МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курганский государственный университет»

Кафедра программного обеспечения автоматизированных систем

# СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

Методические указания
к выполнению лабораторных работ
для магистрантов направления подготовки 09.04.04

Курган 2023

Кафедра: «Программное обеспечение автоматизированных систем».

Дисциплина: «Структуры и алгоритмы обработки данных».

Направление: 09.04.04 «Программная инженерия».

Составил: канд. техн. наук, доцент А. М. Семахин.

Печатается в соответствии с планом издания, утверждённым методическим советом университета «28» декабря 2022 г.

Утверждены на заседании кафедры «3» февраля 2023 г.

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Дисциплина «Структуры и алгоритмы обработки данных» имеет целью дать магистрантам теоретические знания алгоритмов и практические навыки реализации на ПЭВМ методов численной и эволюционной оптимизации.

Предмет дисциплины – алгоритмы методов оптимизации в решении задач.

Задачи дисциплины – изучение алгоритмов численных и эволюционных методов оптимизации.

Практические занятия составляют 24 часа.

Методические указания содержат теоретическое обоснование и варианты заданий для выполнения практических работ по дисциплине «Структуры и алгоритмы данных».

Методические указания разработаны в соответствии с требованиями государственного образовательного стандарта по подготовке магистрантов по направлению 09.04.04 «Программная инженерия», направленность «Методы и алгоритмы интеллектуальной обработки данных в информационно-вычислительных системах».

# 1 МЕТОДЫ ЧИСЛЕННОЙ ОПТИМИЗАЦИИ

## 1.1 Методы линейного программирования

*Линейное программирование* – теоретический аппарат модельного исследования, направленного на отыскание наилучшего способа распределения ограниченных ресурсов. Целевая функция и ограничения математической модели линейного программирования представляются линейными уравнениями и неравенствами. Математическая модель линейного программирования имеет вид

$$\text{extr} \leftarrow Z = \sum_{j=1}^{n} C_j * X_j - \text{целевая функция}$$

при условиях (1.1)

$$\begin{cases} \sum_{j=1}^{n} a_{ij} * X_j = b_i \\ X_j \geq 0, i = \overline{1,m}, j = \overline{1,n} \end{cases}$$

### 1.1.1 Симплекс-метод

*Симплекс-метод* – метод обхода угловых точек области допустимых решений (симплекса) с проверкой на оптимальность.

*Базисное решение* – решение системы уравнений, получаемое приравниванием к нулю $(n - m)$ переменных, где $n$ – количество неизвестных, $m$ – количество уравнений.

*Допустимое базисное решение* – базисное решение, удовлетворяющее требованию неотрицательности правых частей.

*Небазисные переменные* – переменные, имеющие нулевое значение.

*Базисные переменные* – переменные, имеющие ненулевое значение.

*Включаемая переменная* – небазисная переменная, которая будет включена в множество базисных переменных на следующей итерации (при переходе к смежной экстремальной точке).

*Исключаемая переменная* – базисная переменная, которая на следующей итерации подлежит исключению из множества базисных переменных.

Симплекс-алгоритм состоит из следующих шагов.

Шаг 1. Определение начального допустимого базисного решения.

Шаг 2. Определение включаемой переменной из числа небазисных переменных. Если такой переменной нет, то решение оптимально. Иначе осуществляется переход к шагу 3.

Шаг 3. Определение исключаемой переменной из числа базисных переменных.

Шаг 4. Определение нового базисного решения. Переход на шаг 2 [1–4].

### 1.1.2 Метод Гаусса – Жордана

Метод Гаусса – Жордана (метод исключения переменных) определяет оптимальное решение задачи линейного программирования. Метод Гаусса – Жордана выполняется после определения ведущей строки, ведущего столбца и ведущего элемента симплекс-таблицы.

*Ведущая строка* – строка симплекс-таблицы, соответствующая исключаемой переменной.

*Ведущий столбец* – столбец симплекс-таблицы, соответствующий включаемой переменной.

*Ведущий элемент* – элемент таблицы, находящийся на пересечении ведущего столбца и ведущей строки.

*Условие оптимальности*. Включаемой переменной в задаче максимизации (минимизации) является небазисная переменная, имеющая в Z-уравнении

наибольший отрицательный (положительный) коэффициент. В случае равенства коэффициентов для нескольких небазисных переменных выбор делается произвольно. Если все коэффициенты при небазисных переменных в Z-уравнении неотрицательны (неположительны), полученное решение является оптимальным [1–4].

*Условие допустимости.* В задачах максимизации и минимизации в качестве исключаемой переменной выбирается базисная переменная, для которой отношение постоянной в правой части соответствующего ограничения к (положительному) коэффициенту ведущего столбца минимально. В случае равенства этого отношения для нескольких базисных переменных выбор делается произвольно [1-4].

Метод Гаусса – Жордана включает две вычислительные процедуры.

1 Формирование новой ведущей строки.

Новая ведущая строка = Старая ведущая строка / Ведущий элемент.

2 Формирование остальных новых уравнений.

Новое уравнение = Старое уравнение – (Коэффициент ведущего столбца старого уравнения) * (Новая ведущая строка) [1–4].

### 1.1.3 Методы целочисленного программирования

*Целочисленное программирование* – раздел линейного программирования, разрабатывающий и исследующий методы решения задач, искомые переменные которых имеют целочисленные значения [1-4].

Математическая модель целочисленного программирования имеет вид

$$extr \leftarrow Z = \sum_{j=1}^{n} C_j * X_j - \text{целевая функция}$$

при условиях (1.2)

$$\begin{cases} \sum_{j=1}^{n} a_{ij} * X_j = b_i, \; i = \overline{1,m}, \; j = \overline{1,n} \\ X_j \geq 0, \text{ целочисленное значение} \end{cases}$$

#### 1.1.3.1 Метод отсекающих плоскостей

Метод отсекающих плоскостей разработан Р. Гомори в 1957–1958 годах. Алгоритм метода включает этапы:

Этап 1. Ослабленная задача. Целочисленность искомых переменных игнорируется, симплекс-методом определяется оптимальный план.

Этап 2. Расширенная задача. Если план нецелочисленный, составляется дополнительное ограничение, «отсекающее» дробную часть искомой переменной. Дополнительное ограничение включается в систему ограничений и решается расширенная задача [1–4].

### 1.1.3.2 Метод ветвей и границ

Метод ветвей и границ применяется для решения полностью или частично целочисленных задач. Предложен А. Лэндом и Дж. Дойгом в 1960 г.

Дана математическая модель целочисленного линейного программирования

$$\max \leftarrow Z = \sum_{j=1}^{n} c_j * x_j$$

при ограничениях (1.3)

$$\begin{cases} \sum_{j=1}^{n} a_{ij} * x_j = b_i, \ i = \overline{1,m} \\ x_j \geq 0, \ j = \overline{1,n} \\ x_1, x_2, ..., x_p \text{ - целые числа } (p \leq n) \end{cases}$$

Найти вектор $\bar{x} \in E_n$, максимизирующий целевую функцию. Для целочисленной переменной можно указать верхнюю и нижнюю границы, в пределах которых содержатся оптимальные значения

$$V_j \leq x_j \leq W_j, \ j = \overline{1,p}. \quad (1.4)$$

В начале *S*-итерации метода ветвей и границ необходимо иметь:

1. Список задач линейного программирования, каждая из которых должна быть решена в последующих итерациях.

2. Нижнюю границу оптимального значения линейной формы задачи $Z_0^{(S)}$. На первой итерации в качестве $Z_0^{(1)}$ берется значение целевой функции $f(\bar{x})$ в любой целочисленной точке $\bar{x}$. Если точку указать невозможно, то принимают $Z_0^{(1)} = -\infty$.

Пусть в результате *S*-итераций метода получили список из $Z$ задач: $1, 2, ..., Z$ и существует $Z_0^{(S)}$. Алгоритм метода ветвей и границ содержит этапы:

Этап 1. Выбирается из списка задач линейного программирования задача $R$ для решения, $1 \leq R \leq Z$. Задача $R$ решается.

Этап 2. Если задача $R$ имеет решение $\bar{x}_R^{(S)}$, то переходим к этапу 3. Иначе – исключаем задачу $R$ из списка, $Z_0^{(S+1)}=Z_0^{(s)}$. Переход к этапу 1. При $S=0$ делаем вывод, что исходная задача не имеет решения и процесс решения заканчивается.

Этап 3. Если $f(\bar{x}_R^{(S)}) > Z_0^{(S)}$, то переходим к этапу 4. В противном случае задача $R$ исключается из списка. Переход к этапу 1.

Этап 4. Если все компоненты вектора $\bar{x}_R^{(S)}$ удовлетворяют условию целочисленности, то переходим к этапу 5. В противном случае задача $R$ из списка исключается. План $\bar{x}_R^{(S)}$ запоминается, $Z_0^{(S+1)} = f(\bar{x}_R^{(S)})$. Переход к этапу 1. При $S=0$ вектор $\bar{x}^{(1)}$ является решением исходной задачи и процесс решения заканчивается.

