Министерство образования и науки Российской Федерации

СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА М.Ф. РЕШЕТНЕВА

Кафедра системного анализа и исследования операций

УТВЕРЖДАЮ
Заведующий кафедрой
Системного анализа и
исследования операций

_____ И. В. Ковалев
подпись инициалы, фамилия
«____» _____ 20_ _г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

по направлению 553005 - Системный анализ данных и моделей принятия

САМОНАСТРАИВАЮЩИЙСЯ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧ УСЛОВНОЙ И БЕЗУСЛОВНОЙ ОПТИМИЗАЦИИ

(гр. МС)

Научный руководитель _____ / __Е.С. Семенкин_____/
подпись инициалы, фамилия

Магистрант _____ / __А.Б. Сергиенко_____/
подпись инициалы, фамилия

Красноярск 2010 г.

СОЖЕРЖАНИЕ:

Условные обозначения 3

Введение 4

Глава 1. Постановка задачи оптимизации 9

1. 1. Общая постановка задачи оптимизации 9

1. 2. Применимость алгоритмов бинарной оптимизации при решении задач вещественной оптимизации 12

1. 3. Набор тестовых задач оптимизации 20

1. 4. Критерии оценивания алгоритмов 33

1. 5. Выводы 36

Глава 2. Стандартный генетический алгоритм как точка отсчета для сравнения алгоритмов оптимизации 37

2. 1. Общая модель стандартного генетического алгоритма 37

2. 2. Описание стандартного генетического алгоритма на бинарных строках

2. 3. Описание операторов 46

2. 4. Исследование эффективности стандартного генетического алгоритма

2. 6. Выводы 92

Глава 3. Самонастраивающийся генетический алгоритм 92

3. 1. Задача оптимального генетического алгоритма 92

3. 2. Самонастраивающийся генетический алгоритм 94

3. 3. Исследования эффективности самонастраивающегося генетического алгоритма 102

3. 3. Система управления на основе коллектива нечетких правил 104

3. 3. Выводы 107

Список использованных источников: 112

Список публикаций автора 114

Условные обозначения

$a \in A$ – элемент a принадлежит множеству A;

$\bar{x}$ – обозначение вектора;

$\arg f(x)$ – возвращает аргумент x, при котором функция принимает значение $f(x)$;

$Random(X)$ – случайный выбор элемента из множества X с равной вероятностью;

$Random(\{x_i|p_i\})$ – случайный выбор элемента x_i из множества X, при условии, что каждый элемент $x_i \in X$ имеет вероятность выбора равную p_i;

$random(a, b)$ – случайное действительное число из интервала $[a; b]$;

$int(a)$ – целая часть действительного числа a;

$\mu(X)$ – мощность множества X;

$\mathbb{N}$ – множество натуральных чисел.

Замечание. Оператор присваивания обозначается через знак «=», так же как и знак равенства.

Замечание. Индексация всех массивов в начинается с 1. При реализации алгоритмов использовались C-подобные языки программирования, где индексация начинается с нуля.

Замечание. Вызывание трех функций: $Random(X)$, $Random(\{x_i|p_i\})$, $random(a, b)$ – происходит каждый раз, когда по ходу выполнения формул, они встречаются. Если формула итерационная, то нельзя перед ее вызовом один раз определить, например, $random(a, b)$ как константу и потом её использовать на протяжении всех итераций неизменной.

Введение

На данный момент генетический алгоритм (ГА) является одним из наиболее исследуемых и развивающихся алгоритмов глобальной оптимизации прямого поиска.

Но, к сожалению, существует несколько проблем, затрудняющих его применение в решение практических задач и затрудняющих сравнение с другими алгоритмами оптимизации. Во-первых, не существует единообразия в существующих описаниях данного алгоритма. Обычно даны лишь общие рекомендации, которые допускают разночтение в процессе программирования. Либо предложенные схемы применимы лишь для решения тестовых задач. Из-за этого невозможно строго сопоставить друг другу различные исследования по эффективности. Во-вторых, для генетических алгоритмов присуща проблема настройки параметров алгоритма. И настройка параметров сильно чувствительна к эффективности работы алгоритма. Как было выяснено, не существует таких настроек, которые бы были бы одинаково хорошими для всех задач, поэтому выбрать вектор параметров для конкретной задачи сложно. Одной из проблем является и то, что относительно выбора некоторых параметров алгоритма существуют рекомендации, которые при проведении исследований не оправдывают себя, что приводит к искажению информации об эффективности генетического алгоритма.

Поэтому задача создания самонастраивающегося алгоритма, который бы был лишен этих проблем, является актуальной. При этом требуется для проведения исследований, доказывающих эффективность самонастраивающегося алгоритма, провести подробный анализ стандартного генетического алгоритма, предварительно подробно описав его.

Цель работы: разработка и исследование самонастраивающегося генетического алгоритма вещественной и бинарной оптимизации по сравнению со стандартным генетическим алгоритмом.

Для достижения поставленной цели необходимо решить следующие **задачи:**

1. Провести анализ существующих описаний генетического алгоритма и описать вариант, соединяющий единые детали в понимании алгоритма.
2. Программно реализовать и провести анализ сравнительной эффективности стандартного генетического алгоритма на множестве тестовых задач.
3. Провести исследование стандартного генетического алгоритма опровержения или подтверждения рекомендаций некоторых параметров алгоритма.
4. Разработать и программно реализовать самонастраивающийся генетический алгоритм. Показать работоспособность предложенного алгоритма на тестовых и практических задачах, сравнить с стандартным генетическим алгоритмом.

Методы исследования. Для решения поставленных задач использовались методы системного анализа, теории вероятности, математической статистики, псевдобулевой оптимизации и эволюционных алгоритмов.

Научная новизна результатов диссертации:

1. Разработан новый метод решения сложных задач оптимизации – самонастраивающийся генетический алгоритм для задач вещественной и бинарной оптимизации.

2. Проведен сравнительный анализ эффективности самонастраивающегося генетического алгоритма и стандартного генетического алгоритма, и показано, что на большом классе задач самонастраивающийся алгоритм работает лучше, чем стандартный алгоритм с лучшими настройками.

3. Разработана новая система управления, основанная на коллективе нечетких правил, формирование которой является более простой задачей для генетического алгоритма.

Практическая значимость. Предложенный вероятностный генетический алгоритм использован при решении актуальной практической задачи – формирование системы управления на основе коллектива нечетких баз правил для управления автомобиля с прицепом при движении задним ходом. Данная задача решалась в Институте прикладных исследований при Высшей технической школе г. Ульм (Германия). Была показана применимость реализованных алгоритмов. На основе предложенных алгоритмов разработаны современные программные системы, которые позволяют успешно решать задачи оптимизации. Предложенные в диссертации алгоритмы и программные системы используются в учебном процессе при проведении занятий в Гимназии №11 города Красноярска.

В результате работы над диссертацией была сформирована документация, содержащая подробное описание стандартного генетического алгоритма с описанием всей необходимой информацией для проведения сравнительных исследований алгоритмов глобальной оптимизации по эффективности. Был разработан web сервисы: позволяющий в режиме on-line проводить сравнения алгоритмов глобальной оптимизации, и библиотека математических функция на языке C++, содержащая реализованные алгоритмы.

Основные защищаемые положения:

1. Не существует таких параметров стандартного генетического алгоритма, которые были бы лучшими настройками для всех задач оптимизации.

2. Предложенный самонастраивающийся алгоритм позволяет эффективно решать задачи глобальной оптимизации с уменьшением числа настраиваемых параметров алгоритма. При этом для эффективного решения задачи вещественно оптимизации по сравнению со стандартным требуется в 54 раза меньше вычислений целевой функции.

3. Размер турнира равный $T = 2$ или $T = 3$ не является оптимальным для генетического алгоритма. На большом множестве задач оптимальным размером турнира является величина равная половине размера популяции для стандартного генетического алгоритма.

Апробация работы. Основные положения и результаты диссертационной работы докладывались и обсуждались на следующих научно-практических конференциях:

- XXXIX Краевая Научная Студенческая конференция, г. Красноярск, 2006г.
- VII Всероссийская конференция молодых ученых по математическому моделированию и информационным технологиям, г. Красноярск, 2006г.
- X Всероссийская научная конференция с международным участием «Решетневские чтения», 2006.
- XL Краевая научная студенческая конференция, СФУ, г. Красноярск, 2007 г.
- «Актуальные проблемы авиации и космонавтики», СибГАУ, г. Красноярск, апрель 2007 г.

- VIII Всероссийская конференция молодых ученых по математическому моделированию и информационным технологиям, Новосибирск, 2007г.
- Решетнёвские чтения. Материалы XI Международной научной конференции, посвященной памяти генерального конструктора ракетно-космических систем академика М.Ф. Решетнёва, 2007г.
- «Актуальные проблемы авиации и космонавтики», СибГАУ, г. Красноярск, апрель 2008 г.
- «Актуальные проблемы авиации и космонавтики», СибГАУ, г. Красноярск, апрель 2010 г.

Программные системы использующие реализованные автором алгоритмы прошли экспертизу и имеют свидетельство о государственной регистрации программы для ЭВМ № 2009611195 и № 2009611196.

Публикации. По результатам диссертационной работы опубликовано 9 печатных научных работ, список которых приведен в конце диссертации.

Структура

Диссертационная работа состоит из введения, трех глав, заключения, списка литературы из 11 наименований и 1 приложения. Содержание работы изложено на 115 страницах основного текста, включая 13 таблиц и 21 рисунка.

Глава 1. Постановка задачи оптимизации

1. 1. Общая постановка задачи оптимизации

Рассмотрим постановку задачи оптимизации в общем случае, решаемую в данной работе.

Необходимо найти:

$$\bar{x}_{max} = \arg\max_{\bar{x} \in X} f(\bar{x}), \quad (1.1.1)$$

при условии

$g_i(\bar{x}) \leq 0, i = \overline{1, m_1}$;

$h_j(\bar{x}) = 0, j = \overline{1, m_2}$.

Здесь

X – множество всех возможных решений;

$f(\bar{x})$ – функционал, определенный на множестве X, возвращающий действительное число из интервала $(-\infty; \infty)$;

m_1 – число ограничений в виде неравенств;

m_2 – число ограничений в виде равенств.

$\bar{x} \in X$ имеет вид:

$$\bar{x} = (x_1; \dots; x_i; \dots; x_n)^{\mathrm{T}}, \quad (1.1.2)$$

В случае если $m_1 = 0$ и $m_2 = 0$, имеем задачу безусловной оптимизации. В противном случае – условной оптимизации.

В случае если X – множество всех бинарных векторов длины n таких что $x_i \in \{0; 1\}$ $(i = \overline{1, n})$, то имеем задачу бинарной оптимизации (мощность поискового пространства X равна $\mu(X) = 2^n$). Если X – множество всех вещественных векторов длины n таких что $x_i \in \{Left_i; Right_i\}$ $(i = \overline{1, n})$, то имеем задачу вещественной оптимизации.

Будем предполагать в дальнейшем, что $f(\bar{x})$ может представлять собой многоэкстремальный функционал, и вычислению подлежат только значения $f(\bar{x})$ (нет возможности вычислить производные от функционала и т.д.).

В данной работе рассматриваются только задачи вещественной и бинарной оптимизации. При решении задачи на вещественных строках

происходит преобразования исходной задачи к задаче оптимизации на бинарной строках, на основе стандартного представление целого числа – номера узла в сетке дискретизации или стандартным рефлексивным Грей-кодом.

Пример. Требуется найти бинарный вектор длины $n = 8$, сумма компонент которого максимальна.

$$\bar{x}_{max} = \arg\max_{\bar{x} \in X} f(\bar{x}), \tag{1.1.3}$$

где $f(\bar{x}) = \sum_{i=1}^{n} x_i$.

Очевидно, что $\bar{x}_{max} = (1; 1; 1; 1; 1; 1; 1; 1)^{\mathrm{T}}$, но алгоритм этого «не знает», и его задача – найти это решение.

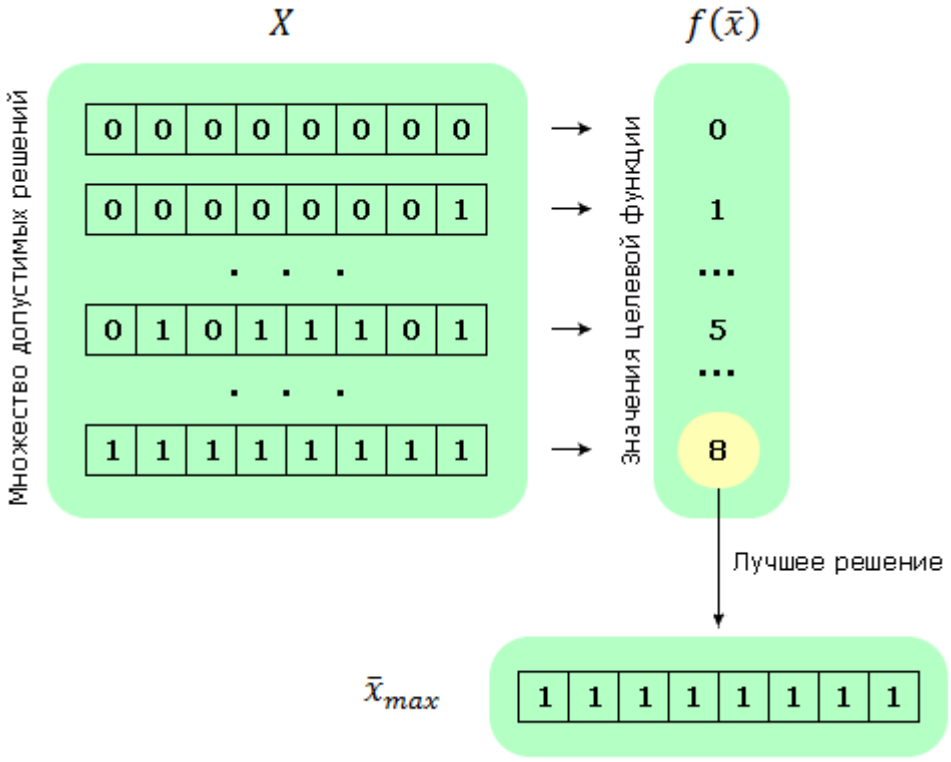

Рисунок 1.1.1. Пример задачи оптимизации

Примечание. Если решается задача минимизации, а не максимизации функционала $f(\bar{x})$, то в алгоритм оптимизации подается преобраованный функционал:

$$f^*(\bar{x}) = -f(\bar{x}), \tag{1.1.4}$$

так как

$$\bar{x}_{min} = \arg\min_{\bar{x}\in X} f(\bar{x}) = \arg\max_{\bar{x}\in X} \left(-f(\bar{x})\right). \quad (1.1.5)$$

Введем некоторые понятия, которыми будем пользоваться в данной работе.

Стандартный генетический алгоритм – простейший генетический алгоритм, описанный в [8]. Он считается базовым для исследования эффективности других ГА. Если не оговорено иное, то под генетическим алгоритмом понимается стандартный генетический алгоритм.

Бинарная строка – вектор-столбец $\bar{x}$ конечной длины n, компоненты которого могут принимать значения из множества $\{0; 1\}$. В принципе не совсем верное название, так как по сути это вектор-столбец, а не вектор-строка.

Бинарный вектор – то же самое, что и бинарная строка.

Вещественная строка – вектор-столбец $\bar{x}$ конечной длины n, компоненты которого могут принимать значения из множества $\{Left_i; Right_i\}$ $(i = \overline{1,n})$. По своей сути не совсем верное название, так как по сути это вектор-столбец, а не вектор-строка.

Вещественный вектор – то же самое, что и вещественная строка.

Решение – любой элемент $\bar{x}$ из множества X.

Точка поискового пространства – то же самое, что и решение.

Возможное решение – любое решение $\bar{x}$ из множества X.

Допустимое решение – решение $\bar{x}$ из множества X, которое удовлетворяет условиям $g_i(\bar{x})$ $(i = \overline{1,m_1})$ и $h_j(\bar{x})$ $(j = \overline{1,m_2})$.

Генотип – то же самое, что и бинарная строка.

Индивид – то же самое, что и бинарная строка.

Хромосома – то же самое, что и бинарная строка.

Ген – компонент x_i бинарного вектора $\bar{x} \in X$.

Бит – компонент x_i бинарного вектора $\bar{x} \in X$.

Фенотип – некий объект, который был закодирован в бинарную, строку при переходе от задачи оптимизации на произвольном пространстве

поиска к задаче оптимизации на бинарных строках. Например, вещественный вектор преобразуется в бинарную строку с помощью кода Грея.

Поколение – одна итерация работы сГА.

Популяция – множество индивидов, обрабатываемых на текущем поколении.

Целевая функция – то же самое, что и функционал $f(\bar{x})$.

Функция пригодности – преобразованная целевая функция для использования в генетическом алгоритме на бинарных строках.

Пригодность – значение функции пригодности.

Поисковое пространство – множество всех возможных векторов X.

Множество решений – то же самое что и поисковое пространство.

Остальные определения будут вводиться по мере их упоминания в тексте.

Замечание. Понятия «генотип», «хромосома», «индивид» обозначают по своей сути одно и то же. Но в связи с тем, что генетический алгоритм имитирует настоящую эволюцию, то на различных этапах имитации более подходящими являются разные понятия из биологии.

1. 2. Применимость алгоритмов бинарной оптимизации при решении задач вещественной оптимизации

Рассматриваемые в работе алгоритмы обычно работают на бинарных строках. Поэтому встает проблема преобразования задачи вещественной оптимизации в бинарную, которая уже и передается алгоритму оптимизации на решение.

Итак, в случае, когда решается задача вещественной оптимизации, добавляются операторы преобразования задачи вещественной оптимизации в задачу бинарной оптимизации ($ConvertingIntoBinary$) и преобразования бинарного решения в вещественное ($BinaryToReal$). При этом к вектору параметров алгоритма оптимизации $Parameters$ добавляется блок $ParametersOfConvertingIntoBinary$.

Существует различные реализации данных операторов. В данной работе рассматривается два из них:

- Стандартное представление целого числа – номер узла в сетке дискретизации;
- Стандартный рефлексивный Грей-код.

Применяемый тип определяется параметров $TypOfConverting$ из $ParametersOfConvertingIntoBinary$.

Итак, имеем:

- пространство поиска X, где $\bar{x} = (x_1; \dots; x_i; \dots; x_n)^{\mathrm{T}}$, $x_i \in \{Left_i; Right_i\}$, $i = \overline{1,n}$;
- ограничения задачи оптимизации $g_i(\bar{x}) \leq 0$ $(i = \overline{1,m_1})$, $h_j(\bar{x}) = 0$ $(j = \overline{1,m_2})$;
- целевую функцию $f(\bar{x})$;

Необходимо произвести с этими данными преобразования.

Стандартное представление целого числа – номер узла в сетке дискретизации

$$TypOfConverting = IntConverting. \quad (1.2.1)$$

Классическая схема, в которой интервал изменения каждой координаты вектора $\bar{x}$ дискретизируется на определенное число интервалов. Каждой точке на оси ставится в соответствие целое неотрицательное число – номер узла в сетке дискретизации (начиная с нуля). Данное целое число кодируется в бинарную строку, длина которой равна наименьшей степени двойки $Length_i$ при которой число $2^{Length_i} \geq NumberOfParts_i + 1$. Этот процесс осуществляется для всех компонент вектора $\bar{x}$. Итоговая бинарная строка, участвующая в работе алгоритма оптимизации на бинарных строках, получается путем склейки бинарных строк всех компонент вектора.

Распишем алгоритм преобразования задачи:

$$ConvertingIntoBinary\begin{pmatrix} X \\ f(\bar{x}) \\ g_i(\bar{x}) \\ h_j(\bar{x}) \\ ParametersOfConvertingIntoBinary \end{pmatrix}$$

• **Начало алгоритма**.

• **Рассчитаем необходимую длину каждой бинарной строки для каждой компоненты вектора** $\bar{x} \in X$:

$Length_i =$ (1.2.2)

• **Начало**

• $S = 1; L = 0;$

• Повторять до тех пор пока $S < NumberOfParts_i + 1$

• **Начало**

$S = S \cdot 2;$

$L = L + 1;$

• **Конец**

• **Возвращаем** L;

• **Конец**

• **Определим множество** X_B**, как множество бинарных векторов** $\bar{x}_B$ **длины** $n_B = \sum_{i=1}^{n} Length_i$.

• **Рассчитаем шаг дискретизации для каждой компоненты вектора** $\bar{x} \in X$:

$h_i = \frac{|Left_i - Right_i|}{2^{Lengt\ h_i}}$;

• **Получим целевую функцию задачи бинарной оптимизации**:

$f_B(\bar{x}_B) = f(\bar{x})$,

где $\bar{x}_i = h_i \cdot \sum_{j=1}^{Lengt\ h_i}\left(2^{j-1} \cdot (\bar{x}_B)_{Begin_i + j}\right) + Left_i$;

$Begin_i = \sum_{j=1}^{i-1} Length_i$, $(i = \overline{1,n})$.

• **Аналогично получим ограничения для задачи бинарной оптимизации:**

Ограничения-неравенства:

$$g_{j\,B}(\bar{x}_B) = g_j(\bar{x}), \quad (1.2.3)$$

где $\bar{x}_i = h_i \cdot \sum_{j=1}^{Lengt\,h_i} \left(2^{j-1} \cdot (\bar{x}_B)_{Begin\,_i+j}\right) + Left_i$;

$Begin_i = \sum_{j=1}^{i-1} Length_i$, $j = \overline{1, m_1}$, $(i = \overline{1, n})$.

Ограничения-равенства:

$$h_{j\,B}(\bar{x}_B) = h_i(\bar{x}), \quad (1.2.4)$$

где $\bar{x}_i = h_i \cdot \sum_{j=1}^{Lengt\,h_i} \left(2^{j-1} \cdot (\bar{x}_B)_{Begin\,_i+j}\right) + Left_i$;

$Begin_i = \sum_{j=1}^{i-1} Length_i$, $j = \overline{1, m_1}$, $(i = \overline{1, n})$.

• **Возвратим вектор** $\begin{pmatrix} X_B \\ f_B(\bar{x}_B) \\ g_{i\,B}(\bar{x}_B) \\ h_{j\,B}(\bar{x}_B) \end{pmatrix}$.

• **Конец алгоритма**.

Здесь $NumberOfParts_i$ $(i = \overline{1, n})$ – число интервалов, на которые предполагается разбивать каждую компоненту вектора $\bar{x}$ в пределах изменения. В большинстве своем настоящее число интервалов будет получаться больше, чем запланировано.

$Begin_i$ – номер элемента (гена) бинарной строки $\bar{x}_B$, с которого начинают идти гены, кодирующую i компоненту вектора $\bar{x}$ $(i = \overline{1, n})$.

$NumberOfParts_i \in ParametersOfConvertingIntoBinaryGA$ $(i = \overline{1, n})$.

Аналогично приведенному методу преобразования бинарного решения в действительное получим формулу для оператора $BinaryToReal$:

$$BinaryToReal \begin{pmatrix} \bar{x}_{subm\,ax\,B} \\ f(\bar{x}_{subm\,ax})_B \\ ParametersOfConvertingIntoBinary \end{pmatrix} = \begin{pmatrix} \bar{x}_{subm\,ax} \\ f(\bar{x}_{subm\,ax}) \end{pmatrix}, \quad (1.2.5)$$

где

$$(\bar{x}_{subm\,ax})_i = h_i \cdot \sum_{j=1}^{Lengt\,h_i} \left(2^{j-1} \cdot \left(\bar{x}_{subm\,ax\,B}\right)_{Begin\,_i+j}\right) + Left_i;$$

$(i = \overline{1,n})$, $Begin_i = \sum_{j=1}^{i-1} Length_i$.

Замечание. При программировании оператора $BinaryToReal$ не вычисляется заново значение целевой функции – используется то значение, которое вычислялось при работе алгоритма на бинарных строках.

Замечание. При реализации алгоритмов оптимизации операция кодирования вещественного числа в бинарную строку не требуется в отличие от операции декодирования.

Замечание. Существует схема, когда итоговая бинарная строка, участвующая в алгоритмах оптимизации на бинарных строках, получается не путем склейки бинарных строк всех компонент вещественного вектора $\bar{x}$, а путем покомпонентной склейки: вначале идут первые биты всех компонент вектора $\bar{x}$, потом вторых и т.д. В данной работе она не рассматривается.

Замечание. Мы используем число $NumberOfParts_i + 1$ для определения длины бинарной строки для кодирования каждой координаты, а не $NumberOfParts_i$ $(i = \overline{1,n})$, так как число точек больше интервалов, на которые мы разбиваем диапазон изменения вещественной координаты, на 1.

