**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: 5**

**Студент гр. 23Б15-пу**

**Бек В.А.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc181568864)
2. [Описание задачи (формализация задачи) 4](#_Toc181568865)
3. [Теоретическая часть 5](#_Toc181568866)

[Основные компоненты генетического алгоритма 5](#_Toc181568867)

1. [Основные шаги программы 7](#_Toc181568868)
2. [Блок схема программы 9](#_Toc181568869)
3. [Описание программы 10](#_Toc181568870)
4. [Рекомендации пользователя 12](#_Toc181568871)
5. [Рекомендации программиста 13](#_Toc181568872)
6. [Контрольный пример 14](#_Toc181568873)
7. [Анализ 17](#_Toc181568874)
8. [Анализ на практике 19](#_Toc181568875)
9. [Вопрос о минимальном количестве телефонов 22](#_Toc181568876)
10. [Вывод 23](#_Toc181568877)
11. [Источники 24](#_Toc181568878)

# Цель работы

Целью данной лабораторной работы является исследование и сравнительный анализ двух основных способов кодирования генотипа хромосом в генетическом алгоритме. Для достижения этой цели необходимо изучить особенности различных методов кодирования, применяемых в генетических алгоритмах, и оценить их влияние на эффективность поиска решения задачи оптимизации. Конкретной задачей является минимизация тестовой функции путём реализации генетического алгоритма с выбранным типом кодирования.

# Описание задачи (формализация задачи)

В рамках данной лабораторной работы необходимо исследовать эффективность генетического алгоритма при решении задачи оптимизации, а именно — при минимизации функции 4 \* (x[1] - 5) \*\* 2 + (x[2] - 6) \*\* 2

Формализация задачи включает следующие компоненты:

1. **Целевая функция**: выбранная тестовая функция, которую необходимо минимизировать с помощью генетического алгоритма. Функция может зависит от двух переменных.
2. **Методы кодирования генотипа**:
   1. **Целочисленное кодирование**: Представление генотипа непосредственно в виде целых чисел, что позволяет избежать дополнительных преобразований и напрямую использовать значения переменных для вычисления функции
   2. **Вещественное кодирование**: представление генотипа непосредственно в виде чисел с плавающей точкой, что позволяет избежать дополнительных преобразований и напрямую использовать значения переменных для вычисления функции.
3. **Цель оптимизации**: нахождение минимального значения функции путём подбора оптимальных значений переменных с использованием генетического алгоритма, который учитывает два способа кодирования.

# Теоретическая часть

Генетические алгоритмы (ГА) представляют собой мощный инструмент оптимизации и поиска, основанный на принципах естественного отбора и эволюции. Эти алгоритмы применяются для решения задач, где необходимо найти экстремумы целевой функции. ГА начинают работу с начальной популяции возможных решений и используют такие операции, как отбор, скрещивание и мутация, чтобы эволюционно улучшить эти решения.

**Основные компоненты генетического алгоритма:**

Генетический алгоритм состоит из нескольких ключевых этапов и компонентов:

1. **Кодирование генотипа**: Каждое возможное решение задачи представлено в виде структуры, называемой хромосомой. Способы кодирования генотипа оказывают значительное влияние на эффективность алгоритма. Наиболее распространены следующие методы:
   * **Бинарное кодирование**: использует битовые строки, где каждый бит кодирует часть значений переменных целевой функции. Этот метод требует преобразования из бинарной формы в вещественное или целочисленное представление.
   * **Вещественное кодирование**: переменные целевой функции представлены числами с плавающей точкой. Этот подход предпочтителен, когда значения переменных непрерывны и их можно использовать непосредственно.
   * **Целочисленное кодирование**: генотип представлен непосредственно целыми числами, что упрощает вычисления и исключает необходимость преобразования значений, как в случае с бинарным кодированием.
2. **Функция приспособленности (fitness function)**: эта функция определяет, насколько хороша хромосома как решение задачи. В данной работе функция приспособленности служит для оценки минимального значения целевой функции. Генетический алгоритм стремится минимизировать её значение, улучшая каждую новую популяцию.
3. **Отбор (Selection)**: один из самых важных этапов алгоритма, на котором происходит выбор хромосом для создания потомства. Методы отбора включают:
   * **Рулеточный отбор (Roulette Wheel Selection)**: вероятность выбора хромосомы пропорциональна её приспособленности. Чем лучше хромосома, тем выше её шанс быть отобранной.
   * **Отбор по турниру (Tournament Selection)**: из случайного подмножества хромосом выбирается лучшая. Этот метод дает высокий шанс отбора для лучших хромосом, что ускоряет эволюцию.
   * **Элитный отбор (Elitism)**: определённое количество лучших хромосом (элита) напрямую переносится в следующее поколение, что сохраняет качество решений на каждом этапе.
4. **Скрещивание (Crossover)**: создаёт новые хромосомы (потомство) из пар родительских хромосом. Основные методы скрещивания включают:
   * **Одноточечное скрещивание**: родительские хромосомы делятся на две части в произвольной точке, и части обменяются между ними.
   * **Двухточечное скрещивание**: две точки определяют диапазон, в пределах которого происходит обмен генетической информацией.
   * **Арифметическое скрещивание**: значения переменных в потомке представляют собой взвешенное среднее родительских значений. Это полезно в задачах с вещественным или целочисленным кодированием, позволяя плавно изменять значения переменных.
5. **Мутация (Mutation)**: изменяет некоторые гены в хромосомах с заданной вероятностью, что позволяет алгоритму исследовать новые решения и избегать преждевременной сходимости к локальным экстремумам. В целочисленном и вещественном кодировании мутация может быть выполнена путём случайного изменения значений переменных в определённых границах.
6. **Критерий остановки**: процесс эволюции продолжается до тех пор, пока не будет достигнуто заданное число поколений.

# Основные шаги программы

* **Импорт библиотек**: Программа использует библиотеки random для случайных операций и tkinter для создания графического интерфейса пользователя.
* **Создание окна**: Инициализируется главное окно приложения с заголовком и фиксированным размером, которое будет использоваться для отображения интерфейса.
* **Определение глобальных переменных**: Создаем ряд глобальных переменных:
  + population: текущая популяция хромосом.
  + best\_solution и best\_value: лучшие найденные решения и их значения.
  + total\_generations: общее количество сгенерированных поколений.
  + integer\_mode и modified\_mode: флаги для активации целочисленного режима и модификаций.
* **Функция set\_generations**: Обновляет количество поколений для генерации.
* **Переключатели режимов**:
  + toggle\_integer\_mode: включает и выключает режим целочисленных значений для хромосом.
  + toggle\_modified\_mode: включает и выключает модифицированный режим, изменяющий процесс отбора и оценки хромосом.
* **Функция create\_initial\_population**: Создает начальную популяцию на основе размера популяции и диапазона значений генов (или целых чисел, если активен целочисленный режим).
* **Функция evaluate\_function**: Определяет целевую функцию, которую необходимо минимизировать. Она оценивает «ценность» хромосомы, определяя ее качество.
* **Функция selection**: Проводит отбор хромосом:
  + В обычном режиме возвращает текущую популяцию.
  + В модифицированном режиме сортирует популяцию и отбирает лучших (по целевой функции).
* **Функция mutate**: Проводит мутацию каждого гена в хромосоме с заданной вероятностью, изменяя значение гена случайным образом в пределах диапазона.
* **Функция crossover**: Реализует кроссовер двух родительских хромосом, объединяя их для получения новой хромосомы (ребенка) с помощью параметра alpha.
* **Функция create\_new\_generation**: Создает новое поколение, проводя отбор, мутацию и кроссовер.
* **Функция calculate\_chromosomes**:
  + Основная функция для запуска алгоритма.
  + Загружает параметры из полей ввода и создает начальную популяцию.
  + В цикле запускает процесс эволюции: отбор, генерацию нового поколения и оценку результатов.
  + Находит лучшее решение и обновляет соответствующие поля интерфейса.
* **Графический интерфейс**:
  + Создаются элементы интерфейса: поля ввода для параметров (количество хромосом, поколений, вероятности мутации, минимальных и максимальных значений генов).
  + Кнопки для переключения режимов, установки числа поколений и запуска алгоритма.
  + Поле для отображения лучшего найденного решения и значений функции.
* **Вывод результатов**:
  + После завершения расчетов программа отображает лучшее решение и значение функции.
  + Таблица показывает хромосомы текущего поколения и их значения функции.
* **Запуск программы**: Инициализация интерфейса и запуск основной программы с помощью root.mainloop(), что позволяет пользователю взаимодействовать с программой через графический интерфейс.

