**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**Факультет прикладной математики – процессов управления**

**отчет**

**по лабораторной работе №5**

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

**на тему «Исследование роевого алгоритма.»**

**вариант**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22.Б16 |  | Гареев К.В. |
| Преподаватель |  | Дик А.Г. |

**Санкт-Петербург**

**2023 г.**

Оглавление

[1.Цель работы 3](#_Toc154455113)

[2. Задачи 3](#_Toc154455114)

[3.Теоритическая часть 3](#_Toc154455115)

[4. Принцип работы алгоритма 4](#_Toc154455116)

[5.Описание функций 5](#_Toc154455117)

[5.1 Описание классов 5](#_Toc154455118)

[5.2 Описание функций 6](#_Toc154455119)

[6.Рекомендации пользователю 9](#_Toc154455120)

[7.Рекомендации программисту 9](#_Toc154455121)

[8. Контрольный пример 10](#_Toc154455122)

[9. Анализ роевого алгоритма и сравнение его с генетическим 10](#_Toc154455123)

[10.Вывод 12](#_Toc154455124)

[11. Листинг 13](#_Toc154455125)

# 1.Цель работы

Основная цель исследования заключается в изучении особенностей алгоритмов роевого интеллекта в контексте решения задач глобальной оптимизации. Данное исследование также предполагает проведение сравнительного анализа этих алгоритмов с генетическим алгоритмом.

# 2. Задачи

Необходимо разработать программу для поиска минимума функции, выбрав тестовую функцию из представленной таблицы. Далее, протестировать данную программу на выбранной тестовой функции и провести сравнительный анализ работы генетического и роевого алгоритмов. Наконец, необходимо подготовить отчет по результатам эксперимента.

# 3.Теоритическая часть

Роевые алгоритмы представляют собой эвристические методы оптимизации, которые черпают вдохновение из природных явлений коллективного поведения, таких как стая птиц, рой пчел или рыб. Основная идея заключается в использовании группы индивидов для эффективного исследования пространства решений в поиске оптимальных значений.

В работе роевого алгоритма для нахождения минимума функции создается и управляется группа частиц в пространстве параметров. Каждая частица представляет собой потенциальное оптимальное решение, а их перемещение в этом пространстве отражает итеративный процесс поиска. Оценка качества каждой частицы осуществляется на основе целевой функции, которую требуется минимизировать.

Роевой алгоритм обладает несколькими параметрами, оказывающими влияние на его эффективность:

- Размер популяции: Определяет количество частиц в рое. Большее число частиц может повысить вероятность нахождения глобального минимума, однако это может увеличить вычислительную сложность.

- Количество поколений: Устанавливает количество итераций, которые выполняет рой. Большее количество поколений может улучшить точность, но также увеличивает временные затраты.

- Коэффициент инерции: Определяет влияние предыдущей скорости на текущую. Высокий коэффициент инерции помогает сохранить текущую траекторию частицы, в то время как низкий может ускорить изменение направления.

- Коэффициент собственного лучшего значения: Вес, определяющий влияние лучшей личной позиции частицы на ее движение. Большее значение усиливает влияние собственного опыта.

- Коэффициент глобального лучшего значения: Вес, определяющий влияние лучшей глобальной позиции на движение частицы. Увеличение этого значения усиливает коллективное воздействие роя.

Изменение указанных параметров существенно влияет на сходимость и эффективность алгоритма. Для достижения оптимальных результатов в конкретной задаче оптимизации часто требуется проведение экспериментов и настройка параметров.

# 4. Принцип работы алгоритма

Алгоритм метода роевой оптимизации:

1. **Инициализация**:
   * Задание параметров алгоритма.
   * Создание начальной популяции частиц.
2. **Основной цикл поколений**:
   * Для каждого поколения:
     + Для каждой частицы в популяции:
       - Рассчет значения целевой функции для текущей позиции.
       - Обновление лучшей позиции частицы, если значение целевой функции лучше предыдущего.
       - Обновление глобального лучшего значения и позиции, если текущее значение лучше глобального лучшего.
     + Для каждой частицы в популяции:
       - Обновление скорости частицы, учитывая заданные параметры
       - Обновление позиции частицы, добавляя к текущей позиции новую скорость.
       - Ограничение позиции в пределах заданных границ.
3. **Вывод результатов**:
   * Отображение результатов на графике, показывая распределение популяции и лучшую найденную точку.
   * Вывод значений лучшей найденной позиции и соответствующего значения функции.
4. **Остановка**:
   * Завершение выполнения алгоритма после завершения всех поколений.
5. **Визуализация**:
   * Визуализация полученных результатов и отметка лучшего решения красной точкой на графике.