Стандартный рефлексивный Грей-код

$TypOfConverting = GrayCodeConverting.$ (1.2.6)

Также, как в предыдущей схеме, интервал изменения каждой координаты вектора $\bar{x}$ дискретизируется на определенное число интервалов. Каждой точке на оси ставится в соответствие целое неотрицательное число – номер узла в сетке дискретизации (начиная с нуля). Данное целое число кодируется в бинарную строку, длина которой равна наименьшей степени двойки $Length_i$ при которой число $2^{Length_i} \geq NumberOfParts_i + 1$. Этот процесс осуществляется для всех компонент вектора $\bar{x}$. Итоговая бинарная строка, участвующая в алгоритме оптимизации на бинарных строках, получается путем склейки бинарных строк всех компонент вектора.

Единственное отличие существует на этапе кодирования целого числа в бинарную строку. Бинарная строка не представляет собой двоичный код целого числа, а представляет код Грея. Его отличительной особенностью является то, что если два целых числа отличаются на единицу, то их коды Грея также будут отличаться только одним битом [4]. Двоичный код не обладает данным свойством.

Существует метод по переводу кода Грея в двоичный код: старший разряд (крайний левый бит) записывается без изменения, каждый следующий символ кода Грея нужно инвертировать, если в двоичном коде перед этим была получена «1», и оставить без изменения, если в двоичном коде был получен «0». Например [5], дан код Грея $a = 1101$ целого числа $x = 9$. Двоичный код после преобразований: $a' = 1001$, что является двоичным представлением числа 9.

Благодаря этому методу мы можем операторы $ConvertingIntoBinary$ и $BinaryToReal$ почти не изменять, а только добавить операцию по переводу бинарной строки из кода Грея в двоичный код.

Распишем алгоритм преобразования задачи вещественной оптимизации:

$$ConvertingIntoBinary\begin{pmatrix} X \\ f(\bar{x}) \\ g_i(\bar{x}) \\ h_j(\bar{x}) \\ ParametersOfConvertingIntoBinaryGA \end{pmatrix}$$

- **Начало алгоритма.**
- **Рассчитаем необходимую длину каждой бинарной строки для каждой компоненты вектора** $\bar{x} \in X$:

$Length_i =$ (1.2.7)

- **Начало**
- $S = 1; L = 0;$
- Повторять до тех пор пока $S < NumberOfParts_i + 1$
  - **Начало**

$S = S \cdot 2;$

$L = L + 1;$

- **Конец**

- **Возвращаем** L;

- **Конец**

- **Определим множество X_B, как множество бинарных векторов $\bar{x}_B$ длины $n_B = \sum_{i=1}^{n} Length_i$.**

- **Рассчитаем шаг дискретизации для каждой компоненты вектора** $\bar{x} \in X$:

$h_i = \frac{|Left_i - Right_i|}{2^{Length\ h_i}}$;

- **Получим целевую функцию задачи бинарной оптимизации**:

$f_B(\bar{x}_B) = f(\bar{x})$,

где $\bar{x}_i = h_i \cdot \sum_{j=1}^{Length\ h_i} \left(2^{j-1} \cdot (\bar{x}_{BG})_{Begin_i + j}\right) + Left_i$;

$Begin_i = \sum_{j=1}^{i-1} Length_i$,

$$(\bar{x}_{BG})_{Begin_i + j} = \begin{cases} (\bar{x}_B)_{Begin_i + j}, \text{если } j = 1; \\ (\bar{x}_B)_{Begin_i + j}, \text{если } j \neq 1 \text{и } (\bar{x}_{BG})_{Begin_i + j - 1} = 0; \\ 1 - (\bar{x}_B)_{Begin_i + j}, \text{если } j \neq 1 \text{и } (\bar{x}_{BG})_{Begin_i + j - 1} = 1. \end{cases}$$

$(i = \overline{1, n})$.

- **Аналогично получим ограничения для задачи бинарной оптимизации:**

Ограничения-неравенства:

$g_{j\,B}(\bar{x}_B) = g_j(\bar{x})$, (1.2.8)

где $\bar{x}_i = h_i \cdot \sum_{j=1}^{Length\ h_i} \left(2^{j-1} \cdot (\bar{x}_{BG})_{Begin_i + j}\right) + Left_i$;

$Begin_i = \sum_{j=1}^{i-1} Length_i, j = \overline{1, m_1}$,

$$(\bar{x}_{BG})_{Begin_i + j} = \begin{cases} (\bar{x}_B)_{Begin_i + j}, \text{если } j = 1; \\ (\bar{x}_B)_{Begin_i + j}, \text{если } j \neq 1 \text{и } (\bar{x}_{BG})_{Begin_i + j - 1} = 0; \\ 1 - (\bar{x}_B)_{Begin_i + j}, \text{если } j \neq 1 \text{и } (\bar{x}_{BG})_{Begin_i + j - 1} = 1. \end{cases}$$

$(i = \overline{1, n})$.

Ограничения-равенства:

$$h_{j\,B}(\bar{x}_B) = h_i(\bar{x}), \qquad (1.2.9)$$

где $\bar{x}_i = h_i \cdot \sum_{j=1}^{Lengt\,h_i}\left(2^{j-1} \cdot (\bar{x}_{BG})_{Begin\,_i+j}\right) + Left_i;$

$Begin_i = \sum_{j=1}^{i-1} Length_i, j = \overline{1, m_1},$

$$(\bar{x}_{BG})_{Begin\,_i+j} = \begin{cases} (\bar{x}_B)_{Begin\,_i+j}, \text{если } j = 1; \\ (\bar{x}_B)_{Begin\,_i+j}, \text{если } j \neq 1 \text{и } (\bar{x}_{BG})_{Begin\,_i+j-1} = 0; \\ 1 - (\bar{x}_B)_{Begin\,_i+j}, \text{если } j \neq 1 \text{и } (\bar{x}_{BG})_{Begin\,_i+j-1} = 1. \end{cases}$$

$(i = \overline{1, n})$.

• **Возвратим вектор** $\begin{pmatrix} X_B \\ f_B(\bar{x}_B) \\ g_{i\,B}(\bar{x}_B) \\ h_{j\,B}(\bar{x}_B) \end{pmatrix}$.

• **Конец алгоритма**.

Здесь $NumberOfParts_i$ $(i = \overline{1, n})$ – число интервалов, на которые предполагается разбивать каждую компоненту вектора $\bar{x}$ в пределах своего изменения. В большинстве своем настоящее число интервалов будет получаться больше.

$Begin_i$ – номер элемента (гена) бинарной строки $\bar{x}_B$, с которого начинают идти гены, кодирующую i компоненту вектора $\bar{x}$ $(i = \overline{1, n})$.

$NumberOfParts_i \in ParametersOfConvertingIntoBinaryGA$ $(i = \overline{1, n})$.

Аналогично приведенному методу преобразования бинарного решения в действительное получим формулу для оператора $BinaryToReal$:

$$BinaryToReal\begin{pmatrix} \bar{x}_{subm\,ax\,B} \\ f(\bar{x}_{subm\,ax})_B \\ ParametersOfConvertingIntoBinaryGA \end{pmatrix} = \begin{pmatrix} \bar{x}_{subm\,ax} \\ f(\bar{x}_{subm\,ax}) \end{pmatrix}, \qquad (1.2.10)$$

где $(\bar{x}_{subm\,ax})_i = h_i \cdot \sum_{j=1}^{Lengt\,h_i}\left(2^{j-1} \cdot \left(\bar{x}_{subm\,ax\,BG}\right)_{Begin\,_i+j}\right) + Left_i;$

$Begin_i = \sum_{j=1}^{i-1} Length_i,$

$$(\bar{x}_{BG})_{Begin_i+j} = \begin{cases} (\bar{x}_B)_{Begin_i+j}, \text{если } j = 1; \\ (\bar{x}_B)_{Begin_i+j}, \text{если } j \neq 1 \text{и } (\bar{x}_{BG})_{Begin_i+j-1} = 0; \\ 1 - (\bar{x}_B)_{Begin_i+j}, \text{если } j \neq 1 \text{и } (\bar{x}_{BG})_{Begin_i+j-1} = 1. \end{cases}$$

$(i = \overline{1,n})$.

1. 3. Набор тестовых задач оптимизации

К сожалению, на данный момент не существует способа теоретического сравнения различных алгоритмов глобальной оптимизации. Во многом это объясняется тем, что они строятся на основе эвристических предположений, которые можно запрограммировать, но построить удобную математическую модель для проверки сходимости и др. крайне сложно.

Поэтому приходится использовать экспериментальную проверку эффективности алгоритмов на конкретных задачах оптимизациях. Для этого используется специальный набор задач оптимизации (тестовые функции), для которых известны точные решения и определены способы вычисления ошибки работы алгоритмов оптимизации.

Введем два понятия, которые будем использовать при описании тестовых задач.

Основная (главная) задача оптимизации – это основная форма тестовой задачи оптимизации.

Подзадача – это производная форма от главной задачи оптимизации. Может отличаться от главной размерностью, коэффициентами и др.

Тестовые задачи безусловной оптимизации

Критерием классификации тестовых задач является тип поискового пространства задачи.

Тестовые задачи на бинарных строках

Сумма всех элементов бинарного вектора.

Обозначение: MHL_TestFuction_SumVector

Наименование: Сумма всех элементов бинарного вектора.

Тип: Задача бинарной оптимизации.

Формула: (целевая функция)

$$f(\bar{x}) = \sum_{i=1}^{n} \bar{x}_i,$$

где $\bar{x} \in X$, $x_j \in \{0; 1\}$ $(j = \overline{1,n})$.

Обозначения: $\bar{x}$ – бинарный вектор;

n – размерность бинарного вектора.

Объем поискового пространства: $\mu(X) = 2^n$.

Решаемая задача оптимизации: $\bar{x}_{max} = \arg\max_{\bar{x} \in X} f(\bar{x})$.

Точка максимума: $\bar{x}_{max} = (1, 1, 1 \dots, 1)^T$,то есть $(\bar{x}_{max})_j = 1$ $(j = \overline{1,n})$.

Максимум функции: $f(\bar{x}_{max}) = n$.

Точка минимума: $\bar{x}_{min} = (0, 0, 0 \dots, 0)^T$,то есть $(\bar{x}_{min})_j = 0$ $(j = \overline{1,n})$.

Минимум функции: $f(\bar{x}_{min}) = 0$.

Основная задача и подзадачи.

Изменяемый параметр: n – размерность бинарного вектора.

Значение в основной задаче: $n = 20$.

Подзадача №2: $n = 30$.

Подзадача №3: $n = 40$.

Подзадача №4: $n = 50$.

Подзадача №5: $n = 60$.

Подзадача №6: $n = 70$.

Подзадача №7: $n = 80$.

Подзадача №8: $n = 90$.

Подзадача №9: $n = 100$.

Подзадача №10: $n = 200.$

Нахождение ошибки оптимизации:

Пусть в результате работы алгоритма оптимизации за N запусков мы нашли решения $(\bar{x}_{subm\,ax})_k$ с значениями целевой функции $\left(f(\bar{x}_{subm\,ax})\right)_k$ соответственно ($k = \overline{1,N}$). Используем три вида ошибок:

Надёжность:

$$R = \frac{\sum_{k=1}^{N} S(\,(\bar{x}_{subm\,ax})_k)}{N},$$

где $S(\,(\bar{x}_{subm\,ax})_k) = \begin{cases} 1, & (\bar{x}_{subm\,ax})_k = \bar{x}_{max}; \\ 0, & (\bar{x}_{subm\,ax})_k \neq \bar{x}_{max}. \end{cases}$

Ошибка по входным параметрам:

$$E_x = \frac{\sum_{k=1}^{N} \sum_{i=0}^{n} |(\bar{x}_{max})_i - ((\bar{x}_{subm\,ax})_i)_k|}{N}.$$

Ошибка по значениям целевой функции:

$$E_y = \sum_{k=1}^{N} \left| f(\bar{x}_{max}) - \left(f(\bar{x}_{subm\,ax})\right)_k \right|.$$

Свойства задачи.

Условной или безусловной оптимизации:	Задача безусловной оптимизации.
Одномерной или многомерной оптимизации:	Многомерной: n.
Функция унимодальная или многоэкстремальная:	Функция унимодальная.
Функция стохастическая или нет:	Функция не стохастическая.
Особенности:	Нет.

Тестовые задачи на вещественных векторах

Эллиптический параболоид.

Обозначение: MHL_TestFuction_ParaboloidOfRevolution

Тип: Задача вещественной оптимизации.

Формула:

(целевая функция)

$$f(\bar{x}) = \sum_{i=1}^{n-1} \bar{x}_i{}^2,$$

где $\bar{x} \in X$, $\bar{x}_j \in [Left_j; Right_j]$,

$Left_j = -2; Right_j = 2, (j = \overline{1,n})$.

Обозначения: $\bar{x}$ – вещественный вектор;

n – размерность вещественного вектора.

Решаемая задача оптимизации: $\bar{x}_{min} = \arg\min_{\bar{x} \in X} f(\bar{x})$.

Точка минимума: $\bar{x}_{min} = (0, 0, 0 \dots, 0)^T$, то есть $(\bar{x}_{min})_j = 0$ $(j = \overline{1,n})$.

Минимум функции: $f(\bar{x}_{min}) = 0$.

График.

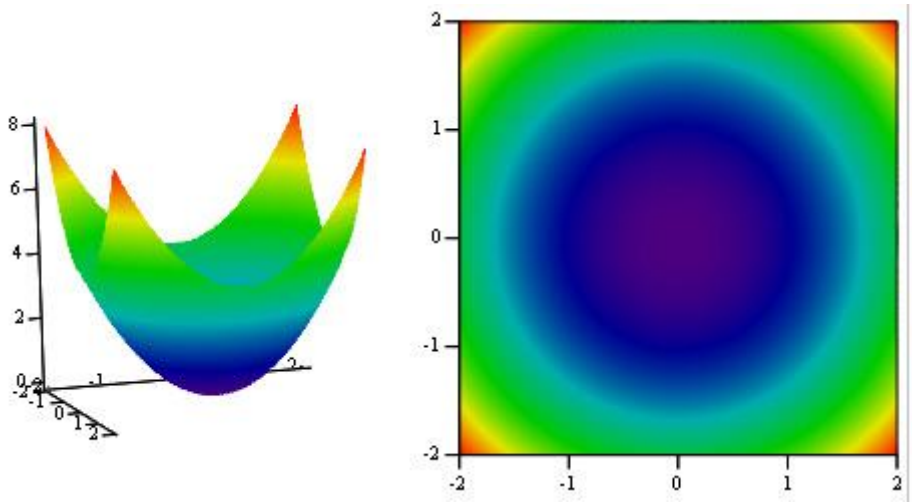

Рисунок 1.5.1.2.1. График эллиптический параболоида при $n = 2$.

Параметры для алгоритмов оптимизации.

Точность вычислений: $\varepsilon = 0.01$.

$NumberOfParts_j = 4095$ $(j = \overline{1,n})$.

Примечание:

$NumberOfParts_j$ выбирается как минимальное число, удовлетворяющее соотношению:

$NumberOfParts_j = 2^{(k_2)_j} - 1 \geq \frac{(Right_j - Left_j)10}{\varepsilon}$, $(k_2)_j \in \mathbb{N}$, $j = \overline{1,n}$.

$NumberOfParts_j$ используется в стандартном генетическом алгоритме для перевода задачи вещественной оптимизации к задаче бинарной оптимизации.

Основная задача и подзадачи.

Изменяемый параметр: n – размерность вещественного вектора.

Значение в основной задаче: $n = 2$.

Подзадача №2: $n = 3$.

Подзадача №3: $n = 4$.

Нахождение ошибки оптимизации:

Пусть в результате работы алгоритма оптимизации за N запусков мы нашли решения $(\bar{x}_{subm\,ax})_k$ со значениями целевой функции $\left(f(\bar{x}_{subm\,ax})\right)_k$ соответственно $(k = \overline{1,N})$. Используем три вида ошибок:

Надёжность: $R = \frac{\sum_{k=1}^{N} S((\bar{x}_{subm\ ax})_k)}{N}$, где

$$S((\bar{x}_{subm\,ax})_k) = \begin{cases} 1, & |((\bar{x}_{subm\,ax})_i)_k - (\bar{x}_{max})_i| \leq \varepsilon\ (i = \overline{1,n}); \\ 0, & \textit{иначе.} \end{cases}$$

Ошибка по входным параметрам:

$$E_x = \frac{\sum_{k=1}^{N} \left(\frac{\sqrt{\sum_{i=1}^{n} ((\bar{x}_{max})_i - ((\bar{x}_{subm\,ax})_i)_k)^2}}{n} \right)}{N}.$$

Ошибка по значениям целевой функции: $E_y = \frac{\sum_{k=1}^{N} \left| f(\bar{x}_{max}) - \left(f(\bar{x}_{subm\ ax})\right)_k \right|}{N}$.

Свойства задачи.

Условной или безусловной оптимизации:	Задача безусловной оптимизации.
Одномерной или многомерной оптимизации:	Многомерной: n.
Функция унимодальная или многоэкстремальная:	Функция унимодальная.
Функция стохастическая или нет:	Функция не стохастическая.

<u>Функция Ackley</u>.

Обозначение:	MHL_TestFuction_Ackley
Тип:	Задача вещественной оптимизации.
Формула: (целевая функция)	$f(\bar{x}) = 20 + e + 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{x}_i{}^2)}} - e^{\frac{1}{n}\sqrt{\sum_{i=1}^{n}\cos(2\pi\cdot\bar{x}_i)}}$, где $\bar{x} \in X$, $\bar{x}_j \in [Left_j; Right_j]$, $Left_j = -5$; $Right_j = 5$, $(j = \overline{1,n})$.
Обозначения:	$\bar{x}$ – вещественный вектор; n – размерность вещественного вектора.
Решаемая задача оптимизации:	$\bar{x}_{min} = \arg\min_{\bar{x}\in X} f(\bar{x})$.
Точка минимума:	$\bar{x}_{min} = (0, 0, 0 \dots, 0)^T$, то есть $(\bar{x}_{min})_j = 0$ $(j = \overline{1,n})$.
Минимум функции:	$f(\bar{x}_{min}) = 0$.

График.

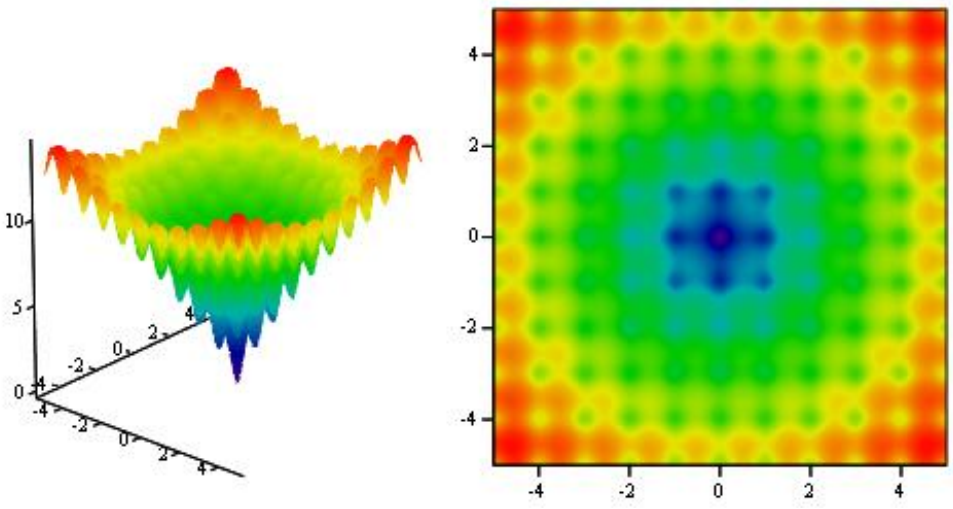

Рисунок 1.5.1.2.2. График функции Ackley при $n = 2$.

Параметры для алгоритмов оптимизации.

Точность вычислений: $\varepsilon = 0.025$.

$NumberOfParts_j = 4095 (j = \overline{1,n})$.

Основная задача и подзадачи.

Изменяемый параметр: n – размерность вещественного вектора.

Значение в основной задаче: $n = 2$.

Подзадача №2: $n = 3$.

Подзадача №3: $n = 4$.

Нахождение ошибки оптимизации:

Пусть в результате работы алгоритма оптимизации за N запусков мы нашли решения $(\bar{x}_{subm\ ax})_k$ со значениями целевой функции $\left(f(\bar{x}_{subm\ ax})\right)_k$ соответственно $(k = \overline{1,N})$. Используем три вида ошибок:

Надёжность: $R = \frac{\sum_{k=1}^{N} S(\ (\bar{x}_{subm\ ax})_k)}{N}$, где

$$S(\ (\bar{x}_{subm\ ax})_k) = \begin{cases} 1, |((\bar{x}_{subm\ ax})_i)_k - (\bar{x}_{max})_i| \le \varepsilon\ (i = \overline{1,n}); \\ 0, \quad \textit{иначе.} \end{cases}$$

Ошибка по входным параметрам: $$E_x = \frac{\sum_{k=1}^{N}\left(\frac{\sqrt{\sum_{i=1}^{n}((\bar{x}_{max})_i - ((\bar{x}_{subm\,ax})_i)_k)^2}}{n}\right)}{N}.$$

Ошибка по значениям целевой функции: $$E_y = \frac{\sum_{k=1}^{N}\left|f(\bar{x}_{max}) - \left(f(\bar{x}_{subm\,ax})\right)_k\right|}{N}.$$

Свойства задачи.

Условной или безусловной оптимизации:	Задача безусловной оптимизации.
Одномерной или многомерной оптимизации:	Многомерной: n.
Функция унимодальная или многоэкстремальная:	Функция многоэкстремальная.
Функция стохастическая или нет:	Функция не стохастическая.

<u>Функция Растригина</u>.

Обозначение:	MHL_TestFuction_Rastrigin
Тип:	Задача вещественной оптимизации.
Формула: (целевая функция)	$f(\bar{x}) = 10n + \sum_{i=1}^{n}\left(\bar{x}_i{}^2 - 10 \cdot \cos(2\pi \cdot \bar{x}_i)\right)$, где $\bar{x} \in X$, $\bar{x}_j \in \left[Left_j; Right_j\right]$, $Left_j = -5$; $Right_j = 5$, $(j = \overline{1,n})$.
Обозначения:	$\bar{x}$ – вещественный вектор; n – размерность вещественного вектора.
Решаемая задача оптимизации:	$\bar{x}_{min} = \arg\min_{\bar{x}\in X} f(\bar{x})$.
Точка минимума:	$\bar{x}_{min} = (0, 0, 0 \ldots, 0)^T$, то есть $(\bar{x}_{min})_j = 0$ $(j = \overline{1,n})$.

Минимум функции: $f(\bar{x}_{min}) = 0.$

График.

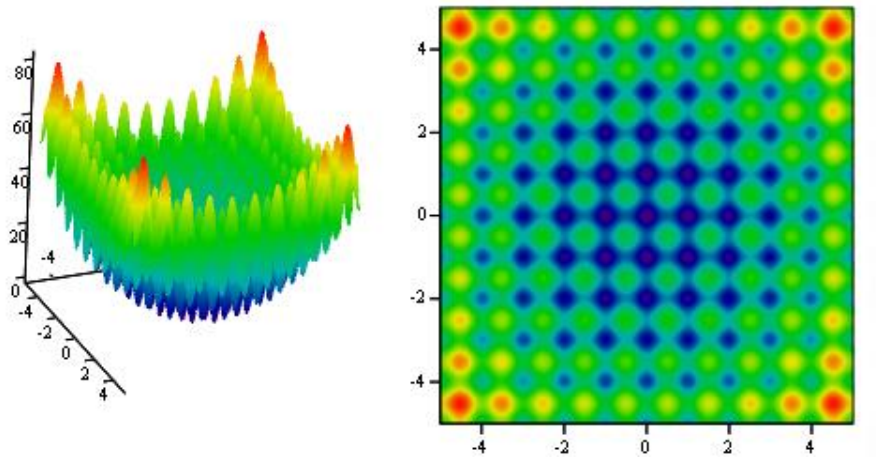

Рисунок 1.5.1.2.3. График функции Растригина при $n = 2$.

Параметры для алгоритмов оптимизации.

Точность вычислений: $\varepsilon = 0.025.$

$NumberOfParts_j = 4095$ $(j = \overline{1,n})$.

Основная задача и подзадачи.

Изменяемый параметр: n – размерность вещественного вектора.

