0	∞	∞	15
0	0	0	0	0	0	0	0	∞	8
0	0	0	0	0	0	0	0	0	∞

Вариант 7:

∞	8	12	7	5	8	11	5	13	∞
0	∞	∞	4	7	4	∞	∞	∞	∞
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	∞	3	3	∞
0	0	0	0	∞	19	12	13	∞	16
0	0	0	0	0	∞	3	16	12	11
0	0	0	0	0	0	∞	6	13	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 8:

∞	13	3	2	5	5	5	7	11	∞
0	∞	7	6	∞	3	3	8	∞	∞
0	0	∞	19	12	13	11	16	∞	12
0	0	0	∞	3	16	12	11	4	∞
0	0	0	0	∞	6	13	1	5	∞
0	0	0	0	0	∞	7	∞	∞	∞
0	0	0	0	0	0	∞	12	∞	16
0	0	0	0	0	0	0	∞	4	∞
0	0	0	0	0	0	0	0	∞	4
0	0	0	0	0	0	0	0	0	∞

Вариант 9:

∞	8	12	7	5	4	11	5	∞	∞
0	∞	∞	4	7	4	15	∞	∞	∞
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	∞	3	3	∞
0	0	0	0	∞	19	12	13	∞	17
0	0	0	0	0	∞	3	16	∞	∞
0	0	0	0	0	0	∞	6	13	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 10:

∞	13	3	2	5	5	5	7	11	∞
0	∞	7	6	∞	3	3	8	8	∞
0	0	∞	19	12	13	15	17	∞	∞
0	0	0	∞	3	16	12	11	∞	∞
0	0	0	0	∞	6	13	1	∞	∞
0	0	0	0	0	∞	7	∞	11	5
0	0	0	0	0	0	∞	12	∞	∞
0	0	0	0	0	0	0	∞	4	∞
0	0	0	0	0	0	0	0	∞	4
0	0	0	0	0	0	0	0	0	∞

Вариант 11:

∞	8	12	7	5	4	1	5	13	∞
0	∞	4	4	7	4	∞	16	∞	∞
0	0	∞	13	3	2	5	9	∞	∞
0	0	0	∞	7	6	∞	3	∞	8
0	0	0	0	∞	19	12	13	15	∞
0	0	0	0	0	∞	3	16	12	11
0	0	0	0	0	0	∞	3	13	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 12:

∞	13	3	2	5	9	5	7	11	∞
0	∞	7	6	∞	3	3	8	8	∞
0	0	∞	19	12	13	15	7	∞	∞
0	0	0	∞	3	16	12	11	∞	∞
0	0	0	0	∞	3	13	1	∞	∞
0	0	0	0	0	∞	7	∞	∞	∞
0	0	0	0	0	0	∞	12	∞	∞
0	0	0	0	0	0	0	∞	∞	∞
0	0	0	0	0	0	0	0	∞	8
0	0	0	0	0	0	0	0	0	∞

Вариант 13:

∞	8	12	7	5	8	11	5	∞	∞
0	∞	∞	4	7	4	17	18	6	∞
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	∞	3	∞	∞
0	0	0	0	∞	9	12	13	∞	∞
0	0	0	0	0	∞	3	16	12	∞
0	0	0	0	0	0	∞	6	∞	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 14 :

∞	13	3	2	5	5	5	7	11	∞
0	∞	7	6	∞	3	3	8	∞	∞
0	0	∞	9	12	13	11	16	∞	∞
0	0	0	∞	3	16	12	11	4	∞
0	0	0	0	∞	6	13	1	∞	∞
0	0	0	0	0	∞	7	∞	11	5
0	0	0	0	0	0	∞	12	15	∞
0	0	0	0	0	0	0	∞	∞	15
0	0	0	0	0	0	0	0	∞	∞
0	0	0	0	0	0	0	0	0	∞

Вариант 15:

∞	8	12	7	5	5	11	5	13	∞
0	∞	∞	4	7	4	5	∞	∞	∞
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	∞	3	∞	∞
0	0	0	0	∞	19	12	13	15	∞
0	0	0	0	0	∞	3	16	∞	∞
0	0	0	0	0	0	∞	6	13	∞
0	0	0	0	0	0	0	∞	7	4
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 16:

∞	8	∞	7	5	∞	11	5	∞	∞
0	∞	∞	4	7	4	∞	∞	16	∞
0	0	∞	13	3	11	5	5	∞	∞
0	0	0	∞	7	6	∞	3	3	∞
0	0	0	0	∞	19	∞	13	∞	∞
0	0	0	0	0	∞	3	16	∞	11
0	0	0	0	0	0	∞	6	13	∞
0	0	0	0	0	0	0	∞	∞	∞
0	0	0	0	0	0	0	0	∞	∞
0	0	0	0	0	0	0	0	0	∞

Вариант 17:

∞	13	3	2	5	5	5	7	11	∞
0	∞	7	6	∞	3	3	8	8	∞
0	0	∞	19	12	13	11	16	1	12
0	0	0	∞	3	16	12	11	∞	∞
0	0	0	0	∞	6	13	1	6	∞
0	0	0	0	0	∞	7	∞	∞	8
0	0	0	0	0	0	∞	12	∞	16
0	0	0	0	0	0	0	∞	4	∞
0	0	0	0	0	0	0	0	∞	4
0	0	0	0	0	0	0	0	0	∞

Вариант 18:

∞	8	12	7	5	4	11	5	13	∞
0	∞	∞	4	7	4	∞	7	6	11
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	7	3	3	∞
0	0	0	0	∞	19	12	13	15	17
0	0	0	0	0	∞	3	16	12	11
0	0	0	0	0	0	∞	6	13	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	13
0	0	0	0	0	0	0	0	0	∞

Вариант 19:

∞	13	3	2	5	5	5	7	11	∞
0	∞	7	6	3	3	3	8	8	∞
0	0	∞	19	12	13	15	17	1	9
0	0	0	∞	3	16	12	11	7	∞
0	0	0	0	∞	6	13	7	2	6
0	0	0	0	0	∞	7	∞	11	∞
0	0	0	0	0	0	∞	12	15	16
0	0	0	0	0	0	0	∞	4	∞
0	0	0	0	0	0	0	0	∞	∞
0	0	0	0	0	0	0	0	0	∞

Вариант 20:

∞	8	12	7	5	4	1	5	13	∞
0	∞	∞	4	7	4	∞	∞	9	∞
0	0	∞	13	3	2	5	9	5	∞
0	0	0	∞	7	6	∞	3	3	8
0	0	0	0	∞	19	12	13	∞	∞
0	0	0	0	0	∞	3	16	12	11
0	0	0	0	0	0	∞	3	3	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 21:

∞	13	3	4	5	9	5	7	∞	∞
0	∞	7	6	∞	3	3	2	8	∞
0	0	∞	19	12	13	15	7	∞	∞
0	0	0	∞	3	16	12	11	∞	∞
0	0	0	0	∞	3	13	5	5	6
0	0	0	0	0	∞	7	∞	11	5
0	0	0	0	0	0	∞	12	15	6
0	0	0	0	0	0	0	∞	∞	15
0	0	0	0	0	0	0	0	∞	8
0	0	0	0	0	0	0	0	0	∞

Вариант 22:

∞	8	12	7	5	8	13	5	13	∞
0	∞	∞	4	7	4	∞	12	∞	11
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	∞	3	3	∞
0	0	0	0	∞	19	12	13	14	16
0	0	0	0	0	∞	3	16	12	11
0	0	0	0	0	0	∞	6	13	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 23:

∞	13	3	2	5	5	5	7	11	∞
0	∞	7	6	∞	5	3	8	∞	∞
0	0	∞	19	12	13	11	16	∞	12
0	0	0	∞	3	16	12	11	4	∞
0	0	0	0	∞	6	13	11	5	∞
0	0	0	0	0	∞	7	∞	∞	∞
0	0	0	0	0	0	∞	12	∞	16
0	0	0	0	0	0	0	∞	4	∞
0	0	0	0	0	0	0	0	∞	4
0	0	0	0	0	0	0	0	0	∞

Вариант 24:

∞	8	12	7	5	4	11	5	∞	∞
0	∞	∞	4	7	4	15	∞	9	∞
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	4	3	3	∞
0	0	0	0	∞	19	12	13	∞	17
0	0	0	0	0	∞	3	16	∞	∞
0	0	0	0	0	0	∞	6	7	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 25:

