**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 – 2**

**Студент гр. 23Б15-пу**

**Черевко М. Е.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc183124285)
2. [Описание задачи (формализация задачи) 4](#_Toc183124286)
3. [Теоретическая часть 5](#_Toc183124287)

[**Модификация выбора родителей: Введение вероятностного ранжирования** 5](#_Toc183124288)

[**Методы кодирования генотипа** 6](#_Toc183124289)

[**Показатели эффективности** 7](#_Toc183124290)

[**Применение генетических алгоритмов** 8](#_Toc183124291)

[**Тестовая функция** 8](#_Toc183124292)

1. [Основные шаги программы 9](#_Toc183124293)
2. [Блок схема программы 12](#_Toc183124294)
3. [Описание программы 13](#_Toc183124295)
4. [Рекомендации пользователя 15](#_Toc183124296)
5. [Рекомендации программиста 17](#_Toc183124297)
6. [Контрольный пример 18](#_Toc183124298)
7. [Анализ результатов работы алгоритма и вводных условий 20](#_Toc183124299)
8. [Вывод 25](#_Toc183124300)
9. [Источники 26](#_Toc183124301)

# Цель работы

Целью данной лабораторной работы является изучение и сравнительный анализ различных методов кодирования генотипа в рамках генетического алгоритма, а также оценка их влияния на эффективность решения задач оптимизации. В частности, рассматривается задача. В ходе выполнения работы будет разработана программная реализация генетического алгоритма, которая поддерживает два типа кодирования генотипа: вещественное и целочисленное.

В процессе исследования планируется изучить, как выбор способа кодирования влияет на ключевые характеристики алгоритма, такие как скорость сходимости, точность нахождения глобального минимума, а также устойчивость алгоритма к попаданию в локальные минимумы. Для этого будет проведён ряд вычислительных экспериментов с использованием тестовой функции.

Полученные результаты будут систематизированы и проанализированы, что позволит сделать выводы о преимуществах и недостатках каждого из рассматриваемых способов кодирования. Итогом работы станет выработка рекомендаций по выбору наиболее подходящего метода кодирования для решения задач оптимизации различной сложности.

# Описание задачи (формализация задачи)

1. **Исследование методов кодирования в генетических алгоритмах:**  
   Рассмотреть и проанализировать особенности различных подходов к кодированию генотипов, включая вещественное и целочисленное кодирование, с акцентом на их преимущества и ограничения.
2. **Создание программы для оптимизации функции:**  
   Разработать программное обеспечение, реализующее генетический алгоритм, с поддержкой различных методов кодирования, для решения задачи нахождения минимума функции.
3. **Экспериментальная проверка на тестовой функции:**  
   Провести тестирование созданной программы на выбранной функции из заданного набора. Оценить производительность алгоритма для каждого способа кодирования и сравнить полученные результаты.
4. **Оценка и интерпретация полученных данных:**  
   Анализировать результаты выполнения алгоритма, исследовать влияние различных подходов к кодированию на скорость и качество нахождения минимума функции. Сделать выводы о целесообразности использования конкретных методов кодирования в различных условиях.

# Теоретическая часть

Генетические алгоритмы (ГА) представляют собой один из ключевых методов оптимизации, основанный на принципах эволюционной биологии, таких как естественный отбор, наследственность и мутация. Их основной принцип заключается в моделировании эволюционного процесса: создается популяция возможных решений задачи (хромосом), которая проходит через этапы мутации, скрещивания и отбора для постепенного улучшения результатов. Эти алгоритмы хорошо зарекомендовали себя в решении сложных задач, где традиционные методы оптимизации оказываются менее эффективными или вовсе неприменимы.

Одной из важных составляющих генетических алгоритмов является способ кодирования генотипа, то есть представления решений в виде цифровых последовательностей. Правильный выбор способа кодирования напрямую влияет на эффективность работы алгоритма, его скорость сходимости и способность находить глобальный оптимум задачи.

### **Модификация выбора родителей: Введение вероятностного ранжирования**

В процессе разработки алгоритма было решено заменить классический подход турнира модифицированным методом, который использует параметр **probablyTheBest**. Эта модификация позволяет вносить элемент вероятности в выбор родителей, улучшая эффективность алгоритма.

* **Особенности метода probablyTheBest**

1. **Основной принцип:**
   * При выборе родителей хромосомы ранжируются по уровню приспособленности.
   * Лучшие хромосомы имеют высокий шанс быть выбранными, однако **с определенной вероятностью выбирается случайная хромосома** из оставшихся, а не самая сильная. Вероятность этого события задается параметром probablyTheBest.
2. **Параметр управления:**
   * Значение probablyTheBest определяет, насколько часто будет происходить выбор случайной хромосомы. Например:
     + При probablyTheBest = 0.2 вероятность выбора случайного родителя составляет 20%.
     + При probablyTheBest = 0.8 – уже 80%.