Значение в основной задаче: $n = 2.$

Подзадача №2: $n = 3.$

Подзадача №3: $n = 4.$

Нахождение ошибки оптимизации:

Пусть в результате работы алгоритма оптимизации за N запусков мы нашли решения $(\bar{x}_{subm\ ax})_k$ со значениями целевой функции $\left(f(\bar{x}_{subm\ ax})\right)_k$ соответственно $(k = \overline{1,N})$. Используем три вида ошибок:

Надёжность: $R = \frac{\sum_{k=1}^{N} S((\bar{x}_{subm\ ax})_k)}{N}$, где

$S((\bar{x}_{subm\ ax})_k) = \begin{cases} 1, |((\bar{x}_{subm\ ax})_i)_k - (\bar{x}_{max})_i| \le \varepsilon\ (i = \overline{1,n}); \\ 0, \quad \textit{иначе.} \end{cases}$

Ошибка по входным параметрам: $E_x = \frac{\sum_{k=1}^{N} \left(\frac{\sqrt{\sum_{i=1}^{n} ((\bar{x}_{max})_i - ((\bar{x}_{subm\ ax})_i)_k)^2}}{n} \right)}{N}.$

Ошибка по значениям целевой функции: $E_y = \frac{\sum_{k=1}^{N} \left| f(\bar{x}_{max}) - \left(f(\bar{x}_{subm\ ax}) \right)_k \right|}{N}.$

Свойства задачи.

Условной или безусловной оптимизации: Задача безусловной оптимизации.

Одномерной или многомерной оптимизации: Многомерной: n.

Функция унимодальная или многоэкстремальная: Функция многоэкстремальная.

Функция стохастическая или нет: Функция не стохастическая.

<u>Функция Розенброка.</u>

Обозначение: MHL_TestFuction_Rosenbrock

Тип: Задача вещественной оптимизации.

Формула:
(целевая функция) $f(\bar{x}) = \sum_{i=1}^{n-1} \left(100\left(\bar{x}_{i+1} - \bar{x}_i^{\,2}\right)^2 + (1 - \bar{x}_i)^2 \right),$

где $\bar{x} \in X$, $\bar{x}_j \in \left[Left_j ; Right_j \right]$,

$Left_j = -2; Right_j = 2, (j = \overline{1,n})$.

Обозначения: $\bar{x}$ – вещественный вектор;

n – размерность вещественного вектора.

Решаемая задача оптимизации: $\bar{x}_{min} = \arg\min_{\bar{x} \in X} f(\bar{x})$.

Точка минимума: $\bar{x}_{min} = (1, 1, 1 \dots, 1)^T$, то есть $(\bar{x}_{min})_j = 1$ $(j = \overline{1, n})$.

Минимум функции: $f(\bar{x}_{min}) = 0$.

График.

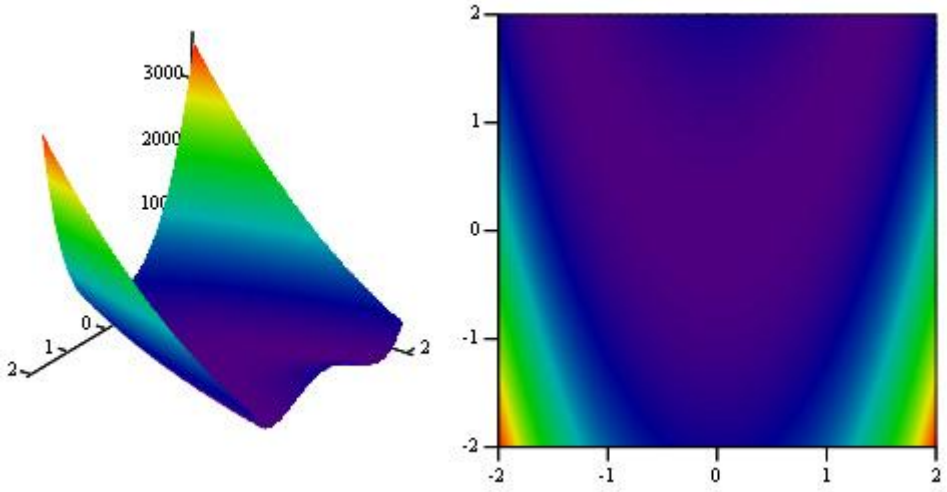

Рисунок 1.5.1.2.4. График функции Розенброка при $n = 2$.

Параметры для алгоритмов оптимизации.

Точность вычислений: $\varepsilon = 0.01$.

$NumberOfParts_j = 4095$ $(j = \overline{1, n})$.

Основная задача и подзадачи.

Изменяемый параметр: n – размерность вещественного вектора.

Значение в основной задаче: $n = 2$.

Нахождение ошибки оптимизации:

Пусть в результате работы алгоритма оптимизации за N запусков мы нашли решения $(\bar{x}_{subm\,ax})_k$ со значениями целевой функции $\left(f(\bar{x}_{subm\,ax})\right)_k$ соответственно ($k = \overline{1,N}$). Используем три вида ошибок:

Надёжность: $R = \frac{\sum_{k=1}^{N} S((\bar{x}_{subm\ ax})_k)}{N}$, где

$$S((\bar{x}_{subm\,ax})_k) = \begin{cases} 1, |((\bar{x}_{subm\,ax})_i)_k - (\bar{x}_{max})_i| \le \varepsilon\ (i = \overline{1,n}); \\ 0, \quad \textit{иначе.} \end{cases}$$

Ошибка по входным параметрам:

$$E_x = \frac{\sum_{k=1}^{N} \left(\frac{\sqrt{\sum_{i=1}^{n} ((\bar{x}_{max})_i - ((\bar{x}_{subm\,ax})_i)_k)^2}}{n} \right)}{N}.$$

Ошибка по значениям целевой функции:

$$E_y = \frac{\sum_{k=1}^{N} \left| f(\bar{x}_{max}) - \left(f(\bar{x}_{subm\,ax})\right)_k \right|}{N}.$$

Свойства задачи.

Условной или безусловной оптимизации:	Задача безусловной оптимизации.
Одномерной или многомерной оптимизации:	Многомерной: n.
Функция унимодальная или многоэкстремальная:	Функция многоэкстремальная.
Функция стохастическая или нет:	Функция не стохастическая.
Особенности:	Наличие полого плато.

Тестовые задачи условной оптимизации

<u>Функция условной оптимизации №1.</u>

Обозначение:	MHL_ConstrainedOptimization_1
Тип:	Задача вещественной оптимизации.
Формула: (целевая функция)	$f(x,y) = 5x + 0.5y,$ $y \le -2x + 5;$

$$y > x - 1.5;$$

$$y \le 2x + 1.$$

где $x, y \in X$, $x, y \in [Left; Right]$,

$Left = -10$; $Right = 10$.

Решаемая задача оптимизации: $(x, y) = \arg\max_{x,y \in X} f(x, y)$.

Точка минимума: $x_{max} = \frac{13}{6}$, $y_{max} = \frac{2}{3}$.

Минимум функции: $f(x_{max}, y_{max}) = \frac{67}{6}$.

Параметры для алгоритмов оптимизации.

Точность вычислений: $\varepsilon = 0.05$.

$NumberOfParts_j = 4095$ $(j = \overline{1, n})$.

Нахождение ошибки оптимизации:

Пусть в результате работы алгоритма оптимизации за N запусков мы нашли решения $(\bar{x}_{subm\,ax})_k$ со значениями целевой функции $\left(f(\bar{x}_{subm\,ax})\right)_k$ соответственно $(k = \overline{1, N})$. Используем три вида ошибок:

Надёжность: $R = \frac{\sum_{k=1}^{N} S((\bar{x}_{subm\ ax})_k)}{N}$, где

$$S((\bar{x}_{subm\,ax})_k) = \begin{cases} 1, |((\bar{x}_{subm\,ax})_i)_k - (\bar{x}_{max})_i| \le \varepsilon \ (i = \overline{1, n}); \\ 0, \quad \textit{иначе.} \end{cases}$$

Ошибка по входным параметрам: $E_x = \frac{\sum_{k=1}^{N} \left(\frac{\sqrt{\sum_{i=1}^{n} ((\bar{x}_{max})_i - ((\bar{x}_{subm\,ax})_i)_k)^2}}{n} \right)}{N}$.

Ошибка по значениям целевой функции: $E_y = \frac{\sum_{k=1}^{N} \left| f(\bar{x}_{max}) - \left(f(\bar{x}_{subm\,ax})\right)_k \right|}{N}$.

Основная задача и подзадачи.

Изменяемый параметр:	Тип учета ограничений.
Значение в основной задаче:	Смертельные штрафы.
Подзадача №2	Статические штрафы

Свойства задачи.

Условной или безусловной оптимизации:	Задача условной оптимизации.
Одномерной или многомерной оптимизации:	Двумерной.
Функция стохастическая или нет:	Функция не стохастическая.
Особенности:	Нет.

1. 4. Критерии оценивания алгоритмов

Для оценивания эффективности работы алгоритмов оптимизации результаты исследований сравниваются по нескольким критериям. Определяются они через три вида ошибок, способ определения каждой из которых рассмотрен в описании тестовых задач.

Если не оговорено иное, в работе значение каждого критерия вычисляется 10 раз и каждый раз значение получается по $N = 100$ запускам алгоритма оптимизации с постоянными параметрами алгоритма.

Критерий №1. Количество вычислений целевой функции для достижения 95% надежности.

Используемый алгоритм для вычисления критерия:

$Criterion1 =$ (1.4.1)

- **Начало алгоритма**.
- $i = 2$;
- $b = false$;

- **Повторять** до тех пор, пока $b = false$:
    - **Начало**
    - **Число вычислений целевой функции** $N = i^2$;
    - **Вычислим** значение надежности R;
    - **Если** $R \geq 0.95$, то $b = true$;
    - **Конец**
- **Возвратим значение** N.
- **Конец алгоритма**.

Критерий №2. Надежность при заданном количестве вычислений целевой функции. Совпадает с ошибкой оптимизации «Надежность» R:

$$Criterion2 = R. \quad (1.4.1)$$

Критерий №3. Ошибка по аргументу при заданном количестве вычислений целевой функции. Совпадает с ошибкой оптимизации «Ошибка по входным параметрам» E_x.

$$Criterion3 = E_x. \quad (1.4.2)$$

Критерий №4. Ошибка по значениям целевой функции при заданном количестве вычислений целевой функции. Совпадает с ошибкой оптимизации «Ошибка по значениям целевой функции» E_y.

$$Criterion4 = E_x. \quad (1.6.3)$$

Критерий Вилкоксона

Для сравнения эффективности алгоритма при различных настроках или нескольких разных алгоритмов используется непараметрический ранговый критерий Вилкоксона. В работе используется враинт, описанный в [9].

«Критерий Вилкоксена предназначен для проверки однородности двух выборок» [9, стр. 93].

Общая идея заключается в том, что из двух выборок составляется общий ряд и ранжируется. При этом одинаковым элементам дается

среднеарифметический ранг. Потом суммируются ранги выборки с меньшим объемом.

Если полученная сумма укладывается в интервал, зависящий от объема выборок и уровня значимости, то считается, что выборки статистически независимы при данном уровне значимости.

В противном случае, смотрятся средние арифметические значения выборок, при сравнении которых определяется какая выборка «превосходит» другую.

В работе используется уровень значимости $\alpha = 0.01$.

Чаще всего в работе используются для сравнения выборки одинокого объема, равного 10. Тогда интервал, при попадании в который выборки считаются однородными равен $[74; 136]$. [9]

При ряде исследований требуется выбрать «лучшую» выборку w^{Best} из множества выборок $\{w^1, \dots, w^K\}$, где K – число выборок, а $w^i = \{w_1^i, \dots, w_{m_i}^i\}$, m_i – объем каждой выборки ($i = \overline{1, K}$).

Путь под «лучшей» выборкой понимается такая выборка, которая по среднему значению больше других выборок.

Предлагается следующий алгоритм. В отличии от обычной практики, алгоритм выдает не одну выборку, а множество выборок W^{Best}, которые неразличимы друг от друга по критерию Вилкоксона.

В алгоритме используется функция, выполняющая критерий Вилкоксона на двух выборках:

$$TestWill(x, y, \alpha) = \begin{cases} 1, \text{если } x \text{ превосходит } y; \\ 0, \text{если } x \text{ и } y \text{ однородны}; \\ -1, \text{если } y \text{ превосходит } x. \end{cases}$$

где x, y – две выборки, а α – уровень значимости.

$BestSample(\{w^1, \dots, w^K\}, \alpha) =$ (1.6.4)

- **Начало алгоритма**.
- $i_{Best} = 0$;
- $W^{Best} = \emptyset$;

- **Цикл от** $i = 0$ до K
    - **Начало**
    - **Если** $TestWill\left(w^i, w^{i_{Best}}, \alpha\right) = 1$, то $i_{Best} = i$;
    - **Конец**
- **Цикл от** $i = 0$ до K
    - **Начало**
    - **Если** $TestWill\left(w^i, w^{i_{Best}}, \alpha\right) = 0$, то $W^{Best} = W^{Best} \cup w^i$;
    - **Конец**
- **Возвратим множество** W^{Best}.
- **Конец алгоритма.**

То есть мы в алгоритме «проходим» по множеству выборок два раза. При первом заходе мы определяем одну выборку, которая превосходит остальные. При этом не обязательно, что по среднему арифметическому данная выборка будет лучшей. При втором заходе мы сравниваем все выборки с выбранной, и если выборки однородны, то заносим их в возвращаемое множество W^{Best}. По этой причине в исследованиях, в которых определяются наилучшие и наихудшие настройки алгоритмов оптимизации, в качестве их выдается множество настроек.

Случай, когда под «лучшей» выборкой понимается выборка, которая по среднему значению меньше других выборок, рассматривается аналогично.

1. 5. Выводы

В первой главе мы рассмотрели подробно постановку задачи, набор тестовых задач и способы сравнения эффективности алгоритмов оптимизации на тестовых задачах. Также был предложен способ использования критерия Вилкосона для сравнения множества выборок.

На основании данной информации теперь мы можем проводить полные исследования с возможностью сравнения эффективности различных алгоритмов глобальной оптимизации.

Глава 2. Стандартный генетический алгоритм как точка отсчета для сравнения алгоритмов оптимизации

2. 1. Общая модель стандартного генетического алгоритма

Генетический алгоритм можно представить как некий стохастический оператор, который на выходе выдает **субоптимальное** решение $\bar{x}_{subm\,ax}$ и значение функционала от этого решения $f(\bar{x}_{subm\,ax})$. То есть мы можем записать, что

$$\begin{pmatrix} \bar{x}_{subm\,ax} \\ f(\bar{x}_{subm\,ax}) \end{pmatrix} = GeneticAlgorithm \begin{pmatrix} X \\ f(\bar{x}) \\ g_i(\bar{x}) \\ h_j(\bar{x}) \\ Parameters \end{pmatrix}, \qquad (2.1.1)$$

где $i = \overline{1, m_1}$, $j = \overline{1, m_2}$;

$GeneticAlgorithm$ – непосредственно сам генетический алгоритм как оператор;

$Parameters$ – параметры генетического алгоритма, которые определяет пользователь (будут рассмотрены ниже).

В виде схемы это можно представить так:

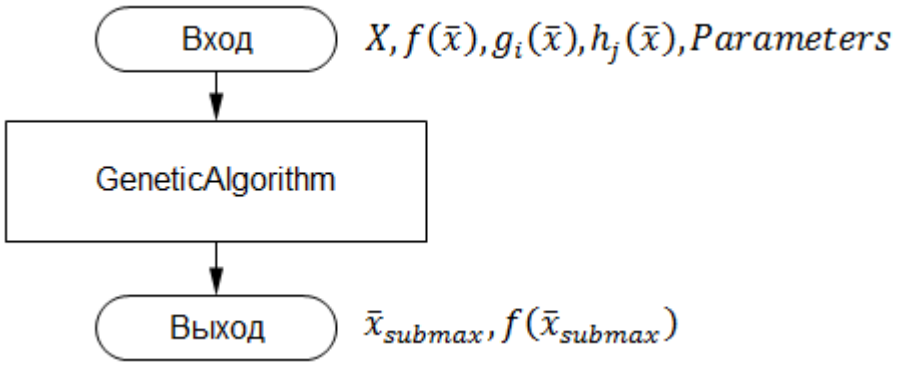

Рисунок 2.1.1. Модель черного ящика генетического алгоритма.

Как было отмечено выше, стандартный генетический алгоритм (сГА) решает задачу оптимизации, как на бинарных строках, так и на вещественных. Поэтому стандартный генетический алгоритм включает в себя два алгоритма:

- Стандартный генетический алгоритм на бинарных строках ($BinaryGeneticAlgorithm$);
- Стандартный генетический алгоритм на вещественных строках.

При этом сГА на вещественных строках включает в себя сГА на бинарных строках, а также блок преобразования задачи вещественной оптимизации к задаче бинарной оптимизации и блок преобразования полученного бинарного решения к вещественному.

По этой причине вектор параметров сГА состоит из двух частей:

$$Parameters = \begin{pmatrix} ParametersOfBinaryGA \\ ParametersOfConvertingIntoBinaryGA \end{pmatrix}. \quad (2.1.2)$$

Здесь $ParametersOfBinaryGA$ – параметры стандартного генетического алгоритма на бинарных строках, а $ParametersOfConvertingIntoBinaryGA$ - параметры преобразования задачи оптимизации на вещественных векторах к задаче оптимизации на бинарных векторах. В случае если в конкретном случае решается задача бинарной оптимизации, то блок параметров $ParametersOfConvertingIntoBinaryGA$ можно опустить.

Построим общую схему сГА, учитывая вышесказанное:

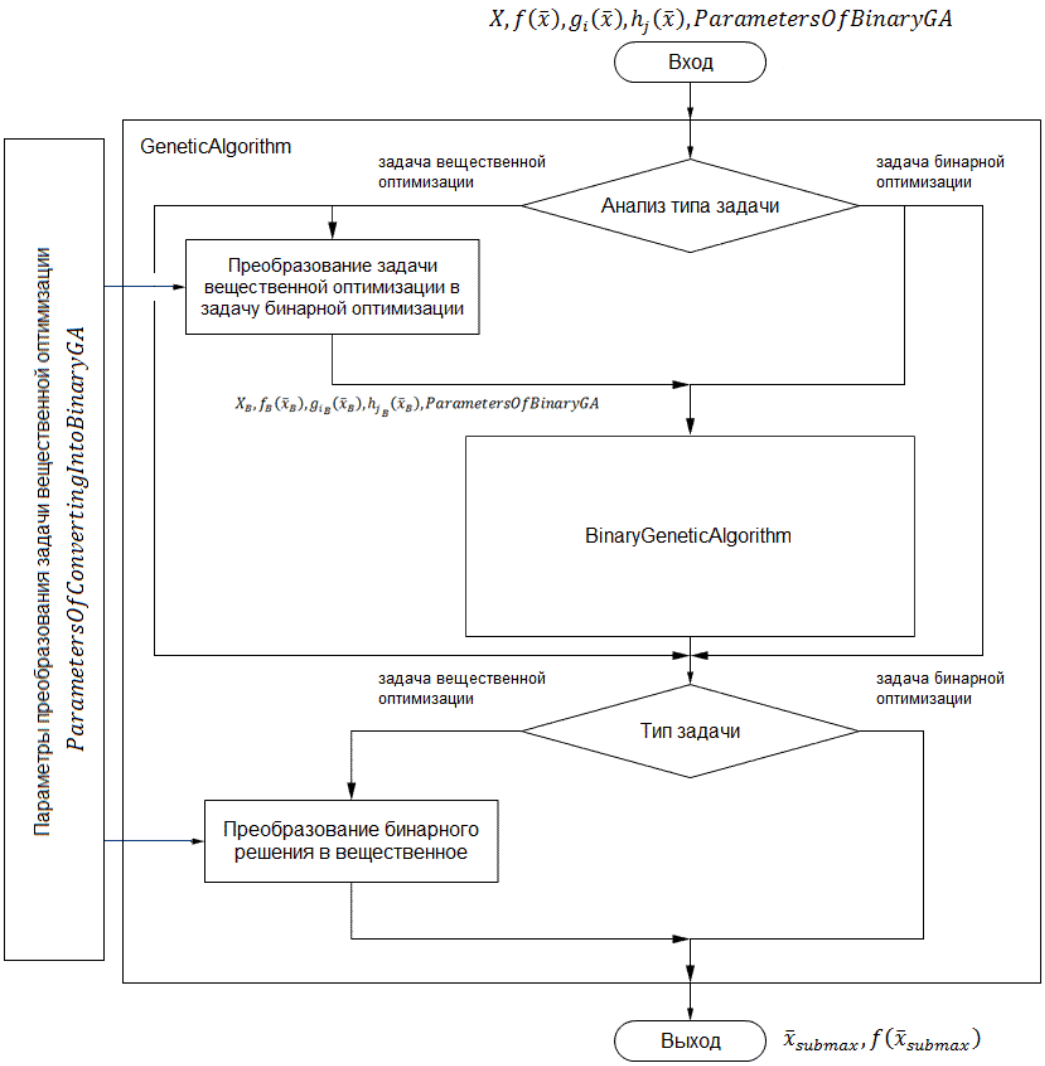

Рисунок 2.1.2. Общая модель стандартного генетического алгоритма

Распишем ее в виде алгоритма $GeneticAlgorithm$:

- **Начало алгоритма**.

- **Анализ типа задачи**. Если X представляет собой множество вещественных векторов, то переходим к следующему шагу, иначе сразу переходим к шагу «Выполнение стандартного генетического алгоритма на бинарных строках».

- **Преобразование задачи вещественной оптимизации в задачу бинарной оптимизации**:

$$\begin{pmatrix} X_B \\ f_B(\bar{x}_B) \\ g_{i_B}(\bar{x}_B) \\ h_{j_B}(\bar{x}_B) \\ ParametersOfBinaryGA \end{pmatrix} =$$

$$ConvertingIntoBinaryGA\begin{pmatrix} X \\ f(\bar{x}) \\ g_i(\bar{x}) \\ h_j(\bar{x}) \\ ParametersOfBinaryGA \\ ParametersOfConvertingIntoBinaryGA \end{pmatrix}.$$

- **Выполнение стандартного генетического алгоритма на бинарных строках**. Если решается задача бинарной оптимизации, то подается вектор входных параметров самого алгоритма $GeneticAlgorithm$ (без $ParametersOfConvertingIntoBinaryGA$, если он присутствует), иначе подается преобразованный вектор, полученный на предыдущем шаге.

$$A = \begin{cases} \begin{pmatrix} X \\ f(\bar{x}) \\ g_i(\bar{x}) \\ h_j(\bar{x}) \\ ParametersOfBinaryGA \end{pmatrix}, \text{если } X - \text{множество бинарных векторов;} \\ \begin{pmatrix} X_B \\ f_B(\bar{x}_B) \\ g_{i_B}(\bar{x}_B) \\ h_{j_B}(\bar{x}_B) \\ ParametersOfBinaryGA \end{pmatrix}, \text{если } X - \text{множество вещественных векторов.} \end{cases}$$

$$\begin{pmatrix} \bar{x}_{subm\ ax\ B} \\ f(\bar{x}_{subm\ ax})_B \end{pmatrix} = BinaryGeneticAlgorithm\ (A).$$

- **Тип задачи**. Используем информацию, полученную на этапе «Анализ типа задачи». Если X представляет собой множество вещественных векторов, то переходим к следующему шагу, иначе **возвращаем** в качестве результата работы алгоритма $GeneticAlgorithm$ вектор $\left(\bar{x}_{subm\ ax\ B}; f(\bar{x}_{subm\ ax})_B\right)^{\mathrm{T}}$.

- **Преобразование бинарного решения в вещественное**. Преобразуем полученное бинарное решение, полученное в ходе выполнения

$BinaryGeneticAlgorithm$ в вещественное, которое требуется в качестве решения задачи оптимизации, и **возвратим** его:

$$\begin{pmatrix} \bar{x}_{subm\ ax} \\ f(\bar{x}_{subm\ ax}) \end{pmatrix} = BinaryToReal \begin{pmatrix} \bar{x}_{subm\ ax\ B} \\ f(\bar{x}_{subm\ ax})_B \\ ParametersOfConvertingIntoBinaryGA \end{pmatrix}.$$

- **Конец алгоритма.**

В основу работы генетического алгоритма заложена имитация процесса эволюции живых организмов. Поэтому алгоритм представляет собой итерационный процесс (имитация смены поколений), работающий одновременно с несколькими взаимодействующими решениями (имитация популяции живых организмов).

Подробная схема, описывающая работу генетического алгоритма на бинарных строках ($BinaryGeneticAlgorithm$), представлена ниже на рисунке.