∞	13	3	2	5	5	5	7	11	∞
0	∞	7	6	∞	3	3	8	8	∞
0	0	∞	19	12	13	15	17	∞	∞
0	0	0	∞	3	16	12	11	11	∞
0	0	0	0	∞	6	13	1	∞	∞
0	0	0	0	0	∞	7	∞	11	5
0	0	0	0	0	0	∞	12	∞	7
0	0	0	0	0	0	0	∞	5	6
0	0	0	0	0	0	0	0	∞	4
0	0	0	0	0	0	0	0	0	∞

Вариант 26:

∞	8	12	7	5	4	9	5	13	∞
0	∞	4	4	7	4	∞	16	∞	∞
0	0	∞	13	3	5	9	9	∞	∞
0	0	0	∞	7	6	∞	3	∞	8
0	0	0	0	∞	19	12	13	15	∞
0	0	0	0	0	∞	3	16	12	11
0	0	0	0	0	0	∞	11	13	∞
0	0	0	0	0	0	0	∞	7	11
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 27:

∞	13	3	2	5	9	5	7	11	∞
0	∞	7	6	9	7	3	8	8	∞
0	0	∞	19	12	13	15	7	∞	7
0	0	0	∞	3	16	12	11	8	∞
0	0	0	0	∞	3	13	1	∞	∞
0	0	0	0	0	∞	7	7	∞	∞
0	0	0	0	0	0	∞	12	11	7
0	0	0	0	0	0	0	∞	11	∞
0	0	0	0	0	0	0	0	∞	8
0	0	0	0	0	0	0	0	0	∞

Вариант 28:

∞	8	12	7	5	8	11	5	∞	∞
0	∞	∞	4	7	4	17	18	6	8
0	0	∞	13	3	7	5	5	∞	7
0	0	0	∞	7	6	∞	3	∞	∞
0	0	0	0	∞	9	2	13	∞	9
0	0	0	0	0	∞	3	16	12	∞
0	0	0	0	0	0	∞	6	∞	∞
0	0	0	0	0	0	0	∞	7	∞
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Вариант 29 :

∞	13	3	2	5	9	5	7	11	∞
0	∞	7	6	∞	3	3	8	6	∞
0	0	∞	9	12	13	11	16	∞	∞
0	0	0	∞	3	16	12	11	4	∞
0	0	0	0	∞	6	13	12	3	∞
0	0	0	0	0	∞	7	∞	11	5
0	0	0	0	0	0	∞	12	15	∞
0	0	0	0	0	0	0	∞	∞	15
0	0	0	0	0	0	0	0	∞	∞
0	0	0	0	0	0	0	0	0	∞

Вариант 30:

∞	8	12	7	5	5	11	5	13	∞
0	∞	∞	4	7	4	5	∞	∞	∞
0	0	∞	13	3	2	5	5	∞	∞
0	0	0	∞	7	6	4	3	∞	8
0	0	0	0	∞	19	12	13	15	∞
0	0	0	0	0	∞	3	16	∞	9
0	0	0	0	0	0	∞	6	13	∞
0	0	0	0	0	0	0	∞	7	4
0	0	0	0	0	0	0	0	∞	12
0	0	0	0	0	0	0	0	0	∞

Лабораторная работа №9.
«Решение задач оптимизации на основе метода Монте-Карло»
(4 часа)

Цель работы: изучение эффективности применения метода Монте-Карло для решения производственно-экономических задач.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Общая постановка задачи

Метод Монте-Карло представляет собой универсальный метод, применяемый для приближенного решения задач самых различных классов: вероятностных и детерминированных, дискретных и непрерывных, задач моделирования и оптимизации и т.д.

Вероятностные задачи – это любые задачи, связанные с анализом случайных явлений (случайных событий, величин, процессов). Детерминированные задачи – это задачи, в постановке которых нет никаких случайных факторов.

Дискретные задачи – это задачи, в которых все анализируемые величины могут принимать значения только из некоторого конечного множества допустимых значений. Непрерывные задачи – это задачи, в которых анализируемые величины могут принимать любые значения из некоторого диапазона (этот диапазон может быть как ограниченным, так и неограниченным).

Задачи моделирования – это задачи, связанные с имитацией некоторого объекта (например, сложной технической системы) или явления (например, процесса развития отрасли) с целью определения его характеристик. Задачи оптимизации – это задачи, в которых требуется выбор лучшего из нескольких возможных вариантов решения.

Большинство практических задач не могут быть однозначно отнесены к одному из приведенных выше классов, а содержат элементы, характерные для нескольких из этих классов.