# 5.Описание функций

# 5.1 Описание классов

Таблица 5.1.1. Описание классов программы

|  |  |
| --- | --- |
| Класс | Описание |
| **ParticleSwarmOptimization (PSO)** | * Класс, представляющий алгоритм роевой оптимизации. * Атрибуты:   + **particles**: Список объектов класса Particle, представляющих частицы в алгоритме.   + **num\_iterations**: Количество итераций алгоритма.   + **min\_range**: Минимальное значение диапазона параметров.   + **max\_range**: Максимальное значение диапазона параметров.   + **inertia\_weight**: Коэффициент инерции.   + **cognitive\_weight**: Коэффициент собственного лучшего значения.   + **social\_weight**: Коэффициент глобального лучшего значения. * Методы:   + **optimize(objective\_function)**: Выполняет оптимизацию с использованием заданной функции. |
| **Particle** | * Класс, представляющий частицу в алгоритме. * Атрибуты:   + **position**: Текущая позиция частицы в пространстве параметров.   + **velocity**: Текущая скорость частицы.   + **personal\_best\_position**: Лучшая позиция частицы.   + **personal\_best\_value**: Лучшее значение функции для частицы. * Методы:   + **new\_velocity(global\_best\_position, inertia\_weight, cognitive\_weight, social\_weight)**: Обновляет скорость частицы.   + **new\_position(global\_best\_position, min\_range, max\_range, inertia\_weight, cognitive\_weight, social\_weight)**: Обновляет позицию частицы.   + **new\_personal\_best(objective\_function)**: Обновляет лучшую позицию и значение функции для частицы. |
| **PSO\_GUI** | * Описание: Класс, представляющий графический интерфейс для взаимодействия с алгоритмом. * Атрибуты:   + Различные виджеты для задания параметров алгоритма. * Методы:   + **perform\_optimization()**: Выполняет оптимизацию с использованием введенных параметров. |

# 5.2 Описание функций

Таблица 5.2.1. Описание функций программы

|  |  |
| --- | --- |
| Функция | Описание |
| **ParticleSwarmOptimization.optimize()** | * Метод класса ParticleSwarmOptimization, выполняющий оптимизацию. * Параметры:   + **objective\_function**: Функция, которую необходимо минимизировать. * Действия:   + Инициализирует глобальные значения лучшей позиции и значения функции.   + Проводит итерации по всем частицам для поиска лучших позиций.   + Возвращает лучшую позицию, лучшее значение функции и историю всех позиций. |
| **Particle.new\_velocity()** | * Метод класса Particle, обновляющий скорость частицы. * Параметры:   + **global\_best\_position**: Глобальная лучшая позиция.   + **inertia\_weight**: Коэффициент инерции.   + **cognitive\_weight**: Коэффициент собственного лучшего значения.   + **social\_weight**: Коэффициент глобального лучшего значения. * Действия:   + Рассчитывает новую скорость частицы на основе коэффициентов и текущих позиций.   + Возвращает новую скорость частицы. |
| **Particle.new\_position()** | * Метод класса Particle, обновляющий позицию частицы. * Параметры:   + **global\_best\_position**: Глобальная лучшая позиция.   + **min\_range**: Минимальное значение диапазона параметров.   + **max\_range**: Максимальное значение диапазона параметров.   + **inertia\_weight**: Коэффициент инерции.   + **cognitive\_weight**: Коэффициент собственного лучшего значения.   + **social\_weight**: Коэффициент глобального лучшего значения. * Действия:   + Обновляет позицию частицы на основе текущей скорости.   + Ограничивает позицию частицы в пределах заданных границ.   + Обновляет текущую позицию частицы. |
| **Particle.update\_personal\_best()** | * Метод класса Particle, обновляющий лучшую позицию частицы. * Параметры:   + **objective\_function**: Функция, которую необходимо минимизировать. * Действия:   + Рассчитывает текущее значение функции.   + Если значение лучше предыдущего, обновляет лучшую позицию и значение функции. |
| **PSO\_GUI.perform\_optimization()** | Метод класса PSO\_GUI, запускающий процесс оптимизации на основе параметров ввода.   * Действия:   + Извлекает параметры алгоритма из элементов виджетов.   + Запускает оптимизацию с использованием введенных параметров.   + Отображает результаты оптимизации на графике в графическом интерфейсе. |