Этап 5. Задача $R$ из списка исключается. В список включаются две новые задачи линейного программирования – задача $Z+1$ и задача $Z+2$, $Z_0^{(S+1)}=Z_0^{(S)}$. Переход к этапу 1. Процесс разбиения задачи $R$ на две новые задачи линейного программирования осуществляется следующим образом. Пусть $\bar{x}_j^{(S)}$ – дробная компонента в полученном оптимальном плане $\bar{x}_R^{(S)}$ и $\left[\bar{x}_j^{(S)}\right]$ целая часть. Тогда задача $Z+1$ имеет вид:

$$f(\bar{x}) = \sum_{j=1}^{n} c_j * x_j \to \max \qquad (1.5)$$

при ограничениях

$$\begin{cases} \sum_{j=1}^{n} a_{ij} * x_j = b_i, \mathrm{i} = \overline{1,\mathrm{m}} \\ \ldots \\ V_j \le x_j \le \left[\bar{x}_j^{(S)}\right] \\ \ldots \\ x_1,\ldots,x_n \ge 0 \end{cases}$$

Тогда задача $Z+2$ имеет вид:

$$f(\bar{x}) = \sum_{j=1}^{n} c_j * x_j \to \max \qquad (1.6)$$

при ограничениях

$$\begin{cases} \sum_{j=1}^{n} a_{ij} * x_j = b_i, \mathrm{i} = \overline{1,\mathrm{m}} \\ \dots \\ \left[ \overline{x_j^{(S)}} \right] + 1 \le x_j \le W_j \\ \dots \\ x_1, \dots, x_n \ge 0 \end{cases}$$

Процесс решения продолжается пока не будут решены все задачи линейного программирования из списка. Решение задачи будет $Z_0^{(S)}$ на последней итерации [1-4].

### 1.1.4 Практическая работа № 1 «Методы решения задач линейного программирования»

**Цель:** получить теоретические знания и практические навыки в моделировании систем методами линейного программирования.

**Используемые приемы и технологии:** линейное программирование, целочисленное программирование.

**Ключевые термины:** математическая модель, симплекс-метод, метод отсекающих плоскостей, метод ветвей и границ.

**Постановка задачи:** разработайте программу, реализующую алгоритм решения оптимизационной задачи методом линейного программирования.

#### 1.1.4.1 Варианты заданий

**Вариант 1**

$extr \leftarrow Z = 2x_1 - 6x_2$

при ограничениях

$$\begin{cases} x_1 + x_2 \ge 2 \\ -x_1 + 2x_2 \le 4 \\ x_1 + 2x_2 \le 8 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 2**

$extr \leftarrow Z = 2x_1 - x_2$

при ограничениях

$$\begin{cases} x_1 + x_2 \ge 4 \\ -x_1 + 2x_2 \le 2 \\ x_1 + 2x_2 \le 10 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 3**

$extr \leftarrow Z = 3x_1 + 2x_2$

при ограничениях

$$\begin{cases} x_1 + 2x_2 \le 6 \\ 2x_1 + x_2 \le 8 \\ -x_1 + x_2 \le 2 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 4**

$extr \leftarrow Z = x_1 + x_2$

при ограничениях

$$\begin{cases} x_1 - 4x_2 \le 4 \\ 3x_1 - x_2 \ge 0 \\ x_1 + x_2 \ge 4 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 5**

$extr \leftarrow Z = 2x_1 + 3x_2$

при ограничениях

$$\begin{cases} 4x_1 + 6x_2 \le 14 \\ 2x_1 + x_2 \le 6 \\ x_1 + 2x_2 \le 12 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 6**

$extr \leftarrow Z = 40x_1 + 50x_2$

при ограничениях

$$\begin{cases} 24x_1 + 8x_2 \le 600 \\ 8x_1 + 8x_2 \le 400 \\ 3x_1 + 8x_2 \le 240 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 7**

$extr \leftarrow Z = 2x_1 + 4x_2$

при ограничениях

$$\begin{cases} 6x_1 + 2x_2 \le 12 \\ -x_1 + 2x_2 \le 8 \\ x_1 + x_2 \le 4 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 8**

$extr \leftarrow Z = 5x_1 + 7x_2$

при ограничениях

$$\begin{cases} 2x_1 + x_2 \le 12 \\ x_1 + 2x_2 \le 8 \\ x_1 + x_2 \le 5 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 9**

$extr \leftarrow Z = 4x_1 + 2x_2$

при ограничениях

$$\begin{cases} 3x_1 + x_2 \le 6 \\ 2x_1 + 4x_2 \le 10 \\ x_1 + 2x_2 \le 8 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 10**

$extr \leftarrow Z = 6x_1 - 4x_2$

при ограничениях

$$\begin{cases} -x_1 + 5x_2 \ge 0 \\ x_1 + 2x_2 \le 14 \\ 2x_1 + 2x_2 \ge 4 \\ -3x_1 + 2x_2 \le 6 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 11**

$extr \leftarrow Z = 14x_1 + 6x_2$

при ограничениях

$$\begin{cases} 7x_1 + 3x_2 \le 21 \\ x_1 - x_2 \le 0 \\ 5x_1 + x_2 \ge 5 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 12**

$extr \leftarrow Z = 6x_1 + 4x_2$

при ограничениях

$$\begin{cases} x_1 + x_2 \le 4 \\ x_1 + 2x_2 \le 6 \\ 3x_1 + x_2 \le 8 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 13**

$extr \leftarrow Z = 10x_1 + 8x_2$

при ограничениях

$$\begin{cases} x_1 + 2x_2 \ge 5 \\ 2x_1 - x_2 \ge 12 \\ x_1 + 3x_2 \ge 4 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 14**

$extr \leftarrow Z = 4x_1 + 6x_2$

при ограничениях

$$\begin{cases} 3x_1 + 4x_2 \le 17 \\ x_1 + x_2 \ge 1 \\ 2x_1 + x_2 \ge 3 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

**Вариант 15**

$extr \leftarrow Z = 3x_1 + 5x_2$

при ограничениях

$$\begin{cases} 2x_1 + x_2 \le 10 \\ x_1 + 2x_2 \le 8 \\ 3x_1 + x_2 \le 18 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$$

### 1.1.4.2 Методические указания

1 Выберите вариант задания по последней цифре номера зачетной книжки.

2 Разработайте алгоритм решения задачи.

3 Разработайте программу решения задачи.

4 Оформите отчет по практической работе, включающий:

– титульный лист;

– теоретическое обоснование;

– постановку задачи;

– описание алгоритма решения задачи;

– оптимальное решение математической модели;

– скриншоты результатов работы программы.

### 1.1.4.3 Контрольные вопросы

1 Что называется математической моделью?

2 Что называется оптимальным решением?

3 Что называется допустимым решением?

4 Какие этапы включает алгоритм метода Гаусса – Жордана?

5 Какие этапы включает алгоритм метода обыкновенного Жорданова исключения?

6 Какие этапы включает алгоритм метода модифицированного Жорданова исключения?

7 Какие этапы включает алгоритм метода отсекающих плоскостей?

8 Какие этапы включает алгоритм метода ветвей и границ?

9 Как формулируется условие оптимальности?

10 Как формулируется условие допустимости?

## 1.2 Методы решения задач нелинейного программирования

Модели линейного программирования не всегда адекватны реальным ситуациям. При решении задач полный и точный учёт зависимостей между факторами и показателями, влияющими на критерий эффективности и ограничительные условия, достигается при разработке нелинейных математических моделей.

### 1.2.1 Постановка задачи нелинейного программирования

Задача нелинейного программирования в общем виде состоит в отыскании вектора искомых переменных $X = (x_1, x_2, ..., x_n)$, при котором целевая функция достигает экстремума (максимума, минимума).

$$extr \leftarrow Z = f(x_1, x_2, ..., x_n)$$

при ограничениях (1.7)

$$\begin{cases} \varphi_i(x_1, x_2, ..., x_n) \le b_i, i = \overline{1, m} \\ \varphi_i(x_1, x_2, ..., x_n) \le b_i, i = b_i, i = \overline{m_1 + 1, m} \\ x_i \ge 0 \end{cases}$$

В отличие от задач линейной оптимизации, для задач нелинейного программирования общего метода, позволяющего решать любые оптимизационные задачи, не существует. Это обусловлено тем, что в задачах нелинейного программирования область допустимых решений может быть невыпуклой, иметь бесконечное число крайних точек, состоять из нескольких частей, а целевая функция может достигать экстремума не только на границе, но и внутри области допустимых решений системы ограничений. Нелинейная целевая функция может иметь несколько локальных экстремумов, среди которых необходимо найти глобальный [1-4].

### 1.2.2 Численные методы поиска безусловного экстремума

Задачей безусловной оптимизации функции нескольких переменных является задача, в которой требуется найти

$$extr \leftarrow Z = f(x_1, x_2, ..., x_n) \quad (1.8)$$

при отсутствии ограничений на $\bar{x} = (x_1, x_2, ..., x_n)$ – вектор управляемых переменных, $extr \leftarrow Z = f(x_1, x_2, ..., x_n)$ – скалярная целевая функция.

Численные методы решения задачи безусловной минимизации подразделяются на три группы: методы нулевого порядка, методы первого порядка, методы второго порядка.

#### 1.2.2.1 Методы нулевого порядка

Методы нулевого порядка (прямого поиска) – методы, основанные на вычислении только значения целевой функции $Z = f(x_1, x_2, ..., x_n)$.

##### 1.2.2.1.1 Методы одномерной оптимизации

Пусть функция $f(x)$ определена на $P \subseteq E_1$. Задача одномерной оптимизации – задача, в которой требуется найти max(min) $f(x)$, $x \in P$. Решение (точ-

ка максимума (минимума)) задачи одномерной оптимизации – точка $X^* \in P$, при условии $f(x^*) \le (\ge) f(x)$ для всех $x \in P$.

$$f(x^*) = \max_{x \in P} (\min) f(x) \tag{1.9}$$

Функция $f(x)$ называется унимодальной на множестве $P$, если существует единственная точка $x^*$ её максимума на $P$ и для любых $x_1, x_2 \in P$,

$$f(x_1) \le f(x_2) \le f(x^*), \text{ если } x_1 \le x_2 \le x^*;$$

$$f(x^*) \ge f(x_1) \ge f(x_2), \text{ если } x^* \le x_1 \le x_2$$

Унимодальная функция монотонно возрастает слева от точки максимума и монотонно убывает справа от неё.

