**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 – 6**

**Студентка гр. 23Б15-пу**

**Беляева А.П.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

**Оглавление**

Цель работы...................................................................................................................3

Краткое описание алгоритма ГА на основании лекционного материала.............................................................................................................3

Cхемы пошагового выполнения алгоритма и блок-схемы.......................................................................................................3

Формализация задачи...........................................................................................5

Листинг программы с детальными комментариями..................................................................................................5

Описание контрольного примера.....................................................................................................5

Анализ результатов работы алгоритма........................................................................................7

Рекомендации пользователя........................................................................................

Вывод.............................................................................................................................8

Источники......................................................................................................................8

# Цель работы

Цель данной работы — изучить основы генетического алгоритма (ГА), реализовать его на языке Python для нахождения минимума функции и исследовать влияние различных модификаций (таких как выбор родителей и тип кодирования генов) на результативность алгоритма.

# Краткое описание алгоритма генетического алгоритма

Генетический алгоритм — это метод оптимизации, основанный на принципах естественного отбора и эволюции. ГА начинается с инициализации случайной популяции решений, которые представляют собой возможные значения переменных. Затем он применяет к ним операторы отбора, кроссинговера и мутации для создания нового поколения решений. Алгоритм повторяет эти шаги до достижения заданного числа поколений или условия остановки. Основные этапы ГА включают:

* Инициализация популяции.
* Оценка приспособленности каждого индивидуума.
* Выбор родителей.
* Применение кроссинговера и мутации для создания потомков.
* Замена старой популяции новой.
* Проверка условия остановки (например, достижение заданного числа поколений).

# Схема пошагового выполнения алгоритма и блок-схема

1. Инициализация: Задаем параметры алгоритма и создаем начальную популяцию.
2. Оценка: Вычисляем приспособленность каждого индивидуума (значение целевой функции).
3. Отбор родителей: Выбираем пары родителей для скрещивания.
4. Кроссовер: Создаем потомков, комбинируя гены родителей.
5. Мутация: Случайным образом изменяем гены потомков с заданной вероятностью.
6. Замена: Создаем новую популяцию потомков и оцениваем ее.
7. Условие остановки: Если достигнуто заданное число поколений, завершить алгоритм.

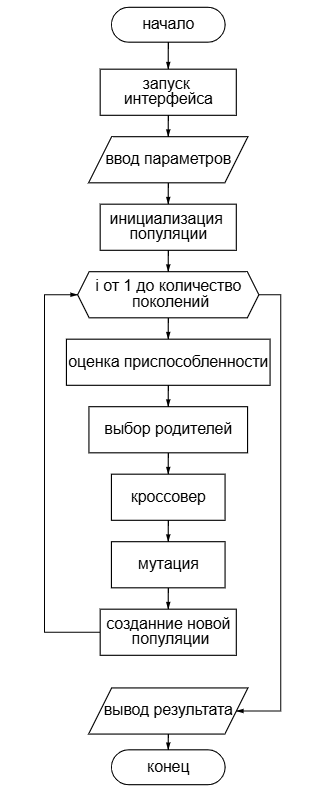


Рис 1. Блок-схема main.py

# Формализация задачи

Целевая функция для минимизации: f(x,y)=(x−2)4+(x−2y)2f(x, y) = (x - 2)^4 + (x - 2y)^2f(x,y)=(x−2)4+(x−2y)2. Задача состоит в нахождении значений переменных x и y, при которых функция достигает минимального значения. Используем язык Python и библиотеку tkinter для создания графического интерфейса. Спецификация программы:

Программа инициализирует популяцию решений.

Позволяет пользователю задавать параметры алгоритма, такие как количество поколений, вероятность мутации и метод выбора родителей.

На каждом поколении программа обновляет таблицу, отображая результаты текущего поколения.

# Листинг программы с детальными комментариями.

```

```

# Описание контрольного примера. Результаты тестирования программы на наборе целевых функций с указанием числа итераций и количества вычислений целевой функции.

