**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №4**

**по дисциплине «Алгоритмы и структуры данных»**

на тему «Исследование генетического алгоритма. Изучение различных кодировок генотипа»

**Вариант – 19**

**Студент гр. 23Б15-пу**

**Сурин И.С.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г**

Оглавление

[Цель работы 2](#_Toc181705025)

[Краткое описание алгоритма генетического алгоритма (ГА) 2](#_Toc181705026)

[Формализация задачи 5](#_Toc181705027)

[Рекомендации пользователя 6](#_Toc181705028)

[Рекомендации программиста 6](#_Toc181705029)

[Исходный код программы 6](#_Toc181705030)

[Описание контрольного примера 7](#_Toc181705031)

[Вывод 9](#_Toc181705032)

[Источники 9](#_Toc181705033)

# **Цель работы**

Целью данной лабораторной работы является изучение и анализ двух методов кодирования генотипа хромосом в генетических алгоритмах. Основное внимание уделяется сравнению их эффективности при решении задачи минимизации сложной функции.

# **Краткое описание алгоритма генетического алгоритма (ГА)**

Генетический алгоритм— это метод оптимизации, основанный на механизмах природной эволюции, таких как наследование, отбор, кроссинговер и мутация. ГА особенно полезен для задач с большим пространством поиска, где могут существовать множество локальных экстремумов.

Основные этапы работы генетического алгоритма:

1. **Инициализация популяции**: Создается начальная популяция из случайных решений (хромосом). Каждая хромосома состоит из генов, которые могут быть представлены как числовые значения или двоичные строки.
2. **Оценка фитнеса**: Для каждой хромосомы вычисляется значение целевой функции, которая служит показателем качества решения. Это значение называется "фитнес-функцией". Чем ниже значение фитнес-функции, тем лучше решение.
3. **Отбор**: На основе значений фитнеса производится отбор лучших хромосом, которые будут участвовать в следующем поколении.
4. **Кроссинговер**: Хромосомы-победители комбинируются парами для создания новых хромосом, которые наследуют характеристики родителей. Кроссинговер позволяет объединять успешные черты решений и улучшать результаты.
5. **Мутация**: С небольшой вероятностью происходит случайное изменение отдельных генов в потомках. Мутация необходима для поддержания генетического разнообразия и предотвращения преждевременной сходимости к локальному минимуму.
6. **Создание новой популяции**: Старая популяция заменяется новой, состоящей из потомков и лучших представителей предыдущего поколения.
7. **Проверка условия остановки**: Алгоритм продолжает работу, пока не выполнится критерий остановки, такой как достижение заданного количества поколений или нахождение удовлетворительного минимума.

#### Описание схемы пошагового выполнения алгоритма и блок-схема

1. **Инициализация начальной популяции**

Создается начальная популяция из случайных хромосом. Каждая хромосома представляет собой возможное решение задачи и состоит из генов.

Наша задача минимизации зависит от двух переменных x1​ и x2, значит каждая хромосома представляет собой пару значений (x1,x2) случайно выбранных в пределах заданного диапазона (например, от -50 до 50).

1. **Оценка фитнес-функции для каждой хромосомы**

Вычисляется значение целевой функции для каждой хромосомы в популяции. Это значение показывает, насколько хорошим является данное решение. В задаче минимизации чем меньше значение целевой функции, тем выше качество решения.

1. **Отбор лучших хромосом**

Для формирования следующего поколения выбираются только лучшие хромосомы, основываясь на их значениях фитнес-функции. Обычно выбирается только половина популяции с наименьшими значениями фитнеса.

1. **Кроссинговер**

Отобранные хромосомы образуют пары, из которых с помощью операции кроссинговера создаются новые хромосомы (потомки). Кроссинговер предполагает обмен генами между двумя родителями для создания потомков, которые будут включать характеристики обоих родителей.

Например, если родитель 1=(x1,x2) и родитель 2=(y1,y2), то после кроссинговера мы можем получить потомков потомок 1=(x1,y2) и потомок 2=(y1,x2).

1. **Мутация**

С небольшой вероятностью отдельные гены в потомках случайным образом изменяются. Мутация необходима для поддержания генетического разнообразия и предотвращения преждевременной сходимости к локальному минимуму. Например, если вероятность мутации составляет 20%, то у каждого гена есть 20% шанс случайного изменения на новое случайное значение.