* **Цели модификации**

1. **Улучшение селекции:**
   * Классический турнирный метод иногда приводит к преждевременному схлопыванию популяции. Это происходит, когда лучшие хромосомы теряются из-за случайностей в процессе селекции.
   * probablyTheBest увеличивает вероятность участия сильных хромосом, что способствует сохранению высокой адаптивности популяции.
2. **Предотвращение зацикливания:**
   * Включение вероятностного компонента позволяет сохранять генетическое разнообразие в популяции.
   * Это предотвращает стагнацию, когда алгоритм зацикливается на подмножестве решений, не исследуя другие области пространства поиска.
3. **Баланс между качеством и разнообразием:**
   * Вероятностный выбор случайных хромосом способствует поиску новых решений, обеспечивая более широкий охват пространства.
   * Одновременно сохраняется шанс для сильных хромосом участвовать в процессе, что поддерживает высокое качество популяции.

### **Методы кодирования генотипа**

1. **Вещественное кодирование**  
   При вещественном кодировании хромосомы представлены в виде последовательностей вещественных чисел, что обеспечивает высокую точность и гибкость при работе с задачами, имеющими непрерывное пространство поиска. Этот метод удобен для оптимизации параметров моделей и функций с высокой точностью.  
   Преимущества вещественного кодирования:
   * Высокая точность представления решений.
   * Возможность работы в непрерывных пространствах.
   * Большая адаптивность к различным задачам.

Однако у этого подхода есть и свои недостатки:

* + Риск "застревания" в локальных минимумах, если параметры отбора, мутации и скрещивания настроены неэффективно.
  + Увеличение вычислительных затрат при больших размерностях задач.

1. **Целочисленное кодирование**  
   Целочисленное кодирование представляет хромосомы в виде последовательностей целых чисел, что делает его предпочтительным для задач с дискретными или комбинаторными решениями. Этот метод часто используется в задачах, где требуется учитывать ограничения, такие как количество доступных ресурсов или дискретные шаги в пространстве поиска.  
   Преимущества целочисленного кодирования:
   * Подходит для задач с дискретными ограничениями.
   * Более высокая скорость работы в задачах с ограниченным пространством поиска.
   * Устойчивость к "переобучению" на малых выборках данных.

Основные недостатки целочисленного кодирования:

* + Меньшая точность по сравнению с вещественным кодированием.
  + Ограниченная применимость в задачах с непрерывными пространствами поиска.

### **Показатели эффективности**

Эффективность генетического алгоритма определяется его способностью находить оптимальные или близкие к оптимальным решения за приемлемое время. Основные показатели:

* **Скорость сходимости**: характеризует время, необходимое алгоритму для достижения заданного уровня качества решения. Более быстрое нахождение оптимума указывает на высокую эффективность алгоритма.
* **Качество решения**: определяет, насколько близко найденное решение к истинному оптимуму. Высокое качество решения свидетельствует о том, что алгоритм справляется с задачей нахождения глобального, а не локального минимума.

Эти показатели напрямую зависят от выбора метода кодирования генотипа, параметров алгоритма (например, вероятности мутации и кроссовера), а также сложности задачи.

### **Применение генетических алгоритмов**

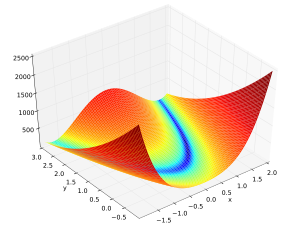
Благодаря своей универсальности и гибкости генетические алгоритмы нашли широкое применение в самых разных областях, включая:

1. **Оптимизация производственных процессов**
   * Минимизация затрат и времени производства.
   * Оптимизация параметров оборудования.
2. **Проектирование и планирование**
   * Разработка сложных инженерных систем.
   * Планирование и оптимизация расписаний.
3. **Машинное обучение и искусственный интеллект**
   * Настройка гиперпараметров моделей.
   * Генерация новых архитектур нейронных сетей.
4. **Экономика и финансы**
   * Управление инвестиционными портфелями.
   * Оптимизация распределения активов.
5. **Биология и медицина**
   * Моделирование эволюционных процессов.
   * Оптимизация биологических экспериментов.

### **Тестовая функция**

В качестве тестовой оптимизационной функции будет использоваться функция Розенброка, имеющая минимум 0 в точке (1,1).

**https://ru.wikipedia.org/wiki/ Функция\_Розенброка**



# Основные шаги программы

1. **Запуск приложения**

* При запуске программы создается основное окно Tkinter, в котором размещены все элементы интерфейса.
* Пользователь видит параметры, которые можно настроить для генетического алгоритма, такие как вероятность мутации, число особей в популяции, ограничения на гены, количество поколений, а также опции выбора целочисленного кодирования и модифицированного отбора.