# Блок схема программы

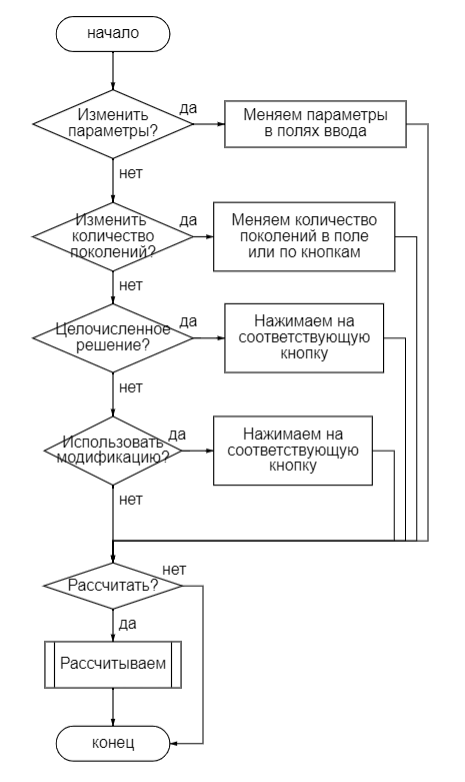
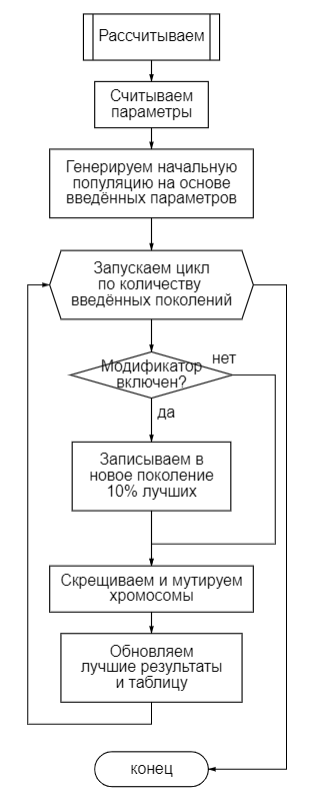
 

Рис. 1 Блок-схема основной программы Рис. 2 Блок-схема подпрограммы

# Описание программы

Программная реализация написана на языке Python 3.12.7 с использованием библиотек random, tkinter, ttk. Программа представляет собой GUI-приложение, реализующее генетический алгоритм для оптимизации функции, с возможностью выбора целочисленного или модифицированного режима работы. Приложение организовано в едином модуле и состоит из 10 функций, каждая из которых выполняет конкретную задачу.

Таблица 1. genetic\_algorithm.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| set\_generations | |  | | --- | |  |  |  | | --- | | Устанавливает значение в поле ввода количества поколений. | | None |
| toggle\_integer\_mode | Переключает режим целочисленного решения (вкл/выкл) и обновляет текст кнопки. | None |
| toggle\_modified\_mode | |  | | --- | |  |  |  | | --- | | Переключает модифицированный режим (вкл/выкл) и обновляет текст кнопки. | | None |
| create\_initial\_population | Создает начальную популяцию хромосом с генами в заданном диапазоне. | list |
| evaluate\_function | |  | | --- | |  |  |  | | --- | | Вычисляет значение функции для переданной хромосомы. | | float |
| selection | Проводит отбор хромосом на основе модифицированного режима. В модифицированном режиме оставляет 10% лучших решений. | list |
| mutate | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Выполняет мутацию генов в хромосоме с заданной вероятностью мутации. | | | list |
| crossover | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Проводит кроссовер (скрещивание) двух родительских хромосом с заданным коэффициентом альфа. | | | list |
| |  | | --- | | create\_new\_generation |  |  |  | | --- | --- | |  |  | | Создает новое поколение, добавляя детей, полученных методом кроссовера, и мутируя их с заданной вероятностью. | list |
| |  | | --- | | calculate\_chromosomes |  |  |  | | --- | --- | |  |  | | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Основная функция, которая инициализирует популяцию, выполняет расчет поколений, сохраняет лучшее решение и обновляет отображаемые данные. | | | None |

# Рекомендации пользователя

Для корректного выполнения программы следуйте приведённым шагам:

* **Заполните параметры**:
  + **"Вероятность мутации, %"** – процент вероятности мутации генов в популяции.
  + **"Количество хромосом"** – размер начальной популяции.
  + **"Минимальное значение гена"** и **"Максимальное значение гена"** – диапазон значений генов.
  + **"Количество поколений"** – число итераций, необходимых для поиска оптимального решения.
* **Настройте режимы алгоритма**:
  + Нажмите на кнопку **"Целочисленное решение: Выкл"** для переключения в целочисленный режим. Включенный режим ограничивает значения генов целыми числами.
  + Нажмите на кнопку **"Модификация: Выкл"** для включения модифицированного режима, который изменяет алгоритм отбора популяции.
* **Выберите количество поколений для быстрого ввода**: Используйте кнопки **"1"**, **"10"**, **"100"** и **"1000"** для быстрой установки числа поколений в соответствующее поле.
* **Запустите алгоритм**: Нажмите **"Рассчитать"** для запуска алгоритма. Программа создаст начальную популяцию, выполнит заданное количество поколений, и отобразит результаты.
* **Просмотрите результаты**:
  + Поле **"Лучшее решение"** покажет значения переменных хромосомы для найденного оптимального решения.
  + Поле **"Значение функции"** отобразит значение целевой функции для лучшего решения.
  + В таблице **"Хромосомы данного поколения"** будут показаны значения функции и генов для всех хромосом текущей популяции.

# Рекомендации программиста

Для корректного функционирования программы рекомендуется выполнить следующие действия:

1. **Установите необходимые библиотеки**:
   * Убедитесь, что у вас установлена библиотека tkinter. Она обычно идет в стандартной поставке Python, но если она отсутствует, вы можете установить ее через пакетный менеджер вашей операционной системы.
2. **Проверьте версию Python**:
   * Рекомендуется использовать Python версии 3.7 или выше, чтобы избежать возможных проблем с совместимостью библиотек и функциональностью tkinter.
3. **Проверка функций и интерфейса**:
   * Убедитесь, что все элементы интерфейса (кнопки, текстовые поля и таблицы) работают корректно. Попробуйте вводить различные значения и проверить, что результаты рассчитываются и отображаются правильно.
   * Обратите внимание на обработку значений в полях ввода — убедитесь, что значения корректно конвертируются в числовые типы.

Следуя этим рекомендациям, вы сможете наладить рабочий процесс и избежать распространенных ошибок при разработке приложения на основе генетического алгоритма с использованием tkinter.

**Код программы:**

**<https://github.com/Kliooo/Algorithms-and-data-structures>**

# Контрольный пример

* **Запуск программы:** Для запуска программы используйте файл genetic\_algorithm.py. Программа откроет графический интерфейс (Рис. 4), в котором можно изменить параметры для запуска генетического алгоритма, выбрать тип решения (целочисленный или вещественный) и указать, использовать ли модификацию, а так же задать количество поколений для расчёта.