В данной работе основное внимание уделяется применению метода Монте-Карло для решения детерминированных задач оптимизации. Решение данных задач методом Монте-Карло основано на многократном случайном выборе вариантов решений и их сравнении. При достаточном количестве испытаний (т.е. вариантов решения, вы-

бранных случайным образом) находится решение, близкое к оптимальному (во многих случаях находится оптимальное решение).

Метод Монте-Карло основан на применении равномерно распределенных псевдослучайных чисел (ПСЧ). Расчет (розыгрыш) ПСЧ выполняется по специальным алгоритмам, позволяющим получать достаточно большую (практически бесконечную) последовательность таких чисел. Эти алгоритмы разработаны таким образом, что ПСЧ всегда принимают значения из диапазона от нуля до единицы. При этом ПСЧ обладают следующими свойствами:

-равномерность: ПСЧ могут принимать значения из любой части диапазона (0;1) с одинаковой частотой;

-случайность: в последовательности ПСЧ нет какой-либо закономерности;

-независимость: любое из значений ПСЧ не зависит от предыдущих ПСЧ.

Все это позволяет рассматривать ПСЧ как случайные величины, распределенные по равномерному закону в диапазоне от нуля до единицы. Алгоритмы для получения ПСЧ реализованы практически во всех языках программирования и во многих прикладных программах обработки данных. В данной работе ПСЧ формируются программно, на основании моделирования работы сдвигового регистра с линейной обратной связью (*LFSR*), функциональная схема которого приведена на рисунке 1.1. Для нормирования ПСЧ целое положительное число в диапазоне от 1 до $2^{32} - 1 = 4294967295$, сформированное *LFSR* в очередном такте работы, необходимо разделить на 4294967296.

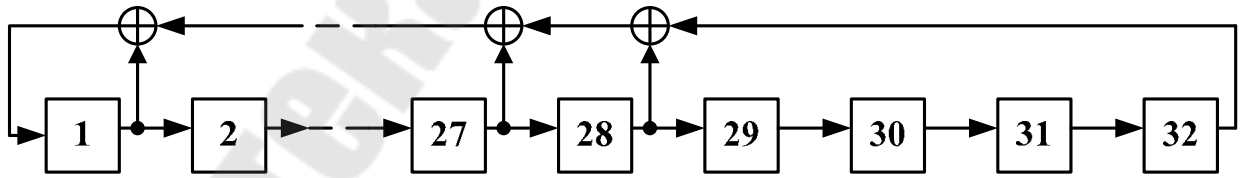

Рисунок 1.1 – LFSR с порождающим полиномом
$x^{32} \oplus x^{28} \oplus x^{27} \oplus x \oplus 1$

ПСЧ применяются для многократного случайного выбора вариантов решения задачи. Из этих вариантов выбирается лучший. Чтобы найти вариант решения, достаточно близкий к оптимальному, обычно требуется выбрать много вариантов решения (от нескольких десятков

до тысяч). Поэтому применение метода Монте-Карло возможно только с использованием программных средств.

Во многих случаях данные задачи могут быть решены другими (аналитическими) методами, позволяющими найти точное оптимальное решение. Это могут быть методы линейного программирования, динамического программирования, поиска экстремумов функций многих переменных и т.д. Однако применение точных аналитических методов, как правило, возможно только для задач небольшой размерности (с небольшим количеством ограничений, переменных и т.д.). Для реальных задач применение аналитических методов нередко оказывается невозможным из-за большого объема сложных вычислений. Реализация и применение таких методов (даже с использованием компьютера) оказывается практически невозможным или нецелесообразным. Поэтому во многих случаях применение метода Монте-Карло может оказаться единственно возможным способом решения задачи. Алгоритм поиска оптимального решения на основе метода Монте-Карло полностью зависит от конкретной задачи.

Рассмотрим пример [10]. На предприятие поступает некоторое сырье для переработки. Известно, что примерно 15% партий сырья имеют повышенное содержание примесей и требуют дополнительной очистки. Часть сырья подвергается контролю при поступлении на предприятие (входной контроль); остальное сырье поступает на переработку без входного контроля. Затраты, связанные с контролем качества одной партии сырья, составляют 8 руб. Если при контроле обнаруживается повышенное содержание примесей, то партия подвергается очистке; затраты на очистку составляют 40 руб. Если партия с повышенным содержанием примесей не подвергалась входному контролю, то необходимость ее очистки обнаруживается в ходе переработки; в этом случае затраты на очистку составляют 70 руб. Требуется найти, какая часть сырья должна подвергаться входному контролю, чтобы общие затраты на контроль и очистку были минимальными.