# 6.Рекомендации пользователю

Поле *«Функция»* позволяет выбрать математическую функцию из выпадающего списка. Поля *«Количество частиц»*, *«Коэффициент инерции»*, *«Коэффициент собственного лучшего значения»*, *«Коэффициент глобального лучшего значения»*, *«Количество итераций»* — позволяют настроить параметры роевого интеллекта.

Кнопка *«Вычислить»* отвечает за запуск работы программы. Далее лучшее решение и значение выводится внизу интерфейса, а также происходит итеративный вывод в график роя частиц.

# 7.Рекомендации программисту

Для запуска программы необходим Python версии не ниже 3.10.6, а также 64-битная операционная система Windows/Linux/macOS. Предварительно необходимо установить библиотеку numpy версии не ниже 1.24.2. Для работы с кодом необходим PyCharm версии не ниже 2022.2.1.

Минимальное необходимое место на диске: 0.5 МБ. Минимальное необходимое количество оперативной памяти: 100 МБ.

# 8. Контрольный пример

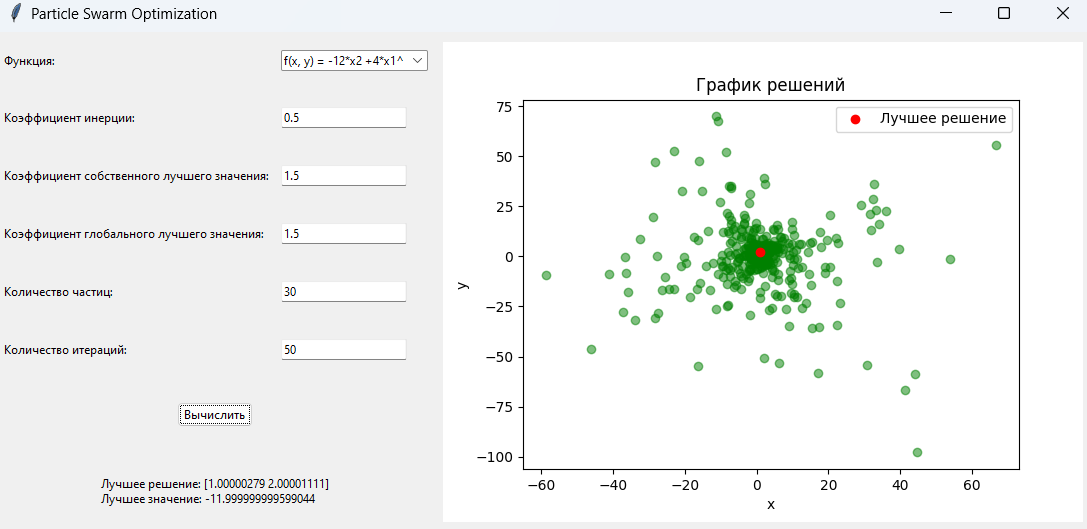


Рисунок 8.1. Пример результата работы алгоритма

# 9. Анализ роевого алгоритма и сравнение его с генетическим

Тесты проводятся при значении коэффициента инерции равным 0.5. Коэффициенты собственного и глобального лучших значений равны 1.5

Таблица 9.1. Тесты алгоритма роя

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Количество частиц | Лучшее решение | Лучшее значение |
| 10 | 50 | (1,77;2,77) | -9,642 |
| 30 | 50 | (0,99;1,99) | -11,999 |
| 50 | 50 | (1;2) | -12 |

Таблица 9.2 Тесты алгоритма роя

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Количество частиц | Лучшее решение | Лучшее значение |
| 50 | 10 | (0,99;1,99) | -11,999 |
| 50 | 30 | (1;2) | -12 |
| 50 | 50 | (1;2) | -12 |

Анализируя полученные результаты можно сделать вывод, что при большом количестве итераций и даже при небольшом количестве частиц, сходимость будет быстрее чем при маленьком количестве итераций и большом количестве частиц

Сравнение генетического и роевого алгоритма:

Для генетического алгоритма будет использоваться вещественное кодирование.