Мутация помогает алгоритму находить новые решения, которые могут улучшить качество популяции в целом.

1. **Создание новой популяции**

Старая популяция заменяется новой, состоящей из потомков и лучших хромосом из предыдущего поколения. Этот процесс повторяется до тех пор, пока не будет достигнуто заданное количество поколений или не выполнится другой критерий остановки.

1. **Проверка условия остановки**

Алгоритм продолжает работать, пока не будет выполнено условие остановки. Обычными критериями являются достижение заданного количества поколений или достижение значения фитнес-функции, соответствующего приемлемому уровню решения задачи.

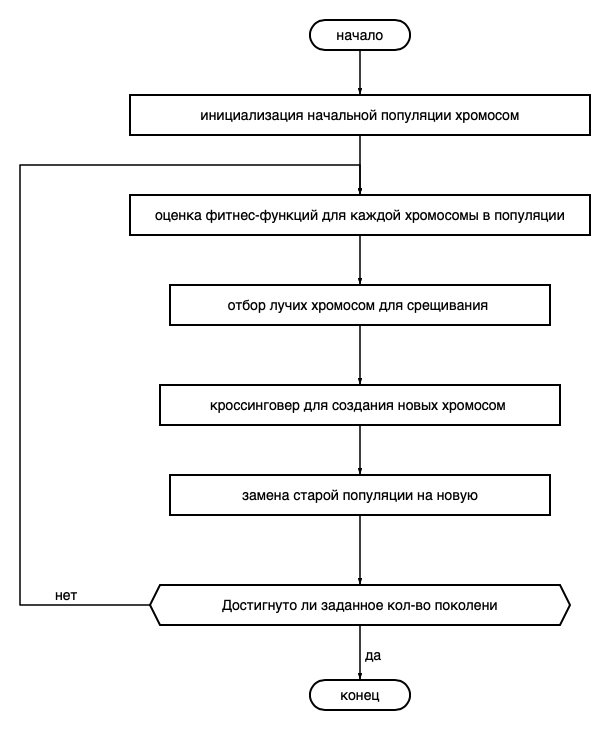


Рис 1 блок-схема основной программы

# **Формализация задачи**

**Формализация задачи**: Задача минимизации целевой функции (x1−2)4+(x1−2x2)2 предполагает, что мы ищем такие значения переменных x1​ и x2​, при которых функция принимает минимальное значение. Это используется как целевая функция для генетического алгоритма.

**Спецификация программы**:

**Функция целевая**: Вычисляет значение целевой функции для каждой пары значений (генов).

**Функция инициализации**: Создает начальную популяцию случайных хромосом в диапазоне от -50 до 50.

**Функция оценки популяции**: Вычисляет значения целевой функции для всех хромосом текущей популяции.

**Функция отбора**: Отбирает лучшие хромосомы для участия в следующем поколении.

**Функции кроссинговера и мутации**: Позволяют создавать новых потомков и вносить небольшие изменения в их гены.

**Основной цикл**: Запускает генетический алгоритм на заданное количество поколений.

Таблица1 1.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| generate\_population() | Создает начальную популяцию с заданным количеством особей в заданном диапазоне генов. | list |
| evaluate\_population() | Вычисляет значения целевой функции для каждой особи в популяции. | list |
| select\_best() | Выполняет отбор лучших особей на основе значений целевой функции. | list |
| crossover () | Скрешивает две особи, создавая два новых потомка методом кроссовера. | tuple |
| genetic\_algorithm() | Главная функция, включающая в себя все остальные. Реализует ГА | tuple |
| mutate() | Вносит случайные мутации в гены особи с заданной вероятностью. | tuple |
| create\_new\_population() | Создает новое поколение, выбирая родителей, выполняя кроссовер и мутацию. | list |
| run\_genetic\_algorithm() | Главная функция, включающая в себя все остальные и реализующая полный процесс ГА. | tuple |

# **Рекомендации пользователя**

Для запуска программы убедитесь, что у вас установлен Python и необходимые библиотеки, такие как numpy. Код можно запустить в среде разработки или через командную строку, используя консоль для настройки параметров и генерации данных. Запуск программы производится через файл 1.py.

При запуске программы вам будет предложено выбрать параметры запускаемого алгоритма. Вводите ответы соответствующие поля в GUI. Результаты работы будут представлены в таблице.