1. **Инициализация параметров**

* Для ввода параметров используются текстовые поля (Entry), а для переключения между режимами (например, целочисленное кодирование) — чекбоксы (Checkbutton).
* При первом запуске программы параметры генетического алгоритма принимают значения по умолчанию:
  + Мутация: 15%
  + Число особей: 20
  + Диапазон генов: от -1 до 1
  + Число поколений: 20
  + Целочисленное кодирование, модифицированный отбор и сохранение истории таблицы отключены.

1. **Генетический алгоритм**

* При запуске алгоритма создается начальная популяция решений (хромосом), размер которой равен числу особей.
* Хромосомы представлены в виде массива из двух генов (x1 и x2), значения которых генерируются:
  + В случае вещественного кодирования — случайные числа с плавающей точкой в пределах заданного диапазона.
  + В случае целочисленного кодирования — случайные целые числа.
* Алгоритм проходит заданное количество поколений. В каждом поколении выполняются следующие шаги:
  1. **Вычисление приспособленности**:  
     Для каждой хромосомы вычисляется значение оптимизируемой функции (целевая функция).
  2. **Сохранение лучшего решения**:  
     Если в текущем поколении найдено решение с меньшим значением функции, оно сохраняется как лучшее.
  3. **Отбор родителей**:  
     С помощью турнира выбираются две родительские хромосомы для скрещивания. При модифицированном отборе лучшие кандидаты имеют 80% вероятность быть выбраны родителями.
  4. **Скрещивание**:  
     Родители комбинируются с вероятностью 90% для создания новой хромосомы (потомка). Если целочисленное кодирование включено, результат округляется до целого числа.
  5. **Мутация**:  
     С заданной вероятностью мутации в потомке случайным образом изменяется один из генов.
  6. Новый потомок добавляется в следующую популяцию.
* После завершения всех операций с текущим поколением создается новая популяция, заменяющая старую.
* Процесс повторяется, пока не будет пройдено заданное число поколений.

1. **Отображение результатов**

* По окончании каждого поколения обновляются данные в интерфейсе:
  1. Отображается количество пройденных поколений.
  2. Показаны лучшие x1, x2 и значение функции для текущей популяции.
  3. В таблицу результатов добавляются данные по всем особям текущего поколения: их значения генов и значения функции.
* Если включено целочисленное кодирование, значения отображаются в формате целых чисел.
* Если не включено сохранение истории таблицы, то будет показываться только последнее поколение в текущей итерации алгоритма.
* После завершения работы алгоритма сохраняются и выводятся лучшие хромосомы и их значения, найденные за все поколения.

1. **Очистка данных**

* При нажатии кнопки "Очистить" происходит очистка всех данных:
  + Удаляется текущая популяция.
  + Лучшие значения функции и хромосом сбрасываются.
  + Очищается таблица результатов.
* Пользователь может настроить новые параметры и запустить алгоритм снова.

1. **Особенности программы**

* **Целочисленное и вещественное кодирование**:  
  Программа поддерживает оба способа кодирования, автоматически адаптируя операции скрещивания и мутации.
* **Гибкость параметров**:  
  Пользователь может задавать диапазон генов, число поколений, вероятность мутации и другие параметры через интерфейс.
* **Модифицированный отбор**:  
  Эта опция увеличивает шансы выбора лучших хромосом, повышая вероятность нахождения глобального минимума.
* **Визуализация**:  
  Результаты работы алгоритма в каждом поколении отображаются в таблице.
* **Возможность отображать данные таблицы результатов, полученные при всех запусках алгоритма**.

1. ***Заключение работы***

* Пользователь может проанализировать результаты работы алгоритма, включая скорость сходимости и качество найденного решения.
* При необходимости можно повторить эксперимент, изменив параметры алгоритма.