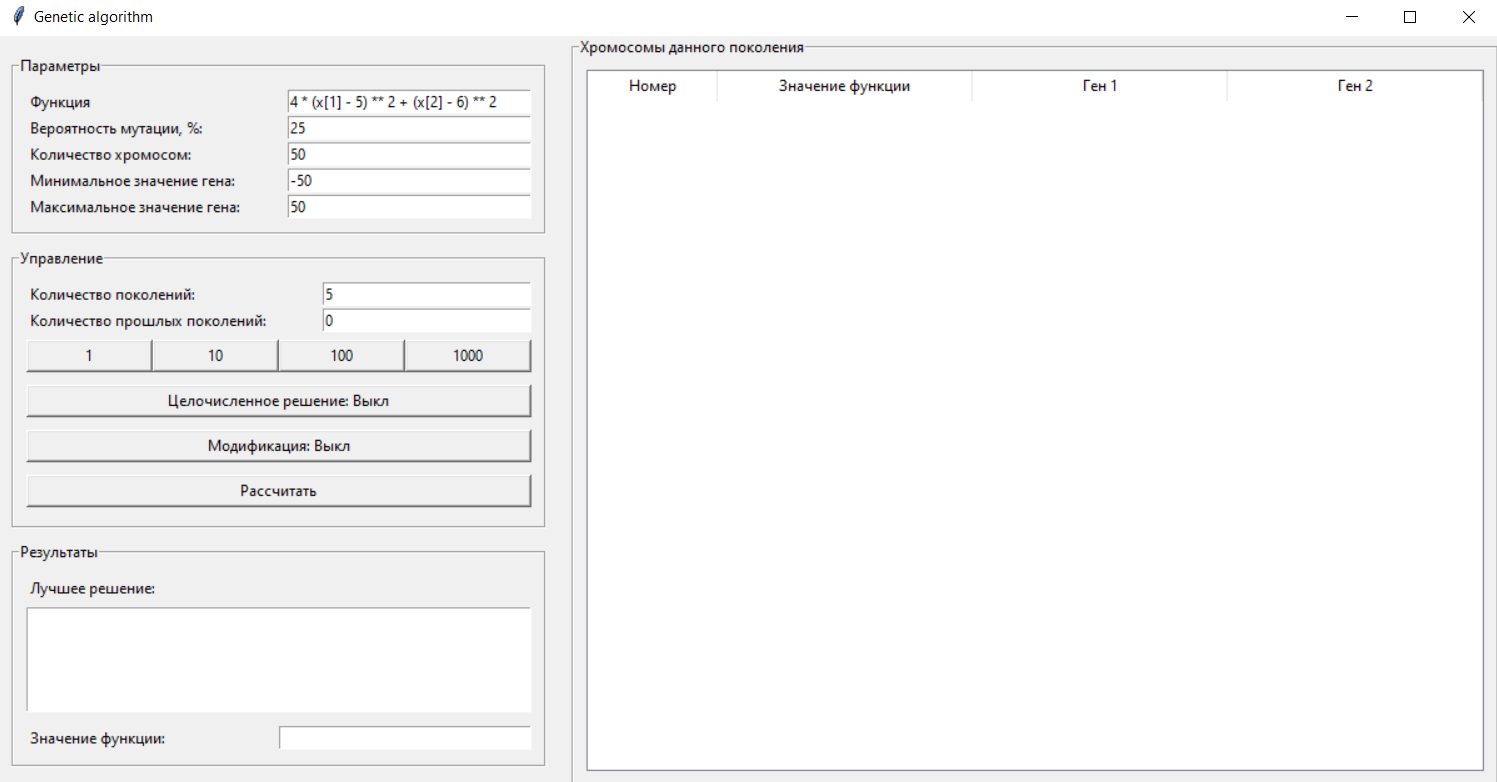


Рис. 3 Графический интерфейс программы

* **Запуск ГА:** Нажимая на кнопку “Рассчитать”, мы запускаем программу, которая выполняет генетический алгоритм. В результате мы получаем лучшее решение и соответствующее значение функции, отображаемые в текстовом поле. Также в таблице хромосомы нового поколения выводятся их номера, значения функции и значения генов. (Рис. 4)

