**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Кафедра “фундаментальная информатика и информационные технологии”**

**ОТЧЕТ**

**по лабораторной работе №5**

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

**на тему «Разработка и реализация алгоритма роевого интеллекта для решения задач глобальной оптимизации»**

**Вариант – 5**

**Студент гр. 22Б16-пу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Олизько С.С.**

**Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дик А.Г.**

**Санкт-Петербург**

**2023 г**

Оглавление

[**Цель работы:** 3](#_Toc153215487)

[**Задача:** 3](#_Toc153215488)

[**Теоритическая часть:** 3](#_Toc153215489)

[**Алгоритм метода:** 5](#_Toc153215490)

[**Описание программы:** 6](#_Toc153215491)

[**Рекомендации для пользователя:** 6](#_Toc153215492)

[**Контрольный пример:** 8](#_Toc153215493)

[**Анализ результатов работы алгоритма и вводных условий:** 8](#_Toc153215494)

[**Вывод:** 10](#_Toc153215495)

[**Источники:** 10](#_Toc153215496)

[**Листинг программы с детальными комментариями:** 11](#_Toc153215497)

# **Цель работы:**

Исследование особенностей алгоритмов роевого интеллекта для решения задач глобальной оптимизации и сравнение с генетическим алгоритмом.

# **Задача:**

Изучить особенности работы роевого алгоритма. Написать программу поиска минимума функции, выбрав вариант тестовой функции из таблицы. Протестировать программу на выбранной тестовой функции. Сравнить работу генетического и роевого алгоритмов. Написать отчет.

# **Теоритическая часть:**

Роевые алгоритмы представляют собой эвристические методы оптимизации, вдохновленные природным поведением коллективов в природе, таких как стаи птиц, рой пчел или рыб. Основная идея заключается в том, что коллективное взаимодействие множества индивидов может привести к эффективному исследованию пространства решений в поиске оптимальных значений.

Для поиска минимума функции роевой алгоритм создает и управляет группой частиц в пространстве параметров. Каждая частица представляет собой кандидата на оптимальное решение, а их движение в пространстве отражает итеративный процесс поиска. Целевая функция определенного вида, которую требуется минимизировать, является основой для оценки качества каждой частицы.

Параметры роевого алгоритма и их влияние:

* **Размер популяции:** Определяет количество частиц в рое. Больший размер популяции может повысить вероятность обнаружения глобального минимума, но при этом увеличит вычислительную сложность.
* **Количество поколений:** Устанавливает количество итераций, которые выполняет рой. Большее количество поколений может улучшить точность, но также увеличит временные затраты.
* **Коэффициент инерции:** Определяет влияние предыдущей скорости на текущую. Высокий коэффициент инерции способствует сохранению текущей траектории частицы, тогда как низкий может способствовать быстрой смене направления.
* **Коэффициент когнитивного влияния:** Вес, определяющий влияние лучшей личной позиции частицы на ее движение. Большее значение увеличит влияние собственного опыта.
* **Коэффициент социального влияния:** Вес, определяющий влияние лучшей глобальной позиции на движение частицы. Увеличение этого значения увеличит коллективное воздействие роя.

Изменение этих параметров может существенно влиять на сходимость и эффективность алгоритма. Эксперименты и настройка параметров обычно требуются для достижения оптимальных результатов в конкретной задаче оптимизации.

# **Алгоритм метода:**

1. **Инициализация:** Задать параметры алгоритма. Создать начальную популяцию частиц.
2. **Основной цикл поколений**:

**Повторить цикл для каждого поколения:**

**Для каждой частицы в популяции:**

* Рассчитать значение целевой функции для текущей позиции.
* Если это значение лучше, чем лучшее ранее, обновить лучшую позицию частицы.
* Если текущее значение лучше, чем глобальное лучшее, обновить глобальное лучшее значение и позицию.

**Для каждой частицы в популяции:**

* Обновить скорость частицы, используя формулу, учитывающую инерцию, когнитивное и социальное влияние.
* Обновить позицию частицы, добавив к текущей позиции новую скорость.
* Произвести ограничение позиции в пределах заданных границ.

1. **Вывод результатов**: Отобразить результаты на графике, показывая распределение популяции и лучшую найденную точку. Вывести значения лучшей найденной позиции и соответствующего значения функции.
2. **Остановка**: Завершить выполнение алгоритма после завершения всех поколений.
3. **Визуализация**: При необходимости, предоставить визуализацию процесса оптимизации, позволяя отслеживать изменение позиций частиц на каждом шаге.

# **Описание программы:**

Программа реализована на языке python 3.10 с использованием следующих

пакетов: tkinter, threading, numpy, matplotlib. В генетическом алгоритме используются 10 функций. В таблицах описание функций генетического алгоритма.