Для решения данной задачи выполним моделирование процесса контроля и очистки сырья при различных долях контролируемых партий сырья: от 0 (входной контроль не выполняется) до 100% (контролируется все сырье) с шагом 5%. Предположим, что доля контролируемых партий сырья равна некоторой заданной величине, например, 5%. Смоделируем процесс контроля и очистки большого количества партий сырья (например, десяти тысяч) и определим величину затрат на эти операции. Обозначим долю контролируемых партий как P (в

данном случае P=0,05). Моделирование осуществляется по следующему алгоритму.

1. Разыгрываются два ПСЧ: R_1 и R_2. Величина R_1 будет использоваться для имитации отбора контролируемых партий, а R_2 – для имитации наличия (или отсутствия) повышенного содержания примесей.

2. Если $R_1<P$, то будем считать, что смоделировано поступление партии сырья на контроль. Общие затраты при этом увеличиваются на 8 руб. (это величина затрат на контроль одной партии). Если при этом $R_2<0,15$ (где 0,15 – доля партий, имеющих повышенное содержание примесей), то будем считать, что смоделировано наличие повышенного содержания примесей в проверяемой партии. В этом случае общие затраты увеличиваются на 40 руб. Если $R_1>P$ (партия сырья не поступает на контроль), и при этом $R_2<0,15$ (партия имеет повышенное содержание примесей), то смоделировано обнаружение повышенного содержания примесей при переработке сырья. Затраты увеличиваются на 70 руб.

Шаги 1 и 2 повторяются многократно, например, 10000 раз (это означает моделирование контроля и очистки 10000 партий сырья). Подсчитываются общие затраты на контроль и очистку. В таблице 1.1 приведен пример десяти испытаний (моделирование контроля и очистки десяти партий).

Таблица 1.1 – Результаты десяти испытаний

Номер испытания	R_1	Контроль	R_2	Повышенное содержание примеси	Затраты на партию	Общие затраты
1	0,0795	да	0,3780	нет	8	8
2	0,0593	да	0,7602	нет	8	16
3	0,2847	нет	0,8197	нет	0	16
4	0,6133	нет	0,5766	нет	0	16
5	0,9595	нет	0,0981	да	70	86
6	0,2410	нет	0,5962	нет	0	86
7	0,2978	нет	0,6458	нет	0	86
8	0,9762	нет	0,0523	да	70	156
9	0,4523	нет	0,8153	нет	0	156
10	0,4286	нет	0,8400	нет	0	156

Аналогично следует выполнить моделирование при других долях контролируемых партий сырья (от 0 до 100% с шагом 10%). Результаты моделирования процесса контроля и очистки сырья приведены в таблице 1.2.

Анализ таблицы показал, что предприятию следует подвергать входному контролю 10% партий сырья. При этом затраты на контроль и очистку сырья будут минимальными.

Таблица 1.2 – Моделирование процесса контроля и очистки

Процент контроля	Затраты на 10000 партий
0	112280
10	**105432**
20	109976
30	115378
40	121354
50	118078
60	126554
70	129020
80	132382
90	137690
100	139120

2 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Изучить метод Монте-Карло.
2. Составить программу для решения дискретной задачи оптимизации методом Монте-Карло.
3. Оформить отчет о выполнении задания с приведением условия задачи, алгоритма и программного кода, результаты работы программы и выводы по результатам.

3 ТРЕБОВАНИЕ К ОТЧЕТУ

В отчете должны быть отображены следующие пункты:
1. Задание.
2. Краткие теоретические сведения.
3. Схема алгоритма программы.
4. Листинг основных частей программы.
5. Результаты работы программы.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Основная идея метода Монте-Карло.
2. Какие задачи могут быть решены методом Монте-Карло.
3. Достоинства метода Монте-Карло.
4. Последовательность решения задачи методом Монте-Карло.
5. Недостатки метода Монте-Карло.
6. Алгоритмы формирования равномерно распределенных псевдослучайных чисел.
7. Сложность метода Монте-Карло.
8. Принципы работы *LFSR*.