В процессе применения методов одномерной оптимизации выделяют два этапа:

- поиск отрезка, содержащего точку максимума;
- уточнение координаты точки максимума на данном отрезке.

### 1.2.2.1.1.1 Алгоритм Свенна

Исходные данные: $X_0$ – начальная точка, $h$ – шаг поиска ($h > 0$).

**Этап 1.** Определить $f(x_0)$, $f(x_0 + h)$, $f(x_0 - h)$, $k = 1$.

**Этап 2.** Если $f(x_0 - h) \le f(x_0) \le f(x_0 + h)$, то $x_1 = x_0 + h$, перейти на 4 этап.

**Этап 3.** Если $f(x_0 - h) \ge f(x_0) \ge f(x_0 + h)$, то $x_1 = x_0 - h$, $h = -h$, перейти на 4 этап, в противном случае $f(x_0 - h) \le f(x_0) \ge f(x_0 + h)$, $a = x_0 - h$, $b = x_0 + h$, конец.

**Этап 4.** $x_{k+1} = x_k + 2^k * h$. Определить $f(x_{k+1})$.

**Этап 5.** Если $f(x_{k+1}) \ge f(x_k)$, то $k = k + 1$, переход на 4 этап.

**Этап 6.** Если $h > 0$, то $a = x_{k-1}$, $b = x_{k+1}$. конец. В противном случае $a = x_{k+1}$, $b = x_{k-1}$, конец.

Случай $f(x_0 - h) \ge f(x_0) \le f(x_0 + h)$ (этап 3) не рассматривается, так как он противоречит предположению о модальности функции $f(x)$.

### 1.2.2.1.1.2 Метод золотого сечения

Золотое сечение отрезка – деление отрезка на две неравные части таким образом, что отношение длины всего отрезка к длине большей части равно отношению длины большей части к длине меньшей части отрезка. Золотое сечение отрезка $[a, \mathrm{b}]$ производится двумя точками $y$ и $z$, симметричными относительно середины отрезка (рисунок 1.1).

$$\frac{b-a}{b-y}=\frac{b-y}{y-a}=\frac{b-a}{z-a}=\frac{z-a}{b-z}=\lambda=\frac{\sqrt{5}+1}{2}=1{,}618$$

$$y=(\lambda-1)*a+(\lambda-1)^2*b=0{,}618*a+0{,}382*b$$

$$z=(\lambda-1)^2*a+(\lambda-1)*b=0{,}382*a+0{,}382*0{,}618*b$$

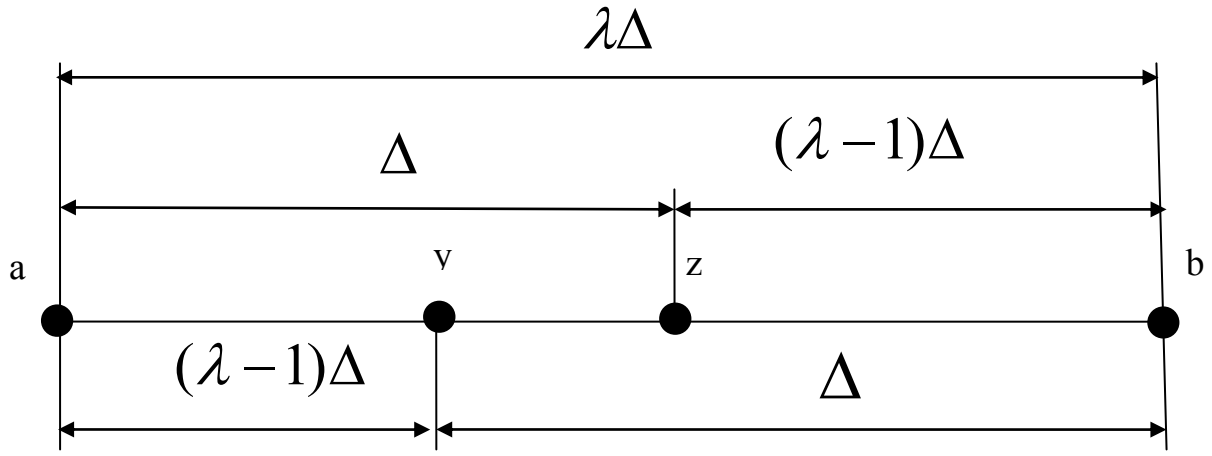


Рисунок 1.1 – Золотое сечение

Точка $y$ производит золотое сечение отрезка $[a, \mathrm{z}]$, а точка $z$ производит золотое сечение отрезка $[y, \mathrm{b}]$. На этом свойстве, позволяющем вычислять значение функции в одной пробной точке, основан алгоритм золотого сечения.

Алгоритм золотого сечения включает этапы:

Исходные данные: $[a, \mathrm{b}]$ – отрезок, содержащий точку максимума, $\varepsilon$ – параметр окончания счёта.

**Этап 1.** $\lambda=\frac{\sqrt{5}+1;}{2}$; $k=1$, $a_k=a$, $b_k=b$;

$y=(\lambda-1)*a_k+(\lambda-1)^2*b_k$; $A=f(y)$;

$z=(\lambda-1)^2*a_k+(\lambda-1)*b_k$; $B=f(z)$.

**Этап 2.** Если $A>B$, то перейти на 4 этап.

**Этап 3.** $a_{k+1}=y$; $b_{k+1}=b_k$;

$y=z$; $A=B$;

$z = (\lambda - 1)^2 * a_{k+1} + (\lambda - 1) * b_{k+1}$;

$B = f(z)$, перейти на 5 этап.

**Этап 4.** $a_{k+1} = a_k$; $b_{k+1} = z$; $z = y$; $B = A$;

$y = (\lambda - 1) * a_{k+1} + (\lambda - 1)^2 * b_{k+1}$;

$A = f(y)$.

**Этап 5.** Если $b_{k+1} - a_{k+1} < \varepsilon$, то $X^* \in [a_{k+1}, b_{k+1}]$, конец.

**Этап 6.** $k = k + 1$, перейти на этап 2.

### 1.2.2.1.2 Метод сопряжённых направлений (метод Пауэлла)

Постановка задачи: необходимо найти безусловный минимум $f(x)$ многих переменных, т. е. такую точку $x^* \in R^n$, что $f(x^*) = \min f(x)$, при $x \in R^n$.

Определение. Пусть $H$ – симметричная матрица размеров $(n * n)$. Векторы $d_1, d_2, ..., d_n$ называются H-сопряжёнными (просто сопряжёнными), если $d_i^T H d_j = 0$ при всех $i \neq j$.

В методе Пауэлла используется факт, что минимум квадратичной функции может быть найден не более, чем за n шагов при условии, что поиск ведётся вдоль сопряжённых относительно матрицы Гессе направлений.

Алгоритм метода Пауэлла

**Этап 1.** Задать начальную точку $x^0$, число $\varepsilon > 0$ для окончания алгоритма, начальные направления поиска $d_1 = \begin{pmatrix} 1 \\ 0 \\ ... \\ 0 \end{pmatrix}$, $d_2 = \begin{pmatrix} 0 \\ 1 \\ ... \\ 0 \end{pmatrix}$,…, $d_n = \begin{pmatrix} 0 \\ 0 \\ ... \\ 1 \end{pmatrix}$.

Положим $d_0 = d_n$, $i = 0$, $y^0 = x^0$, $k = 0$.

**Этап 2.** Найти $y^{i+1} = y^i + t_i d_i$, где параметр $t_i$ (неограниченный по знаку) находится в результате поиска минимума функции $f(y^i + t_i d_i)$ по $t_i$ одним из методов одномерной оптимизации.

**Этап 3.** Проверить выполнение условий:

a) если $i < n - 1$, положить $i = i + 1$ и перейти к этапу 2;

b) если $i = n - 1$, проверить успешность поиска по $n$ первым направлениям. Если $y^n = y^0$, то поиск завершить: $x^n = y^n$, иначе $i = i + 1$ и перейти к этапу 2;

c) если $i = n$, то проверить успешность поиска по $n$ последним направлениям. Если $y^{n+1} = y^1$, поиск завершить: $x^* = y^n$, иначе перейти к этапу 4 для построения сопряжённого направления.

**Этап 4.** Положить $x^{k+1} = y^{n+1}$ и проверить условие окончания:

a) если $\left|x^{k+1} - x^k\right| < \varepsilon$, то поиск завершить: $x^* = x^{k+1}$;

b) иначе положить: $\bar{d}_0 = \bar{d}_n = y^{n+1} - y^1$ (новое направление); $\bar{d}_i = \bar{d}_{i+1}$, $i = \overline{1, n-1}$ (исключается старое направление). Если при этом $rg(\bar{d}_{1,\ldots,\bar{d}_n}) = n$, то новая система направлений линейно зависима. Тогда следует продолжить поиск в старых направлениях. Для этого положить: $\bar{d}_i = \bar{d}_i$, $i = \overline{0, n}$, $y^0 = x^{k+1}$, $k = k+1$, $i = 0$ и перейти к этапу 2.

## 1.2.2.1.3 Практическая работа № 2 «Методы Свенна и золотого сечения в определении решения на экстремум унимодальной функции»

**Цель:** закрепить теоретические знания и получить практические навыки в разработке программы решения задач методами одномерной оптимизации.

**Используемые приемы и технологии:** технология визуального проектирования и событийного программирования

**Ключевые термины:** нелинейное моделирование, математическая модель, оптимальный план, допустимый план, алгоритм Свенна, метод золотого сечения.