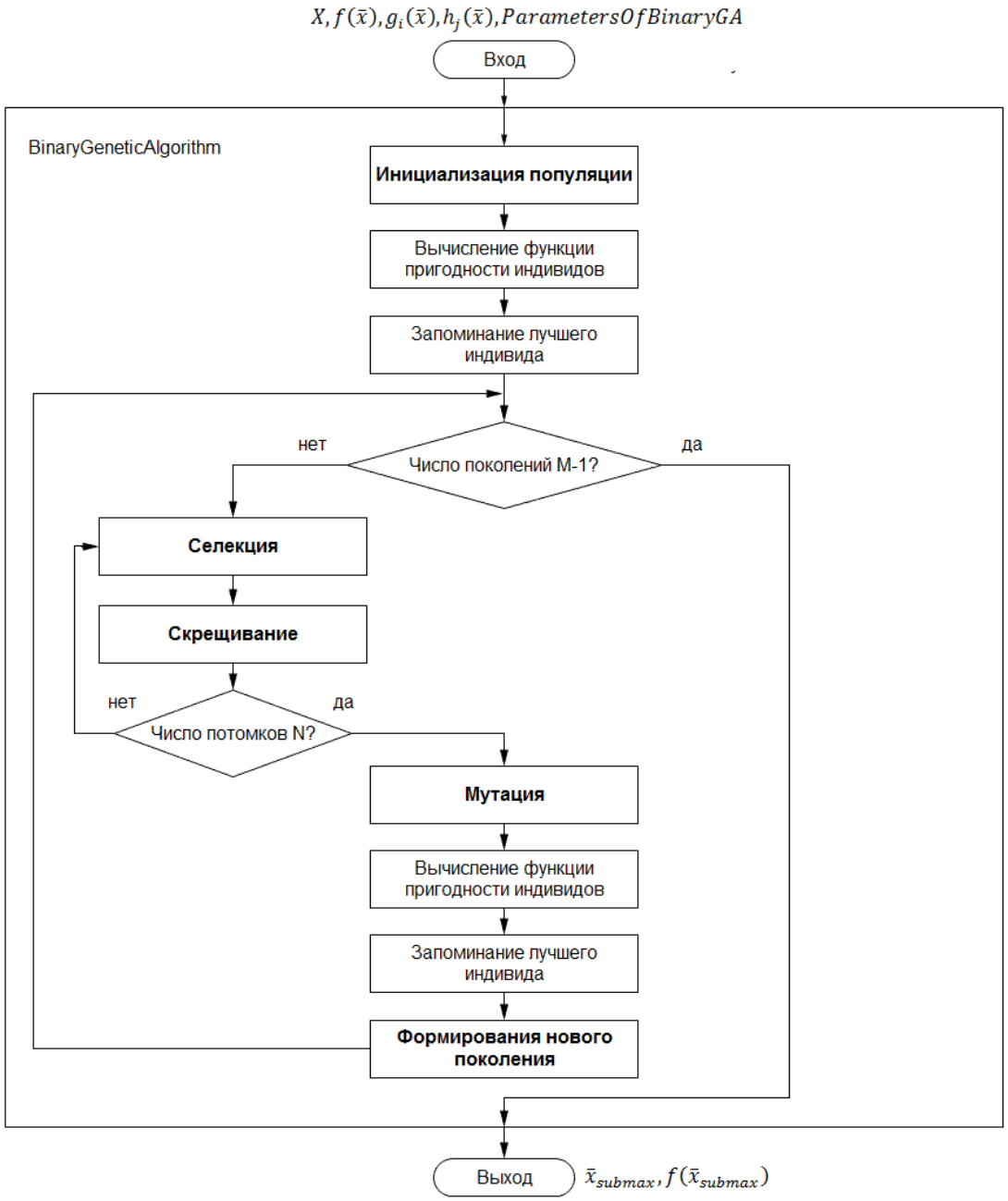

Рисунок 2.1.3. Схема генетического алгоритма

2. 2. Описание стандартного генетического алгоритма на бинарных строках

Теперь приступим непосредственно к описанию самого алгоритма согласно схеме, приведенной выше.

- **Начало алгоритма**.
- **Инициализация популяции**. Создается массив индивидов:

$Population = Initialization(X)$,

где $Population = \{\bar{x}^1; \bar{x}^2; \dots; \bar{x}^N\}$, $\bar{x}^i \in X$, $i = \overline{1, N}$.

- **Вычисление функции пригодности** индивидов. Создается массив:

$Fitness = \{f_{fit}(\bar{x}^1); f_{fit}(\bar{x}^2); \dots; f_{fit}(\bar{x}^N)\}$.

- **Запоминание лучшего индивида** и его значение целевой функции:

$\overline{Best} = \arg \max_{\bar{x} \in Population} f(\bar{x})$;

$BestFitness = \max_{\bar{x} \in Population} f(\bar{x})$.

- Повторять $M - 1$ раз
    - **Начало**
    - Цикл от $j = 1$ до N
        - **Начало**
        - **Селекция**. Выбрать двух родителей:

          $\overline{Parent}^1 = Selection(Population, Fitness, DataOfSel)$;

          $\overline{Parent}^2 = Selection(Population, Fitness, DataOfSel)$, где

          $\overline{Parent}^1 \in X, \overline{Parent}^2 \in X$.
        - **Скрещивание**. Получение потомка из родителей:

          $\overline{Child}^j = Crossover(\overline{Parent}^1, \overline{Parent}^2, DataOfCros)$.
        - **Конец**
    - **Мутация** всех потомков.

      $MutChildPopulation = Mutation(ChildPopulation, DataOfMut)$,

      где

      $ChildPopulation = \{\overline{Child}^1; \dots; \overline{Child}^N\}, \overline{Child}^i \in X, i = \overline{1, N}$;

      $MutChildPopulation = \{\overline{MutChild}^1; \dots; \overline{MutChild}^N\}$,

      $\overline{MutChild}^i \in X, i = \overline{1, N}$.
    - **Вычисление функции пригодности** потомков:

$$FitnessOfMutChild = \{f_{fit}(\overline{MutChild}^1); \dots; f_{fit}(\overline{MutChild}^N)\}.$$

- **Запоминание**. Если есть потомок лучше сохраненного лучшего индивида $\overline{Best}$, то заменяем его новым индивидом с его значением целевой функции:

$$\overline{Best} = \arg \max_{\bar{x} \in MutChildPopulation \cup \{\overline{Best}\}} f(\bar{x});$$

$$BestFitness = \max_{\bar{x} \in MutChildPopulation \cup \{\overline{Best}\}} f(\bar{x}).$$

- **Формирование нового поколения**. Из потомков и родителей формируется новое поколение, которое заменяет предыдущее поколение.

$$\begin{pmatrix} NewPopulation \\ NewFitness \end{pmatrix} = Forming \begin{pmatrix} Population \\ MutChildPopulation \\ Fitness \\ FitnessOfMutChild \\ DataOfForm \end{pmatrix};$$

$$Population = NewPopulation;$$

$$Fitness = NewFitness.$$

- **Конец**

- **Выдаем лучшего индивида** и его значение целевой функции, то есть возвращаем пару $\begin{pmatrix} \overline{Best} \\ BestFitness \end{pmatrix}$.

- **Конец алгоритма**.

Здесь,

N – размер популяции;

M – число поколений;

$DataOfSel$ – вектор параметров оператора селекции;

$DataOfCros$ – вектор параметров оператора скрещивания;

$DataOfMut$ – вектор параметров оператора мутации;

$DataOfForm$ – вектор параметров оператора формирования нового поколения.

При решении задачи нам известно количество вычислений целевой функции $CountOfFitness$, которые мы можем реализовать. Данная величина определяется по затратам по вычисление целевой функции в одной точке, временными ограничениями, экономической целесообразностью и другими причинами.

Вышеописанные переменные являются параметрами генетического алгоритма, варьируя которые мы получаем разные по эффективности реализации ГА. Во всех векторах параметров операторов главным параметром является тип данного оператора, так как существует несколько реализаций каждого из операторов.

При этом в сГА на бинарных строках число поколений и размер популяции берутся в таком соотношении, чтобы они были приблизительно одинаковыми. В таком случае мы можем исключить параметры M и N из числа определяемых пользователем, воспользовавшись формулой:

$$M = int\left(\sqrt{CountOfFitness}\right); \quad (2.2.1)$$
$$N = int\left(\frac{CountOfFitness}{M}\right).$$

Итак, определим вектор всех изменяемых параметров стандартного генетического алгоритма на бинарных строках:

$$ParametersOfBinaryGA = \begin{pmatrix} CountOfFitness \\ DataOfSel \\ DataOfCros \\ DataOfMut \\ DataOfForm \end{pmatrix}. \quad (2.2.2)$$

Замечание. В литературе часто в качестве критерия остановки алгоритма используется не число вычислений функции $CountOfFitness$, а время работы алгоритма или достижение заданной точности задачи или самого алгоритма.

Если взять в качестве критерия время работы алгоритма, то исследования по сравнению алгоритмов будут жестко привязаны к компьютерам, на которых проводились вычисления. При этом критерий будет характеризовать скорее ЭВМ, чем алгоритм, и результаты будет сложно проверить другими исследователями. Следует помнить, что часто в практических задачах основное время работы занимает вычисление целевой функции, а не работы самого алгоритма, которое пренебрежительно мало.

Если же взять в качестве критерия остановки достижение заданной точности алгоритмом (например, K поколений проходят без улучшения наилучшего решения $\overline{Best}$), то сравнивать между собой можно только эволюционные алгоритмы, так как данный критерий не приемлем для других алгоритмов глобальной оптимизации.

2. 3. Описание операторов

Рассмотрим подробно каждый из операторов генетического алгоритма из его описания.

Инициализация популяции

Инициализация популяции – процесс формирования случайного массива с фиксированным числом N элементов из множества X. При этом должно выполняться условие, что для любых $\bar{x}, \bar{y} \in X$ вероятности попадания в популяцию один или более раз одинаковы.

Итак, данный оператор определяется формулой:

$$Initialization(X) = \{\bar{x}^1; \bar{x}^2; \dots; \bar{x}^N\}, \bar{x}^i = Random(X), \bar{x}^i \in X, i = \overline{1,N}. \quad (2.3.1)$$

Пример. Допустим, решаем задачу из примера, приведенного в постановке задачи оптимизации. Зададим размер популяции $N = 6$. Тогда в результате работы оператора инициализации мы можем получить популяцию:

$$Population = \begin{Bmatrix} (0;0;1;1;1;1;0;1)^{\mathrm{T}} \\ (1;1;1;0;0;0;1;1)^{\mathrm{T}} \\ (1;0;1;1;0;1;0;1)^{\mathrm{T}} \\ (1;0;0;1;1;1;1;0)^{\mathrm{T}} \\ (0;1;0;1;0;0;0;0)^{\mathrm{T}} \\ (0;1;1;1;0;0;0;0)^{\mathrm{T}} \end{Bmatrix}.$$

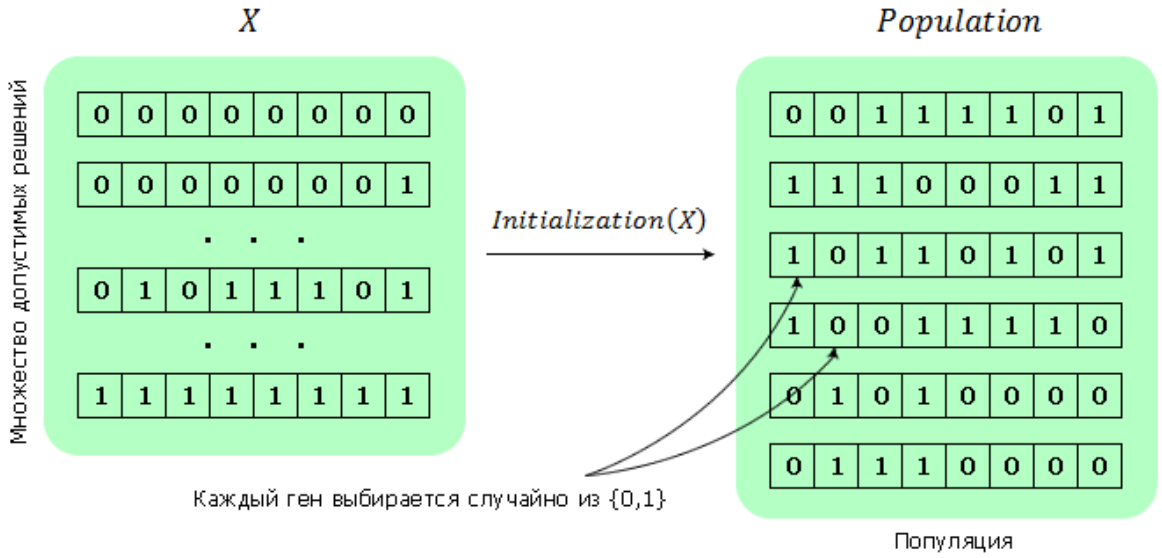

Рисунок 2.2.1. Инициализация популяции

Вычисление функции пригодности

Генетический алгоритм не работает с функционалом $f(\bar{x})$, определяющий эффективность найденных решений. Как видно на схеме (Рисунок 4.1), ГА работает с **функцией пригодности**, которая для каждого индивида из популяции определяется по формуле:

$$f_{fit}(\bar{x}^k) = f_1\left(f_2\left(f(\bar{x}^k), g_i(\bar{x}^k), h_j(\bar{x}^k)\right), Y\right), \qquad (2.3.2)$$

где $\bar{x}^k \in Y$, $Y \in \{Population; MutChildPopulation\}$, $k = \overline{1,N}$, $i = \overline{1,m_1}$, $j = \overline{1,m_2}$.

Запишем более кратко для простоты последующих выкладок:

$$f_{fit}(\bar{x}^k) = f_1^k\left(f_2^k\right), \qquad (2.3.3)$$

где $f_2^k = f_2\left(f(\bar{x}^k), g_i(\bar{x}^k), h_j(\bar{x}^k)\right)$, $f_1^k = f_1\left(f_2^k, Y\right)$.

То есть значение функционала $f(\bar{x}^k)$ претерпевает два преобразования.

f_2^k – преобразование с учетом ограничений, накладываемых на задачу оптимизацию. Например, это может быть штрафы (смертельные, адаптивные и др.). В случае если $m_1 = 0$ и $m_2 = 0$ (задача безусловной оптимизации), то:

$$f_2^k = f(\bar{x}^k), k = \overline{1, N}. \quad (2.3.4)$$

f_1^k – преобразование непосредственно по определению функции пригодности. Ее значение требуется в операторе селекции. При использовании пропорциональной селекции выдвигается требование, чтобы показатель эффективности решения был неотрицательной величиной. Функционал $f(\bar{x})$ таким свойством не обладает. В литературе не описывается способов приведения к соответствующему виду $f(\bar{x}^k)$ (или точнее f_2^k). Обычно сразу оговаривается, что $f(\bar{x})$ обладает необходимыми условиями. Но тестовые функции, на которых проводятся исследования, этими свойствами не обладают. Из-за этого не адекватно сравнивать результаты исследования генетических алгоритмов. Например, при проверке одинаковых модификаций ГА у разных исследователей результаты могут отличаться, так как условия проведения экспериментов разные – по своей сути решаются разные задачи оптимизации. Поэтому предлагается следующая формула, при использовании которой будет всегда выполняться условие $f_1^k \geq 0$:

$$I^{min} = \min_{\bar{x}^k \in Y} f_2^k;$$

$$I^{max} = \max_{\bar{x}^k \in Y} f_2^k;$$

$$f_1^k = \begin{cases} \frac{f_2^k - I^{min}}{I^{max} - I^{min}}, \text{если } I^{min} \neq I^{max}; \\ 1, \text{если } I^{min} = I^{max}. \end{cases} \quad (2.3.5)$$

$$k = \overline{1, N}.$$

Отметим, что условие $f_2^k \geq 0$ не означает удовлетворения целевой функции требованиям пропорциональной селекции на всем множестве решений, так как условие рассматривает только решения из текущей популяции.

Замечание. Для других типов селекции (ранговая и турнирная) нет требования о неотрицательности функции пригодности. Но монотонное преобразование f_1^k не изменяет вероятности выбора индивидов из популяции ввиду того, что в данных типах селекции используется только попарное сравнение пригодностей, а не их численных значений.

Замечание. При преобразовании (2.3.5) обязательно найдется индивид с вероятностью выбора равной нулю в пропорциональной селекции. То есть в таком случае, всегда хотя бы один индивид в текущем поколении не имеет шансов стать родителем. Существуют другие преобразования f_1^k, лишенные этого недостатка, но они не рассматриваются в данной работе.

Замечание. При использовании пропорциональной селекции в случае, когда целевая функция (точнее ее преобразование с учетом ограничений) может принимать отрицательные значения, то пригодность одного и того же индивида может изменяться в разных поколениях из-за преобразования (2.4.2.4).

Пример. Допустим, решаем некую задачу условной оптимизации. Размер популяции $N = 10$. Преобразования функционала можно показать на рисунке:

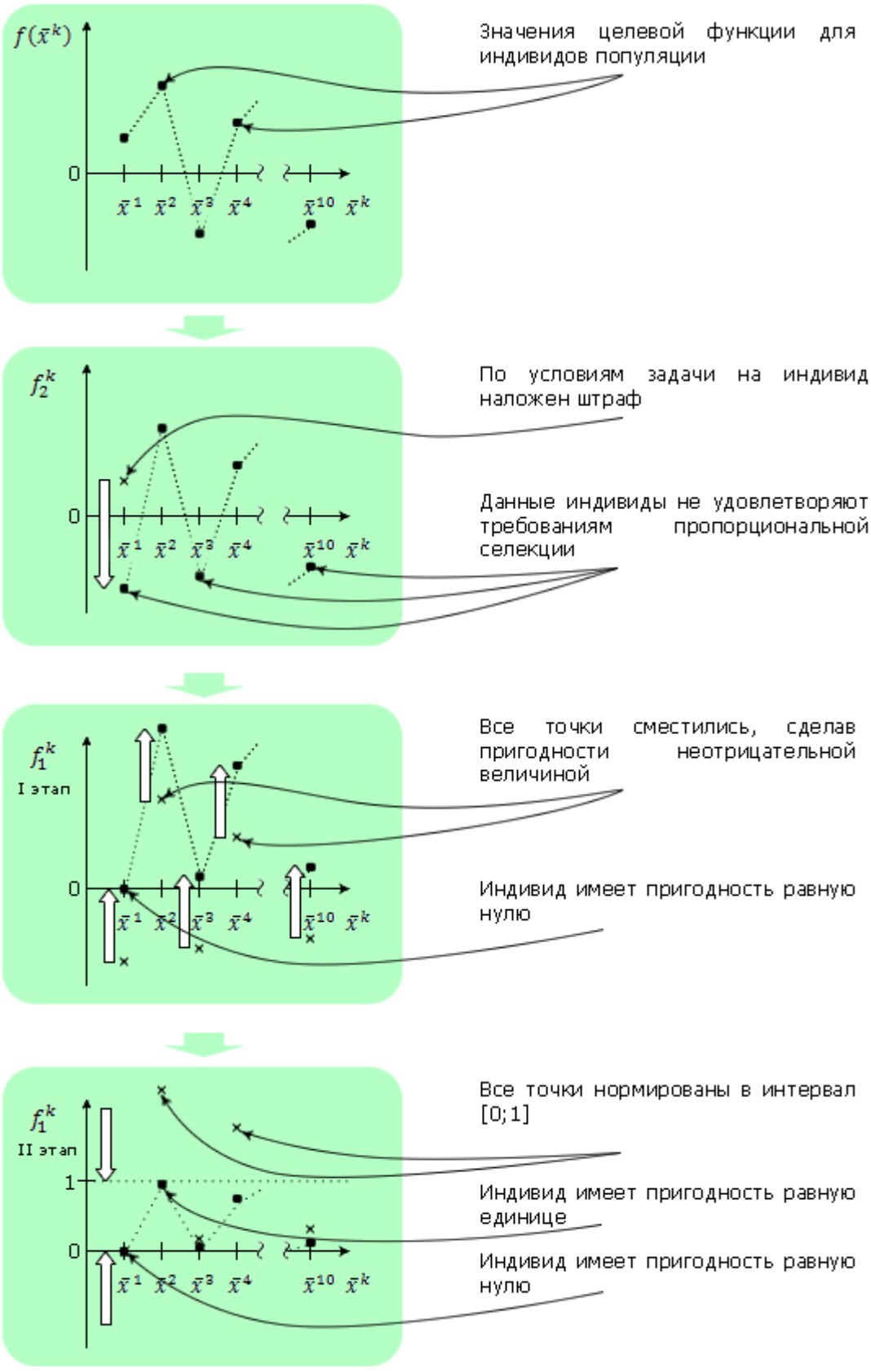

Рисунок 2.3.2 Получение пригодности индивидов популяции

Селекция

Селекция – оператор случайного выбора одного индивида из популяции, основываясь на значениях функции пригодности всех индивидов текущей популяции, для использования его в операторе скрещивания. При этом вероятность выбора у индивидов с более высокой пригодностью выше, чем у индивидов с более низкой пригодностью.

В обычном генетическом алгоритме рассматривается три типа селекции, который определяется параметром $TypeOfSelection$ из вектора $DataOfSel$.

Пропорциональная селекция.

$$TypeOfSelection = ProportionalSelection. \quad (2.3.6)$$

Вероятность выбора элемента пропорциональна значению пригодности индивида. Данный вид селекции может работать только с неотрицательными значениями пригодности, чем, и вызвано преобразование (2.3.5).

Пропорциональная селекция определяется формулой:

$$Selection(Population, Fitness, DataOfSel) = Random(\{\bar{x}_i | p_i\}), \quad (2.3.7)$$

где

$$p_i = \begin{cases} \frac{f_{fit}(\bar{x}_i)}{\sum_{j=1}^{N} f_{fit}(\bar{x}_j)}, \text{если } \exists\, f_{fit}(\bar{x}_k) \neq 0\ (k = \overline{1,N}); \\ \frac{1}{N}, \text{если } \nexists\, f_{fit}(\bar{x}_k) \neq 0\ (k = \overline{1,N}). \end{cases}$$

$\bar{x}_i \in Population,\ i = \overline{1,N}.$

Как видим, формула определения вероятности выбора индивида имеет составной вид. Второе условие ($\nexists\, f_{fit}(\bar{x}_k) \neq 0$) предназначено для маловероятного случая, когда в популяции все индивиды будут иметь пригодность равную нулю.

$DataOfSel$ не содержит каких-либо параметров относительно данного типа селекции.

Пример. Пусть $Fitness = \{0{,}5; 0{,}2; 0{,}1; 0{,}6; 0{,}2; 0{,}4\}$. Тогда вероятности выбора индивидов равны:

$$p_1 = \frac{50}{50+20+10+60+20+40} = 0{,}25;$$

$$p_2 = \frac{20}{50+20+10+60+20+40} = 0{,}1;$$

$$p_3 = \frac{10}{50+20+10+60+20+40} = 0{,}05;$$

$$p_4 = \frac{60}{50+20+10+60+20+40} = 0{,}3;$$

$$p_5 = \frac{20}{50+20+10+60+20+40} = 0{,}1;$$

$$p_6 = \frac{40}{50+20+10+60+20+40} = 0{,}2.$$

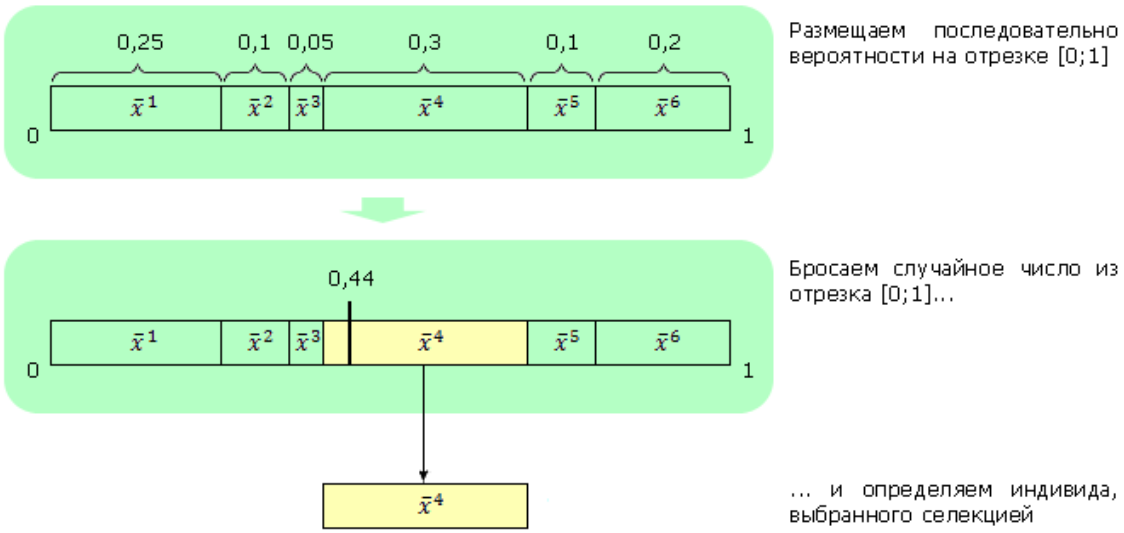

Рисунок 2.3.3 Механизм работы пропорциональной селекции

Ранговая селекция.

$$TypeOfSelection = RankSelection. \qquad (2.3.8)$$

Работает не с массивом пригодностей напрямую, а массивом нормированных рангов, присваиваемых индивидам на основе значений пригодности. Используется функция, которая проставляет ранги для элементов несортированного массива пригодностей, то есть номера, начиная с 1, в отсортированном массиве. Если в массиве есть несколько одинаковых элементов, то ранги им присуждаются как среднеарифметические ранги этих элементов в отсортированном массиве. Если это не сделать, то вероятность выбора индивидов одинаковых по функции пригодности будет не равна друг другу, что противоречит идеи оператора селекции. Далее для выбора индивидов используется пропорциональная селекция, работающая с массивом рангов.