Таблица 9.3 Сравнение генетического и роевого алгоритма

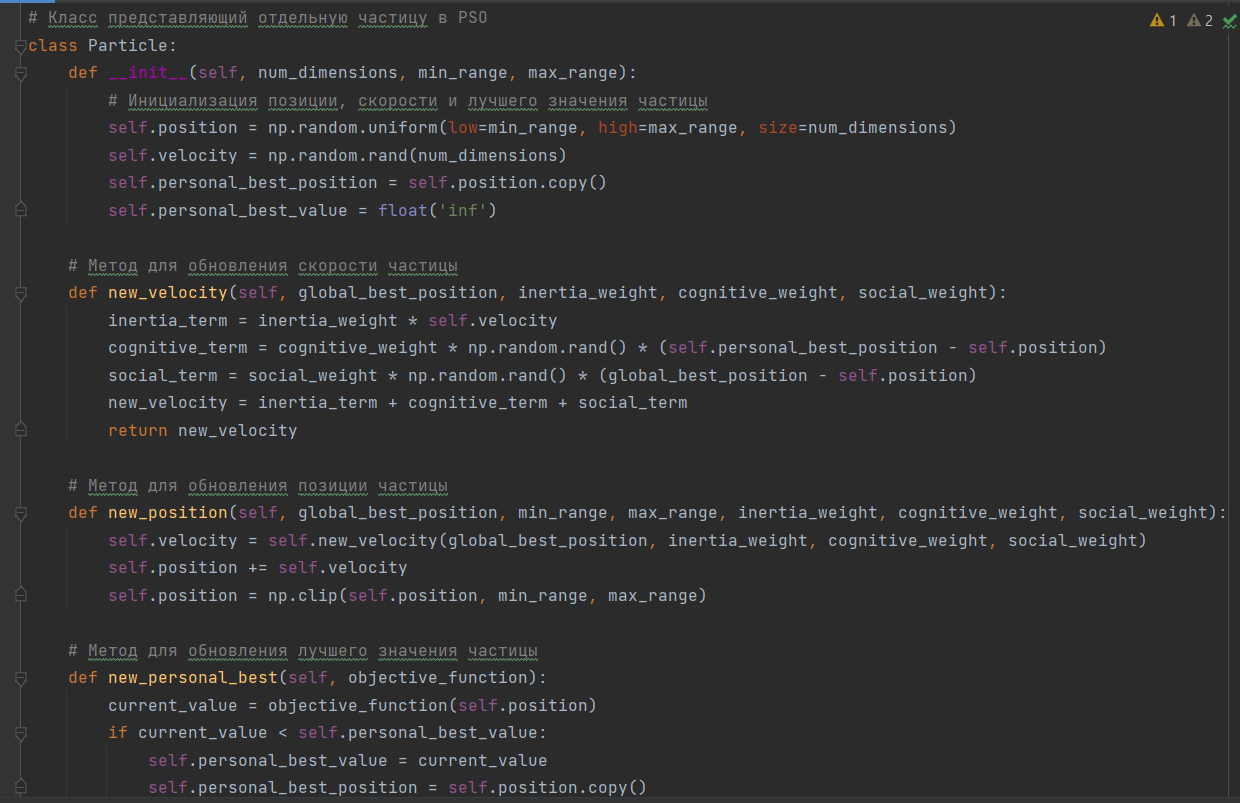
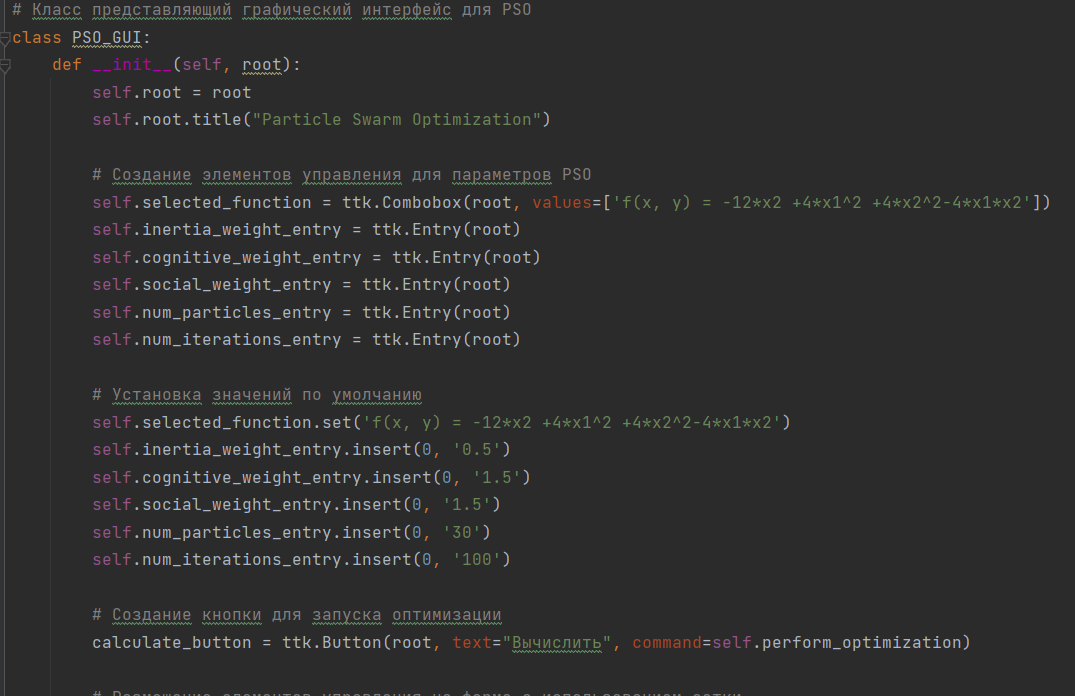
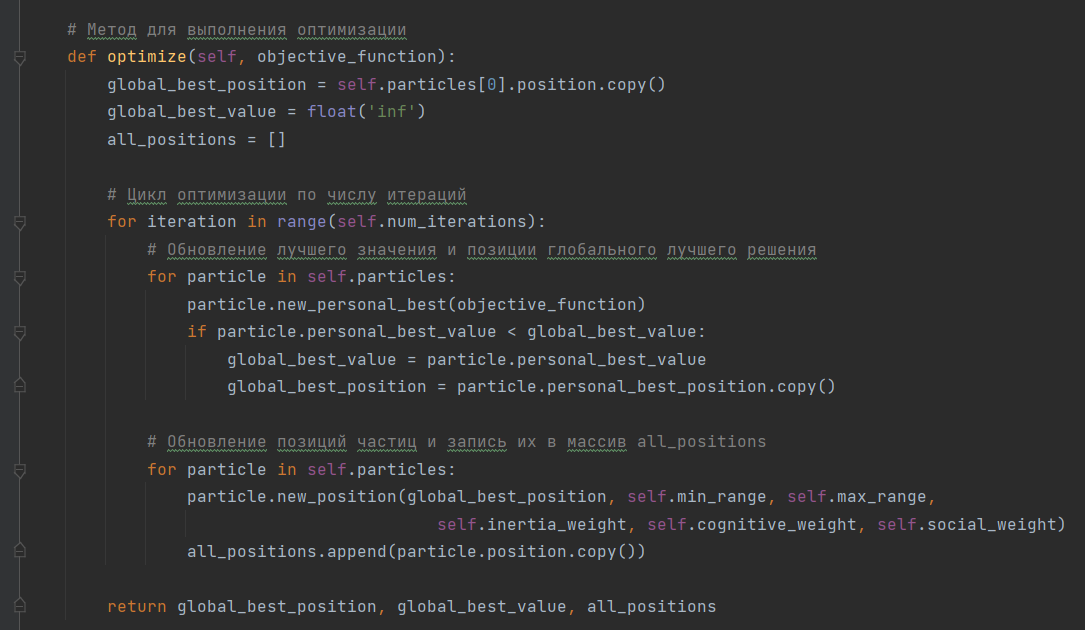