**Постановка задачи:** разработайте визуальное приложение решения задачи методами одномерной оптимизации.

### 1.2.2.1.3.1 Варианты заданий

Методом Свенна найти отрезок, содержащий точку экстремума унимодальной функции $f(x)$, уточнить точку экстремума методом золотого сечения, $\varepsilon = 0{,}05$.

**Вариант 1.** $f(x) = x^2 + 1 \to \min$

**Вариант 2.** $f(x) = 3x - x^2 - 1 \to \max$

**Вариант 3.** $f(x) = 2x - x^2 - 1 \to \max$

**Вариант 4.** $f(x) = 2x^2 + 3 \to \min$

**Вариант 5.** $f(x) = x^2 + x + 1 \to \min$

**Вариант 6.** $f(x) = 10x^2 + 7x + 1 \to \min$

**Вариант 7.** $f(x) = 15x - 2x^2 + 5 \to \max$

**Вариант 8.** $f(x) = 3 + 7x - 2x^2 \to \max$

**Вариант 9.** $f(x) = 4 - 3x^2 \to \max$

**Вариант 10.** $f(x) = 5 - 4x - x^2 \to \max$

**Вариант 11.** $f(x) = 12 + 6x + x^2 \to \min$

**Вариант 12.** $f(x) = 14 - 6x + x^2 \to \min$

**Вариант 13.** $f(x) = 8 - 3x - 2x^2 \to \max$

**Вариант 14.** $f(x) = 3 + 4x + x^2 \to \min$

**Вариант 15.** $f(x) = 10 + 5x + 2x^2 \to \min$

### 1.2.2.1.3.2 Методические указания

1 Разработайте алгоритм программы.

2 Разработайте визуальное приложение, формализующее алгоритм методов одномерной оптимизации.

3 Оформите отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- теоретическое обоснование;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

### 1.2.2.1.3.3 Контрольные вопросы

1 Что называется нелинейным программированием?

2 Какая постановка задачи нелинейного программирования?

3 Что называется унимодальной функцией?

4 Что понимается под нелинейной моделью?

5 Какие методы одномерной оптимизации?

6 Сколько и какие этапы включает алгоритм Свенна?

7 Что понимается под золотым сечением?

8 Какое свойство золотого сечения?

9 Какие исходные данные содержит алгоритм метода золотого сечения?

10 Сколько и какие этапы включает алгоритм метода золотого сечения?

### 1.2.2.2 Методы первого порядка

Методы первого порядка (градиентные методы) – методы, в которых используются значения функции $f(x)$ и её градиента, т. е. вектора, компонентами которого являются частные производные первого порядка.

**Определение.** Решением (точкой минимума) задачи безусловной оптимизации называется вектор $\bar{x}^* \in E_n$, такой, что $f(\bar{x}^*) \le f(\bar{x})$ для всех $\bar{x} \in E_n$, $f(\bar{x}^*) = \min f(\bar{x}),\ \bar{x} \in E_n$.

**Определение.** Вектор $\bar{S}$ называется направлением спуска функции $f(\bar{x})$ в точке $\bar{x}$, если существует такое $\delta > 0$, что $f(\bar{x} + \lambda \bar{S}) \le f(\bar{x})$, для всех $\lambda \in (0; \delta)$.

Сущность рассматриваемых методов решения задачи (1.8) состоит в определении последовательности точек $\bar{x}_1, \bar{x}_2, ..., \bar{x}_k$, принадлежащих $E_n$, монотонно уменьшающих значение функции $f(\bar{x})$. Такие методы называются методами спуска.

Градиентные методы – методы безусловной оптимизации, в которых в качестве направления поиска берётся функция $f(\bar{x})$. Последовательность точек генерируется градиентным методом в соответствии с формулой

$$\bar{x}_{k+1} = \bar{x}_k + \lambda_k \nabla f(\bar{x}_k), \qquad (1.10)$$

где $\lambda_k$ – параметр, характеризующий величину шага вдоль направления. Величина шага $\lambda_k$ может выбираться разными способами. Если значение параметра $\lambda_k$ вычисляется путём решения задачи одномерной оптимизации, то градиентный метод называется методом наискорейшего спуска, или методом Коши.

#### 1.2.2.2.1 Метод наискорейшего градиентного спуска

Метод наискорейшего градиентного спуска (метод Коши) – градиентный метод, у которого параметр $\lambda_k$ в формуле (1.10) вычисляется путём решения задачи одномерной оптимизации.

Алгоритм метода Коши

*Начальный этап.* Выбрать $\bar{x}_1$ – начальную точку, $\varepsilon > 0$ – параметр окончания счёта, положить $k = 1$.

*Основной этап.*

Шаг 1. Если $\left\| \nabla f(\bar{x}_k) \right\| < \varepsilon$, то $\bar{x}^* = x_k$, остановиться.

Шаг 2. Положить $\bar{S}_k = -\nabla f(\bar{x}_k)$, вычислить $\lambda_k = \arg\min f(\bar{x}_{k+\lambda\bar{S}_k})$, положить $k = k+1$ и перейти к шагу 1.

### 1.2.2.2.2 Практическая работа № 3
### «Метод Коши в решении задач безусловной оптимизации»

**Цель:** закрепить теоретические знания и получить практические навыки в разработке программы решения задач градиентным методом наискорейшего спуска.

**Используемые приемы и технологии:** технология визуального проектирования и событийного программирования.

**Ключевые термины:** нелинейное моделирование, математическая модель, оптимальный план, градиентный метод наискорейшего спуска.

**Постановка задачи:** разработайте визуальное приложение решения задачи методом Коши.

#### 1.2.2.2.2.1 Варианты заданий

Методом Коши решить задачу безусловной оптимизации с точностью $\varepsilon = 0{,}05$. Решение сопроводить геометрической интерпретацией.

**Вариант 1.** $f(x) = -x_1 + 6x_2 - 2x_1^2 - 3x_2^2 + 3x_1x_2 \to \max$

**Вариант 2.** $f(x) = 6x_1 + 4x_2 - x_1^2 - 0{,}5x_2^2 - x_1x_2 \to \max$

**Вариант 3.** $f(x) = 3x_1 - 2x_2 - 0{,}5x_1^2 - x_2^2 + x_1x_2 \to \max$

**Вариант 4.** $f(x) = -4x_1 + 8x_2 - x_1^2 - 1{,}5x_2^2 + 2x_1x_2 \to \max$

**Вариант 5.** $f(x) = x_1 + 4x_2 - x_1^2 - 3x_2^2 + 2x_1x_2 \to \max$

**Вариант 6.** $f(x) = -2x_1 + x_2 - 3x_1^2 - 2x_2^2 + x_1x_2 \to \max$

**Вариант 7.** $f(x) = x_1 - 2x_2 - x_1^2 - 3x_2^2 - x_1x_2 \to \max$

**Вариант 8.** $f(x) = 3x_1 + 6x_2 - x_2^2 + 2x_1x_2 \to \max$

**Вариант 9.** $f(x) = -3x_1 + 2x_2 - 2x_1^2 - x_2^2 + 2x_1x_2 \to \max$

**Вариант 10.** $f(x) = 4x_1 + x_2 - 3x_1^2 - x_2^2 + x_1x_2 \to \max$

**Вариант 11.** $f(x) = x_1^3 - x_1x_2 + x_2^2 - 2x_1 + 3x_2 - 4 \to \min$, $x^0 = (0,0)^T$

**Вариант 12.** $f(x) = (x_2 - x_1^2)^2 + (1 - x_1)^2 \to \min$, $x^0 = (0,0)^T$

**Вариант 13.** $f(x) = (x_2^2 + x_1^2 - 1)^2 + (x_1 + x_2 - 1)^2 \to \min$, $x^0 = (0,3)^T$

**Вариант 14.** $f(x) = 4(x_1 - 5)^2 + (x_2 - 6)^2 \to \min$, $x^0 = (8,9)^T$

**Вариант 15.** $f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \to \min$, $x^0 = (0,0)^T$

### 1.2.2.2.2.2 Методические указания

1 Разработайте алгоритм программы.

2 Разработайте визуальное приложение, реализующее градиентный алгоритм наискорейшего спуска.

3 Оформите отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- теоретическое обоснование;
- скриншоты программы;
- код программы;
- выводы.

### 1.2.2.2.2.3 Контрольные вопросы

1 Что называется градиентным методом?

2 Что понимается под направлением спуска функции $f(x)$ в точке $x$?

3 Как определяется Евклидова норма матрицы?

4 На какие группы разделяются методы решения задач безусловной оптимизации в зависимости от уровня используемой в методе информации о целевой функции?

5 Какие этапы включает алгоритм метода спуска?

6 Сколько и какие этапы включает градиентный алгоритм наискорейшего спуска?

7 Что понимается под точкой минимума задачи безусловной оптимизации?

8 Каким образом определяется параметр, характеризующий величину шага вдоль направления спуска?

9 Какие исходные данные содержит начальный этап алгоритма метода Коши?

10 Что понимается под градиентным методом наискорейшего спуска?

## 1.2.2.3 Методы второго порядка

Методы второго порядка решения задач безусловной оптимизации – методы, в которых используются первые и вторые производные функции $f(x)$,

т. е. $f(x)$, $\nabla f(x)$, $H(x)$, где $H(x)$ – матрица Гессе, элементами которой являются частные производные второго порядка функции $f(x)$.

### 1.2.2.3.1 Метод Ньютона

Постановка задачи. Пусть дана функция, ограниченная снизу на множестве $R^n$ и имеющая непрерывные частные производные во всех его точках. Необходимо найти локальный минимум функции $f(x)$ на множестве допустимых решений $X = R^n$, т. е. найти такую точку $x^* \in R^n$, что $f(x^*) = \min f(x)$, $x \in R^n$, $f(x) \in C^2$.