Значит, ранговая селекция определяется формулой:

$$Selection(Population, Fitness, DataOfSel) = Random(\{\bar{x}_i | p_i\}), \quad (2.3.9)$$

где $p_i = \frac{Rank\left(f_{fit}(\bar{x}_i)\right)}{\sum_{j=1}^{N} Rank\left(f_{fit}(\bar{x}_j)\right)}$, $\bar{x}_i \in Population$, $i = \overline{1,N}$.

$$Rank\left(f_{fit}(\bar{x}_i)\right) = \frac{\sum_{j=1}^{N} NumberOfSorting\ \left(f_{fit}(\bar{x}_i), Fitness\right) \cdot S\left(f_{fit}(\bar{x}_i), f_{fit}(\bar{x}_j)\right)}{\sum_{j=1}^{N} S\left(f_{fit}(\bar{x}_i), f_{fit}(\bar{x}_j)\right)}, \quad (2.3.10)$$

где $S\left(f_{fit}(\bar{x}_i), f_{fit}(\bar{x}_j)\right) = \begin{cases} 1, \text{если } f_{fit}(\bar{x}_i) = f_{fit}(\bar{x}_j); \\ 0, \text{если } f_{fit}(\bar{x}_i) \neq f_{fit}(\bar{x}_j). \end{cases}$

$NumberOfSorting\left(f_{fit}(\bar{x}_i), Fitness\right)$ – функция, возвращающая номер элемента $f_{fit}(\bar{x}_i)$ в отсортированном массиве $Fitness$ в порядке возрастания.

Формула (2.4.3.5) подсчитывает средние арифметические ранги при условии, что в массиве $Fitness$ могут встречаться одинаковые элементы.

$DataOfSel$ также не содержит каких-либо параметров относительно данного типа селекции.

Пример. Пусть $Fitness = \{0{,}5; 0{,}2; 0{,}1; 0{,}6; 0{,}2; 0{,}4\}$. Тогда вероятности выбора индивидов равны:

$$p_1 = \frac{5}{5+2{,}5+1+6+2{,}5+4} = 0{,}238;$$

$$p_2 = \frac{2{,}5}{5+2{,}5+1+6+2{,}5+4} = 0{,}119;$$

$$p_3 = \frac{1}{5+2{,}5+1+6+2{,}5+4} = 0{,}047;$$

$$p_4 = \frac{6}{5+2{,}5+1+6+2{,}5+4} = 0{,}286;$$

$$p_5 = \frac{2{,}5}{5+2{,}5+1+6+2{,}5+4} = 0{,}119;$$

$$p_6 = \frac{4}{5+2{,}5+1+6+2{,}5+4} = 0{,}190.$$

Процесс получения этих вероятностей показан на рисунке ниже:

Рисунок 2.3.4 Механизм работы ранговой селекции

Турнирная селекция.

$$TypeOfSelection = TournamentSelection. \quad (2.3.11)$$

Из популяции с равной вероятностью выбираются индивиды в количестве T (размер турнира), где $2 \le T \le N$. При этом каждый индивид может попасть в группу (**турнир**) только один раз (турнирная селекция без возвращения). Из данной группы выбирается индивид с наибольшей пригодностью.

Значит, турнирная селекция определяется формулой:

$$Selection(Population, Fitness, DataOfSel) = \arg\max_{\bar{x} \in H} f_{fit}(\bar{x}), \quad (2.3.12)$$

где $H = \{h_i | h_i = Random(Population / (\{h_1\} \cup \{h_2\} \cup \ldots \cup \{h_{i-1}\}))\}$, $i = \overline{1, T}$.

Турнирная селекция является единственным типом данного оператора, который добавляет в $DataOfSel$ дополнительный параметр – размер турнира T. Обычно выбирают значение этого параметра равное $T = 2$.

Пример. Массив возьмем тот же, что и в предыдущих примерах $Fitness = \{0{,}5; 0{,}2; 0{,}1; 0{,}6; 0{,}2; 0{,}4\}$. Размер турнира равен $T = 3$. Пример работы оператора показан на рисунке:

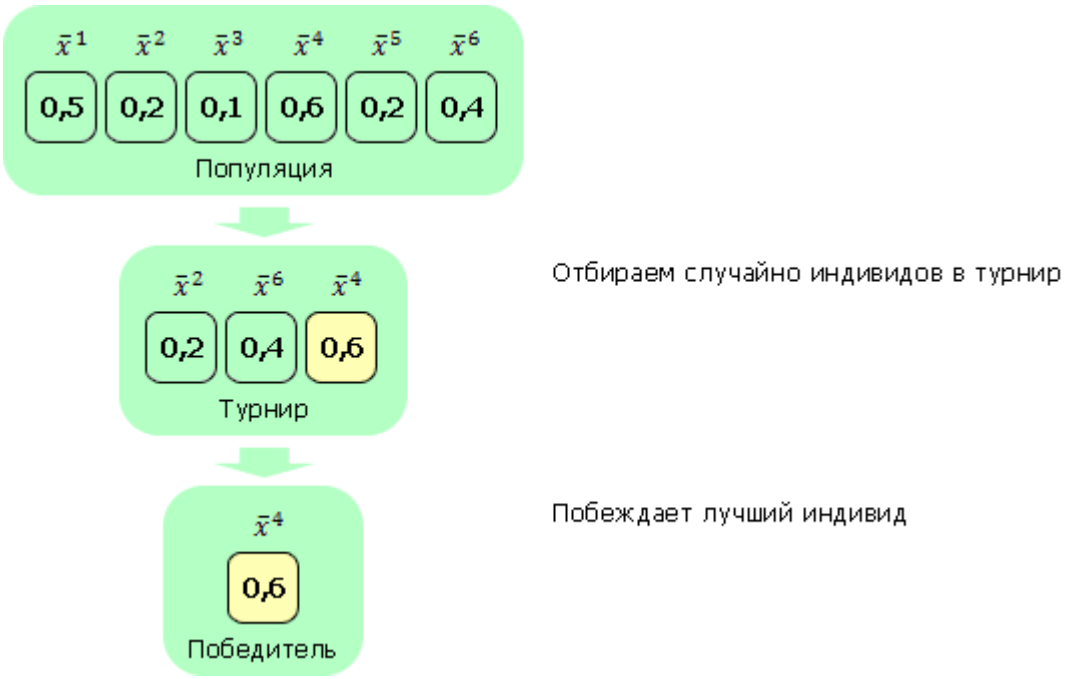

Рисунок 2.3.5 Механизм работы турнирной селекции

Итак, мы рассмотрели используемые варианты селекции. Тогда можно составить вектор параметров оператора селекции $DataOfSel$:

$$DataOfSel = \begin{pmatrix} TypeOfSel \\ T \end{pmatrix} \quad (2.3.13)$$

Замечание. Часто в литературе наряду с рассмотренными ниже тремя видами селекции упоминается элитарная селекция (элитная селекция, селекция элитизма). Этот оператор **не является** селекцией по определению этого оператора. В сГА включен в состав одного из вида оператора формирования нового поколения из родителей и потомков.

Замечание. Если при выполнении селекции сравниваются два разных индивида с одинаковыми значениями функций пригодности, то выбирается первый из них.

Скрещивание

Скрещивание (кроссовер) – оператор случайного формирования нового индивида из двух выбранных родителей с сохранением признаков обоих родителей.

В обычном генетическом алгоритме рассматривается три типа скрещивания, который определяется параметром $TypeOfCrossover$ из вектора $DataOfCros$.

Одноточечное скрещивание.

$$TypeOfCrossover = SinglepointCrossover. \quad (2.3.14)$$

Пусть имеется два родителя (родительские хромосомы) $\overline{Parent}^1$ и $\overline{Parent}^2$. В случайном месте происходит разрыв между двумя позициями генов в обеих хромосомах. После этого хромосомы обмениваются частями, в результате чего образуются два потомка. Из них выбирается случайно один потомок, который и передается в качестве результата оператора скрещивания. То есть скрещивание происходит по формулам:

$$Crossover(\overline{Parent}^1, \overline{Parent}^2, DataOfCros) = Random(\{\overline{Offspring}^1; \overline{Offspring}^2\}), \quad (2.3.15)$$

где

$R = Random(\{2; 3; \dots; N\})$;

$\overline{Offspring}_i^1 = \overline{Parent}_i^1, i = \overline{1, R-1}$;

$\overline{Offspring}_l^1 = \overline{Parent}_l^2, l = \overline{R, N}$;

$\overline{Offspring}_i^2 = \overline{Parent}_i^2, i = \overline{1, R-1}$;

$\overline{Offspring}_l^2 = \overline{Parent}_l^1, l = \overline{R, N}$;

$\overline{Offspring}^1 \in X, \overline{Offspring}^2 \in X$.

$DataOfCros$ не содержит каких-либо параметров относительно данного типа скрещивания.

Пример. Для всех видов скрещивания будем использовать двух родителей: $\overline{Parent}^1 = \{0; 1; 0; 1; 1; 1; 0; 0\}$ и $\overline{Parent}^2 = \{1; 1; 0; 0; 1; 0; 1\}$. Одноточечное скрещивание показано на рисунке:

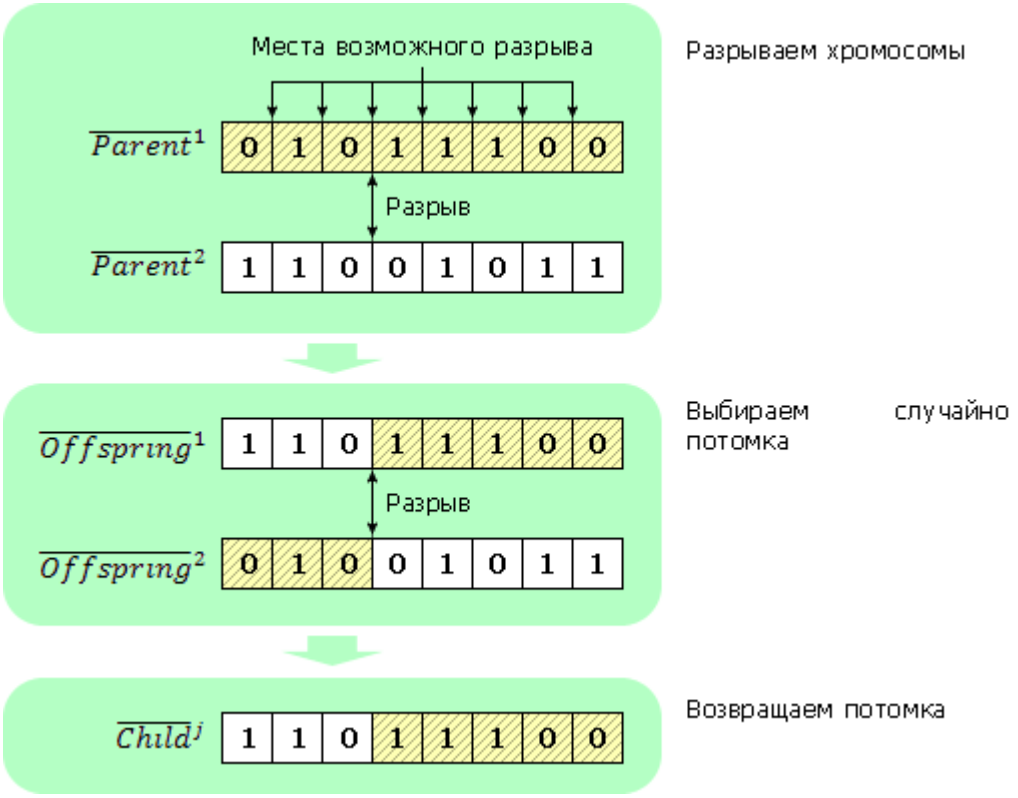

Рисунок 2.3.6 Одноточечное скрещивание

Двухточечное скрещивание.

$$TypeOfCrossover = TwopointCrossover. \quad (2.3.16)$$

Пусть имеется два родителя (родительские хромосомы) $\overline{Parent}^1$ и $\overline{Parent}^2$. В двух случайных местах происходят разрывы между двумя позициями генов в обеих хромосомах. После этого хромосомы обмениваются частями, в результате чего образуются два потомка. Из них выбирается случайно один потомок, который и передается в качестве результата оператора скрещивания. То есть скрещивание происходит по формулам:

$$Crossover(\overline{Parent}^1, \overline{Parent}^2, DataOfCros) = Random(\{\overline{Offspring}^1; \overline{Offspring}^2\}), \quad (2.3.17)$$

где

$R_1 = Random(\{2; 3; \dots; N\});$

$R_2 = Random(\{2; 3; \dots; N\});$

$\overline{Offspring}_i^1 = \overline{Parent}_i^1, i = \overline{1, R_1 - 1};$

$\overline{Offspring}_l^1 = \overline{Parent}_l^2, l = \overline{R_1, R_2 - 1};$

$\overline{Offspring}_k^1 = \overline{Parent}_k^1, k = \overline{R_2, N};$

$\overline{Offspring_i^2} = \overline{Parent_i^2}, i = \overline{1, R_1 - 1};$

$\overline{Offspring_l^2} = \overline{Parent_l^1}, l = \overline{R_1, R_2 - 1};$

$\overline{Offspring_k^2} = \overline{Parent_k^2}, k = \overline{R_2, N};$

$\overline{Offspring}^1 \in X, \overline{Offspring}^2 \in X.$

$DataOfCros$ не содержит каких-либо параметров относительно данного типа скрещивания.

Пример. Двухточечное скрещивание показано на рисунке:

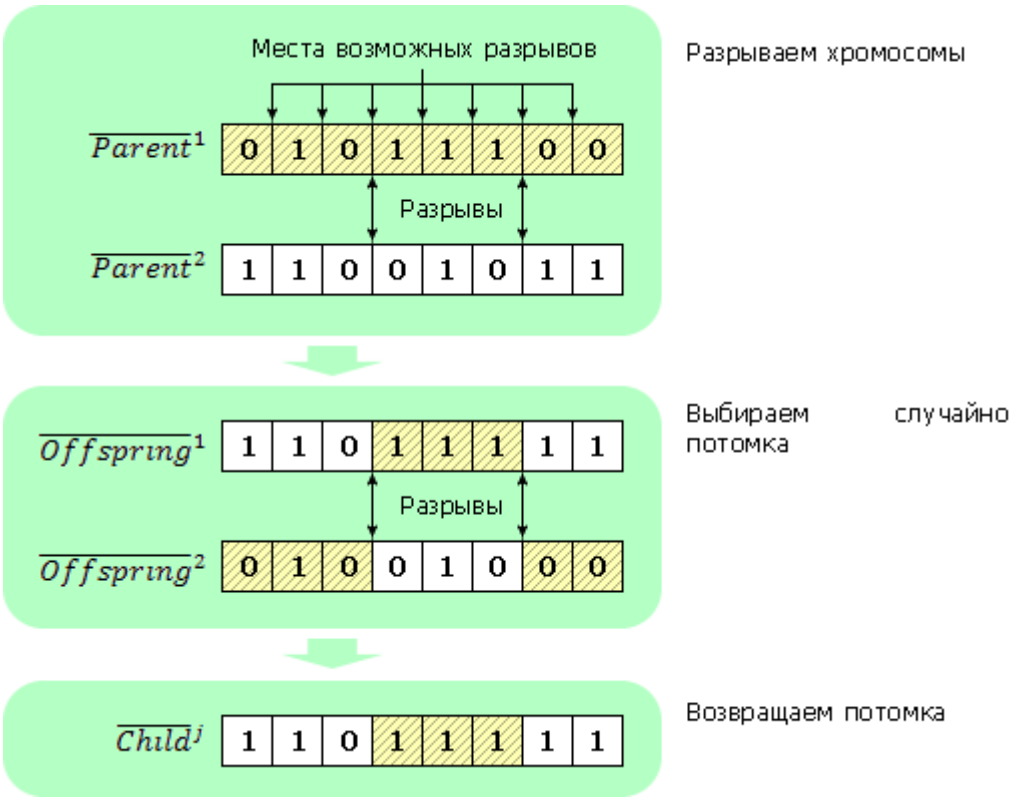

Рисунок 2.3.7 Двухточечное скрещивание

Равномерное скрещивание.

$TypeOfCrossover = UniformCrossover.$ (2.3.18)

Пусть имеется два родителя (родительские хромосомы) $\overline{Parent}^1$ и $\overline{Parent}^2$. Поток состоит из генов, каждый из которых выбран случайно из генов родителей на соответствующих позициях. То есть скрещивание происходит по формулам:

$Crossover(\overline{Parent}^1, \overline{Parent}^2, DataOfCros) = \overline{Offspring}$, (2.3.19)

где

$\overline{Offspring_i} = Random(\{\overline{Parent_i^1}; \overline{Parent_i^2}\}), i = \overline{1, N};$

$\overline{Offspring} \in X$.

$DataOfCros$ не содержит каких-либо параметров относительно данного типа скрещивания.

Пример. Равномерное скрещивание показано на рисунке:

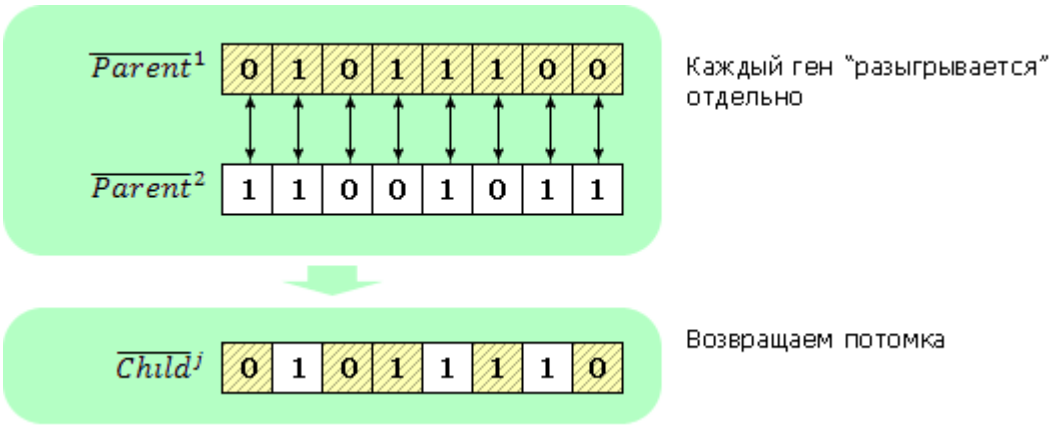

Рисунок 2.3.8 Равномерное скрещивание

Итак, мы рассмотрели используемые варианты скрещивания. Тогда можно составить вектор параметров оператора скрещивания $DataOfCros$:

$$DataOfCros = (TypeOfCros) \quad (2.3.20)$$

Мутация

Мутация – оператор случайного изменения всех потомков из популяции. Цель данного оператора не получить более лучшее решение, а разнообразить многообразие рассматриваемых индивидов. Обычно мутация предполагает незначительное изменение потомков. При выполнении оператора каждый ген каждого индивида с некоторой заданной вероятностью $ProbabilityOfMutation$ мутирует, то есть меняет свое значение на противоположное. Мутация происходит по формулам:

$$Mutation(ChildPopulation, DataOfMut) = \{\overline{MutChild}^1; \dots; \overline{MutChild}^N\}, \quad (2.3.21)$$

где

$$\overline{MutChild}_j^i = \begin{cases} \overline{Child}_j^i, \text{если } random(0,1) > ProbabilityOfMutation; \\ 1 - \overline{Child}_j^i. \end{cases}$$

$\overline{Child}^i \in ChildPopulation, i = \overline{1,N}$;

$\overline{MutChild}^i \in X, i = \overline{1,N}$.

Обычно в генетическом алгоритме вероятность мутации выбирается из трех вариантов: слабая ($Weak$), средняя ($Average$) и сильная ($Strong$) мутация.

Отсюда вероятность мутации определяется формулой:

$$ProbabilityOfMutation(TypeOfMutation) = \quad (2.3.22)$$

$$= \begin{cases} \frac{1}{3n}, \text{если } TypeOfMutation = WeakFor_n; \\ \frac{1}{n}, \text{если } TypeOfMutation = AverageFor_n; \\ \min\left(\frac{3}{n}, 1\right), \text{если } TypeOfMutation = StrongFor\_n. \end{cases}$$

Здесь

$$TypeOfMutation \in \begin{Bmatrix} WeakFor\_n, \\ AverageFor\_n, \\ StrongFor\_n, \end{Bmatrix},$$

n – длина вектора $\bar{x} \in X$ бинарной задачи оптимизации.

Составим вектор параметров оператора мутации $DataOfMut$:

$$DataOfMut = (TypeOfMutation). \quad (2.3.23)$$

Замечание. Определение сильной мутации через $\min\left(\frac{3}{n}, 1\right)$ связано с тем, что при $n < 3$ вероятность мутации становится больше 1, что недопустимо. Но при программировании это не вызывает отклонений в работе генетического алгоритма, к тому же использование ГА для столь малых размерностей нецелесообразно.

Пример. На рисунке показана мутация одного из индивидов из популяции. С точки зрения теории, представленной в данной работе, можно предположить, что популяция состоит из одного индивида.

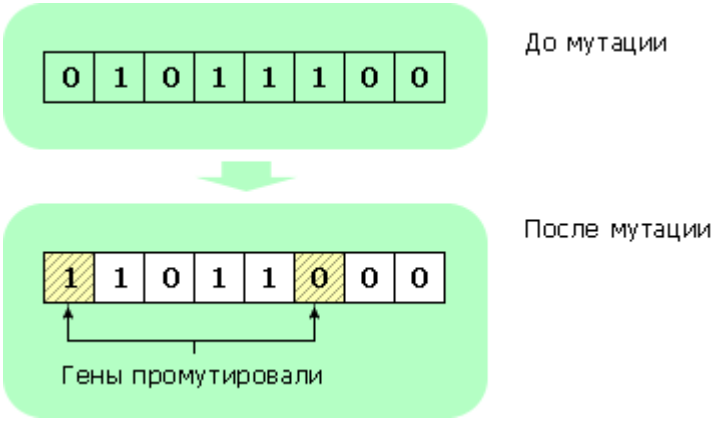

Рисунок 2.3.9 Мутация

Формирование нового поколения

Формирование нового поколения – оператор формирования нового поколения из массива родителей и получившихся потомков с использованием уже известных значений функции пригодности, как родителей, так и потомков.

В обычном генетическом алгоритме используется два типа формирования нового поколения, который определяется параметром $TypeOfGenerationForming$ из вектора $DataOfForm$.

Только потомки.

$$TypeOfGenerationForming = OnlyOffspringGenerationForming. \quad (2.3.24)$$

Данная схема предполагает формирование нового поколения из потомков и родителей таким, что в новое поколение попадают только потомки. Данный тип формирования нового поколения определяется формулой:

$$Forming\begin{pmatrix} Population \\ MutChildPopulation \\ Fitness \\ FitnessOfMutChild \\ DataOfForm \end{pmatrix} = \begin{pmatrix} MutChildPopulation \\ FitnessOfMutChild \end{pmatrix}. \quad (2.3.25)$$

$DataOfForm$ не содержит каких-либо параметров относительно данного типа формирования нового поколения.