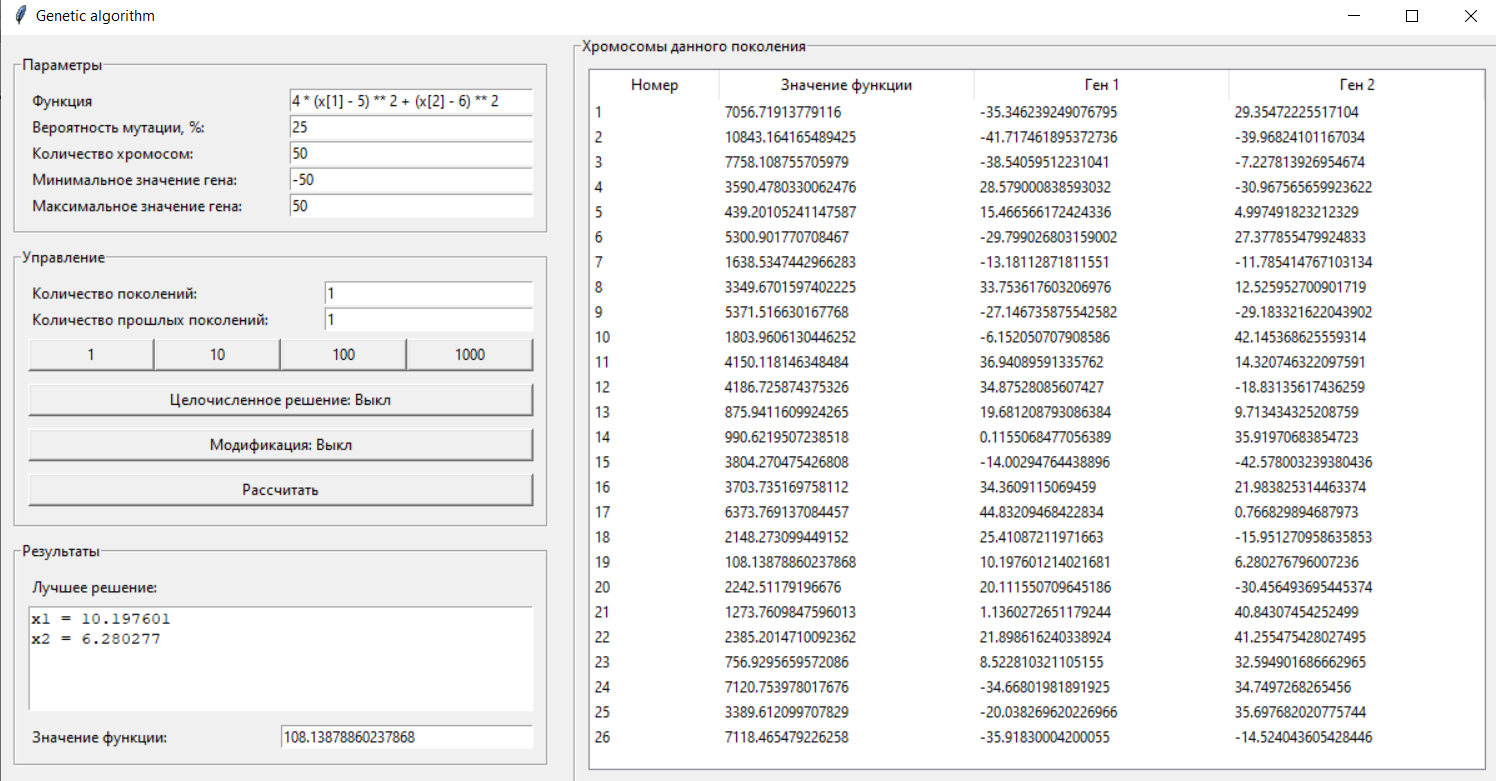


Рис. 4 Первый запуск

* **Повторный запуск ГА:** При повторном нажатии на кнопку “Рассчитать” и вводе большего числа поколений лучшее решение постепенно будет сходиться к истинному, а значение функции — к 0 (Рис. 5).