Стратегия поиска состоит в построении последовательности точек $\{x^k\}$, $k = 0,1,...$, таких что $f(x^{k+1}) < f(x^k)$, $k = 0,1,....$ Точки последовательности вычисляются по правилу

$$x^{k+1} = x^k + d^k,\ k = 0,1,... \qquad (1.11)$$

где $x^0$ – задаётся пользователем, а направление спуска $d^k$ определяется для каждого значения $k$ по формуле

$$d^k = -H^{-1}(x^k)\nabla f(x^k) \qquad (1.12)$$

Выбор $d^k$ по формуле (1.12) гарантирует выполнение требований $f(x^{k+1}) < f(x^k)$ при условии, что $H(x^k) > 0$. При выполнении условия $H(x^k) > 0$ последовательность является последовательностью точек минимумов квадратичных функций $F_k$, $k = 0,1,...$. Построение $\{x^k\}$ заканчивается в точке $x^k$, для которой $\left\|\nabla f(x^k)\right\| < \varepsilon_1$, где $\varepsilon_1$ – заданное малое положительное число, или при $k > M$, где $M$ – предельное число итераций, или при двукратном одновременном выполнении двух неравенств $\left\|x^{k+1} * x^k\right\| < \varepsilon_2$, $\left|f(x^{k+1}) - f(x^k)\right| < \varepsilon_2$, где $\varepsilon_2$ – малое положительное число.

Алгоритм метода Ньютона

**Этап 1.** Задать исходные данные $x^0$, $\varepsilon_1 > 0$, $\varepsilon_2 > 0$, $M$ – предельное число итераций. Найти градиент $\nabla f(x)$, матрицу Гессе $H(x)$.

**Этап 2.** Положить $k = 0$.

**Этап 3.** Вычислить градиент $\nabla f(x^k)$.

**Этап 4.** Проверить выполнение условия $\left\|\nabla f(x^k)\right\| < \varepsilon_1$:

а) если условие выполняется, то расчёт окончен и $x^* = x^k$;

b) иначе перейти к этапу 5.

**Этап 5.** Проверить выполнение неравенства $k \geq M$ :

a) если неравенство выполнено, то расчёт окончен и $x^* = x^k$;

b) иначе перейти к этапу 6.

**Этап 6.** Вычислить матрицу $H(x)$.

**Этап 7.** Вычислить матрицу $H^{-1}(x^k)$.

**Этап 8.** Проверить выполнение условия $H^{-1}(x^k) > 0$:

a) если $H^{-1}(x^k) > 0,$ то перейти к этапу 9;

b) иначе перейти к этапу 10, положив $d^k = -\nabla f(x^k)$.

**Этап 9.** Определить $d^k = -H^{-1}(x^k)\nabla f(x^k)$.

**Этап 10.** Найти точку $x^{k+1} = x^k + t_k d^k$, положив $t_k = 1,$ если $d^k = -H^{-1}(x^k)\nabla f(x^k),$ или выбрав $t_k$ из условия $f(x^{k+1}) < f(x^k),$ если $d^k = -\nabla f(x^k)$.

**Этап 11.** Проверить выполнение условий $\left\|x^{k+1} - x^k\right\| < \varepsilon_2,$ $\left|f(x^{k+1}) - f(x^k)\right| < \varepsilon_2$:

a) если оба условия выполнены при текущем значении $k$ и $k = k - 1,$ то расчёт окончен, $x^k = x^{k+1}$;

b) иначе положить $k = k + 1$ и перейти к этапу 3.

### 1.2.2.4 Практическая работа № 4
### «Метод Ньютона в решении задач безусловной оптимизации»

**Цель:** закрепить теоретические знания и получить практические навыки в разработке программы решения задач методом Ньютона.

**Используемые приемы и технологии:** технология визуального проектирования и событийного программирования.

**Ключевые термины:** нелинейное моделирование, математическая модель, оптимальный план, метод Ньютона.

**Постановка задачи:** разработайте визуальное приложение решения задачи методом Ньютона.

#### 1.2.2.4.1 Варианты заданий

Методом Ньютона решить задачу безусловной оптимизации. Решение сопроводить геометрической интерпретацией.

**Вариант 1.** $x_1^2 + x_1x_2 + 2x_2^2 \to \min$

**Вариант 2.** $7x_1^2 + 2x_1x_2 + 5x_2^2 + x_1 - 10x_2 \to \min$

**Вариант 3.** $3x_1^2 - 3x_1x_2 + 5x_2^2 - 2x_1 + x_2 \to \min$

**Вариант 4.** $x_1^2 + 4x_1x_2 + 17x_2^2 + 5x_1 \to \min$

**Вариант 5.** $5x_1^2 - 4x_1x_2 + 5x_2^2 - x_1 - x_2 \to \min$

**Вариант 6.** $4x_1^2 - 4x_1x_2 + 6x_2^2 - 17x_1 \to \min$

**Вариант 7.** $2x_1^2 - 2x_1x_2 + 3x_2^2 + x_1 - 3x_2 \to \min$

**Вариант 8.** $10x_1^2 + 3x_1x_2 + x_2^2 + 10x_2 \to \min$

**Вариант 9.** $x_1^2 - 2x_1x_2 + 6x_2^2 + x_1 - x_2 \to \min$

**Вариант 10.** $x_1^2 + x_2^2 + x_1 + x_2 \to \min$

**Вариант 11.** $x_1^2 + x_1x_2 + x_2^2 \to \min$

**Вариант 12.** $x_1^2 + 4x_2^2 + 3x_1 - 2x_2 \to \min$

**Вариант 13.** $100(x_2 - x_1^2) + (1 - x_1)^2 \to \min$, $x^0 = (2,3)^T$

**Вариант 14.** $3x_1^2 + x_1x_2 + 2x_2^2 \to \min$

**Вариант 15.** $x_1^2 + x_1x_2 + 4x_2^2 \to \min$

### 1.2.2.4.2 Методические указания

1 Разработайте алгоритм программы.

2 Разработайте визуальное приложение, реализующее алгоритм метода Ньютона.

3 Оформите отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- теоретическое обоснование;
- скриншоты программы;
- код программы;
- выводы.

### 1.2.2.4.3 Контрольные вопросы

1 Как формулируется постановка задачи решения методами второго порядка?

2 Каким образом можно повысить эффективность процесса поиска при решении задач нелинейного программирования методами второго порядка?

3 Какая стратегия поиска при использовании метода Ньютона?

4 Какие этапы включает алгоритм метода Ньютона?

5 Какие методы являются модификацией метода Ньютона?

6 Какие методы решения задач нелинейного программирования называются квазиньютоновскими методами?

7 Какая стратегия поиска при использовании метода Ньютона – Рафсона?

8 Какие этапы включает алгоритм метода Ньютона – Рафсона?

9 Какая стратегия поиска при использовании метода Марквардта?

10 Какие этапы включает алгоритм методов Марквардта?

### 1.2.3 Численные методы поиска условного экстремума

Методы решения задач условной оптимизации можно представить как итерационный процесс, в котором, исходя из начальной точки $x_0 \in P$, получают последовательность точек $x_k \in P$, монотонно изменяющих значение целевой функции $f(x)$.

Общая схема методов условной оптимизации состоит из нескольких этапов.

*Начальный этап.* Задать $\varepsilon > 0$ и выбрать начальную точку $x_0 \in P$.

*Основной этап.*

**Этап 1.** Выбрать $S_k$ ($k$-я итерация) – возможное направление спуска функции $f(x)$ в точке $x_k$. Если такого направления нет, то $x_k^* = x_k$ – решение задачи. В противном случае перейти к этапу 2.

**Этап 2.** Положить $x_{k+1} = x_k + \beta S_k$, где $\beta_k$ находим, решая задачу

$$f(x_k + \beta_k S_k) \to \min$$

$$\beta > 0,\ (x_k + \beta_k S_k) \in P$$

**Этап 3.** Заменить $k$ на $k+1$ и перейти к этапу 1.

Методы условной оптимизации различаются способом выбора возможного направления спуска $S_k$ функции $f(x)$ в точке $x$.

#### 1.2.3.1 Метод Зойтендейка

Постановка задачи: найти минимум дважды непрерывно дифференцируемой функции $f(x)$ при условии, что вектор $x \in R^n$ удовлетворяет условию $g(x) \le 0,\quad j = \overline{1,m}$, при котором функции $g_j(x),\quad j = \overline{1,m}$ есть также дважды непрерывно дифференцируемые функции по $x$, т. е. такую точку $x^* \in X$, что

$$f(x^*) = \min\ f(x),\ x \in X$$
$$X = \{x \mid g_j(x) \le 0,\ j = \overline{1,m}\} \qquad (1.13)$$

Стратегия решения задачи методом Зойтендейка состоит в построении последовательности допустимых точек $\{x^k\}$, таких, что $f(x^{k+1}) < f(x^k)$, $k = 0,1,...$ Правило построения последовательности $\{x^k\}$:

$$x^{k+1} = x^k + t_k d^k,\ k = 0,1,..., \qquad (1.14)$$

где $x^k$ – такая допустимая точка, что выполняется условие

$$-\varepsilon_k < g_{j(x^k)\le 0},\ j \in J_a, \qquad (1.15)$$

где $J_{a`}$ – множество индексов $j$ активных ограничений, для которых выполнено условие;

$t_k > 0$ – величина шага, определяемая в результате решения задачи одномерной оптимизации

$$f(x^k + t_k d^k) \to \min; \qquad (1.16)$$
$$g_j(x^k + t_k d^k) \le 0,\ j = \overline{1,m}$$

Алгоритм метода Зойтендейка включает несколько этапов.