1. Запуск программы и ввод параметров

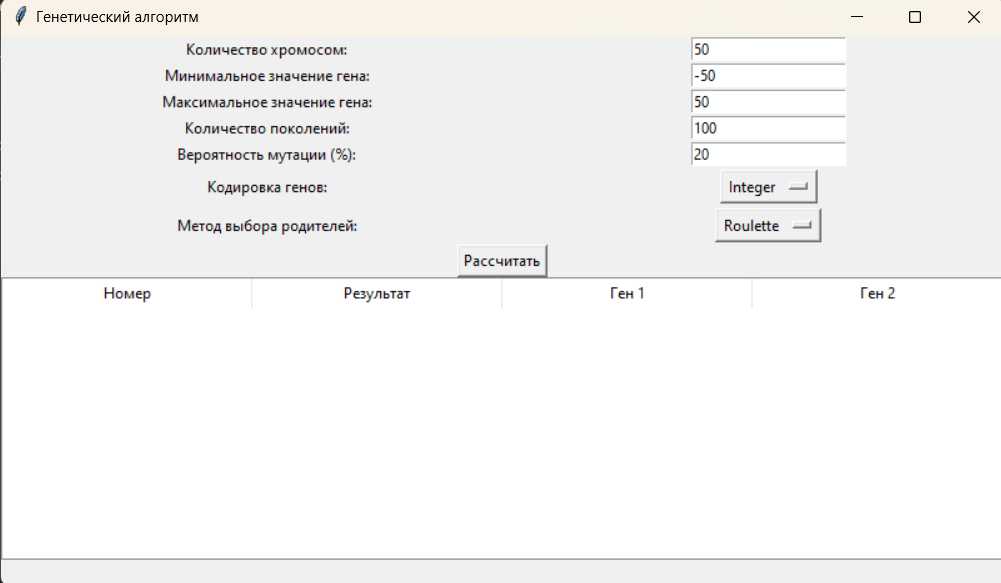


Рис 2. Запуск программы

1. Ввод начальных параметров

При запуске программы откроется окно, в котором можно задать параметры для работы генетического алгоритма (Рис. 2). Введите значения для следующих параметров:

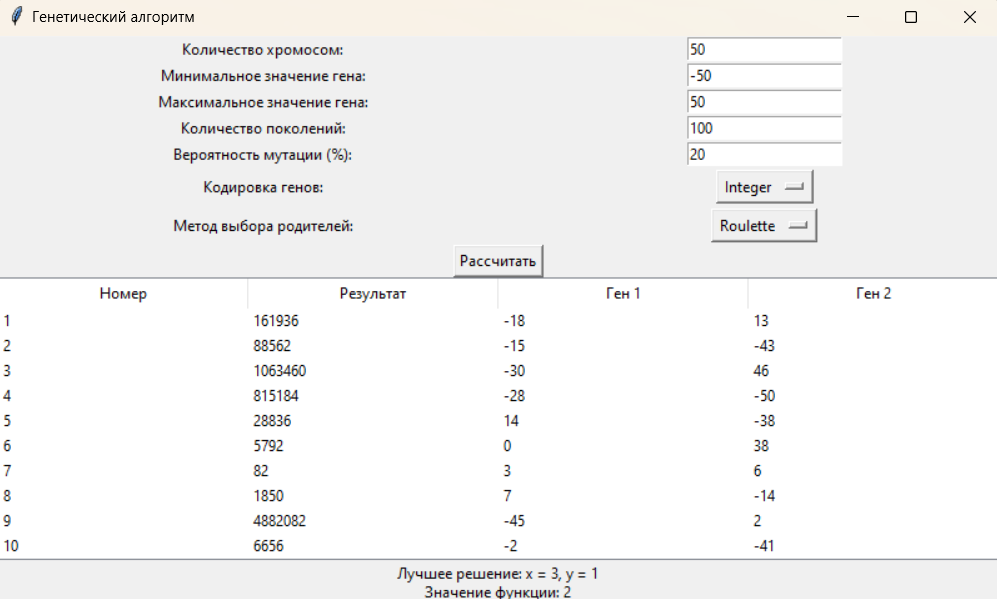
1. Вероятность мутации (%) — определяет, как часто будут изменяться гены.
2. Количество хромосом — задает размер популяции, участвующей в оптимизации.
3. Минимальное и максимальное значения генов — устанавливает диапазон возможных значений для генов.
4. Количество поколений — определяет, сколько раз алгоритм обновит популяцию в процессе поиска оптимального решения.
5. Тип кодировки — выберите, использовать целочисленную или вещественную кодировку для генов.
6. Запуск алгоритма  
   

Рис 3. Пример запуска программы

1. Просмотр результатов

Программа была протестирована на функции f(x,y)=(x−2)^4+(x−2y)^2 при различных значениях параметров, таких как количество поколений, вероятность мутации и методы выбора родителей. Программа достигала минимума функции на 1000 поколениях при вероятности мутации 20%.