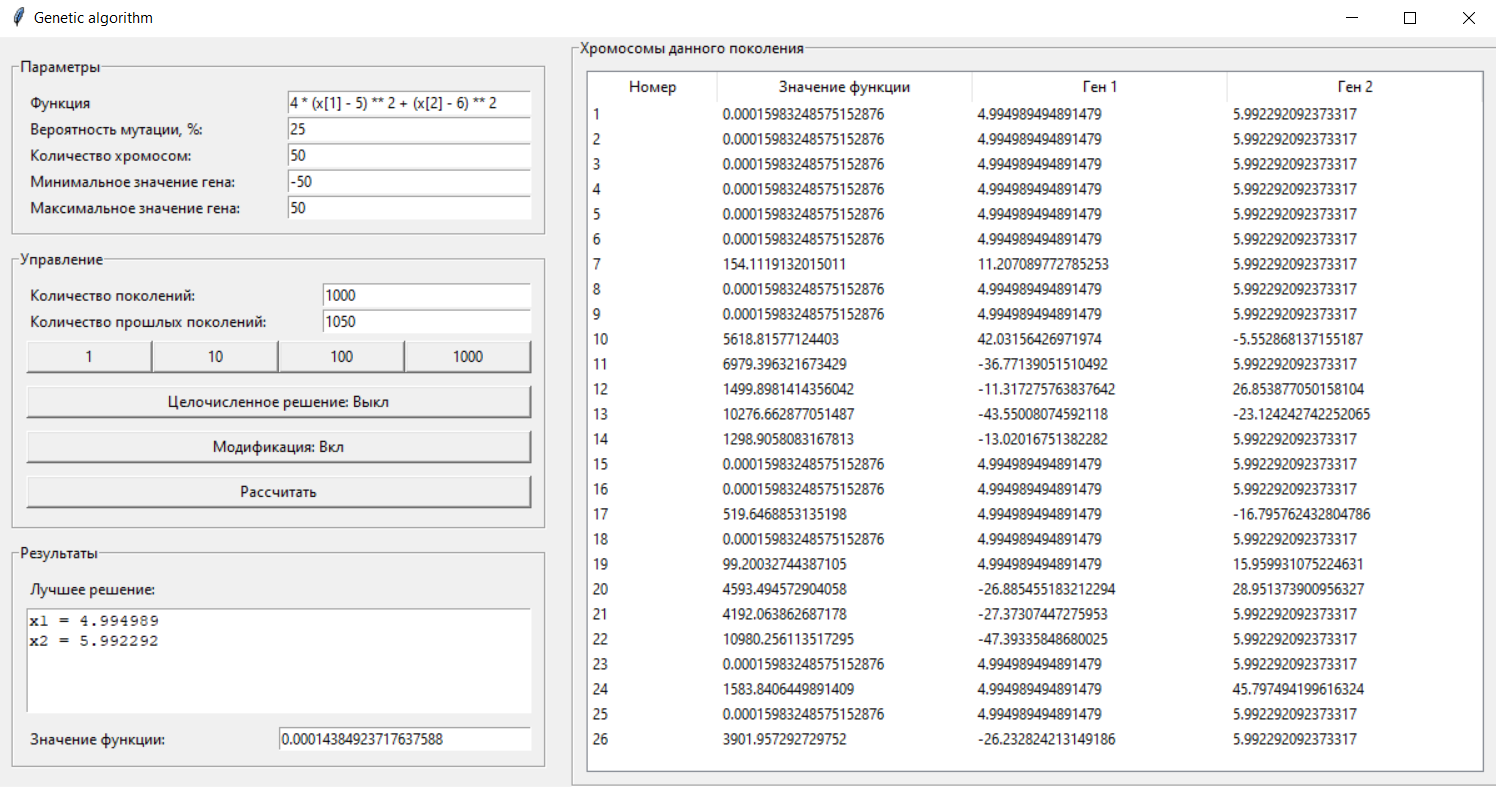


Рис. 5 Повторные запуски

# Анализ

Тесты проводились при вероятности мутации: 25%, минимальном значении гена: -50, максимальном значении гена: 50.

**Вещественный геном (с модификацией):**

Таблица 2: тесты вещественного генома с модификацией

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Количество хромосом | Лучшее решение | Значение в лучшей точке |
| 10 | 50 | (3.397153, 6.200638) | 9.2850053092505546 |
| 20 | 50 | (4.781539, 6,173672) | 0.1989559119258956 |
| 50 | 50 | (4.781539, 6,173672) | 0.0454505513501322 |
| 150 | 50 | (4.985548, 6.041756) | 0.0023211339912342 |
| 1500 | 50 | (4.998936, 5.999048) | 4.893563263232e-06 |

Таблица 3: тесты вещественного генома с модификацией

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Количество хромосом | Лучшее решение | Значение в лучшей точке |
| 10 | 20 | (2.798647, 8.372789) | 24.285005392505546 |
| 20 | 20 | (3.279449, 7,564791) | 14.289760342353235 |
| 30 | 20 | (5.363765, 6,754023) | 0.9880644888612312 |
| 100 | 20 | (5.055819, 6.021756) | 0.0132113391234223 |
| 1500 | 20 | (5.000569, 6.003851) | 1.453534534332e-05 |

**Вещественный геном (без модификации):**

Таблица 4: тесты вещественного генома без модификации

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Количество хромосом | Лучшее решение | Значение в лучшей точке |
| 10 | 50 | (-2.453426, 3.733579) | 227.35234432452135 |
| 20 | 50 | (8.623526, 9.398447) | 64.069196424442338 |
| 50 | 50 | (5.072840, 7.394164) | 1.7220672323424234 |
| 640 | 50 | (5.248162, 5.556513) | 0.4430176315460318 |
| 1500 | 50 | (5.248162, 5.556513) | 0.4430176315460318 |

Таблица 5: тесты вещественного генома без модификации

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Количество хромосом | Лучшее решение | Значение в лучшей точке |
| 10 | 20 | (3.341608, -11.117513) | 304.01235235255677 |
| 20 | 20 | (0.723803, -3.6371673) | 166.01843745745743 |
| 30 | 20 | (0.723803, -3.6371673) | 166.01843745745743 |
| 80 | 20 | (5.565096, 5.1041723) | 2.0743532468778787 |
| 640 | 20 | (5.565096, 5.1041723) | 2.0743532468778787 |