**Этап 1.** Задать $\varepsilon_0$, $M$ – предельное число итераций, $x^0 \in X$ – допустимую начальную точку, в которой $\nabla f(x^0) \ne 0$.

**Этап 2.** Положить $k = 0$.

**Этап 3.** Проверить выполнение условия $k < M$:

a) если $k = M$, расчёт закончен;

b) если $k < M$, перейти к этапу 4.

**Этап 4.** Выполнить $g_j(x^k),\ j = \overline{1,m}$.

**Этап 5.** Проверить выполнение условия $-\varepsilon_k \le g_j(x^k) \le 0,\ j = \overline{1,m}$. Сформировать множество $J_a$ индексов $j$, для которых условие выполнено. Если условие выполнено хотя бы для одного $j \in J_a$, то перейти к этапу 6. Иначе положить $\varepsilon_k = 2\varepsilon_k$ и повторить вычисления на этапе 5.

**Этап 6.** Записать систему неравенств

$$\nabla f(x^k)^T d^k < 0; \qquad (1.17)$$
$$\nabla g_j(x^k)^T d^k < 0,\ j \in J_a$$

**Этап 7.** Сформировать задачу линейного программирования

$$Z \to \min$$
$$\nabla f(x^k)^T d^k < Z; \qquad (1.18)$$

$$\nabla g_j(x^k)^T d^k < Z;\ j \in J_a;$$

$$|\, d_k \,| \le 1,\ i = \overline{1,m}$$

**Этап 8.** Решить задачу линейного программирования, сформированную на этапе 7. В результате находится искомое возможное направление спуска $d^k$ и $Z^*$ – минимальное значение $Z$.

**Этап 9.** Вычислить шаг $t_k$, решив задачу

$$f(x^k + t_k d^k) \to \min; \qquad (1.19)$$

$$g_j(x^k + t_k d^k) \le 0,\ j = \overline{1,m},$$

или из соотношения $t_k = \min\{t_k^* \ge 0; t_k^{**} \ge 0\}$ для чего следует:

a) найти величину $t_k^*$ из условия $f(x^k + t_k^* d^k) = \min f(x^k + t_k d^k);\ t_k > 0$.

b) определить величину $t_k^j$, $j = \overline{1,m}$ из условий $g_j(x^k + t_k \Delta x^k) = 0,\ t_k \ge 0.$ Если условие $g_j(x^k + t_k \Delta x^k) = 0$ выполняется только при $t_k < 0$, то $t_k^j$ не вычисляется. Если в точке $x^k$ ограничение с номером $j$ активно и $t_k = 0$, то значение $t_k^j$ не вычисляется;

c) найти $t_k^{**} = \min\{t_k^j\}$;

d) вычислить значение $t_k = \min\{t_k^*; t_k^{**}\}$.

**Этап 10.** Найти точку $x^{k+1} = x^k + t_k d^k$.

**Этап 11.** Проверить условие окончания:

a) если $Z^* \ge -\varepsilon_k$, то расчёт может быть либо закончен, если точность $\varepsilon_k$ удовлетворительна, либо продолжен при $\varepsilon_{k+1} = q\varepsilon_k$, $0 < q < 1$. В первом случае $x^{k+1}$ – искомое приближённое решение задачи;

b) $f(x^k + t_k d^k) \to \min;$

c) $g_j(x^k + t_k d^k) \le 0,\ \ j = \overline{1,m}.$

Во втором случае перейти к этапу 3.

### 1.2.3.2 Практическая работа № 5
### «Метод Зойтендейка в решении задач условной оптимизации»

**Цель:** закрепить теоретические знания и получить практические навыки в разработке программы решения задач методом Зойтендейка.

**Используемые приемы и технологии:** технология визуального проектирования и событийного программирования.

**Ключевые термины:** нелинейное моделирование, математическая модель, оптимальный план, метод Зойтендейка.

**Постановка задачи:** разработайте визуальное приложение решения задачи методом Зойтендейка.

### 1.2.3.2.1 Варианты заданий

Методом Зойтендейка решить задачу условной оптимизации с точностью $\varepsilon = 0{,}05$. Решение сопроводить геометрической интерпретацией.

**Вариант 1**

$$\min \leftarrow Z = (x_1 - 4)^2 + (x_2 - 5)^2$$

при ограничениях

$$\begin{cases} x_1 + x_2 - 1 \le 0 \\ -x_1 \le 0 \\ -x_2 \le 0 \end{cases}$$

**Вариант 2**

$$\min \leftarrow Z = (x_1 - 3)^2 + (x_2 - 6)^2$$

при ограничениях

$$\begin{cases} 2x_1 + x_2 - 1 \le 0 \\ -2x_1 \le 0 \\ -x_2 \le 0 \end{cases}$$

**Вариант 3**

$$\min \leftarrow Z = (x_1 - 2)^2 + (x_2 - 4)^2$$

при ограничениях

$$\begin{cases} x_1 + 3x_2 - 1 \le 0 \\ -x_1 \le 0 \\ -3x_2 \le 0 \end{cases}$$

**Вариант 4**

$$\min \leftarrow Z = (x_1 - 5)^2 + (x_2 - 7)^2$$

при ограничениях

$$\begin{cases} x_1 + 2x_2 - 1 \le 0 \\ -x_1 \le 0 \\ -2x_2 \le 0 \end{cases}$$

**Вариант 5**

$$\min \leftarrow Z = (x_1 - 2)^2 + (x_2 - 3)^2$$

при ограничениях

$$\begin{cases} 3x_1 + x_2 - 1 \le 0 \\ -3x_1 \le 0 \\ -x_2 \le 0 \end{cases}$$

**Вариант 6**

$$\min \leftarrow Z = (x_1 - 7)^2 + (x_2 - 8)^2$$

при ограничениях

$$\begin{cases} 4x_1 + x_2 - 1 \le 0 \\ -2x_1 \le 0 \\ -x_2 \le 0 \end{cases}$$

**Вариант 7**

$$\min \leftarrow Z = (x_1 - 9)^2 + (x_2 - 10)^2$$

при ограничениях

$$\begin{cases} x_1 + 4x_2 - 1 \le 0 \\ -x_1 \le 0 \\ -2x_2 \le 0 \end{cases}$$

**Вариант 8**

$$\min \leftarrow Z = (x_1 - 6)^2 + (x_2 - 9)^2$$

при ограничениях

$$\begin{cases} x_1 + 5x_2 - 1 \le 0 \\ -2x_1 \le 0 \\ -x_2 \le 0 \end{cases}$$

**Вариант 9**

$\min \leftarrow Z = (x_1 - 10)^2 + (x_2 - 12)^2$

при ограничениях

$$\begin{cases} 2x_1 + 3x_2 - 1 \le 0 \\ -x_1 \le 0 \\ -x_2 \le 0 \end{cases}$$

**Вариант 10**

$\min \leftarrow Z = (x_1 - 12)^2 + (x_2 - 11)^2$

при ограничениях

$$\begin{cases} x_1 + x_2 - 1 \le 0 \\ -2x_1 \le 0 \\ -3x_2 \le 0 \end{cases}$$

### 1.2.3.2.2 Методические указания

1 Разработайте алгоритм программы.

2 Разработайте визуальное приложение, реализующее алгоритм метода Зойтендейка.

3 Оформите отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- теоретическое обоснование;
- скриншоты программы;
- код программы;
- выводы.

### 1.2.3.2.3 Контрольные вопросы

1 Какая постановка задачи нелинейного программирования условной оптимизации?

2 Какие этапы включает общий алгоритм методов условной оптимизации?

3 Какие методы нелинейного программирования относятся к методам последовательной безусловной оптимизации?

4 Какие методы нелинейного программирования относятся к методам возможных направлений?

5 В чём заключается сущность методов последовательной безусловной оптимизации?

6 В чём заключается сущность методов возможных направлений?

7 Какие этапы включает метод штрафов?

8 Какие особенности сходимости метода штрафов?

9 Какие этапы включает метод Зойтендейка?

10 Какая стратегия поиска у метода Зойтендейка?

# 2 МЕТОДЫ ЭВОЛЮЦИОННОЙ ОПТИМИЗАЦИИ

## 2.1 Простой бинарный генетический алгоритм

Простой бинарный генетический алгоритм решает задачу, возможное решение задачи можно представить в виде битовой строки. Решение задачи называется особью. Группа особей называется популяцией.

Простой бинарный генетический алгоритм включает несколько этапов.

*Родители <–{случайная популяция}*

***While not*** *{критерий останова}*

*Рассчитать приспособленность каждого родителя в популяции*

*Дети <–0*

***While*** *|Дети|<|Родители|*

*Применить приспособленности для вероятностного отбора пары родителей с целью их спаривания*

*Спарить родителей для создания детей C1 и C2*

*Дети <–Дети U{C1, C2}*

***Loop***

*Случайно мутировать несколько детей*

*Родители<–Дети*

***Next*** *поколение*

**Задача One-max.** Задача нахождения подстроки заключается в поиске подстроки из n бит с максимально возможным числом единиц. Приспособленность битовой строки – это число единиц. Разработайте программное приложение решения задачи генетическим алгоритмом. Используйте n = 30, лимит поколений = 100, размер популяции = 20 и скорость мутации = 1%.

## 2.2 Метод симуляции восстановления

Симулированное закаливание (simulated annealing, SA) – алгоритм оптимизации, основанный на охлаждающем и кристаллизующем поведении химических веществ.

Разработан Скоттом Киркпатриком, Чарльзом Гелаттом и Марко Векки в 1983 г. для оптимального решения задач, связанных с компьютерным проектированием, таких как размещение компонентов схем и маршрутизация проводов.