Только потомки и копия лучшего индивида.

$$TypeOfGenerationForming = OnlyOffspringWithBestGenerationForming. \quad (2.3.26)$$

Данная схема предполагает формирование нового поколения из потомков и родителей таким, что в новое поколение попадают только потомки (без одного) и копия лучшего индивида $\overline{Best}$ (лучшего за всё время работы генетического алгоритма, а не только текущего поколения). В русской литературе данный способ часто называют селекцией элитизма. Данный тип формирования нового поколения определяется формулой:

$$Forming\begin{pmatrix} Population \\ MutChildPopulation \\ Fitness \\ FitnessOfMutChild \\ DataOfForm \end{pmatrix} = \begin{pmatrix} \bar{x}^i \\ f^i \end{pmatrix}. \quad (2.3.27)$$

где

$$\bar{x}^i = \begin{cases} \overline{Best}, \text{если } i = 0; \\ \overline{MutChild}^i, \text{если } i \neq 0 \end{cases}$$

$$f^i = f(\bar{x}^i);$$

$DataOfForm$ не содержит каких-либо параметров относительно данного типа формирования нового поколения.

Итак, мы рассмотрели используемые варианты формирования нового поколения. Составим вектор параметров оператора формирования нового поколения:

$$DataOfForm = (TypeOfForm). \quad (2.3.28)$$

Вектор параметров генетического алгоритма

Как мы помним вектор параметров сГА состоит из двух частей:

$$Parameters = \begin{pmatrix} ParametersOfBinaryGA \\ ParametersOfConvertingIntoBinaryGA \end{pmatrix}.$$

Распишем два этих компонента на основании предыдущего материала.

Вначале рассмотрим $ParametersOfBinaryGA$, компоненты которого определяют параметры стандартного генетического алгоритма на бинарных строках:

$$ParametersOfBinaryGA = \begin{pmatrix} CountOfFitness \\ TypeOfSel \\ T \\ TypeOfCros \\ TypeOfMutation \\ TypeOfForm \end{pmatrix}.$$

Полагая, что в стандартном генетическом алгоритме при турнирной селекции размер турнира равен $T = 2$, то можно сократить вектор:

$$ParametersOfBinaryGA = \begin{pmatrix} CountOfFitness \\ TypeOfSel \\ TypeOfCros \\ TypeOfMutation \\ TypeOfForm \end{pmatrix}, \qquad (2.3.29)$$

где

$CountOfFitness \in \mathbb{N}$;

$TypeOfSel \in \{ProportionalSelection; RankSelection; TournamentSelection\}$;

$TypeOfCros \in \{SinglepointCrossover; TwopointCrossover; UniformCrossover\}$;

$$TypeOfMutation \in \begin{Bmatrix} WeakFor\_n, \\ AverageFor\_n, \\ StrongFor\_n, \end{Bmatrix};$$

$$TypeOfForm \in \begin{Bmatrix} OnlyOffspringGenerationForming, \\ OnlyOffspringWithBestGenerationForming \end{Bmatrix}.$$

Так как $\mu(TypeOfSel) = 3$, $\mu(TypeOfCros) = 3$, $\mu(TypeOfMutation) = 3$, $\mu(TypeOfForm) = 2$, то при фиксированных $CountOfFitness$ и N (что обычно делают при исследованиях генетического алгоритма) число различных вариантов настроек генетического алгоритма равно:

$CountOfSettings = \mu(TypeOfSel) \cdot \mu(TypeOfCros) \cdot \mu(TypeOfMutation) \cdot \mu(TypeOfForm) = 54$.

Это может быть использовано для определения эффективных настроек генетического алгоритма.

Теперь рассмотрим вектор $ParametersOfConvertingIntoBinaryGA$, определяющий преобразование задачи вещественной оптимизации в задачу бинарной оптимизации.

$$ParametersOfConvertingIntoBinaryGA = \begin{pmatrix} TypOfConverting \\ NumberOfParts_i \end{pmatrix}, \quad (2.3.30)$$

где

$TypOfConverting \in \{IntConverting; GrayCodeConverting\}$;

$NumberOfParts_i \in \{1;\ 2; \dots\}$ $(i = \overline{1,n})$.

Рекомендации. Для некоторых параметров существуют рекомендации по выбору значений. Если выбирать не из предложенных вариантов, то алгоритм будет работать, но заданные величины не будут соответствовать реальным, которые используются внутри алгоритма (будут пересчитаны).

$CountOfFitness = {k_1}^2$, где $k_1 \in \mathbb{N}$, например, $k_1 = 100$. (2.3.31)

$$NumberOfParts_i = 2^{(k_2)_i} - 1, \quad (2.3.32)$$

где $(k_2)_i \in \mathbb{N}$, например, $(k_2)_i = 14$ $(i = \overline{1,n})$.

Также предлагается способ определения конкретных значений $NumberOfParts_i$, а точнее $(k_2)_i$. Для задач вещественной оптимизации в качестве одного из параметра выступает задаваемая точность вычислений ε. Даная величина несет в себе следующий смысл: запуск генетического алгоритма считается удачным, если найденное алгоритмом решение $\bar{x}_{subm\,ax}$ удовлетворяет условию:

$$|(\bar{x}_{subm\,ax})_i - (\bar{x}_{max})_i| \le \varepsilon, \quad (2.3.33)$$

где $(i = \overline{1,n})$.

Шаг дискретизации в генетическом алгоритме берется в 10 раз меньше ε, чтобы при переходе к задаче бинарной оптимизации ГА мог найти точку рядом с глобальным оптимумом. Итак:

$h_i \le \frac{\varepsilon}{10}$;

$\frac{Right_i - Left_i}{2^{(k_2)_i} - 1} \le \frac{\varepsilon}{10}$;

$$2^{(k_2)_i} \geq \frac{10(Right_i - Left_i)}{\varepsilon} + 1, \qquad (2.3.34)$$

где $(i = \overline{1,n})$, $(k_2)_i \in \mathbb{N}$.

Тогда $(k_2)_i$ определяется как минимальное натуральное число, которое удовлетворяет последнему неравенству (2.5.7). На практике выбор решается простым перебором натуральных чисел, начиная с 1. Это некритично с точки времени выполнения, так как перебор выполняется за всё время работы алгоритма только один раз, да и просмотреть нужно обычно не более пару десятков натуральных чисел начиная с 1.

Отметим, что мы не можем использовать только одно значение k_2, а нужно использовать целый вектор $(k_2)_i$ $(i = \overline{1,n})$. Это объясняется тем, что ε для всех вещественных координат одно и то же. Однако границы изменения каждой координаты могут быть различны, а, следовательно, разбиваться интервал каждой координаты при переходе к бинарной задаче оптимизации будет на разное число частей.

2. 4. Исследование эффективности стандартного генетического алгоритма

Автором было проведено исследование описанного стандартного генетического алгоритма на множестве тестовых задач при полном переборе всех возможных настроек алгоритма. Результаты исследования приведены в Приложении №1. На каждый из 4 критериев для каждой тестовой задачи были определены наилучшие и наихудшие настройки генетического алгоритма.

Объем вычислений целевой функции подбирался таким образом для критериев №2, №3, №4, чтобы надежность при лучших настройках в среднем находился в диапазоне от 0.5 до 0.8.

Таблица 2.4.1. Число вычислений целевой функции для решения задач оптимизации стандартным генетическим алгоритмом (сГА).

Задача	Число вычислений целевой функции	Число поколений
Сумма всех элементов бинарного вектора. Размерность n=20 (Главная задача)	196	14
Сумма всех элементов бинарного вектора. Размерность n=30 (Подзадача №2)	400	20
Сумма всех элементов бинарного вектора. Размерность n=40 (Подзадача №3)	576	24
Сумма всех элементов бинарного вектора. Размерность n=50 (Подзадача №4)	784	28
Сумма всех элементов бинарного вектора. Размерность n=60 (Подзадача №5)	1024	32
Сумма всех элементов бинарного вектора. Размерность n=70 (Подзадача №6)	1369	37
Сумма всех элементов бинарного вектора. Размерность n=80 (Подзадача №7)	1444	38
Сумма всех элементов бинарного вектора. Размерность n=90 (Подзадача №8)	1681	41
Сумма всех элементов бинарного вектора. Размерность n=100 (Подзадача №9)	2116	46
Сумма всех элементов бинарного вектора. Размерность n=200 (Подзадача №10)	4624	68
Эллиптический параболоид. Размерность n=2 (Главная задача)	361	19
Эллиптический параболоид. Размерность n=4 (Подзадача №2)	1225	35
Функция Розенброка. Размерность n=2 (Главная задача)	5041	71
Функция Ackley. Размерность n=2 (Подзадача №2)	289	17
Функция Ackley. Размерность n=4 (Подзадача №3)	1521	39
Функция Растригина. Размерность n=2 (Главная задача)	1024	32
Функция Растригина. Размерность n=3 (Подзадача №2)	3025	55
Функция Растригина. Размерность n=4 (Подзадача №3)	6084	78

Обозначим для удобства и уменьшения объема текста в работе задачи оптимизации номерами.

Таблица 2.4.1. Номера исследуемых задач оптимизации.

Задача	Номер задачи
Сумма всех элементов бинарного вектора. Размерность n=20 (Главная задача)	1

Сумма всех элементов бинарного вектора. Размерность n=30 (Подзадача №2)	2
Сумма всех элементов бинарного вектора. Размерность n=40 (Подзадача №3)	3
Сумма всех элементов бинарного вектора. Размерность n=50 (Подзадача №4)	4
Сумма всех элементов бинарного вектора. Размерность n=60 (Подзадача №5)	5
Сумма всех элементов бинарного вектора. Размерность n=70 (Подзадача №6)	6
Сумма всех элементов бинарного вектора. Размерность n=80 (Подзадача №7)	7
Сумма всех элементов бинарного вектора. Размерность n=90 (Подзадача №8)	8
Сумма всех элементов бинарного вектора. Размерность n=100 (Подзадача №9)	9
Сумма всех элементов бинарного вектора. Размерность n=200 (Подзадача №10)	10
Эллиптический параболоид. Размерность n=2 (Главная задача)	11
Эллиптический параболоид. Размерность n=4 (Подзадача №2)	12
Функция Розенброка. Размерность n=2 (Главная задача)	13
Функция Ackley. Размерность n=2 (Подзадача №2)	14
Функция Ackley. Размерность n=4 (Подзадача №3)	15
Функция Растригина. Размерность n=2 (Главная задача)	16
Функция Растригина. Размерность n=3 (Подзадача №2)	17
Функция Растригина. Размерность n=4 (Подзадача №3)	18
Функция условной оптимизации №1 (Главная задача)	19
Функция условной оптимизации №1 (Подзадача №2)	20

Обозначим для удобства и уменьшения объема текста в работе настройки стандартного генетического алгоритма номерами.

Таблица 2.4.3. Номера всех возможных настроек для стандартного генетического алгоритма для бинарных задач.

№	ПАРАМЕТРЫ АЛГОРИТМА
1	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки;
2	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
3	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;
4	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
5	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;

6	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
7	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки;
8	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
9	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;
10	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
11	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;
12	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
13	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки;
14	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
15	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;
16	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
17	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;
18	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
19	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация;

	Тип формирования нового поколения = Только потомки;
20	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
21	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;
22	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
23	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;
24	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
25	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки;
26	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
27	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;
28	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
29	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;
30	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
31	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки;
32	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
33	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание;

	Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;
34	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
35	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;
36	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
37	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки;
38	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация**;** Тип формирования нового поколения = Только потомки и копия лучшего индивида;
39	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;
40	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
41	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;
42	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
43	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки;
44	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
45	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;
46	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
47	Тип селекции = Турнирная селекция;

	Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;
48	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
49	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки;
50	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
51	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;
52	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;
53	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;
54	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;

Таблица 2.4.4. Номера всех возможных настроек для стандартного генетического алгоритма для вещественных задач.

№	ПАРАМЕТРЫ АЛГОРИТМА
1	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
2	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
3	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации =

	Стандартное представление целого числа – номер узла в сетке дискретизации;
4	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
5	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
6	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
7	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
8	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
9	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
10	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
11	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
12	Тип селекции = Пропорциональная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
13	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание;

	Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
14	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
15	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
16	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
17	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
18	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
19	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
20	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
21	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
22	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;

23	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
24	Тип селекции = Пропорциональная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
25	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
26	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
27	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
28	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
29	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
30	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
31	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
32	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация;

	Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
33	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
34	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
35	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
36	Тип селекции = Пропорциональная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
37	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
38	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
39	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
40	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
41	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;

42	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
43	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
44	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
45	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
46	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
47	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
48	Тип селекции = Ранговая селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
49	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
50	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
51	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего

	индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
52	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
53	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
54	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
55	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
56	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
57	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
58	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
59	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
60	Тип селекции = Ранговая селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;

61	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
62	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
63	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
64	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
65	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
66	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
67	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
68	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
69	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
70	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки;

	Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
71	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
72	Тип селекции = Ранговая селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
73	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
74	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
75	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
76	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
77	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
78	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
79	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
80	Тип селекции = Турнирная селекция;

	Тип скрещивания = Одноточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
81	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
82	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
83	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
84	Тип селекции = Турнирная селекция; Тип скрещивания = Одноточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
85	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
86	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
87	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
88	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
89	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки;

	Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
90	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
91	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
92	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
93	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
94	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
95	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
96	Тип селекции = Турнирная селекция; Тип скрещивания = Двуточечное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
97	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
98	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
99	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание;

	Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
100	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Слабая мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
101	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
102	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
103	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
104	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Средняя мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
105	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
106	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;
107	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида; Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартное представление целого числа – номер узла в сетке дискретизации;
108	Тип селекции = Турнирная селекция; Тип скрещивания = Равномерное скрещивание; Тип мутации = Сильная мутация; Тип формирования нового поколения = Только потомки и копия лучшего индивида;

	Тип преобразования задачи вещественной оптимизации в бинарной оптимизации = Стандартный рефлексивный Грей-код;

Какие выводы можно сделать, проанализировав таблицы из Приложения 1?

Невозможно выбрать для любой задачи один вариант лучших или худших настроек стандартного генетического алгоритма: их почти всегда больше 1, если сравнивать по критерию Вилкоксона.

Ниже приведена таблица, по которой был сделан соответствующий вывод.

Таблица 2.4.5. Количество лучших и худших настроек для различных тестовых задач по критерию №2.

Задача	**Количество лучших настроек**	**Количество худших настроек**
Сумма всех элементов бинарного вектора. Размерность n=20	7	9
Сумма всех элементов бинарного вектора. Размерность n=30	3	16
Сумма всех элементов бинарного вектора. Размерность n=40	1	23
Сумма всех элементов бинарного вектора. Размерность n=50	3	28
Сумма всех элементов бинарного вектора. Размерность n=60	2	30
Сумма всех элементов бинарного вектора. Размерность n=70	3	30
Сумма всех элементов бинарного вектора. Размерность n=80	5	38
Сумма всех элементов бинарного вектора. Размерность n=90	6	42
Сумма всех элементов бинарного вектора. Размерность n=100	4	42
Сумма всех элементов бинарного вектора. Размерность n=200	2	43
Эллиптический параболоид. Размерность n=2	10	8
Эллиптический параболоид. Размерность n=4	2	58
Функция Розенброка. Размерность n=2	2	11
Функция Ackley. Размерность n=2	6	15
Функция Ackley. Размерность n=4	4	47
Функция Растригина. Размерность n=2	8	7
Функция Растригина. Размерность n=3	9	30
Функция Растригина. Размерность n=4	4	49

Из этой таблицы можно сделать также следующий вывод.

Число лучших настроек чаще всего в несколько раз меньше числа худших настроек стандартного генетического алгоритма при условии,

что объем вычислений целевой функции такой, что надежность при лучших настройках в среднем находился в диапазоне от 0.5 до 0.8. И при увеличении размерности дисбаланс увеличивается.

В выводе оговаривается объем вычислений целевой функции, так как возможны крайние ситуации: если объем вычислений целевой функции очень мал, то все настройки будут давать очень плохие результаты; если объем очень велик, то в большинстве настройки будут одинаковы хороши.

Построим график для функции «Сумма всех элементов бинарного вектора» числа худших настроек в зависимости от размерности.

Данный график иллюстрирует относительный характер поведения данного параметра, что свидетельствует о том, что при увеличении размерности набор хороших настроек резко уменьшается, а значит «угадать» хорошие настройки сложнее.

Рисунок 2.4.1. Количество худших настроек при увеличение размерности задачи.

При увеличении размерности количество худших настроек увеличивается при условии, что объем вычислений целевой функции такой, что надежность при лучших настройках в среднем находился в диапазоне от 0.5 до 0.8. Количество же лучших настроек обычно остается на приблизительно одинаковом уровне.

При увеличении размерности увеличивается дифференциация вариантов настроек генетического алгоритма по надежности.

В качестве примера приведем график надежности генетического алгоритма на задаче «Сумма элементов бинарного вектора» при разной размерности ($n = 20$ и $n = 200$). Точки на графике соответствуют различным настройкам алгоритма. При этом на каждой кривой все точки отсортированы в порядке возрастания, то есть настройки для конкретной точки по сои Ox разные.

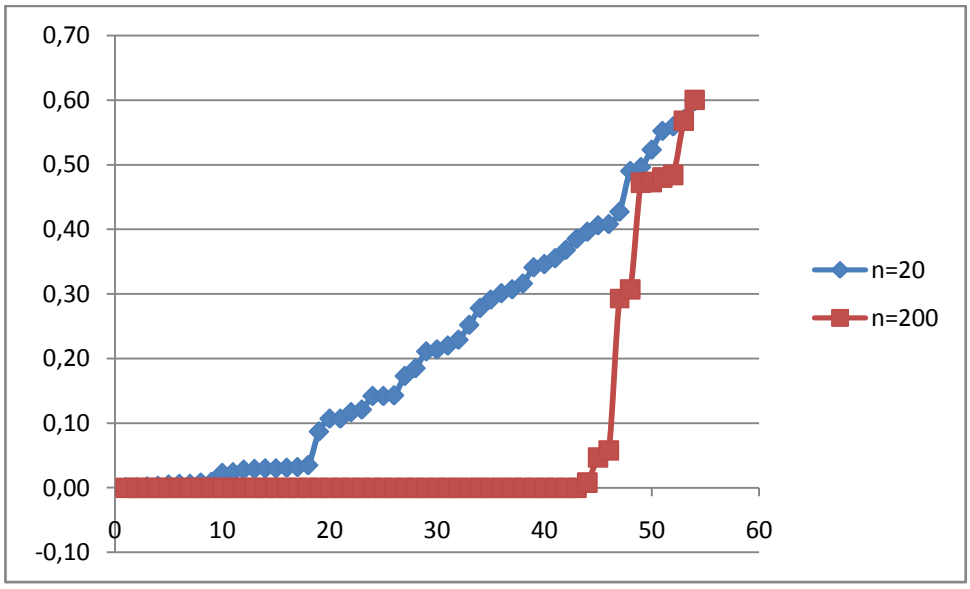

Рисунок 2.4.2. Сравнение общей картины настроек генетического алгоритма при разной размерности задачи «Сумма элементов бинарного вектора» (По оси Ox данные на каждой кривой отсортированы).

Если мы посмотрим на неотсортированный график, где каждй точки на оси Ox будут соответствовать одна и та же настройка генетичсекого алгоритм, то увидим, что:

При увеличении размерности задачи хорошие настройки (с высоким значением надежности по сравнению с другими настройками) остаются хорошими, а плохие (с низким значением надежности по сравнению с другими настройками) остаются плохими.

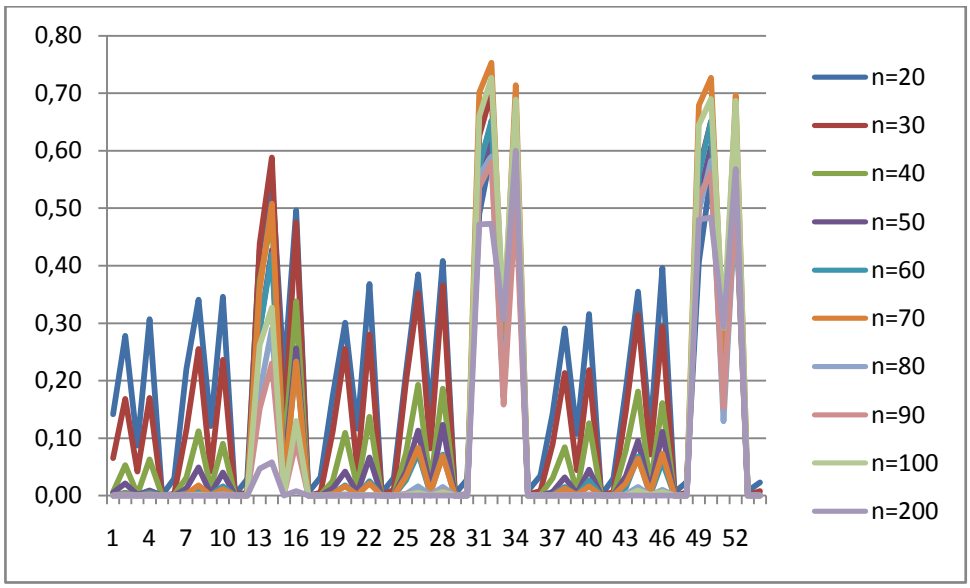

Рисунок 2.4.23 Сравнение общей картины настроек генетического алгоритма при разной размерности задачи «Сумма элементов бинарного вектора».

Для одной и той же задачи по разным критериям множество лучших и худших настроек перекрывается, но не всегда совпадает.

Приведем примеры:

Таблица 2.4.6. Худшие и лучшие настройки для задачи Сумма всех элементов бинарного вектора по трем критериям (Главная задача).

Главная задача. Сумма всех элементов бинарного вектора.			
№	Критерий 2	Критерий 2	Критерий 3
5	Worst	Worst	Worst
11	Worst	Worst	Worst
14	Best	Best	Best
16	Best	Best	Best
17	Worst		
23	Worst		
29	Worst		
31	Best	Best	Best
32	Best	Best	Best
34	Best	Best	Best
35	Worst		
41	Worst	Worst	Worst
47	Worst	Worst	Worst
50	Best	Best	Best
52	Best	Best	Best
53	Worst		

Таблица 2.4.7. Худшие и лучшие настройки для задачи «Функция Розенброка» (Главная задача).

Главная задача. Функция Розенброка.			
№	Критерий 2	Критерий 2	Критерий 3
10	Worst		
14	Worst		
27			Best
37		Worst	Worst
38	Worst	Worst	Worst
40		Worst	
43			Best
49		Worst	
50	Worst	Worst	Worst
55		Best	Best
62	Worst		
66	Worst		
67	Best	Best	Best
73		Worst	Worst
74	Worst	Worst	Worst
76		Worst	
79		Best	Best
85	Worst	Worst	
86	Worst	Worst	Worst
91		Best	Best
98	Worst		
102	Worst		
103	Best	Best	Best

Не существует таких настроек генетического алгоритма, которые бы для всех задач были бы лучшими или худшими.

Таблица 2.4.8. Лучшие и худшие настройки по критерию №2 (надежность)

Лучшие и худшие настройки								
	Функция Растригина			Эллиптический параболоид		Функция Ackley		Функция Розенброка
№	n=2	n=3	n=4	n=2	n=4	n=2	n=4	n=2
1					Worst		Worst	
2					Worst		Worst	
3					Worst			
4					Worst			
5			Worst		Worst		Worst	

6			Worst		Worst		Worst	
7								
8		Best			Worst			
9		Worst	Worst	Worst	Worst	Worst	Worst	
10		Worst	Worst	Worst	Worst	Worst	Worst	Worst
11			Worst		Worst		Worst	
12			Worst		Worst		Worst	
13					Worst		Worst	
14			Worst		Worst		Worst	Worst
15					Worst			
16					Worst			
17			Worst		Worst		Worst	
18			Worst		Worst		Worst	
19					Worst			
20					Worst			
21	Worst	Worst	Worst	Worst	Worst	Worst	Worst	
22		Worst	Worst		Worst		Worst	
23			Worst		Worst		Worst	
24			Worst		Worst		Worst	
25			Worst		Worst		Worst	
26		Worst	Worst		Worst		Worst	
27								
28					Worst			
29		Worst	Worst	Worst	Worst		Worst	
30		Worst	Worst		Worst		Worst	
31					Worst			
32					Worst			
33	Worst	Worst	Worst	Worst	Worst	Worst	Worst	
34		Worst	Worst	Worst	Worst	Worst	Worst	
35		Worst	Worst		Worst		Worst	
36			Worst		Worst		Worst	
37								
38								Worst
39								
40								
41								
42								
43	Best	Best		Best		Best		
44	Best	Best	Best					
45	Worst	Worst	Worst		Worst	Worst	Worst	
46		Worst	Worst		Worst		Worst	
47			Worst		Worst		Worst	
48					Worst		Worst	
49								
50								Worst
51				Best			Best	

52								
53								
54								
55	Best	Best		Best		Best		
56	Best	Best	Best					
57	Worst	Worst	Worst		Worst		Worst	
58		Worst	Worst		Worst	Worst	Worst	
59			Worst		Worst		Worst	
60					Worst		Worst	
61			Worst					
62		Worst	Worst					Worst
63				Best	Best	Best	Best	
64								
65			Worst					
66		Worst	Worst		Worst			Worst
67				Best		Best		Best
68								
69	Worst	Worst	Worst	Worst	Worst	Worst	Worst	
70		Worst	Worst		Worst	Worst	Worst	
71		Worst	Worst		Worst		Worst	
72			Worst		Worst		Worst	
73								
74								Worst
75								
76								
77								
78								
79	Best	Best		Best		Best		
80	Best	Best	Best					
81		Worst	Worst		Worst	Worst	Worst	
82		Worst	Worst		Worst	Worst	Worst	
83			Worst		Worst		Worst	
84							Worst	
85								Worst
86								Worst
87				Best			Best	
88								
89								
90								
91	Best	Best		Best				
92	Best	Best	Best					
93	Worst	Worst	Worst	Worst	Worst	Worst	Worst	
94		Worst	Worst		Worst	Worst	Worst	
95			Worst		Worst		Worst	
96					Worst		Worst	
97		Worst	Worst					

98		Worst	Worst					Worst
99				Best	Best		Best	
100								
101		Worst	Worst					
102		Worst	Worst		Worst			Worst
103				Best		Best		Best
104								
105	Worst	Worst	Worst		Worst	Worst	Worst	
106		Worst	Worst		Worst	Worst	Worst	
107		Worst	Worst		Worst		Worst	
108			Worst		Worst		Worst	

По вышеприведенной таблице, проанализировав, можно сделать такой вывод.