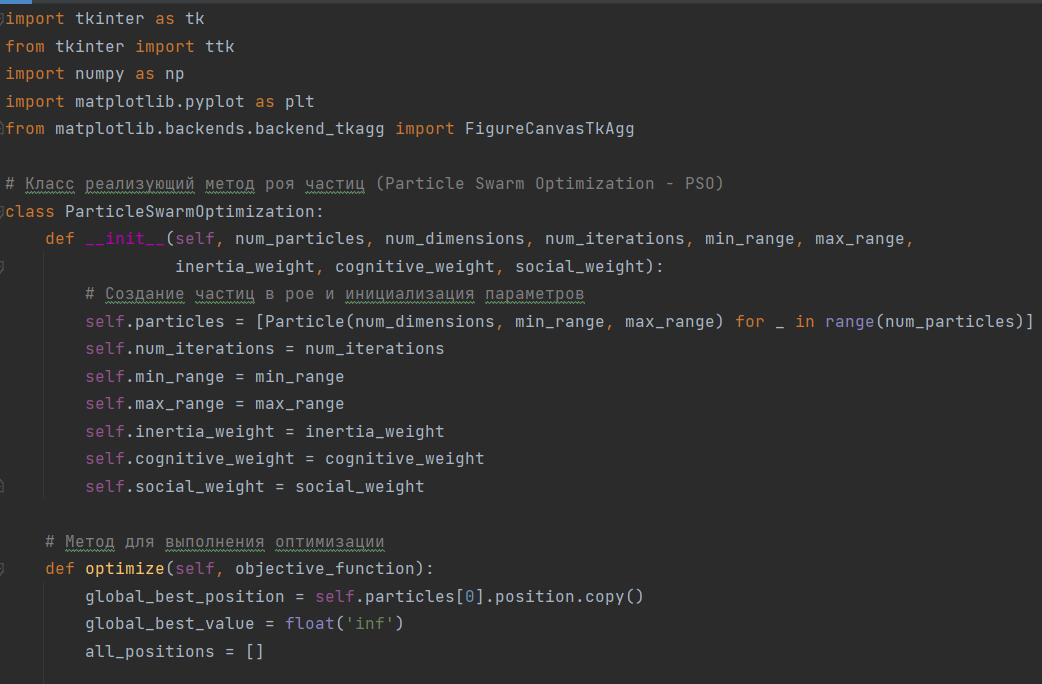
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество итераций | Лучшее решение генетика | Лучшее решение рой | Лучшее значение генетика | Лучшее значение рой |
| 4 | (1,85;0,96) | (-1,46;-1,07) | -1,228 | 19,75 |
| 8 | (0,71;0,99) | (0,75;1,89) | -8,722 | -11,819 |
| 12 | (1,35;2,36) | (1,22;1,96) | -11,489 | -11,77 |