Задание

На предприятие радиоэлектронной промышленности поступают комплектующие изделия – резисторы с номиналом невысокой точности (15%). Известно, что примерно *A*% резисторов не подходит для изготовления продукции и требуют дополнительной подгонки. Чтобы выявить такие резисторы, необходим входной контроль. Стоимость контроля одного резистора составляет *B* руб. Стоимость подгонки составляет *C* руб. Если резистор установили в изделие, то стоимость его замены составляет *D* руб. Требуется найти, какую часть резисторов необходимо подвергнуть входному контролю, чтобы общие затраты на контроль и подгонку были минимальными.

Псевдослучайные числа формируются на основе *LFSR* с заданным полиномом.

Таблица 1.3 – Варианты задания

Вариант	***A*, %**	***B***	***C***	***D***	**№ полинома**
1	5	2	50	150	5
2	5	3	55	145	7
3	10	4	60	140	9
4	10	5	65	135	11
5	15	6	70	130	13
6	15	7	50	125	15
7	20	8	55	120	2
8	20	9	60	115	4
9	25	2	65	110	6
10	25	3	70	105	8
11	7	4	50	145	10
12	7	5	55	140	12
13	17	6	55	135	14
14	17	7	60	130	4
15	19	5	65	125	6
16	15	6	70	120	8
17	20	7	50	145	10
18	20	8	55	140	12
19	25	9	65	135	1
20	25	2	70	130	3
21	15	7	50	125	15
22	20	8	55	145	2
23	20	9	60	140	4
24	25	2	65	135	6
25	25	3	70	130	8
26	7	4	50	125	10
27	7	5	55	120	12
28	17	6	60	90	14
29	17	7	65	85	1
30	19	8	70	80	3

Варианты полиномов

1. $x^{32} \oplus x^{28} \oplus x^{27} \oplus x^{25} \oplus x^{22} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{18} \oplus x^{12} \oplus x^{10} \oplus x^{9} \oplus x^{6} \oplus 1.$

2. $x^{32} \oplus x^{30} \oplus x^{29} \oplus x^{26} \oplus x^{25} \oplus x^{24} \oplus x^{23} \oplus x^{20} \oplus x^{18} \oplus x^{9} \oplus x^{5} \oplus x \oplus 1.$

3. $x^{32} \oplus x^{28} \oplus x^{27} \oplus x^{26} \oplus x^{23} \oplus x^{22} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{18} \oplus x^{17} \oplus x^{12} \oplus x^9 \oplus x^5 \oplus x^4 \oplus x^2 \oplus 1$.

4. $x^{32} \oplus x^{30} \oplus x^{28} \oplus x^{25} \oplus x^{24} \oplus x^{23} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{18} \oplus x^{17} \oplus x^{13} \oplus \oplus x^{11} \oplus x^9 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x \oplus 1$.

5. $x^{32} \oplus x^{31} \oplus x^{30} \oplus x^{29} \oplus x^{27} \oplus x^{26} \oplus x^{25} \oplus x^{24} \oplus x^{23} \oplus x^{21} \oplus x^{20} \oplus x^{17} \oplus \oplus x^{16} \oplus x^{14} \oplus x^{12} \oplus x^9 \oplus x^8 \oplus x^7 \oplus 1$.

6. $x^{32} \oplus x^{31} \oplus x^{30} \oplus x^{29} \oplus x^{27} \oplus x^{25} \oplus x^{23} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{13} \oplus x^{10} \oplus \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x \oplus 1$.

7. $x^{32} \oplus x^{30} \oplus x^{28} \oplus x^{27} \oplus x^{25} \oplus x^{24} \oplus x^{23} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{18} \oplus x^{16} \oplus \oplus x^{15} \oplus x^{14} \oplus x^8 \oplus x^6 \oplus x^5 \oplus x^3 \oplus 1$.

8. $x^{32} \oplus x^{29} \oplus x^{27} \oplus x^{26} \oplus x^{25} \oplus x^{23} \oplus x^{21} \oplus x^{17} \oplus x^{16} \oplus x^{15} \oplus x^{13} \oplus x^{12} \oplus \oplus x^8 \oplus x^5 \oplus x^4 \oplus x^3 \oplus 1$.