При увеличении размерности на большинстве рассмотренных тестовых задачах существуют такие настройки генетического алгоритма, которые на данном семействе тестовых задач всегда являются лучшими или худшими. Но это касается не любой настройки, являющейся лучшей или худшей.

Было также предложена модификация стандартного генетического алгоритма. Во всех исследованиях число вычислений целевой функции для критериев №2, №3, №4 для задач оптимизации бралось таким же, как и в исследованиях на выявление наилучших настроек стандартного генетического алгоритма (Таблица 2.4.1).

Требуется определить оптимальный размер турнира в турнирной селекции по критерию №3. Выбор производился из следующего множества:

1. $T = 2$ (используется в стандартном генетическом алгоритме);
2. $T = 3$;
3. $T = \frac{1}{5}N$;
4. $T = \frac{1}{4}N$;
5. $T = \frac{1}{3}N$;

6. $T = \frac{1}{2}N$;

7. $T = N$,

где N – размер популяции, а T – размер турнира.

Для каждой задачи оптимизации проводился запуск $n = 1000$ серий генетического алгоритма с турнирной селекцией. В каждой серии по 1 запуску генетического алгоритма с разными размерами турнира. Остальные параметры генетического алгоритма (за исключением числа вычислений целевой функции) выбираются в каждой серии случайным образом. В итоговой таблице для каждого размера турнира показана доля серий, в которых соответствующий размер турнира был наилучшим. Нужно отметить, что сумма долей в столбцах не равна 1, так как возможны случаи, когда одинаковую ошибку показывают несколько размеров турнира.

Результаты оказались не теми, которые предполагалось получить. На таблице закрашены лучшие варианты.

Таблица 2.4.9. Эффективность генетического алгоритма по критерию №3 (ошибка по аргументу) при различном размере турнира.

Турнир		$T = 2$	$T = 3$	$T = \frac{1}{5}N$	$T = \frac{1}{4}N$	$T = \frac{1}{3}N$	$T = \frac{1}{2}N$	$T = N$
Номер задачи	1	0,229	0,472	0,202	0,454	0,545	0,663	0,717
	2	0,219	0,472	0,549	0,632	0,647	0,758	0,78
	3	0,14	0,373	0,495	0,605	0,656	0,726	0,724
	4	0,127	0,378	0,547	0,607	0,663	0,728	0,735
	5	0,129	0,343	0,576	0,637	0,696	0,744	0,738
	6	0,15	0,402	0,627	0,667	0,746	0,799	0,768
	7	0,111	0,267	0,536	0,59	0,663	0,743	0,69
	8	0,098	0,244	0,554	0,591	0,639	0,742	0,685
	9	0,12	0,269	0,624	0,664	0,748	0,758	0,749
	10	0,093	0,19	0,557	0,602	0,634	0,677	0,494
	11	0,027	0,098	0,099	0,132	0,26	0,317	0,399
	12	0,001	0,025	0,181	0,17	0,226	0,319	0,407
	13	0,143	0,144	0,196	0,231	0,203	0,165	0,155
	14	0,025	0,097	0,086	0,149	0,172	0,292	0,349
	15	0,002	0,041	0,204	0,245	0,295	0,364	0,397
	16	0,106	0,154	0,214	0,23	0,247	0,331	0,314
	17	0,097	0,18	0,224	0,241	0,299	0,332	0,318
	18	0,115	0,168	0,27	0,306	0,336	0,335	0,291

Анализируя таблицу можно сделать несколько выводов:

Размер турнира равный $T = 2$ или $T = 3$ не является оптимальным.

На большом множестве задач оптимальным размером турнира является величина равная половине размера популяции.

2. 6. Выводы

Во второй главе рассмотрен стандартный генетический алгоритм и проведены исследования по сравнению его эффективности, таблицы которых приведены в Приложении 1.

Было установлено, что не существует таких настроек, генетического алгоритма, которые бы лучшими для всех алгоритмов. Также было установлено, что обычно рекомендуемый размер турнира равный 2 не является оптимальным.

Определены особенности поведения настроек генетического алгоритма для различных задач при различной размерности.

Глава 3. Самонастраивающийся генетический алгоритм

3. 1. Задача оптимального генетического алгоритма

$$\begin{pmatrix} \bar{x}_{subm\ ax} \\ f(\bar{x}_{subm\ ax}) \end{pmatrix} = GeneticAlgorithm \begin{pmatrix} X \\ f(\bar{x}) \\ g_i(\bar{x}) \\ h_j(\bar{x}) \\ Parameters \end{pmatrix}$$

Считаем, что в данный момент нас не интересует нахождение значения $f(\bar{x}_{subm\ ax})$. Параметры $X, f(x), g_i(\bar{x}), h_j(\bar{x})$ и $CountOfFitness$ задаются условиями задачи, поэтому считаем их фиксированными величинами. Введем обозначение:

$$Parameters^* = Parameters \cap \{CountOfFitness\}. \qquad (3.1.1)$$

Вектор $Parameters^*$ является варьируемым. Значит, для конкретной решаемой задачи оптимизации мы можем записать:

$\bar{x}_{subm\ ax} = GeneticAlgorithm(Parameters^*)$, где $\bar{x}_{subm\ ax} \in X$. (3.1.2)

Желательно, чтобы выполнялось условие:

$$|\|\bar{x}_{subm\ ax}\| - \|\bar{x}_{max}\|| = 0. \quad (3.1.3)$$

То есть мы вводим требование того, что множество X было нормируемым. В работе рассматриваются множество бинарных векторов и вещественных векторов. Поэтому в качестве нормы можно использовать критерий № 2 (надежность) и №3 (ошибка по аргументу).

Так как генетический алгоритм является стохастической процедурой, то x_{submax} – случайная величина. Поэтому требование (3.1.3) перепишем как

$$|\mathrm{M}(\|\bar{x}_{submax}\|) - \|\bar{x}_{max}\|| = 0, \quad (3.1.4)$$

где M – математическое ожидание нормы случайной величины.

Пусть D – допустимая область изменения вектора параметров генетического алгоритма $Parameters^*$.

Отсюда можно сформулировать задачу оптимизации вида:

$$Parameters^*_{min} = \arg\min_{P \in D} |\mathrm{M}(\|GeneticAlgorithm(P)\|) - \|x_{max}\|| \quad (3.1.5)$$

Данная задача возникает на том основании, что для каждой конкретной задачи оптимизации вектор $Parameters^*_{min}$ различный, как было показано выше.

Усложним поставленную задачу (3.1.4). По схеме ГА можно видеть, что алгоритм представляет собой итеративную процедуру. Значит, мы можем записать (3.1.4) в виде:

$$ArrayP_{min} = \arg\min_{AP \in AD} \left|\mathrm{M}\left(\left\|GeneticAlgorithm_{adap}\,(AP)\right\|\right) - \|x_{max}\|\right| \quad (3.1.6)$$

Здесь

$$AD = \{ap: ap = (P_1, \dots, P_{M-1})^{\mathrm{T}}, P_i \in D, i = \overline{1, M-1}\}, \quad (3.1.7)$$

$ArrayP_{min} \in AD$.

То есть, представлена такая модель генетического алгоритма $GeneticAlgorithm_{adap}$, структура и операторы которого совпадают с

соответствующим генетическим алгоритмом на том же множестве X, но вектор параметров алгоритма изменяется на каждом поколении. Такой вариант генетического алгоритма будем называть адаптивным генетическим алгоритмом.

На практике вместо вектора $ArrayP_{min}$ будем получать вектор $ArrayP_{submin}$, который не должен сильно отличаться от $ArrayP_{min}$ по эффективности относительно критерия из (3.1.5).

Естественно предположить, что элементы вектора определяются по некоторому закону:

$$P_i = \gamma(P_1, \dots P_{i-1}, Fitness_1, \dots, Fitness_{i-1}),\ i = \overline{2, M-1}. \qquad (3.1.8)$$

В данном случае, вектор параметров будет изменяться на основе информации о значениях параметров и состояниях популяции на предыдущих поколениях. Важно отметить, что для работы алгоритма требуется начальное приближение P_1.

3. 2. Самонастраивающийся генетический алгоритм

На основании выше поставленной задачи адаптивного генетического алгоритма и исследований стандартного генетического алгоритма был предложено несколько вариантов самонастраивающегося алгоритма. Многие из них показали низкую эффективность. Ниже приводится схема наиболее удачного варианта.

Отметим, что по сравнению со стандартным генетическим алгоритмом изменению на данный момент подвергся только часть, связанная с решением задачи бинарной оптимизации. Поэтому общая схема алгоритма почти тождественна схеме работы сГА. Данная особенность определила, что в самонастраивающемся генетическом алгоритме не подлежат автоматической настройке параметры, ответственные за преобразования задачи вещественной оптимизации в задачу бинарной оптимизации. А именно это параметр:

$$TypeOfForm \in \begin{Bmatrix} OnlyOffspringGenerationForming, \\ OnlyOffspringWithBestGenerationForming \end{Bmatrix}$$

Поэтому самонастраивающийся алгоритм на вещественных и бинарных строках будет иметь 1 настраиваемый пользователем параметр.

Распишем алгоритм $SelfAdaptivGeneticAlgorithm$:

- **Начало алгоритма**.
- **Анализ типа задачи**. Если X представляет собой множество вещественных векторов, то переходим к следующему шагу, иначе сразу переходим к шагу «Выполнение стандартного генетического алгоритма на бинарных строках».
- **Преобразование задачи вещественной оптимизации в задачу бинарной оптимизации**:

$$\begin{pmatrix} X_B \\ f_B(\bar{x}_B) \\ g_{i\,B}(\bar{x}_B) \\ h_{j\,B}(\bar{x}_B) \end{pmatrix} = ConvertingIntoBinaryGA \begin{pmatrix} X \\ f(\bar{x}) \\ g_i(\bar{x}) \\ h_j(\bar{x}) \\ ParametersOfConvertingIntoBinary \\ ParametersOfBinaryGA \end{pmatrix}.$$

- **Выполнение самонастраивающегося генетического алгоритма на бинарных строках**. Если решается задача бинарной оптимизации,.

$$A = \begin{cases} \begin{pmatrix} X \\ f(\bar{x}) \\ g_i(\bar{x}) \\ h_j(\bar{x}) \\ ParametersOfBinaryGA \end{pmatrix}, \text{если } X - \text{множество бинарных векторов;} \\ \begin{pmatrix} X_B \\ f_B(\bar{x}_B) \\ g_{i\,B}(\bar{x}_B) \\ h_{j\,B}(\bar{x}_B) \\ ParametersOfBinaryGA \end{pmatrix}, \text{если } X - \text{множество вещественных векторов.} \end{cases}$$

$$\begin{pmatrix} \bar{x}_{subm\,ax\,B} \\ f(\bar{x}_{subm\,ax})_B \end{pmatrix} = BinarySelfAdaptivGeneticAlgorithm\,(A).$$

- **Тип задачи**. Используем информацию, полученную на этапе «Анализ типа задачи». Если X представляет собой множество вещественных

векторов, то переходим к следующему шагу, иначе **возвращаем** в качестве результата работы алгоритма $SelfAdaptivGeneticAlgorithm$ вектор $\left(\bar{x}_{subm\ ax\ B}; f(\bar{x}_{subm\ ax})_B\right)^{\mathrm{T}}$.

- **Преобразование бинарного решения в вещественное.** Преобразуем полученное бинарное решение, полученное в ходе выполнения $BinarySelfAdaptivGeneticAlgorithm$ в вещественное, которое требуется в качестве решения задачи оптимизации, и **возвратим** его:

$$\begin{pmatrix} \bar{x}_{subm\ ax} \\ f(\bar{x}_{subm\ ax}) \end{pmatrix} = BinaryToReal \begin{pmatrix} \bar{x}_{subm\ ax\ B} \\ f(\bar{x}_{subm\ ax})_B \\ ParametersOfBinaryGA \\ ParametersOfConvertingIntoBinary \end{pmatrix}.$$

- **Конец алгоритма.**

Заметим, что самонастраивающийся генетический алгоритм не имеет вектора настраиваемых параметров, а сама схема использует операторы стандартного генетического алгоритма.

Вектор $ParametersOfConvertingIntoBinary$ по своей структуре совпадает с вектором $ParametersOfConvertingIntoBinaryGA$ из стандартного генетического алгоритма, поэтому мы не будем его здесь расписывать.

Распишем самонастраивающийся генетический алгоритм на бинарных строках $BinarySelfAdaptivGeneticAlgorithm$:

- **Начало алгоритма.**
- **Инициализация популяции**. Создается массив индивидов:

$Population = Initialization(X)$,

где $Population = \{\bar{x}^1; \bar{x}^2; \dots; \bar{x}^N\}$, $\bar{x}^i \in X$, $i = \overline{1, N}$.

- **Вычисление функции пригодности** индивидов. Создается массив:

$Fitness = \{f_{fit}(\bar{x}^1); f_{fit}(\bar{x}^2); \dots; f_{fit}(\bar{x}^N)\}$.

- **Запоминание лучшего индивида** и его значение целевой функции:

$\overline{Best} = \arg \max_{\bar{x} \in Population} f(\bar{x})$;

$BestFitness = \max_{\bar{x} \in Population} f(\bar{x})$.

- **Создание дополнительных массивов**:

$F_k^{Sel} = 0, N_k^{Sel} = 0, k = \overline{1,3}$.

$F_k^{Cros} = 0, N_k^{Cros} = 0, k = \overline{1,3}$.

$F_k^{Muta\,tion} = 0, N_k^{Mutation} = 0, k = \overline{1,3}$.

- Цикл от $I = 0$ до $I = M - 1$
    - **Начало**
    - **Определение значений** параметров $TypeOfSel$, $TypeOfCros, TypeOfMutation$.
    - Цикл от $j = 1$ до N
        - **Начало**
        - **Селекция**. Выбрать двух родителей:

          $\overline{Parent}^1 = Selection(Population, Fitness, DataOfSel)$;

          $\overline{Parent}^2 = Selection(Population, Fitness, DataOfSel)$, где

          $\overline{Parent}^1 \in X, \overline{Parent}^2 \in X$.
        - **Скрещивание**. Получение потомка из родителей:

          $\overline{Child}^j = Crossover(\overline{Parent}^1, \overline{Parent}^2, DataOfCros)$.
        - **Конец**
    - **Мутация** всех потомков.

      $MutChildPopulation =$

      $Mutation(ChildPopulation, DataOfMut)$,

      где

      $ChildPopulation = \{\overline{Child}^1; \dots; \overline{Child}^N\}, \overline{Child}^i \in X, i = \overline{1,N}$;

      $MutChildPopulation = \{\overline{MutChild}^1; \dots; \overline{MutChild}^N\}$,

      $\overline{MutChild}^i \in X, i = \overline{1,N}$.
    - **Если** $I = \max\{7; 0.1 \cdot N\}$, то пересоздание популяции потомков:

      $MutChildPopulation = Initialization(X)$.
    - **Вычисление функции пригодности** потомков:

$$FitnessOfMutChild = \{f_{fit}(\overline{MutChild}^1); \dots; f_{fit}(\overline{MutChild}^N)\}.$$

- **Запоминание**. Если есть потомок лучше сохраненного лучшего индивида $\overline{Best}$, то заменяем его новым индивидом с его значением целевой функции:

$$\overline{Best} = \arg \max_{\bar{x} \in MutChildPopulation \cup \{\overline{Best}\}} f(\bar{x});$$

$$BestFitness = \max_{\bar{x} \in MutChildPopulation \cup \{\overline{Best}\}} f(\bar{x}).$$

- **Формирование нового поколения**. Из потомков и родителей формируется новое поколение, которое заменяет предыдущее поколение.

$$\begin{pmatrix} NewPopulation \\ NewFitness \end{pmatrix} = Forming \begin{pmatrix} Population \\ MutChildPopulation \\ Fitness \\ FitnessOfMutChild \\ DataOfForm \end{pmatrix};$$

$$Population = NewPopulation;$$

$$Fitness = NewFitness.$$

- **Уточнение значений** массивов F^{Sel}, F^{Cros}, $F^{Mutation}$, N^{Sel}, N^{Cros}, $N^{Mutation}$.
- **Конец**

- **Выдаем лучшего индивида** и его значение целевой функции, то есть возвращаем пару $\begin{pmatrix} \overline{Best} \\ BestFitness \end{pmatrix}$.

- **Конец алгоритма**.

Здесь,

N – размер популяции;

M – число поколений;

$DataOfSel$ – вектор параметров оператора селекции, аналогичный вектору параметров стандартного генетического алгоритма;

$DataOfCros$ – вектор параметров оператора скрещивания, аналогичный вектору параметров стандартного генетического алгоритма;

$DataOfMut$ – вектор параметров оператора мутации, аналогичный вектору параметров стандартного генетического алгоритма;

$DataOfForm$ – вектор параметров оператора формирования нового поколения, аналогичный вектору параметров стандартного генетического алгоритма.

$$M = 2 \cdot int\left(\sqrt{CountOfFitness}\right); \quad (3.2.1)$$

$$N = int\left(\frac{CountOfFitness}{M}\right).$$

Размер турнира для турнирной селекции:

$$T = \frac{N}{2}. \quad (3.2.2)$$

Тип формирования нового поколения:

$$TypeOfForm = OnlyOffspringWithBestGenerationForming. \quad (3.2.3)$$

Вектор всех изменяемых параметров стандартного генетического алгоритма на бинарных строках содержит только число вычислений целевой функции:

$$ParametersOfBinaryGA = (CountOfFitness). \quad (3.2.4)$$

Распишем два новых оператора в самонастраивающемся алгоритме.

Определение значений параметров $TypeOfSel$, $TypeOfCros$, $TypeOfMutation$:

- **Начало алгоритма**.
- **Если** $I < \max\{7; 0.1 \cdot N\}$, то
  - **Начало**
  - $TypeOfSel = Random(\{0, 1, 2\})$;
  - $TypeOfCros = Random(\{0, 1, 2\})$;

- $TypeOfMutation = Random(\{0, 1, 2\})$;
- **Конец**

- **Иначе**
  - **Начало**
  - $TypeOfSel = RankSelection4(\{0; 1;\ 2\}, F^{Sel})$;
  - $TypeOfCros = RankSelection4(\{0;\ 1; 2\}, F^{Cros})$;
  - $TypeOfMutation = RankSelection4\big(\{0;\ 1;\ 2\}, F^{Mutation}\ \big)$
  - **Конец**

- **Конец алгоритма**.

Здесь I – счетчик в цикле генетического алгоритма по поколениям.

То есть первые поколения значения выбираются случайно, а все последующие выбираются с помощью ранговой селекции на основании значений из массивов F^{Sel}, F^{Cros} и $F^{Mutation}$:

$$RankSelection4(X, F) = Random\big(\{\bar{x}_i | p_i\}\big),$$

где $p_i = \frac{Rank\ (f(\bar{x}_i))}{\sum_{j=1}^{n} Rank\ \left(f(\bar{x}_j)\right)}$, $\bar{x}_i \in X$, $f(\bar{x}_i) \in F$, $i = \overline{1, n}$.

$$Rank\big(f(\bar{x}_i)\big) = 4\frac{\sum_{j=1}^{n} NumberOfSorting\ \ (f(\bar{x}_i), Fitness) \cdot S\big(f(\bar{x}_i), f(\bar{x}_j)\big)}{\sum_{j=1}^{n} S\big(f(\bar{x}_i), f(\bar{x}_j)\big)},$$

где $S\left(f(\bar{x}_i), f\big(\bar{x}_j\big)\right) = \begin{cases} 1, \text{если } f(\bar{x}_i) = f\big(\bar{x}_j\big); \\ 0, \text{если } f(\bar{x}_i) \neq f\big(\bar{x}_j\big). \end{cases}$

Числа 0, 1, 2 соответствуют соответствующим вариантам операторов генетического алгоритма:

$TypeOfSel \in \{ProportionalSelection; RankSelection; TournamentSelection\} = \{0; 1; 2\}$;

$TypeOfCros \in \{SinglepointCrossover; TwopointCrossover; UniformCrossover\} = \{0; 1; 2\}$;

$$TypeOfMutation \in \begin{Bmatrix} WeakFor\_n, \\ AverageFor\_n, \\ StrongFor\_n, \end{Bmatrix} = \{0; 1; 2\}.$$

Уточнение значений массивов F^{Sel}, F^{Cros}, $F^{Mutatio\ n}$, N^{Sel}, N^{Cros}, $N^{Mutation}$.

- **Начало алгоритма**.

$$F^{Sel}_{TypeOfSel} = \frac{N^{Sel}_{TypeOfSel} \cdot F^{Sel}_{TypeOfSel} + MaxFitness}{N^{Sel}_{TypeOfSel} + 1};$$

$$N^{Sel}_{TypeOfSel} = N^{Sel}_{TypeOfSel} + 1;$$

$$F^{Cros}_{TypeOf\ Cros} = \frac{N^{Cros}_{TypeOf\ \ Cros} \cdot F^{Cros}_{TypeOf\ \ Cros} + MaxFitness}{N^{Cros}_{TypeOf\ \ Cros} + 1};$$

$$N^{Cros}_{TypeOf\ Cros} = N^{Cros}_{TypeOf\ Cros} + 1;$$

$$F^{Sel}_{TypeOf\ Mutation} = \frac{N^{Mutation}_{TypeOf\ \ Mutation} \cdot F^{Mutation}_{TypeOf\ \ Mutation} + MaxFitness}{N^{Mutation}_{TypeOf\ \ Mutation} + 1};$$

$$N^{Mutation}_{TypeOf\ Mutation} = N^{Mutation}_{TypeOf\ Mutation} + 1,$$

где $MaxFitness = \max\limits_{\bar{x} \in MutChildPopulation} f(\bar{x})$

- **Конец алгоритма**.

Мы используем в качестве значения индекса, например $TypeOfSel$, так как выше ввели соответствие между типами операторов и числами.

Итак, распишем основные идеи, закладываемые в самонастраивающийся алгоритм:

1. В первые поколения ($\max\{7; 0.1 \cdot N\}$) типы операторов выбираются случайно.
2. После этих поколений популяция сбрасывается, чтобы не использовать популяцию, которая могла из-за неправильных значений типов операторов стать непригодной для эффективной работы ГА.
3. Типы операторов выбираются с помощью ранговой селекции с увеличенными значениями рангов в 4 раза на основании массивов, содержащих среднее значение наилучшего значения

целевой функции, при условии, что в поколении был выбран соответствующий тип оператора.

4. Тип формирования нового поколения всегда равен: только потомки и лучший индивид.
5. В турнирной селекции турнир равен половине популяции.
6. Число поколений в два раза меньше размера популяции.

3. 3. Исследования эффективности самонастраивающегося генетического алгоритма

Было проведено исследование эффективности самонастраивающегося генетического алгоритма на множестве тестовых задач, которые рассматривались при исследовании стандартного генетического алгоритма.