После сравнительного анализа генетического и роевого алгоритмов с использованием популяции в 30 особей и параметров по умолчанию выявлены определенные закономерности в их эффективности. На первых трех итерациях генетический алгоритм демонстрирует более высокие результаты по сравнению с роевым. Это может быть связано с тем, что в начальной стадии генетический алгоритм быстрее сходится к локальным оптимумам, обладая более эффективной стратегией. Однако, на более поздних итерациях роевой алгоритм начинает превосходить генетический, достигая более низких значений функции в оптимальных точках. Возможно, это объясняется тем, что роевой алгоритм более эффективно избегает застревания в локальных оптимумах и обеспечивает более полное исследование пространства поиска решений. Таким образом, каждый из методов демонстрирует свои преимущества на различных этапах оптимизации. Выбор между генетическим и роевым алгоритмами может зависеть от конкретных требований задачи и особенностей оптимизируемой функции. Генетический алгоритм может быть более эффективным на начальных этапах из-за быстрой сходимости, в то время как роевой алгоритм проявляет свои преимущества при продолжительной оптимизации за счет более полного исследования пространства поиска решений.

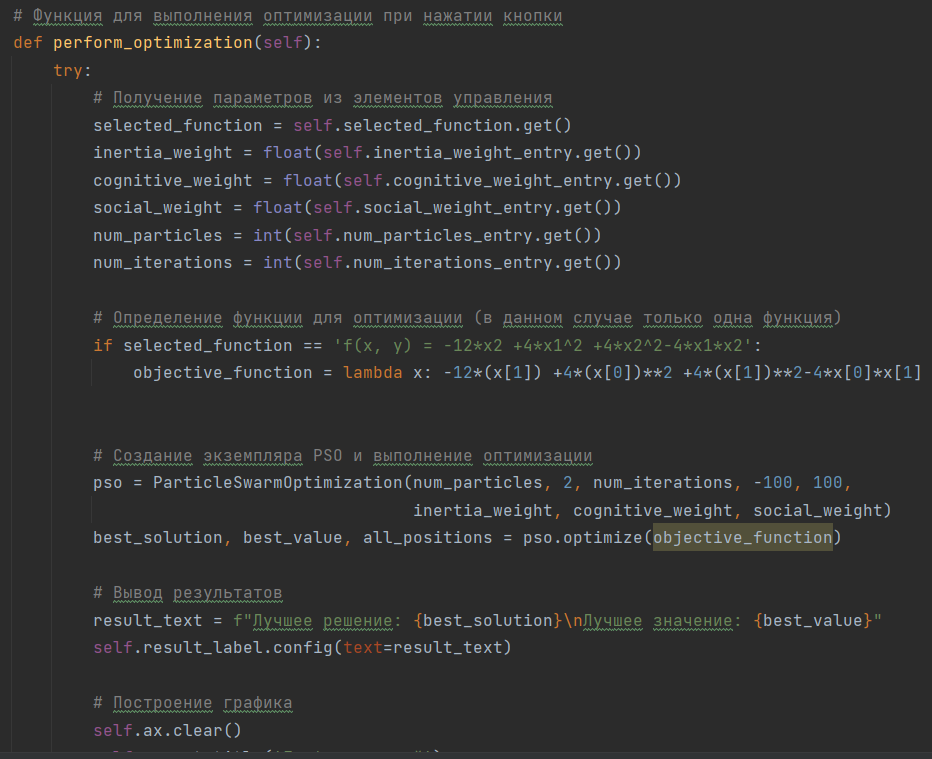
# 10.Вывод

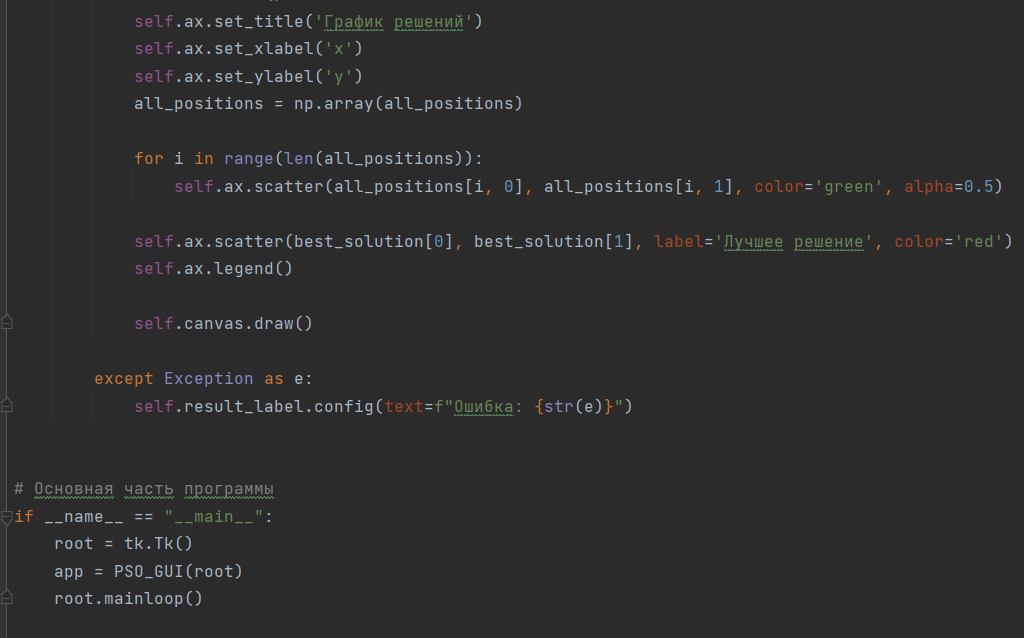
В ходе исследования особенностей алгоритмов роевого интеллекта и их сопоставительного анализа с генетическим алгоритмом были обнаружены важные моменты. Роевой алгоритм продемонстрировал свою эффективность как метод глобальной оптимизации, достигая сходимости к оптимальным решениям при увеличении числа итераций и размера популяции. В ходе сравнения с генетическим алгоритмом выявлено, что на начальных этапах оптимизации генетический алгоритм демонстрирует более высокую эффективность. Однако с увеличением числа итераций роевой алгоритм превосходит его, обеспечивая более низкие значения функции в оптимальных точках. Выводы подчеркивают значимость выбора алгоритма, исходя из конкретных требований задачи, и выявляют уникальные преимущества каждого метода в различных условиях оптимизации.

# 11. Листинг









11.1. Листинг программы