*Таблица 1: функции*

|  |  |  |
| --- | --- | --- |
| **Имя функции** | **Тип возвращаемого значения** | **Описание функции** |
| Particle.\_\_init\_\_ | None | Инициализация частицы. Принимает границы поиска для каждой переменной в виде списка пар (минимум, максимум). |
| objective\_function | float | Целевая функция для оптимизации. Принимает значения переменных x и y, возвращает значение функции в заданных точках. |
| update\_velocity | numpy.ndarray | Обновление скорости частицы. Принимает частицу, глобальную лучшую позицию, коэффициент инерции, когнитивное и социальное влияние. Возвращает новую скорость частицы. |
| run\_swarm\_algorithm | tuple | Запуск алгоритма роя частиц для оптимизации функции. Принимает размер популяции, количество поколений, границы поиска, коэффициент инерции, коэффициенты когнитивного и социального влияния. Возвращает глобальную лучшую позицию, соответствующее значение функции и популяцию (для вывода на график). |
| SwarmAlgorithmGUI.\_\_init\_\_ | None | Инициализация графического интерфейса пользователя для алгоритма роя частиц. Принимает окно приложения master. |
| create\_widgets | None | Создание и размещение элементов управления в графическом интерфейсе. |
| run\_iterations | None | Запуск алгоритма роя частиц с указанным числом итераций. |
| run\_specified\_iterations | None | Запуск алгоритма роя частиц с указанным числом итераций в отдельном потоке. |
| display\_results\_on\_graph | None | Отображение результатов на графике. Принимает глобальную лучшую позицию, значение функции, границы поиска и популяцию. |
| update\_best\_result\_label | None | Обновление меток для лучших результатов. Принимает глобальную лучшую позицию и значение функции. |

# **Рекомендации для пользователя:**

**Запуск программы:**

* Убедитесь, что на вашем компьютере установлен Python.
* Установите необходимые библиотеки: tkinter (для графического интерфейса), numpy, matplotlib и threading. Выполните команду pip install numpy matplotlib.
* Запустите код для активации графического интерфейса и начала использования роевого алгоритма.

**Настройка параметров роевого алгоритма:**

* Введите параметры алгоритма, такие как размер популяции, количество поколений, коэффициент инерции, коэффициент когнитивного влияния, коэффициент социального влияния и границы поиска.
* Удостоверьтесь, что значения параметров корректны перед запуском.

**Запуск роевого алгоритма:**

* Нажмите кнопку "Выполнить итерации", чтобы начать выполнение роевого алгоритма с заданными параметрами.

**Отображение результатов:**

* После завершения выполнения алгоритма, вы увидите лучший результат (координаты точки и значение функции) в соответствующем разделе интерфейса.

**Исходный код программы доступен по ссылке ниже:**

<https://github.com/StephanOlizko?tab=repositories>

# **Контрольный пример:**

*Рис 1: пример окна программы*

# **Анализ результатов работы алгоритма и вводных условий:**

Тесты проводились с коэффициентами инерции 0.5 и коэффициентами когнитивного и социального влияния 1.5 .

*Таблица 2: тесты алгоритма роя*

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 10 | 50 | (5.04848116, 6.05542939) | 0.012474108971066544 |
| 20 | 50 | (4.99944347, 6.00101631) | 0.000002271803393026 |
| 30 | 50 | (4.99993144, 5.99999587) | 0.000000018821671070 |
| 40 | 50 | (4.999994, 6.00002147) | 0.000000000604934810 |

*Таблица 3: тесты алгоритма роя*

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 50 | 10 | (5.00000921, 5.99995735) | 0.0000000021584418 |
| 50 | 20 | (5.00000013, 5.99999997) | 0.0000000000000653 |
| 50 | 30 | (5.00000015, 6.00000072) | 0.0000000000006142 |
| 50 | 40 | (4.99999985, 6.00000011) | 0.0000000000001086 |

Сравнения проводились с дефолтными параметрами и с размером популяции 50. Генетический алгоритм кодировался вещественным геномом.

*Таблица 4: сравнение роевого и генетического алгоритмов*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество итераций | Наилучшее решение г.а | Значение в лучшей точке г.а. | Наилучшее решение роя | Значение в лучшей точке роя |
| 3 | (5.099103169465771, 6.0894321560813705) | 0.04728386333400766 | [5.01981423 6.51909805] | 0.2710332010567511 |
| 6 | (4.917540572847525, 5.88848094781864) | 0.012551213573619246 | [4.91351151 6.0982658 ] | 0.03957720088181515 |
| 9 | (4.968726002218322, 5.757372789566975) | 0.06278021499150524 | [5.00131432 6.02012506] | 0.0004119275991664096 |
| 12 | (4.993684329655814, 6.00083239507777) | 0.00016024364915123016 | [5.01305455 5.97230065] | 0.0014489385120142256 |

**Роевой алгоритм:**

Анализ результатов тестов роевого алгоритма позволяет выделить ключевые зависимости. При увеличении числа итераций (таблица 2) наблюдается сходимость алгоритма, проявляющаяся в значительном улучшении наилучших решений и значений в лучших точках к 40-й итерации. Увеличение размера популяции при фиксированном числе итераций (таблица 3) приводит к общему улучшению результатов, что может быть связано с более интенсивным исследованием пространства поиска более крупными популяциями.