# Блок схема программы

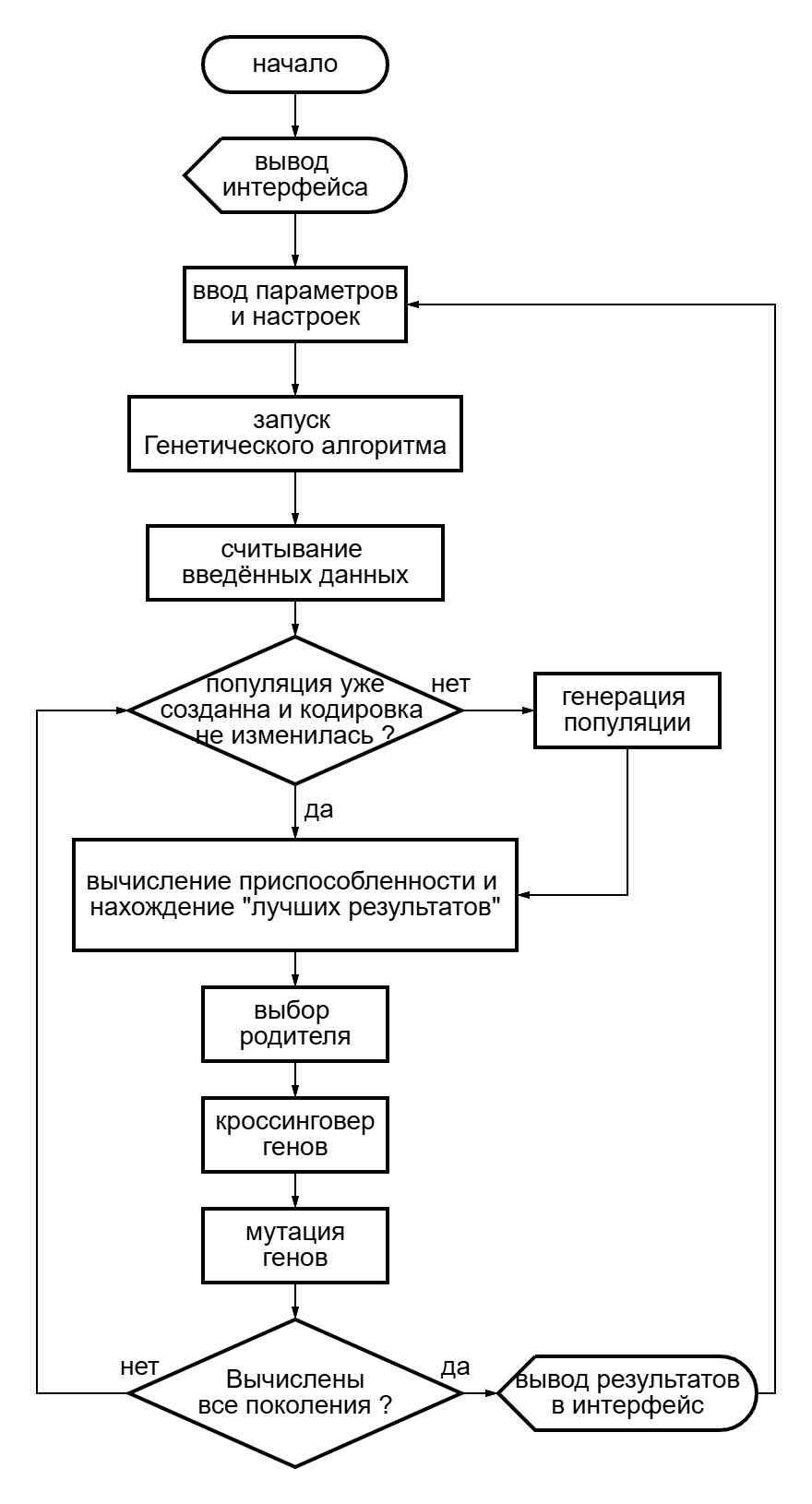


Рис 1. Блок-схема «Genetic\_algorithm.py»

# Описание программы

Программная реализация написана на языке Python 3.12.6 с использованием следующих библиотек: **numpy**, **tkinter** и **random**. Программа представляет собой графический интерфейс, предназначенный для решения оптимизационных задач с использованием генетического алгоритма. Основной функционал программы реализован в классе **GAApplication**, который включает 12 методов, каждый из которых выполняет строго определённую функцию:

Таблица 1. «Genetic\_algorithm.py»

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| \_\_init\_\_ | Инициализирует графический интерфейс, создаёт все элементы окна, устанавливает начальные параметры алгоритма. | None |
| \_add\_parameter\_input() | Добавляет поле ввода с меткой для ввода числового параметра алгоритма. | None |
| \_add\_checkbox() | |  | | --- | |  |  |  | | --- | | Добавляет чекбокс с меткой для выбора логических параметров алгоритма (например, целочисленное кодирование). | | None |
| \_add\_label() | Добавляет текстовую метку на интерфейс. | None |
| fitness\_function(x1, x2) | Вычисляет значение функции для заданных значений x1 и x2 | float или int |
| generate\_population(size, min\_val, max\_val) | Создаёт начальную популяцию особей (массив хромосом) в заданном диапазоне значений. | Массив numpy с координатами (x1, x2) для каждой особи. |
| select\_parents(population, fitness\_scores) | Выбирает двух родителей для скрещивания на основе турнирного отбора (с учётом модифицированного выбора, если включён). | Массив numpy из двух выбранных родителей. |
| crossover(p1, p2, crossover\_prob=0.9) | Выполняет операцию кроссовера между двумя родителями с заданной вероятностью. Если включена целочисленная кодировка, результат округляется. | Массив numpy, представляющий нового потомка. |
| mutate(individual, mutation\_rate, min\_val, max\_val) | Мутирует одну особь с заданной вероятностью. Изменяет одно случайное значение в пределах диапазона, с учётом текущей кодировки. | Массив numpy, представляющий мутированного потомка |
| genetic\_algorithm() | |  | | --- | |  |  |  | | --- | | Основной цикл генетического алгоритма: генерирует популяцию, вычисляет приспособленность, отбирает родителей, скрещивает, мутирует и обновляет популяцию. | | None |
| reset\_data() | |  | | --- | |  |  |  | | --- | | Сбрасывает данные программы (очищает популяцию, историю, лучшие значения и возвращает интерфейс к исходному состоянию). | | None |
| \_update\_results(gen) | |  | | --- | |  |  |  | | --- | | Обновляет интерфейс с информацией о текущем состоянии алгоритма, включая лучшие значения и историю поколений. | | None |