# Анализ результатов работы алгоритма и вводных условий

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Кодировка генов | Наилучшее решение | Значение в лучшей точке |
| 50 | int | (2, 0) | 4 |
| 50 | float | (2.1692,1.1138) | 0.0042 |
| 500 | int | (2,1) | 0 |
| 500 | float | (2.1344, 1.0798) | 0.0009 |

Табл 1. Тестирование метода кодирования гена

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Выбор родителя | Наилучшее решение | Значение в лучшей точке |
| 50 | рулетка | (3,1) | 2 |
| 50 | турнир | (2,1) | 0 |
| 500 | рулетка | (1,0) | 2 |
| 500 | турнир | (2,1) | 0 |

Табл 2. Тестирование выбора родителя с int

|  |  |  |  |
| --- | --- | --- | --- |
| Количество поколений | Выбор родителя | Наилучшее решение | Значение в лучшей точке |
| 50 | рулетка | (2.4154,0.9477) | 0.3001 |
| 50 | турнир | (1.9423,0.9720) | 1.42e-05 |
| 500 | рулетка | (1.9699,1.1247) | 0.078 |
| 500 | турнир | (2.1181,1.0596) | 1.19e-05 |

Табл 3. Тестирование выбора родителя с float

Анализ показал, что метод выбора родителей влияет на скорость достижения минимума функции. Турнирный отбор оказался более эффективным по сравнению с рулеткой, так как обеспечивал более стабильный выбор лучших индивидов. Добавление вещественной кодировки также улучшило результат, так как дало большее разнообразие возможных решений.

# Рекомендации пользователя

**Запуск программы**:

* Откройте программу и введите параметры генетического алгоритма.

**Настройка параметров**:

* **Вероятность мутации (%)** — обычно 5-20%, чтобы избежать застревания в локальных минимумах.
* **Количество хромосом** — 20-50 для начала.
* **Диапазон значений генов** — укажите минимальные и максимальные значения генов.
* **Количество поколений** — 50-100 для тестирования; больше для улучшения результатов.
* **Тип кодировки** — целочисленный или вещественный, в зависимости от задачи.

**Выбор метода отбора**:

* Выберите рулетку или турнирный отбор для выбора родителей.

**Запуск алгоритма**:

* Нажмите **«Рассчитать»**, чтобы запустить алгоритм и отслеживать результаты в таблице.

**Анализ результата**:

* Проверьте лучшее найденное решение и значение целевой функции. Если результат неудовлетворителен, настройте параметры и запустите заново.

**Повторный запуск (при необходимости)**:

* Увеличьте количество поколений, измените вероятность мутации или метод отбора, если хотите улучшить результат.

# Рекомендации программиста

Для корректной работы программы убедитесь, что установлены:

* Python версии 3.12.0 или выше;
* Библиотеки: numpy, tkinter, random.

# Вывод

В ходе работы была создана программа, которая позволяет применять генетический алгоритм для поиска минимума функции. Изучение различных методов выбора родителей и типов кодировки генов позволило улучшить результаты.

# Источники

1. Numpy - numpy documentation // Numpy URL: [https://numpy.org/](https://numpy.org/" \t "_new) (дата обращения: 06.11.2024).
2. Tkinter - Python interface to Tcl/Tk // Tkinter URL: [https://docs.python.org/3/library/tkinter.html](https://docs.python.org/3/library/tkinter.html" \t "_new) (дата обращения: 05.11.2024).
3. random - random documentation // Random URL: [https://docs.python.org/3/library/random.html](https://docs.python.org/3/library/random.html" \t "_new) (дата обращения: 05.11.2024).