Объем вычислений целевой функции во всех исследованиях равен объему вычислений целевой функции, который использовался в стандартном генетическом алгоритме.

В результате были получены результаты, часть которых приведена ниже в таблице. В таблице приведены усредненные значения по 1000 запускам. Для стандартного генетического алгоритма значения берутся из Приложения №1.

Таблица 3.3.1. Эффективность по надежности на тестовых задачах вещественной оптимизации стандартного и самонастраивающегося алгоритма.

№ задачи	Стандартный генетический алгоритм		Самонастраивающийся генетический алгоритм	
	При худшей настройке	При лучшей настройке	При худшей настройке	При лучшей настройке
11	0,01	0,694	0,676	0,811
12	0	0,632	0,637	0,834
13	0,03	0,664	0,335	0,582
14	0,012	0,552	0,535	0,69
15	0	0,802	0,801	0,897
16	0,032	0,627	0,495	0,657
17	0	0,606	0,417	0,636

18	0	0,577	0,31	0,635
19	0,015	0,591	0,141	0,405
20	0,024	0,536	0,172	0,366

Как видно из таблицы, почти на всех задачах вещественной безусловной оптимизации самонастраивающийся генетический алгоритм с лучшими настройками работает лучше, чем стандартный генетический алгоритм с лучшими настройками, и на всех задачах самонастраивающийся генетический алгоритм с худшими настройками работает лучше, чем стандартный генетический алгоритм с лучшими настройками.

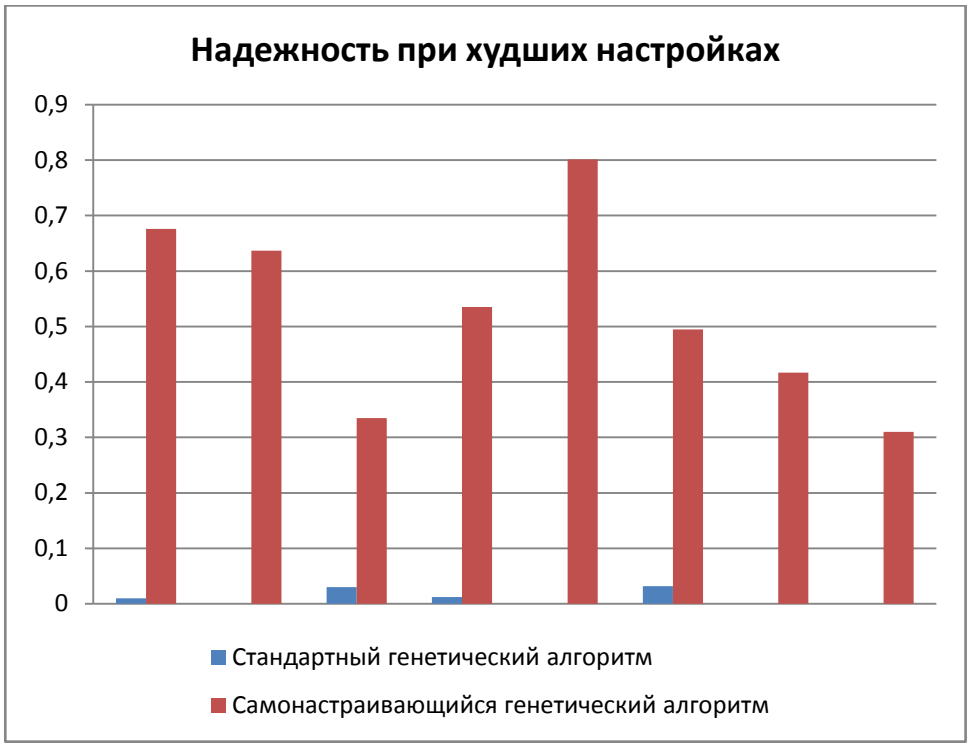

Рисунок 3.3.1. Надежность на разных тестовых задачах при худших настройках самонастраивающегося и стандартного генетического алгоритма.

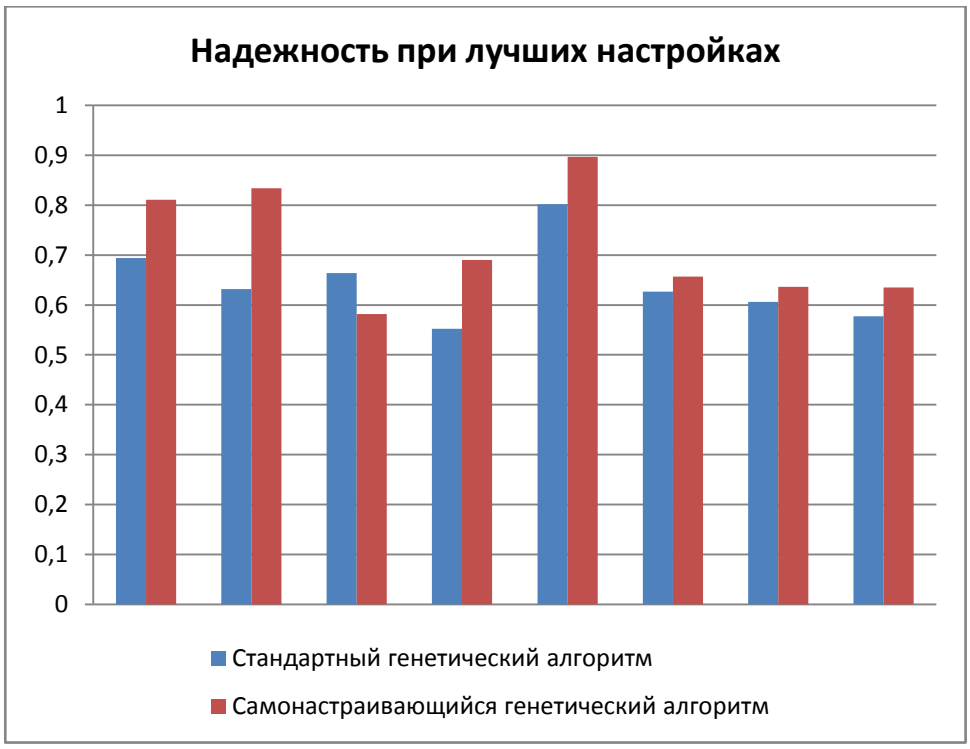

Рисунок 3.3.2. Надежность на разных тестовых задачах при худших настройках самонастраивающегося и стандартного генетического алгоритма.

На задачах бинарной оптимизации, как показали исследования, самонастраивающийся алгоритм не превосходит стандартный генетический алгоритм. Это будет выяснено в последующих исследованиях.

Далее рассмотрим пример применения самонастраивающегося алгоритма для решения практической задачи.

3. 3. Система управления на основе коллектива нечетких правил

Пусть у нас есть объект управления с n входами и m выходами.

Пусть $\bar{y}(t_i) = \{y_1, \dots, y_m\}$ – вектор выходных наблюдаемых параметров, а $\bar{u}(t_i) = \{u, \dots, u_n\}$ – вектор входных управляемых параметров в момент t_i $(i = 1,2,3, \dots)$. Требуется к времени T перевести систему из состояния $\bar{y}(0)$ в состояние $\bar{y}^*(T)$.

В качестве системы управления будет выступать система на нечеткой логике с конечным числом правил. Для ее формирования используется предлагаемый в работе генетический алгоритм. Длина хромосомы равна

$4(n+m)k+4m$, где k – число нечетких правил. Способ декодирования вещественной строки приводится в работе [11].

В качестве конкретного примера для объекта управления был взята система «автомобиль с прицепом». Требуется задним ходом перевести машину из одного положения в другое, то сеть осуществить процесс автопарковки.

Для моделирования использовалась «велосипедная модель», то есть рассматривается только одно воображаемое колесо на каждой оси посередине (вместо двух настоящих колес).

На рисунке показана принципиальная схема велосипедной модели.

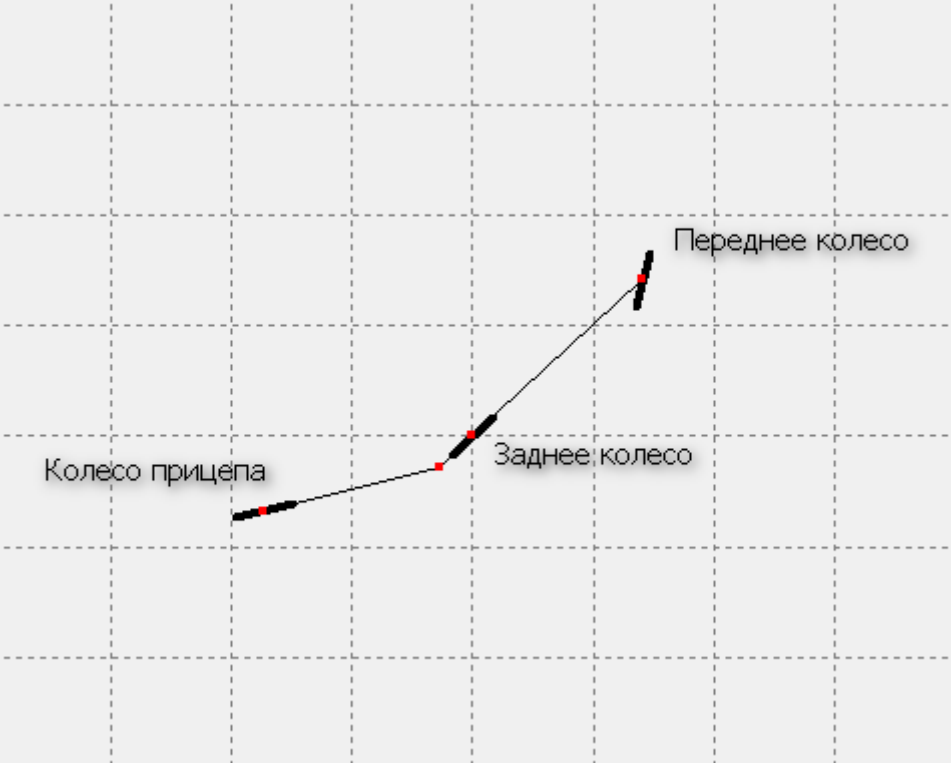

Рисунок 3.3.1. Принципиальная схема системы «автомобиль с прицепом»

В модели не учитываются динамические свойства системы – только кинематические. Движение прицепа не влияет на движение автомобиля. Имеется максимальный угол поворота прицепа относительно автомобиля. При достижении его система «автомобиль с прицепом» останавливается. Управление осуществляется только одним параметром $(n=1)$: угол

поворота передних коле. Скорость заднего колеса является постоянной величиной.

Задается конечное положение автомобиля с прицепом. Относительно начального и конечного положения колеса трейлера задается отрезок прямой линии. Мы способны измерять в каждый момент времени 4 параметра ($m = 4$):

- Угол поворота прицепа относительно автомобиля.
- Угол между отрезком предполагаемой траектории и прицепом.
- Расстоянием между колесом прицепа и его конечным желаемым положением.
- Расстоянием между колесом прицепа до ближайшей точки на отрезке предполагаемой траектории.

Требуется перевести систему «автомобиль с прицепом» в конечное положение, с условием, чтобы угол прицепа относительно автомобиля в конечном положении не превышал указанного.

Как показали исследования, для построения нечеткой базы правил, способной управлять данной системой, требуется большое количество вычислительных ресурсов, и генетический алгоритм не справляется за отведенное ему время. Поэтому была предложена схема по генерированию коллектива небольших баз правил, которые могут осуществить управление в нешироком диапазоне.

Принцип действия предложенной системы управления заключается в том, что в каждый момент времени t_i осуществляется прогнозирование положения системы при использовании каждой небольшой базы правил. И в каждый момент времени выбирается база правил, которая по прогнозу даст минимальную ошибку.

Оказалось, что данная схема способна решить поставленную задачу, а генерирование простых баз правил осуществляется с более высокой эффективностью с помощью предложенного генетического алгоритма.

Использовались базы правил по 10 правил каждая (генетический алгоритм работал с хромосомами длины 216).

На рисунке ниже показаны траектории движения системы при выполнении автопарковки с помощью сгенерированного коллектива нечетких баз правил.

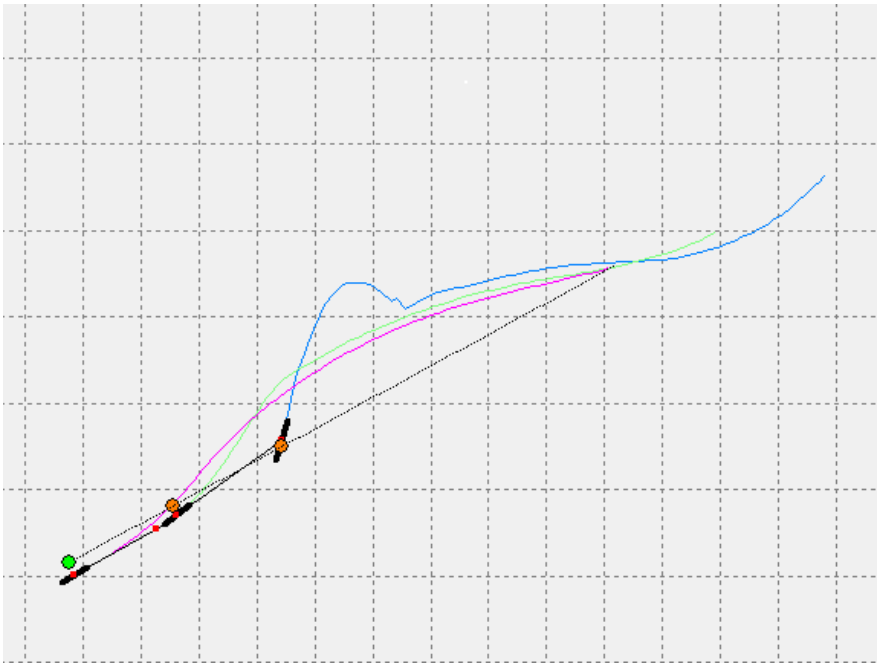

Рисунок 3.3.2. Траектория движения системы «автомобиль с прицепом»

3. 3. Выводы

В третьей главе был рассмотрен самонастраивающийся алгоритм. Было приведено описание операторов и основных идей, на которых он строится. Для решения задач вещественной оптимизации остается только один настраиваемый параметр, а для задач бинарной оптимизации настраиваемых параметров нет.

Для эффективного решения задачи вещественной оптимизации стандартным алгоритмом требуется $108 \cdot CountOfFitness$ вычислений целевой функции (так как не существует лучших одинаковых настроек для всех задач). Для самонастраивающегося требуется только $2 \cdot CountOfFitness$, то есть в 54 раза меньше.

Также было показана принципиальная возможность применения рассматриваемых автором алгоритмов для решения задачи формирования системы управления на основе коллективов нечетких правил.

Заключение

В результате диссертационного исследования был проведен анализ стандартного генетического алгоритма и самонастраивающегося алгоритма на множестве задач оптимизации. Было показана эффективность разработанного алгоритма. Также было проведено подробное исследование стандартного генетического алгоритма при полном рассмотрении всех возможных настроек. Ниже приведен список основных результатов диссертационного исследования:

1. Предложенный и задокументированный стандартный генетический алгоритм лишен параметров, выбор которых не оговаривается ни в алгоритме, ни в задаче оптимизации. Поэтому можно обосновано утверждать, что на тестовых задачах был проведен полный перебор возможных настроек.

2. Невозможно выбрать для любой задачи один вариант лучших или худших настроек стандартного генетического алгоритма: их почти всегда больше 1, если сравнивать по критерию Вилкоксона.

3. При увеличении размерности задачи увеличивается дифференциация вариантов настроек генетического алгоритма по надежности.

4. Для одной и той же задачи по разным критериям множество лучших и худших настроек перекрывается, но не всегда совпадает, то есть существуют такие настройки, которые являются лучшими или худшими по всем критериям, но существуют такие настройки, которые являются лучшими или худшими по одному критерию, но по другим не являются.

5. Не существует таких настроек генетического алгоритма, которые для всех задач были бы лучшими или худшими.

6. Число лучших настроек чаще всего в несколько раз меньше числа худших настроек стандартного генетического алгоритма при условии, что объем вычислений целевой функции такой, что надежность при лучших

настройках в среднем находился в диапазоне от 0.5 до 0.8. И при увеличении размерности дисбаланс увеличивается.

7. При увеличении размерности количество худших настроек увеличивается при условии, что объем вычислений целевой функции такой, что надежность при лучших настройках в среднем находился в диапазоне от 0.5 до 0.8. Количество же лучших настроек обычно остается на приблизительно одинаковом уровне.

8. При увеличении размерности задачи хорошие настройки (с высоким значением надежности по сравнению с другими настройками) остаются хорошими, а плохие (с низким значением надежности по сравнению с другими настройками) остаются плохими.

9. При увеличении размерности на большинстве рассмотренных тестовых задачах существуют такие настройки генетического алгоритма, которые на данном семействе тестовых задач всегда являются лучшими или худшими. Но это касается не любой настройки, являющейся лучшей или худшей.

10. Размер турнира равный $T = 2$ или $T = 3$ не является оптимальным для стандартного генетического алгоритма.

11. На большом множестве задач оптимальным размером турнира является величина равная половине размера популяции для стандартного генетического алгоритма.

12. Разработан самонастраивающийся генетический алгоритм на бинарных строках без настраиваемых параметров, который показывает высокую эффективность.

13. Показана применимость самонастраивающегося алгоритма на задачах условной оптимизации.

14. Почти на всех задачах вещественной безусловной оптимизации самонастраивающийся генетический алгоритм с лучшими настройками работает лучше, чем стандартный генетический алгоритм с лучшими

настройками, и на всех задачах самонастраивающийся генетический алгоритм с худшими настройками работает лучше, чем стандартный генетический алгоритм с лучшими настройками. Для эффективного решения задачи вещественной оптимизации стандартным алгоритмом требуется $108 \cdot CountOfFitness$ вычислений целевой функции (так как не существует лучших одинаковых настроек для всех задач). Для самонастраивающегося требуется только $2 \cdot CountOfFitness$, то есть в 54 раза меньше.

15. Разработан способ формирования системы управления объекта, основанной на композиции множества нечетких баз правил, генерируемой генетическим алгоритмом.

Список использованных источников:

1. Матвеев, М.Г. Модели и методы искусственного интеллекта. Применение в экономике: учеб. Пособие [Текст] / М.Г. Матвеев, А.С. Свиридов, Н.А. Алейникова. – М.: Финансы и статистика; ИНФРА-М, 2008. – 448с.
2. Разработка простого генетического алгоритма [Internet] http://www.wikiznanie.ru/ru-wz/index.php/Разработка_простого_генетического_алгоритма
3. Джонс, М. Т. Программирование искусственного интеллекта в приложениях [Текст] / М. Тим Джонс. – М.: ДМК Пресс, 2006. – 312с.
4. Код Грея [Internet] http://ru.wikipedia.org/wiki/Код_Грея
5. Оптимальное кодирование http://www.ie.tusur.ru/books/COI/page_07.htm
6. Rastrigin function [Internet] http://en.wikipedia.org/wiki/Rastrigin_function
7. The Ackley Problem [Internet] http://tracer.lcc.uma.es/problems/ackley/ackley.html
8. Сергиенко, А.Б. Генетический алгоритм. Стандарт. Часть I. Описание стандартного генетического алгоритма (сГА) [Internet] / А.Б. Сергиенко, П.В. Галушин, В.В. Бухтояров, Р.Б. Сергиенко, Е.А. Сопов, С.А. Сопов. – http://www.harrix.org/files/project_standart_ga/Geneticheskii_algoritm_Standart_v_1_5_Release_Candidate.pdf
9. Большев, Л.Н. Таблицы математиечкой статистики. [Текст] / Л.Н. Большев, Н.В. Смирнов. – М.: Наука. Главная редакция физико-математической литературы, 1983. – 416с.
10. Сергиенко, А.Б. Модель генетического алгоритма с самонастраивающимися параметрами [Текст] / А.Б. Сергиенко // Решетнёвские чтения. Материалы XI Международной научной

конференции, посвященной памяти генерального конструктора ракетно-космических систем академика М.Ф. Решетнёва (6-10 ноября 2007 года, г. Красноярск) / под ред. И. В. Ковалева; Сибирский государственный аэрокосмический университет. Красноярск, 2007.

11. Сергиенко, А.Б. Разработка адаптивного генетического алгоритма для решения задач оптимизации со сложной структурой генотипа [Текст] / А.Б. Сергиенко – Квалификационная работа бакалавра, 2008. – С.92
12. Yoshihiro, M. Agent Oriented Self Adaptive Genetic Algorithm. [Internet] / M. Yoshihiro, S. Naoki, Y. Keiichi, I. Minoru. – http://ito-lab.naist.jp/themes/pdffiles/031209.eiichi-t.cec2003.pdf

Список публикаций автора

1. Сергиенко, А.Б. О решении задач комбинаторной оптимизации методом генетического алгоритма и о сравнении эффективных настроек генетического алгоритма [Текст] / А.Б. Сергиенко // Молодёжь и наука: начало XXI века: материалы Всероссийской научно-технической конференции студентов, аспирантов и молодых учёных: в 3 ч. Ч. 2. – Красноярск: ИПЦ КГТУ, 2006. – С. 3.
2. Сергиенко, А.Б. О решении задач комбинаторной оптимизации методом генетического алгоритма и о сравнении эффективных настроек генетического алгоритма [Текст] / А.Б. Сергиенко // Решетнёвские чтения. Тезисы докладов X Всероссийской научной конференции, посвященной памяти Генерального конструктора ракетно-космических систем академика М.Ф. Решетнёва (8-10 ноября 2006 года, г. Красноярск). Редакционно-издательский отдел СибГАУ, Красноярск, 2006. – С. 266.
3. Сергиенко, А.Б. Особенности настройки генетических алгоритмов при решении задач оптимизации на перестановках [Текст] / А.Б. Сергиенко // VII Всероссийская конференция молодых ученых по математическому моделированию и информационным технологиям (с участием иностранных ученых). Программа и тезисы докладов, Институт вычислительных технологий СО РАН, г. Красноярск, 2006. – С. 67.
4. Сергиенко, А.Б. Модель генетического алгоритма с самонастраивающимися параметрами [Текст] / А.Б. Сергиенко // Решетнёвские чтения. Материалы XI Международной научной конференции, посвященной памяти генерального конструктора ракетно-космических систем академика М.Ф. Решетнёва (6-10 ноября 2007 года, г. Красноярск) / под ред. И. В. Ковалева; Сибирский государственный аэрокосмический университет. Красноярск, 2007. – С. 252-253.
5. Сергиенко, А.Б. Разработка автоматизированной информационно-аналитической системы поддержки принятии решения для задач

бюджетирования и оперативного управления предприятием [Текст] / Е.А. Сопов, В.Г. Жуков, В.Г. Королев, А.Б. Сергиенко // Решетнёвские чтения. Материалы XI Международной научной конференции, посвященной памяти генерального конструктора ракетно-космических систем академика М.Ф. Решетнёва (6-10 ноября 2007 года, г. Красноярск) / под ред. И. В. Ковалева; Сибирский государственный аэрокосмический университет. Красноярск, 2007. – С. 255.

6. Сергиенко, А.Б. Модель генетического алгоритма с самонастраивающимися параметрами [Текст] / А.Б. Сергиенко // VIII Всероссийская конференция молодых ученых по математическому моделированию и информационным технологиям. Программа и тезисы докладов, Институт вычислительных технологий СО РАН, г. Новосибирск, 2007. – С. 72-73.
7. Сергиенко, А.Б. Влияние циклического сдвига на эффективность генетического алгоритма предприятием [Текст] / А.Б. Сергиенко // Актуальные проблемы авиации и космонавтики. Тезисы Всероссийской научно-практической конференции. студентов, аспирантов и молодых специалистов (2–6 апреля 2007, г. Красноярск) : в 2 т. Т. 1. – Красноярск: Редакционно-издательский отдел СибГАУ, 2007. – С. 194.
8. Сергиенко, А.Б. Самонастраивающийся генетический алгоритм [Текст] / А.Б. Сергиенко // Актуальные проблемы авиации и космонавтики. Тезисы Всероссийской научно-практической конференции студентов, аспирантов и молодых специалистов (2008, г. Красноярск) : в 2 т. Т. 1. – Красноярск: Редакционно-издательский отдел СибГАУ, 2008. – С. 207.
9. Сергиенко, А.Б. Сравнение PSO и генетического алгоритмов оптимизации [Текст] / С.С. Бежитский, Е.А. Бежитская, А.Б. Сергиенко // Труды Конгресса по интеллектуальным системам и информационным технологиям «AIS-IT’09»: в 4 т. Т. 1. – М: Физматлит, 2009. – С. 7.