# Рекомендации пользователя

1. Перед запуском алгоритма заполните параметры:

* Мутация (%): задайте вероятность мутации в процентах. Рекомендуется начинать с 5–15%.
* Число особей: определите количество хромосом в популяции (например, 20–50).
* Мин. значение гена и Макс. значение гена: задайте диапазон возможных значений генов.
* Число поколений: укажите максимальное количество итераций для алгоритма.

Убедитесь, что диапазоны генов и другие параметры настроены в соответствии с вашей задачей оптимизации.

1. **Выбор кодировки**

* Для задач с дискретными (целочисленными) решениями активируйте опцию Целочисленная кодировка.
* Для задач с непрерывными значениями оставьте эту опцию отключенной.

1. **Использование модифицированного отбора**  
   Если требуется повысить вероятность выбора как лучших, так и худших хромосом в процессе селекции, включите опцию Использовать модифицированный отбор. Это может быть полезно для предотвращения застревания алгоритма в локальных минимумах.
2. **Запуск алгоритма**  
   Нажмите кнопку Запустить алгоритм, чтобы инициировать процесс оптимизации. Программа начнет рассчитывать значения функций для каждой особи, отображая информацию о лучших результатах и прогресс по поколениям.
3. **Просмотр результатов**

* Поля x1, x2, и Значение функции отображают текущие лучшие найденные значения.
* Таблица справа позволяет просмотреть историю поколений, включая координаты хромосом и их значение приспособленности.

1. **Очистка данных**  
   Для повторного использования программы с новыми параметрами нажмите Очистить. Это сбросит текущее состояние популяции и результаты, сохранив заданные параметры алгоритма.
2. **Рекомендации по настройке параметров**

* При выборе больших значений для числа особей или поколений алгоритм может работать медленнее, но это увеличивает вероятность нахождения оптимального решения.
* Регулируйте вероятность мутации для баланса между исследованием новых решений и сохранением текущих лучших решений.

1. **Анализ результатов**  
   Сравните лучшие найденные значения функции при разных параметрах и типах кодировок для анализа эффективности алгоритма.

# Рекомендации программиста

1. **Требования для корректной работы программы:**

* **Python:** Убедитесь, что установлена версия Python 3.12.0 или выше.
* **Библиотеки:** Проверьте наличие следующих библиотек:
  + numpy (для работы с массивами и вычислений),
  + tkinter (для графического интерфейса),
  + random (для генерации случайных чисел).

1. **Подготовка:**

* Скачайте исходный код программы и убедитесь, что все файлы находятся в одной директории.
* Запустите программу через терминал с помощью команды: **python Genetic\_algorithm.py** или с помощью IDE (например VScode)

1. **Работа с интерфейсом:**

* Проверьте корректность ввода параметров через графический интерфейс (настройки числа поколений, вероятности мутации, диапазонов значений и других параметров).
* Убедитесь, что галочка "Сохранять историю таблицы" работает корректно: при её отключении история очищается между запусками.

1. **Отладка:**

* Если результаты не соответствуют ожидаемым, проверьте корректность диапазонов ввода и настроек.
* Для анализа работы можно временно включить отладочные сообщения, добавив print() в ключевых местах алгоритма.

1. **Исходный код программы доступен в репозитории**:

<https://github.com/FraaaM/Genetic_Algorithm>

# Контрольный пример

1. **Запуск программы и ввод параметров**

Для работы программы запустите файл Genetic\_algorithm.py. Программа открывает графический интерфейс для настройки и запуска генетического алгоритма для функции Розенброка: (100\*(x2 - x1^2)^2 + (1 - x1)^2).

1. **Ввод начальных параметров и запуск алгоритма**

При запуске программы откроется окно с полями для ввода параметров генетического алгоритма (Рис. 2). Пользователь может задать параметры алгоритма в соответствующих полях:

* **Вероятность мутации (%)** — задает вероятность случайного изменения генов в процессе мутации. Значение задается в процентах (например, 15%).
* **Количество хромосом** — определяет размер популяции, участвующей в процессе оптимизации. Рекомендуется выбирать значения от 10 до 100 для оптимального результата.
* **Минимальное и максимальное значения генов** — задают диапазон, в котором могут находиться значения генов хромосом. Например, [-1; 1] для вещественных значений.
* **Количество поколений** — определяет, сколько итераций выполнит алгоритм. Это значение влияет на точность и время работы программы.
* **Тип кодировки** — выберите тип кодирования генов:
  + **Целочисленная кодировка** — значения генов будут целыми числами, что подходит для дискретных задач.
  + **Вещественная кодировка** — значения генов могут быть дробными, что **Модифицированный отбор** — позволяет использовать динамическое ранжирование родителей для более точного поиска решения.
* **Сохранять историю таблицы** — при включении программа будет отображать данные всех поколений в таблице, что позволяет анализировать процесс эволюции.