9. $x^{32} \oplus x^{31} \oplus x^{27} \oplus x^{26} \oplus x^{23} \oplus x^{22} \oplus x^{19} \oplus x^{18} \oplus x^{17} \oplus x^{16} \oplus x^{15} \oplus x^{12} \oplus \oplus x^9 \oplus x^8 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus x \oplus 1$.

10. $x^{32} \oplus x^{31} \oplus x^{30} \oplus x^{29} \oplus x^{27} \oplus x^{24} \oplus x^{23} \oplus x^{22} \oplus x^{18} \oplus x^9 \oplus x^7 \oplus x^6 \oplus \oplus x^3 \oplus x^2 \oplus 1$.

11. $x^{32} \oplus x^{31} \oplus x^{30} \oplus x^{28} \oplus x^{27} \oplus x^{24} \oplus x^{23} \oplus x^{21} \oplus x^{18} \oplus x^{16} \oplus x^{15} \oplus x^{13} \oplus \oplus x^{12} \oplus x^{11} \oplus x^{10} \oplus x^9 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^2 \oplus 1$.

12. $x^{32} \oplus x^{29} \oplus x^{28} \oplus x^{23} \oplus x^{22} \oplus x^{21} \oplus x^{16} \oplus x^{13} \oplus x^{10} \oplus x^6 \oplus x^5 \oplus x^4 \oplus 1$.

13. $x^{32} \oplus x^{31} \oplus x^{26} \oplus x^{25} \oplus x^{24} \oplus x^{22} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{16} \oplus x^{13} \oplus x^{12} \oplus \oplus x^8 \oplus x^2 \oplus 1$.

14. $x^{32} \oplus x^{28} \oplus x^{22} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{18} \oplus x^{16} \oplus x^{10} \oplus x^8 \oplus x^7 \oplus x^3 \oplus 1$.

15. $x^{32} \oplus x^{30} \oplus x^{27} \oplus x^{26} \oplus x^{25} \oplus x^{23} \oplus x^{22} \oplus x^{21} \oplus x^{20} \oplus x^{19} \oplus x^{18} \oplus x^{17} \oplus \oplus x^{13} \oplus x^6 \oplus x^5 \oplus x^4 \oplus 1$.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Андронов, С. А. Методы оптимального проектирования / С. А. Андронов. – СПб.: СПб ГУАП, 2001. – 169 с.
2. Банди, Б. Методы оптимизации: Вводный курс / Б. Банди. – М.: Радио и связь, 1988.
3. Галлеев, Э.М. Оптимизация: теория, примеры, задачи / Э.М. Галлеев, В.М. Тихомиров. – М.: Эдиториал УРСС, 2000.
4. Гилл, Ф., Практическая оптимизация / Ф. Гилл, У. Мюррей, М. Райт. М. – М: Мир,1985.
5. Кузнецов, А.В. Высшая математика: Математическое программирование / А.В. Кузнецов, В.А. Сакович, Н.И. Холод. – Минск: Вышэйшая школа, 2001.
6. Лоу, А.М. Имитационное моделирование. Классика CS. 3-е изд. / А.М. Лоу, Б.Д. Кельтон. – СПб.:Питер, 2004. – 847 с.
7. Мельников, О.И. Моделирование оптимизационных задач в системе Maple / О.И. Мельников, А.А. Морозов. – Мн.: Нац. ин-т образования, 2010. – 88 с.
8. Просветов Г.И. Дискретная математика: задачи и решения: учебное пособие / Г.И. Просветов. – М. БИНОМ. Лаборатория знаний, 2008. – 222 с.
9. Сборник задач и упражнений по высшей математике. Математическое программирование: Учебное пособие / А.В. Кузнецов, В.А. Сакович, Н.И. Холод и др.; под общ. ред. А.В. Кузнецова, Н.И. Рутковскогою – Мн.: Выш. шк., 2002. – 447 с.
10. Смородинский, С.С. Методы анализа и принятия решений в слабоструктурированных задачах / С.С. Смородинский, Н.В. Батин. – Мн.: БГУИР, 2002.
11. Таха, Х. А. Введение в исследование операций / Х. А. Таха. – М.: Вильямс, 2005.

Мурашко Игорь Александрович

ОПТИМИЗАЦИЯ ПРОЕКТНЫХ РЕШЕНИЙ

Лабораторный практикум по одноименной дисциплине для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» дневной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 04.07.11.
Рег. № 23Е.
E-mail: ic@gstu.by
http://www.gstu.by