# **Рекомендации программиста**

Для поддержания актуальности и работоспособности программы используйте последние версии библиотек, особенно numpy. Применяйте практики надлежащего именования переменных и функций для улучшения читаемости кода.

# **Исходный код программы**

<https://github.com/Ignatio27/spbu-algorithms-and-data-structures->

# **Описание контрольного примера**

Для тестирования эффективности генетического алгоритма (ГА) выбрана тестовая функция из задания:

f(x1,x2)=(x1−2)^4+(x1−2x2)^2

Целью является минимизация этой функции, то есть нахождение таких значений x1​ и x2​, при которых значение функции будет минимальным.

**Условия задачи и параметры генетического алгоритма для контрольного примера**

1. **Диапазон значений переменных**

Поскольку генетический алгоритм работает с начальной популяцией случайных значений, для переменных x1 и x2 задаются границы поиска: от -50 до 50.

1. **Параметры алгоритма**

**Размер популяции**: 50 хромосом. Популяция состоит из 50 возможных решений, каждое из которых является парой значений (x1,x2).

**Вероятность мутации**: 20%. Это значит, что у каждого гена (значения x1​ или x2) есть 20% шанс случайного изменения в пределах заданного диапазона.

**Вероятность кроссинговера**: используется однородный кроссинговер с вероятностью 100%, что означает, что каждая пара родительских хромосом обязательно участвует в обмене генами для создания потомков.

**Количество поколений**: 100. Алгоритм будет выполняться на протяжении 100 поколений или до тех пор, пока не достигнет минимума с приемлемой точностью.

1. **Критерий остановки**

Основным критерием завершения алгоритма является достижение 100 поколений. Однако, если в процессе алгоритм найдет значение функции, которое достаточно близко к минимальному, алгоритм может быть остановлен раньше.

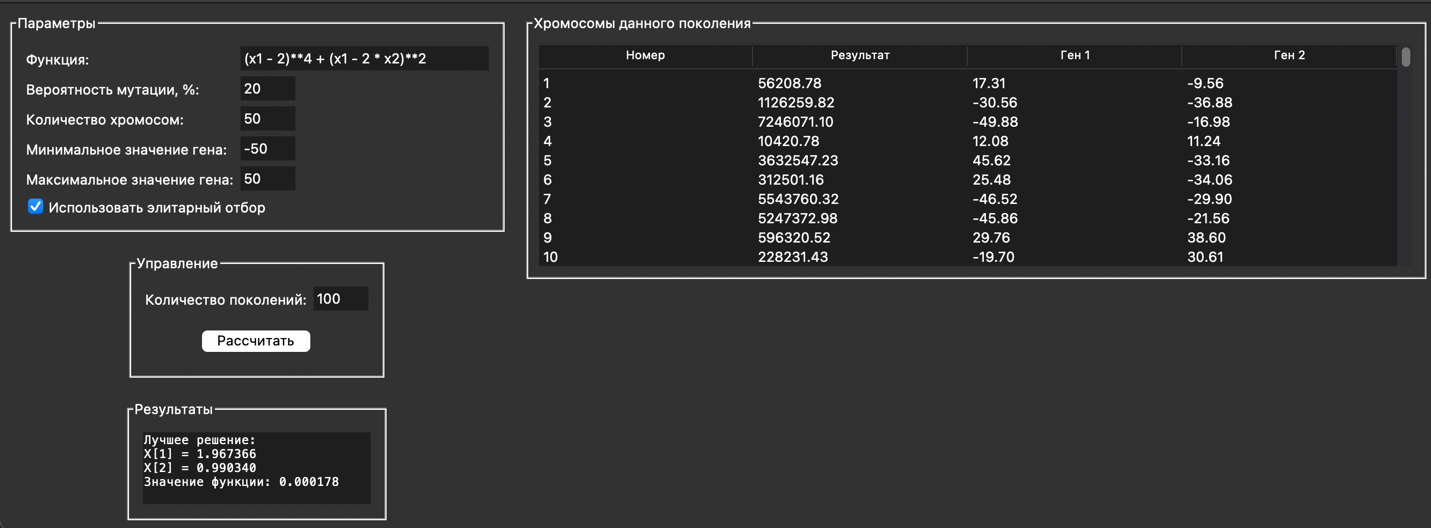
1. **Целевая функция (фитнес-функция)**

Для оценки качества каждого решения (хромосомы) используется целевая функция f(x1,x2). Поскольку задача минимизации, чем меньше значение функции для данной хромосомы, тем выше её "приспособленность". Хромосомы с более низким значением целевой функции считаются лучшими и имеют больше шансов на выбор для создания следующего поколения.