После ввода параметров нажмите кнопку **"Запустить алгоритм"**. Программа выполнит расчет и отобразит результаты: значения оптимальных параметров (x1, x2) и минимальное значение целевой функции. Если нужно сбросить данные, нажмите кнопку **"Очистить"**.

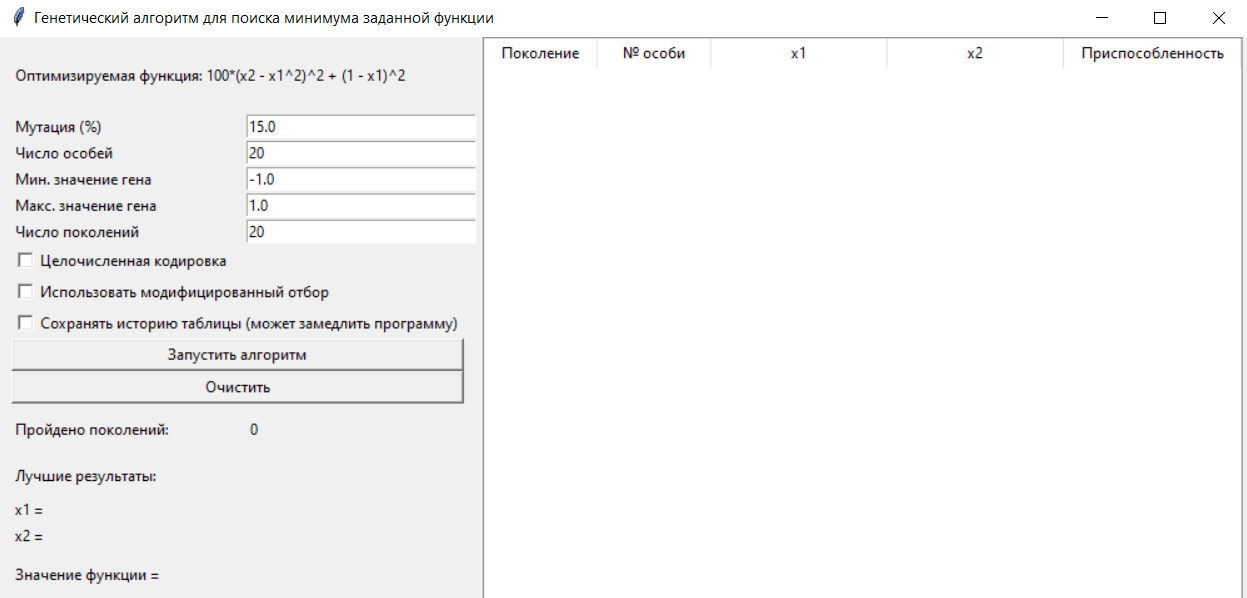


Рис 2. Пример начального окна программы

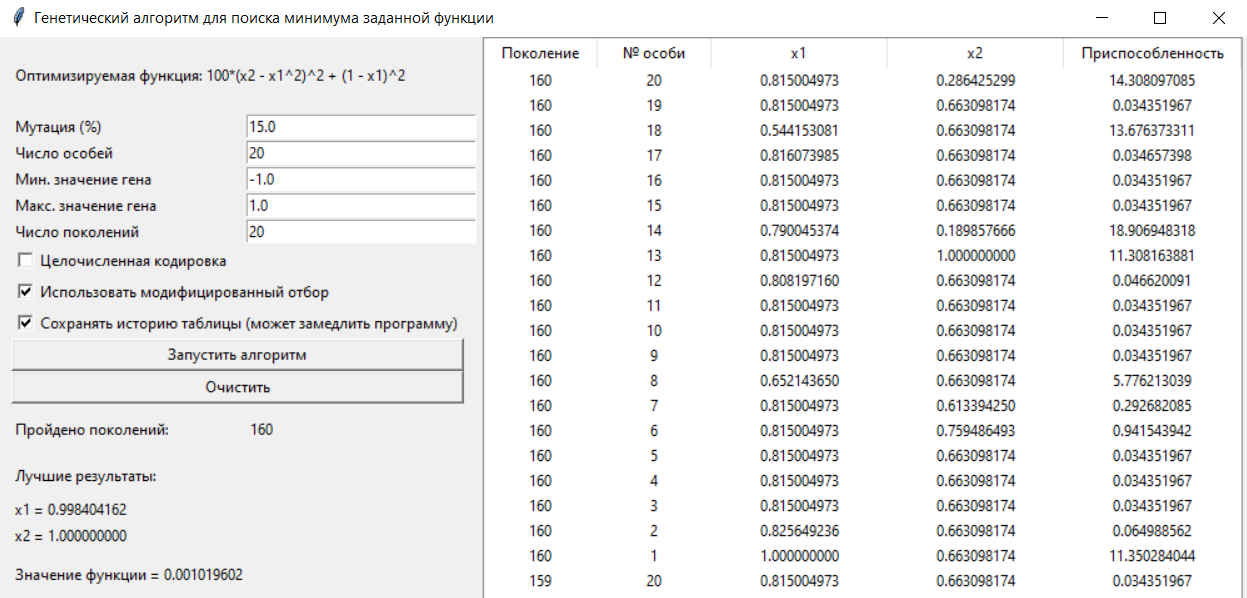


Рис 3. Пример работы программы

1. **Просмотр результатов (Рис 3)**

**После выполнения расчетов на экране отобразятся:**

* Количество пройденных поколений.
* Наилучшее значение функции и переменные, которые её задают.
* Таблица, через которую можно отслеживать вычисленные данные.

# Анализ результатов работы алгоритма и вводных условий

* **Тесты проводились с 10% вероятностью мутации.**
* **Рассмотрим решения без модификации выбора родителя.**

Таблица 2: Тесты вещественного генома с изменением числа поколений

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 20 | 20 | (0.555611617, 0.356846690) | 0.43805685 |
| 100 | 20 | (0.448554946, 0.194380263) | 0.308744526 |
| 120 | 20 | (0.519751766, 0.272902774) | 0.231400609 |
| 440 | 20 | (0.755721498, 0.583081463) | 0.073991650 |

1. Увеличение размера популяции от 20 до 40 приводит к значительному улучшению качества решений: значение целевой функции снижается до **0.02963**.
2. Однако дальнейшее увеличение популяции (до 60 и 100) не приводит к значительному улучшению результата, а в некоторых случаях даже ухудшает значение функции.
3. Оптимальный баланс наблюдается при размере популяции в 40, где достигается наименьшее значение функции. Это может быть связано с достаточным разнообразием популяции и эффективной работой алгоритма.

Таблица 3: Тесты вещественного генома с разным размером популяции