**Целочисленный геном (с модификацией):**

Таблица 6: тесты целочисленного генома с модификацией

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Количество хромосом | Лучшее решение | Значение в лучшей точке |
| 1 | 50 | (3.0, 5.0) | 15.3 |
| 2 | 50 | (6.0, 6.5) | 3.825 |
| 10 | 50 | (5.5, 6.0) | 1.8 |
| 13 | 50 | (5.0, 6.0) | 0.0 |

Таблица 7: тесты целочисленного генома с модификацией

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Количество хромосом | Лучшее решение | Значение в лучшей точке |
| 1 | 20 | (8.5, 2.5) | 55.125 |
| 5 | 20 | (7.0, 7.0) | 15.3 |
| 10 | 20 | (6.0, 5.0) | 4.5 |
| 25 | 20 | (5.0, 6.0) | 0.0 |

**Целочисленный геном (без модификации):**

Таблица 8: тесты целочисленного генома без модификации

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Количество хромосом | Лучшее решение | Значение в лучшей точке |
| 1 | 50 | (0.0, 2.0) | 116.0 |
| 2 | 50 | (6.0, 12.0) | 40.0 |
| 13 | 50 | (4.0, 9.0) | 13.0 |
| 50 | 50 | (4.0, 6.0) | 4.0 |
| 3000 | 50 | (5.0, 6.0) | 0.0 |

Таблица 9: тесты целочисленного генома без модификации

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Количество хромосом | Лучшее решение | Значение в лучшей точке |
| 1 | 20 | (0.0, 15.0) | 181.0 |
| 4 | 20 | (7.0, 14.0) | 80.0 |
| 7 | 20 | (6.0, 1.0) | 29.0 |
| 2500 | 20 | (5.0, 4.0) | 4.0 |

#### **Вещественный геном**

1. **С модификацией**: В таблицах, отражающих результаты тестов вещественного генома с модификацией, наблюдается значительное улучшение результатов по мере увеличения количества поколений. Значения в лучших точках стремительно приближаются к нулю. Например, при 1500 поколениях и 50 хромосомах значение достигло 4.89e-06.
2. **Без модификации**: В тестах без модификации наблюдается меньшая эффективность. Лучшие решения часто имеют нехорошие значения, такие как 227.35 и 64.07. Следовательно, модификация алгоритма положительно сказывается на его эффективности.

#### **Целочисленный геном**

1. **С модификацией**: Результаты тестов целочисленного генома с модификацией также показывают тенденцию к улучшению по мере увеличения числа поколений. Наиболее оптимальное решение с значением 0.00 достигнуто при 13 поколениях, следовательно, модификация позволяет достигать оптимальных решений быстрее и эффективнее.
2. **Без модификации**: Тесты целочисленного генома без модификации продемонстрировали менее впечатляющие результаты. Только при 3000 поколениях минимальное значение достигло 0.0.

#### **Общие наблюдения**

* **Число хромосом**: Наличие 50 хромосом оказалось более эффективным по сравнению с 20 в большинстве случаев, что указывает на то, что большее количество хромосом может способствовать нахождению более качественных решений.

# Вывод

В данной работе проведено исследование генетических алгоритмов (ГА) как метода оптимизации, акцентируя внимание на их применении в задачах поиска оптимальных решений. Были проанализированы различные реализации ГА, включая вещественный и целочисленный геномы, а также их модификации.

Результаты показали, что вещественные геномы с модификациями обеспечивают лучшие результаты по сравнению с целочисленными, подтверждая важность выбора подходящего представления для конкретной задачи. Исследование также подчеркивает сложность оптимизационных задач и необходимость адаптивности в настройках алгоритмов.

# Источники

* Редактор блок-схем.

[*https://programforyou.ru/block-diagram-redactor*](https://programforyou.ru/block-diagram-redactor)

*дата обращения: (02.11.2024)*

* tkinter — Библиотека для создания графических интерфейсов в Python.

[*https://docs.python.org/3/library/tkinter.html*](https://docs.python.org/3/library/tkinter.html%20)

*дата обращения: (27.10.2024)*

* random — Генерация псевдослучайных чисел*.* [*https://docs.python.org/3/library/random.html*](https://docs.python.org/3/library/random.html)

*дата обращения: (27.10.2024)*