### 2.2.1 Алгоритм метода отжига

Алгоритм отжига включает пять этапов: начальное решение, оценка решения, случайный поиск решения, критерий допуска, снижение температуры.

**Этап 1.** Начальное решение. Для большинства задач начальное решение является случайным. Оно помещается в текущее решение (current solution). Возможно загрузить в качестве начального решения существующее решение, найденное во время предыдущего поиска. Это предоставляет алгоритму базу, на основании которой выполняется поиск оптимального решения задачи.

**Этап 2.** Оценка решения. Оценка решения состоит из декодировки текущего решения и выполнения действий, позволяющих понять его целесообразность для решения задачи. Закодированное решение может состоять из набора переменных. Они будут декодированы из существующего решения, а затем производится оценка эффективности решения.

**Этап 3.** Случайный поиск решения. Производится копирование текущего решения в рабочее решение (working solution). Произвольно изменяется рабочее решение. После выполнения поиска рабочего решения осуществляется оценивание решения. Поиск нового решения основан на методе Монте-Карло.

**Этап 4.** Критерий допуска. Имеются два решения:

1) текущее решение;

2) найденное (рабочее) решение.

С каждым решением связана определённая энергия, представляющая собой эффективность решения. Например, чем меньше энергия, тем выше эффективность решения.

Рабочее решение сравнивается с текущим решением. Если рабочее решение имеет меньшую энергию, чем текущее решение, то производится копирование рабочего решения в текущее решение и осуществляется переход к этапу 5. Если рабочее решение хуже текущего решения, то определяется критерий допуска, чтобы выяснить, что следует сделать с текущим рабочим решением. Вероятность допуска основывается на уравнении

$$P(\delta E) = \exp(-\delta E / T), \tag{2.1}$$

где $T$ – температура;

$\delta E$ – дельта энергии.

При высокой температуре плохие решения принимаются чаще, чем отбрасываются. Если энергия меньше, вероятность принятия решения выше. При снижении температуры вероятность принятия худшего решения снижается. При этом более высокий уровень энергии способствует уменьшению вероятности принятия худшего решения.

При высоких температурах симулированное закаливание позволяет принимать худшие решения для того, чтобы произвести более полный поиск решений. При снижении температуры диапазон поиска уменьшается, пока не достигает равенства при температуре 0 градусов по Цельсию.

**Этап 5.** Снижение температуры. Существует множество вариантов снижения температуры: линейное, экспоненциальное, обратное, логарифмическое, обратное линейное, размерно-зависимое охлаждения.

В качестве варианта снижения температуры может использоваться геометрическая функция

$$T_{i+1} = \alpha T_i, \tag{2.2}$$

где $T_i$ – температура;

$\alpha$ – коэффициент, $0 < \alpha < 1$.

При одной температуре выполняется несколько итераций. После завершения итераций температура будет понижена. Процесс продолжается пока температура не достигнет нуля.

### 2.2.2 Практическая работа № 6
### «Метод симуляции восстановления в решении задачи N шахматных ферзей (NQP)»

Задачи с ограничениями – разновидность поисковых задач, в которых состояния должны удовлетворять ряду ограничений. Если выразить ограничения в виде затрат, а затем минимизировать затраты, то задача с ограничениями сведётся к общей задаче поиска.

Задачи с ограничениями применяются в искусственном интеллекте, исследовании операций и распознавании образов. Решение задач с ограничениями производится с помощью генетических алгоритмов.

#### 2.2.2.1 Варианты заданий

Реализовать задачу N шахматных ферзей. Варианты заданий представлены в таблице 2.1.

Таблица 2.1 – Варианты заданий

| Вариант | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| n | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |

**Постановка задачи.** Требуется расставить на доске размером N*N N ферзей так, чтобы никакие два не били друг друга. На одной горизонтали, на одной вертикали и на одной диагонали не может находиться два ферзя.

Задача имеет решение для любого натурального числа n, кроме n = 2, n = 3. В задаче о восьми ферзях решений 92, из них 12 уникальных, не сводимых друг к другу симметрией.

Реализовать генетический алгоритм решения задачи на ПЭВМ.

Провести эксперименты с параметрами алгоритма.

#### 2.2.2.2 Методические указания

1 Разработайте алгоритм программы.

2 Разработайте программное приложение, реализующее алгоритм отжига.

3 Проведите эксперимент с параметрами алгоритма.

4 Оформите отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- теоретическое обоснование;
- скриншоты программы;
- код программы;
- выводы.

#### 2.2.2.3 Контрольные вопросы

1 Что лежит в основе симулированного закаливания?

2 Какой вид имеет математическое уравнение вероятности того, что система атомов находится в конфигурации s?

3 Какие этапы включает алгоритм отжига?

4 Какие режимы охлаждения имеет алгоритм отжига?

5 Какие математические выражения описывают режимы охлаждения метода симулированного закаливания?

6 Каким образом влияет температура на результативность метода отжига?

7 Каким образом влияет режим охлаждения на результативность метода отжига?

8 Каким образом влияет стратегия кандидатного решения на каждой итерации на результативность метода отжига?

9 Какие преимущества имеет метод симулированного закаливания?

10 Какая область применения метода отжига?

### 2.3 Метод оптимизации на основе муравьиной кучи

Оптимизация на основе муравьиной кучи (ant colony optimization, ACO) – алгоритм, обусловленный феромонным поведением муравьев. Алгоритм оптимизации на основе муравьиной кучи разработан Марко Дориго в 1991 г. Первоначально предполагалось применение алгоритма муравья для решения задач

комбинаторной оптимизации (задача о рюкзаке, коммивояжере, маршрутизации транспорта), но алгоритм муравья был использован для решения непрерывных оптимизационных задач (оптимизация функций Eggholder, Химмельблау, Симионеску). После модификации алгоритма муравья были получены алгоритмы минимаксной муравьиной системы (MMAS) и системы муравьиной кучи (ACS).

### 2.3.1 Алгоритм метода оптимизации на основе муравьиной кучи

Алгоритм муравья, независимо от модификаций, представляется в следующем виде. (***Пока условия выхода не выполнены.)***

**1. Создаём муравьёв.**

**2. Ищем решения.**

**3. Обновляем феромон.**

**4. Дополнительные действия (опционально).**

Рассмотрим каждый шаг в цикле более подробно.

**1. Создаём муравьёв.**

Стартовая точка, куда помещается муравей, зависит от ограничений, накладываемых условиями задачи. Потому что для каждой задачи способ размещения муравьёв является определяющим. Либо все они помещаются в одну точку, либо в разные с повторениями, либо без повторений.

На этом же этапе задаётся начальный уровень феромона. Он инициализируется небольшим положительным числом для того, чтобы на начальном шаге вероятности перехода в следующую вершину не были нулевыми.

**2. Ищем решения.**

Вероятность перехода из вершины i в вершину j определяется по следующей формуле

$$P_{ij,k(t)} = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum\limits_{j \in J_{i,k}} [\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}, \qquad (2.3)$$

где $\tau_{ij}(t)$ – уровень феромона;

$\eta_{ij}(t)$ – эвристическое расстояние;

$\alpha, \beta$ – константные параметры.

Необходим компромисс между этими величинами (чтоб смягчать жадность алгоритма и не застревать в локальных минимумах), который находится экспериментально.

**3. Обновляем феромон.**

Уровень феромона обновляется в соответствии с приведённой формулой

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k \in \{used(i,j)\}} \frac{Q}{L_k(t)}, \quad (2.4)$$

где $\rho$ – интенсивность испарения;

$L_k(t)$ – цена текущего решения для k-го муравья;

$Q$ – параметр, имеющий значение порядка цены оптимального решения, то есть $Q/L_k(t)$ – феромон, откладываемый k-м муравьём, использующим ребро (i, j).

**4. Дополнительные действия (опционально).**

Обычно здесь используется алгоритм локального поиска, однако он может также появиться и после поиска всех решений.

Анализируются результаты экспериментов.

## 2.3.2 Практическая работа № 7 «Метод оптимизации на основе муравьиной кучи в решении задачи коммивояжера (TSP)»

### 2.3.2.1 Варианты заданий

Реализовать задачу коммивояжера (Traveling salesman problem, TSP). Варианты заданий представлены в таблице 2.2.

Таблица 2.2 – Варианты заданий

| Вариант | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| n | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |

**Постановка задачи.** Дан список городов и известны расстояния между каждыми двумя городами. Найти кратчайший путь, проходящий через все города и возвращающийся в исходную точку.

Реализовать генетический алгоритм решения задачи на ПЭВМ.

Провести эксперименты с параметрами алгоритма.

### 2.3.2.2 Методические указания

1 Разработайте алгоритм программы.

2 Разработайте программное приложение, реализующее алгоритм решения задачи коммивояжера методом оптимизации на основе муравьиной кучи.

3 Оформите отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- теоретическое обоснование;
- скриншоты программы;
- код программы;
- выводы.

### 2.3.2.3 Контрольные вопросы

1 Что лежит в основе метода оптимизации на основе муравьиной кучи?

2 Какие этапы включает алгоритм муравья?

3 Что понимается под феромоном?

4 Какой вид имеет математическая модель нанесения и испарения феромона?

5 Какой вид имеет алгоритм муравьиной системы для решения задачи коммивояжера?

6 Какой вид имеет алгоритм непрерывной оптимизации на основе муравьиной кучи?

7 Какие параметры включает муравьиная система?

8 Какие существуют модификации метода оптимизации на основе муравьиной кучи?

9 Какие преимущества имеет метод оптимизации на основе муравьиной кучи?