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 100 | 20 | (0.277638147, 0.036961759) | 0.682777568 |
| 100 | 40 | (0.9297739071, 0.688545186) | 0.029631932 |
| 100 | 60 | (0.6983031151, 0.498760451) | 0.103411585 |
| 100 | 100 | (0.7192322394, 0.516406691) | 0.078909400 |

1. **Увеличение размера популяции** от 20 до 40 значительно улучшает результаты: значение функции уменьшается до **0.02963**.
2. Дальнейшее увеличение популяции до 60 и 100 приводит к снижению качества решения. Это может быть связано с увеличением сложности управления большим количеством хромосом при фиксированном числе поколений.
3. Оптимальный размер популяции для данной задачи составляет **40**.

Таблица 4: Тесты бинарного генома с разным количеством поколений

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 10 | 5 | (0, 0) | 1 |
| 20 | 5 | (1, 1) | 0 |
| 30 | 5 | (1, 1) | 0 |
| 40 | 5 | (1, 1) | 0 |

1. Для бинарной кодировки оптимальное решение (значение функции **0**) достигается уже при 20 поколениях.
2. Увеличение числа поколений до 30 и 40 не улучшает результаты, что свидетельствует о стабильности алгоритма для простых задач.
3. Бинарная кодировка более эффективна для данной функции: она быстрее достигает глобального минимума.

Таблица 5: Тесты бинарного генома с разным размером популяции

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 10 | 5 | (0, 0) | 1 |
| 10 | 10 | (1, 1) | 0 |
| 10 | 20 | (1, 1) | 0 |
| 10 | 30 | (1, 1) | 0 |

1. При увеличении размера популяции до 10, решение достигает глобального минимума (**значение функции 0**).
2. Дальнейшее увеличение размера популяции не влияет на результат, так как решение уже найдено.
3. Бинарная кодировка более эффективна для данной функции: она быстрее достигает глобального минимума.