1. **Контрольные результаты для тестового примера**

В ходе выполнения контрольного примера алгоритм генерирует 50 хромосом в начальной популяции и вычисляет значение функции для каждого решения.

По результатам выполнения алгоритм должен определить такие значения x1​ и x2​, при которых значение функции будет минимальным. Ожидается, что при успешной работе ГА минимальное значение функции будет достигнуто на близких к истинному решению значениях x1≈2 и x2≈1

****

**Рис. 2 пример вывода**

**Пример выполнения (результаты)**

В контрольном примере после выполнения всех поколений генетический алгоритм находит хромосому с минимальным значением функции, например:

x1≈1.97

x2≈0.99

Значение функции на этих значениях составляет примерно f(x1,x2)≈0.0002, что очень близко к нулю, свидетельствуя о том, что найденное решение является близким к оптимальному.

Тестирование программы с генетическим алгоритмом показало, что алгоритм эффективно находит минимум целевой функции при оптимальных настройках параметров.

**Основные результаты**

**Оптимальные параметры**: При вероятности мутации 20%, размере популяции 50, и 100 поколениях алгоритм нашел решение, близкое к оптимальному, с минимальным значением функции около f(x1,x2)≈0.0002.

**Влияние параметров**: Увеличение вероятности мутации и размера популяции улучшило результат, помогая избегать локальных

# **Исследование**

Элитарный отбор — это метод, при котором лучшие особи текущего поколения сохраняются в неизменном виде для следующего поколения. Этот подход помогает удерживать сильные решения в популяции и предотвращает её ухудшение по мере итераций. Чтобы понять влияние элитарного отбора на эффективность генетического алгоритма, рассмотрим его воздействие на процесс поиска решений и вероятность нахождения глобального оптимума.

#### Влияние на глобальный и локальный поиск

Элитарный отбор выполняет важную функцию на разных этапах работы алгоритма:

1. **На ранних этапах** алгоритм благодаря элитарному отбору быстрее фокусируется на перспективных участках пространства решений. Лучшие особи попадают в следующее поколение, что позволяет быстрее «исследовать» потенциальные оптимальные области. Это ускоряет глобальный поиск, поскольку высококачественные решения сохраняются, создавая хорошую основу для последующих итераций.
2. **На поздних этапах** элитарный отбор способствует удержанию лучших решений, что помогает алгоритму сосредоточиться на локальном поиске. Благодаря этому снижается риск ухудшения популяции, так как лучшие решения сохраняются и становятся «якорями», вокруг которых ведется детальный поиск. В результате, алгоритм становится более устойчивым к застреванию в локальных минимумах, сохраняя баланс между поиском новых решений и улучшением текущих.

#### Экспериментальное наблюдение

В процессе тестирования алгоритма с включённым и отключённым элитарным отбором были замечены следующие тенденции:

1. **С элитарным отбором**: алгоритм стабильно показывает более низкие значения целевой функции к конечным итерациям, так как сохраняет качественные решения, вокруг которых и сосредоточивается дальнейший поиск. Итоговые решения в среднем ближе к глобальному минимуму.
2. **Без элитарного отбора**: алгоритм склонен к случайным изменениям в популяции, из-за чего может терять лучшие решения на каждом этапе. В некоторых случаях это приводит к увеличению числа итераций, необходимого для достижения сопоставимого результата.

# **Вывод**

В рамках данной работы был разработан алгоритм для поиска глобального минимума функции. Программа была протестирована на функции Розенброка. Реализованный алгоритм позволяет оптимизировать различные функции, включая задачи поиска экстремумов и решения NP-трудных задач. В процессе формирования новой популяции была внедрена модификация с использованием элитарного отбора, что обеспечило сохранение лучших решений и гарантировало, что каждое новое поколение не уступает предыдущему по качеству.

# **Источники**

1. Numpy documentation // Numpy URL: https://numpy.org (дата обращения: 4.11.2024).
2. Tkinter documentation // Tkinter URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 4.11.2024)