10 Какая область применения метода оптимизации на основе муравьиной кучи?

## 2.4 Алгоритм оптимизации на основе роя частиц

Алгоритм роя частиц был разработан в 1995 году Джеймсом Кеннеди и Расселом Еберхартом.

### 2.4.1 Этапы алгоритма оптимизации на основе роя частиц

Идея алгоритма была заимствована из исследований поведения скоплений животных (косяков рыб, стай птиц и т. п.), модель была упрощена, добавлены элементы поведения толпы людей.

Блок-схема алгоритма выглядит следующим образом.

**Этап 1.** Создание роя частиц.

**Этап 2.** Нахождение лучшего решения для каждой частицы.

**Этап 3.** Нахождение лучшего решения среди всех частиц.

**Этап 4.** Коррекция скорости каждой частицы.

**Этап 5.** Перемещение каждой частицы.

**Этап 6.** Проверка выполнения условия: критерий останова:

a) если условие истинно, то переход на 7 этап;

b) иначе переход на 2 этап.

**7 Этап.** Вывод результата работы алгоритма.

Пусть дано n-мерное пространство (пространство поиска), в котором двигаются частицы (агенты алгоритма). В начале частицы разбросаны случайным образом по всей области поиска и каждая частица имеет случайный вектор скорости. В каждой точке, где побывала частица, рассчитывается значение целевой функции. Каждая частица запоминает, какое (и где) лучшее значение целевой функции она лично нашла, а также каждая частица знает, где расположена точка, являющаяся лучшей среди всех точек, которые разведали частицы. На каждой итерации частицы корректируют свою скорость (модуль и направление), чтобы, с одной стороны, быть поближе к лучшей точке, которую частица нашла сама и, с другой стороны, приблизиться к точке, которая в данный момент является глобально лучшей. Через некоторое количество итераций частицы должны собраться вблизи наиболее хорошей точки, хотя возможно, что часть частиц останется где-то в относительно неплохом локальном экстремуме, но главное, чтобы хотя бы одна частица оказалась вблизи глобального экстремума.

Сходимость алгоритма зависит от коррекции скорости. В первоначальном виде алгоритма коррекция скорости выглядела следующим образом:

$$v_{i,t+1} = v_{\mathrm{i,t}} + \varphi_p r_p (p_i - x_{i,t}) + \varphi_g r_g (g_i - x_{i,t}), \qquad (2.5)$$

где $v_{i,t}$ – *i-я* компонента скорости при *t*-й итерации алгоритма;

$x_{i,t}$ – *i*-я координата частицы при *t*-й итерации алгоритма;

$p_i$ – *i*-я координата лучшего решения, найденного частицей;

$g_i$ – *i*-я координата лучшего решения, найденного всеми частицами;

$r_p, r_g$ – случайные числа в интервале (0, 1);

$\varphi_p, \varphi_g$ – весовые коэффициенты, которые надо подбирать под конкретную задачу.

Корректируем текущую координату каждой частицы по формуле

$$x_{i,t+1} = x_{i,t} + v_{i,t+1} \qquad (2.6)$$

После этого рассчитываем значение целевой функции в каждой новой точке, каждая частица проверяет, не стала ли новая координата лучшей среди всех точек, где она побывала. Затем среди всех новых точек проверяем, не нашли ли мы новую глобально лучшую точку, и, если нашли, запоминаем ее координаты и значение целевой функции в ней.

### 2.4.2 Модификации алгоритма

Формула для коррекции скорости частиц является сутью алгоритма, на подбор коэффициентов $\varphi_p$ и $\varphi_g$ были направлены многие исследования, также были предложены модифицированные алгоритмы роя частиц, коротко рассмотрим некоторые из них.

#### 2.4.2.1 Учет инерции

Одна из модификаций алгоритма состоит в том, чтобы добавить еще один весовой коэффициент $\omega(t)$ перед текущей скоростью (коэффициент инерции), благодаря которому скорость изменяется более плавно.

$$v_{i,t+1} = \omega(t)v_{i,t} + \varphi_p r_p(p_i - x_{i,t}) + \varphi_g r_g(g_i - x_{i,t}) \qquad (2.7)$$

Этот коэффициент может быть константой, а может зависеть от номера итерации $t$, например, линейно уменьшаться, начиная от небольшой величины, меньшей 1, и до какой-то другой величины, отличной от нуля.

#### 2.4.2.2 Канонический алгоритм

Один из самых распространенных вариантов алгоритма вводит нормировку коэффициентов $\varphi_p$ и $\varphi_g$, чтобы сходимость не так сильно зависела от их выбора.

$$v_{i,t+1} = k[v_{i,t} + \varphi_p r_p(p_i - x_{i,t}) + \varphi_g r_g(g_i - x_{i,t})], \qquad (2.8)$$

где $k$ – коэффициент, лежащий в интервале (0, 1);

$\varphi$ – сумма весовых коэффициентов $\varphi_p$ и $\varphi_g$, $\varphi = \varphi_p + \varphi_g$, $\varphi > 4$.

### 2.4.3 Преимущества и недостатки

Недостатки алгоритма:

1. Не гарантирована сходимость алгоритма при конечном числе частиц, отсюда и увеличение необходимого количества расчетов целевой функции.

2. Ориентация на одну лучшую точку, что увеличивает вероятность остановки алгоритма в неплохом, но не глобальном минимуме.

Преимущества алгоритма:

1. Понятность.

2. Возможность быстро перестроить алгоритм на другую формулу для расчета скорости частиц.

## 2.4.4 Практическая работа № 8
## «Метод оптимизации на основе роя частиц»

### 2.4.4.1 Варианты заданий

Реализовать на языке программирования задачу по вариантам заданий (таблица 2.3).

**Постановка задачи.** Минимизировать функцию $f(x)$.

Таблица 2.3 – Варианты заданий

| Номер варианта | Функция |
|---|---|
| 1 | Функция Экли |
| 2 | Функция Розенброка |
| 3 | Функция Швефеля абсолютной величины |
| 4 | Синусоидальная функция Швефеля |
| 5 | Функция синусоидальной огибающей |
| 6 | Функция Eggholder |
| 7 | Функция двойной суммы Швефеля |
| 8 | Десятистепенная функция |
| 9 | Функция Флетчера |
| 10 | Тестовая функция Экли |

### 2.4.4.2 Методические указания

1 Разработайте алгоритм программы.

2 Разработайте программное приложение, реализующее алгоритм решения задачи методом оптимизации на основе роя частиц.

3 Оформите отчет по лабораторной работе, включающий разделы:

- постановка задачи;
- теоретическое обоснование;
- скриншоты программы;
- код программы;
- выводы.

### 2.4.4.3 Контрольные вопросы

1 Что лежит в основе метода оптимизации на основе роя частиц?

2 Какие этапы включает базовый алгоритм оптимизации на основе роя частиц?

3 Что понимается под скоростью особи?

4 Какие факторы влияют на изменение скорости особи?

5 Какие регулировочные параметры используются в алгоритме оптимизации на основе роя частиц?

6 Какие существуют виды топологии роя частиц?

7 Какие существуют способы ограничения скорости особи?

8 Какие существуют модификации метода оптимизации на основе роя частиц?

9 Какие преимущества и недостатки имеет метод оптимизации на основе роя частиц?

10 Какая существует область применения метода оптимизации на основе роя частиц?

## ЗАКЛЮЧЕНИЕ

Развитие современных систем невозможно без эффективного управления, обеспечивающего переход из одного качественного состояния в другое. Процесс управления системой предусматривает выработку управленческих решений. Для выработки эффективных решений управления используют методы численной и эволюционной оптимизации.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Аттетков А. В. Методы оптимизации : учебник для вузов / А. В. Аттетков, С. В. Галкин, В. С. Зарубин ; под ред. В. С. Зарубина, А. П. Крищенко. – Москва : Изд-во МГТУ им. Н. Э. Баумана, 2001. – 440 с.

2 Пантелеев А. В. Методы оптимизации. Практический курс : учебное пособие / А. В. Пантелеев, Т. А. Летова. – Москва : Логос, 2017. – 424 с.

3 Пантелеев А. В. Летова Т. А. Методы оптимизации в примерах и задачах : учебное пособие / А. В. Пантелеев, Т. А. Летова. – 4-е изд., испр. – Санкт-Петербург : Издательство «Лань», 2015. – 512 с.

4 Таха Хемди. Введение в исследование операций / Хемди Таха. – Москва : Издательский дом «Вильямс», 2006. – 912 с.

5 Дэн Саймон. Алгоритмы эволюционной оптимизации / Саймон Дэн ; пер. с англ. А. В. Логунова. – Москва : ДМК Пресс, 2020. – 940 с.

6 Гладков А. А. Генетические алгоритмы / А. А. Гладков, В. В. Курейчик, В. М. Курейчик; Под ред. В. М. Курейчика. – Москва : Физматлит, 2016. – 368 с.

7 Вирсански Эйал. Генетические алгоритмы на Python / Эйал Вирсански. – Москва: ДМК Пресс, 2020. – 286 с.

8 Джонс М. Т. Программирование искусственного интеллекта в приложениях / М. Т. Джонс ; пер. с англ. А. И. Осипова. – Москва : ДМК Пресс, 2013. – 312 с.

Семахин Андрей Михайлович

# СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

Методические указания
к выполнению лабораторных работ
для магистрантов направления подготовки 09.04.04

Редактор В. С. Никифорова

Подписано в печать 15.03.2023 Формат 60×84 1/16 Бумага 80 г/м$^2$
Печать цифровая Усл. печ. л. 2,6 Уч.-изд. л. 2,6
Заказ 12 Тираж 25

Библиотечно-издательский центр КГУ.
640020, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.