* **Общий вывод для кодировок**

1. **Вещественная кодировка**:
   * Увеличение количества поколений улучшает результаты, однако эффективность снижается после 120 поколений (без модификации).
   * Размер популяции существенно влияет на результат. Оптимум достигается при размере популяции 40.
2. **Бинарная кодировка**:
   * **Для данной функции бинарная кодировка быстрее достигает оптимума, это также относится и к результатам при включенной модификации (Таб. 6 и 7).**
   * Небольшой размер популяции (10–20) и количество поколений (20) достаточны для достижения глобального минимума.
3. **Сравнение кодировок**:
   * Бинарная кодировка лучше подходит для задач с простыми целевыми функциями, тогда как вещественная может быть более универсальной для сложных задач

* **Рассмотрим решения с включённой модификацией выбора родителя**

Модификация выбора родителей с параметром **probablyTheBest** делает генетический алгоритм более гибким и устойчивым к локальным минимумам. Этот подход улучшает баланс между **эксплуатацией** (использованием текущих лучших решений) и **исследованием** (поиском новых решений), что особенно важно для сложных задач оптимизации.

**В данной работе разработан модифицированный отбор: выбирается 80% лучших и 20% худших родителей, для нивелирования локальных минимумов.**

Таблица 6: Тесты веществ. генома с изменением числа поколений (мод.)

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 20 | 20 | (0.698351489, 0.487258471) | 0.091010863 |
| 40 | 20 | (0.709457356, 0.501258874) | 0.084843877 |
| 320 | 20 | (0.968899063, 0.943512671) | 0.003221882 |
| 580+ | 20 | (1.000000000, 1.000000000) | 0.000000000 |

1. С увеличением числа поколений значение в лучшей точке стремится к глобальному минимуму (0). Уже при 320 поколениях достигается решение с высокой точностью (0.003221882), а начиная с 580 поколений зачастую встречается идеальный результат (0.00000000000).
2. Увеличение числа поколений улучшает точность поиска, позволяя алгоритму избежать локальных минимумов. Использование модификации выбора родителей ускоряет достижение глобального оптимума за счёт сохранения генетического разнообразия.

Таблица 7: Тесты веществ. генома с разным размером популяции (мод.)

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 50 | 20 | (0.629196554, 0.387378871) | 0.144736241 |
| 50 | 40 | (0.701241520, 0.489761783) | 0.089645541 |
| 50 | 60 | (0.814088726, 0.657713437) | 0.037090092 |
| 50 | 100 | (1.000000000, 0.998529130) | 0.000216346 |
| 50 | 100+ | (1.000000000, 1.000000000) | 0.000000000 |

1. С увеличением размера популяции значение в лучшей точке стабильно уменьшается. Это связано с тем, что большая популяция повышает шансы на нахождение глобального минимума.
2. При размере популяции 100 алгоритм достигает оптимального решения.
3. Увеличение размера популяции способствует улучшению результатов, однако после достижения определённого порога (в данном случае 100) прирост эффективности становится незначительным.

**Результат работы алгоритма при целочисленной кодировке совпал со значениями, полученными с выключенной модификацией (Таб. 4 и 5)**

* **Общие выводы**

1. **Преимущества модификации:**
   * Метод с выбором 80% лучших и 20% худших родителей способствует сохранению разнообразия в популяции.
   * Гибкость отбора позволяет избежать зацикливания на локальных минимумах, что особенно важно при сложных условиях.
2. **Рекомендации:**
   * Для задач, требующих высокой точности, рекомендуется увеличивать количество поколений (320+), сохраняя умеренный размер популяции (40–60).
   * Для ускорения поиска глобального минимума в задачах с большим пространством решений предпочтителен увеличенный размер популяции (100+).
3. **Сравнение с классическим подходом:**
   * Включение модификации делает алгоритм более быстрым и устойчивым к локальным минимумам.

# Вывод

В ходе проекта был разработан и исследован генетический алгоритм, способный работать с двумя типами кодировки: вещественной и бинарной. Проведённые тесты подтвердили, что эффективность алгоритма существенно зависит от выбранных параметров: размера популяции, числа поколений и типа кодировки генов.

Для вещественного генома показано, что увеличение числа поколений и размера популяции способствует достижению более точного глобального минимума, особенно при использовании модифицированного выбора родителей. Бинарная кодировка, несмотря на ограниченную точность представления данных, показала высокую эффективность на малых размерах популяции и при небольшом числе поколений.

Применение модификации отбора с параметром **probablyTheBest** позволило улучшить баланс между поиском новых решений и использованием текущих лучших хромосом, что особенно важно для задач, склонных к локальным минимумам. Эта модификация показала себя особенно полезной при работе с вещественным геномом, ускоряя достижение глобального оптимума. В процессе тестирования выяснилось, что алгоритм успешно находит глобальный минимум функции Розенброка в тестовых условиях.

Таким образом, проект демонстрирует, как адаптация алгоритма к условиям задачи и внедрение улучшений, таких как модифицированный отбор, могут существенно повысить эффективность оптимизации. Разработанный алгоритм является мощным инструментом для решения задач глобальной оптимизации и может быть расширен для более сложных приложений.

# Источники

* Редактор блок-схем.

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

*дата обращения: (21.11.2024)*

* 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: (21.11.2024)*

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

*дата обращения: (21.11.2024)*