**Сравнение алгоритмов:**

В результате проведенного сравнительного анализа генетического и роевого алгоритмов с использованием дефолтных параметров и популяции размером 50, можно выделить определенные тенденции в их производительности. На первых трех итерациях генетический алгоритм демонстрирует лучшие результаты, превосходя роевой алгоритм. Это может быть связано с тем, что на начальных этапах генетический алгоритм эффективнее и быстрее сходится к локальным оптимумам.

Однако, на более поздних итерациях (6, 9 и 12), роевой алгоритм начинает проявлять свою превосходство, приводя к более низким значениям функции в лучших точках. Возможно, роевой алгоритм лучше справляется с проблемой выхода из локальных оптимумов и обеспечивает более эффективное исследование пространства поиска.

Таким образом, на разных этапах оптимизации каждый из методов демонстрирует свои преимущества, и 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, в то время как роевой алгоритм может обеспечивать лучшие результаты при продолжительной оптимизации.

# **Вывод:**

В ходе исследования особенностей алгоритмов роевого интеллекта и их сравнительного анализа с генетическим алгоритмом были выявлены ключевые моменты. Роевой алгоритм проявил себя как эффективный метод глобальной оптимизации, демонстрируя сходимость к оптимальным решениям при увеличении числа итераций и размера популяции. Сравнение с генетическим алгоритмом выявило, что на начальных этапах оптимизации генетический алгоритм показывает более высокую эффективность, однако с увеличением числа итераций роевой алгоритм превосходит его, предоставляя более низкие значения функции в лучших точках. Выводы подчеркивают важность выбора алгоритма в зависимости от конкретных требований задачи и подчеркивают уникальные преимущества каждого метода в различных условиях оптимизации.

# **Источники:**

<https://habr.com/ru/articles/105639/>

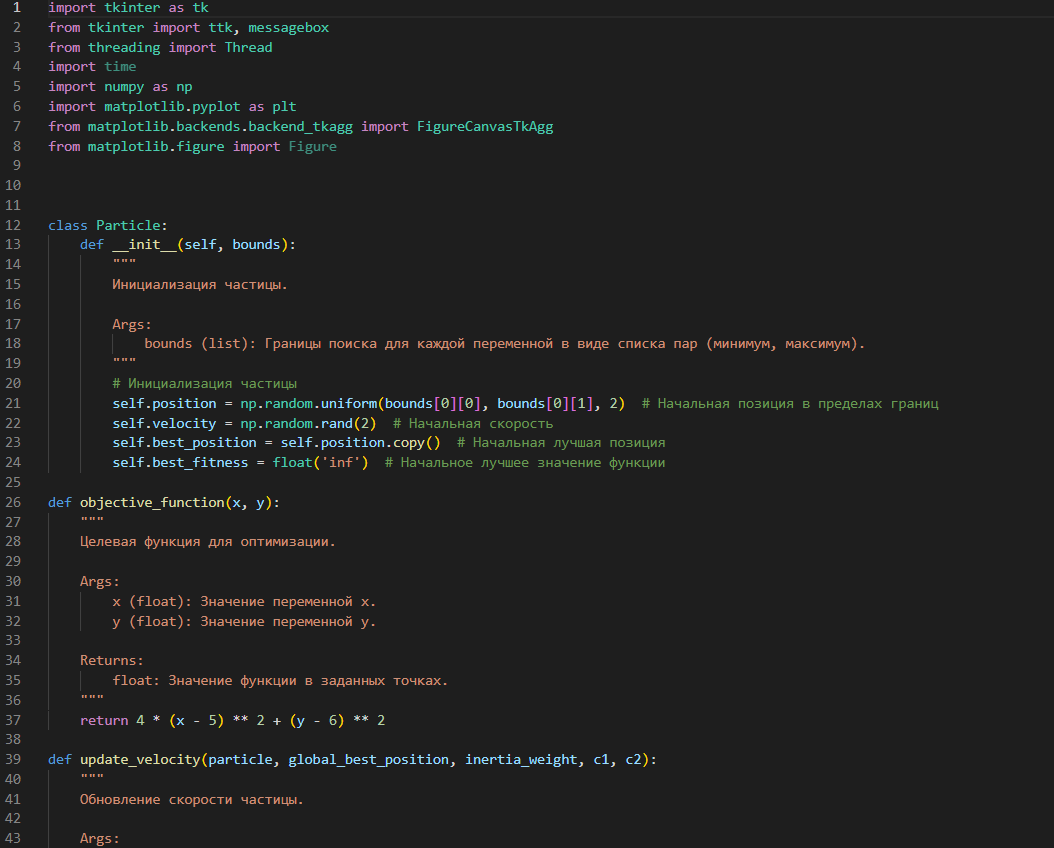
<https://docs.python.org/3/library/tkinter.html>

<https://docs.python.org/3/library/threading.html>

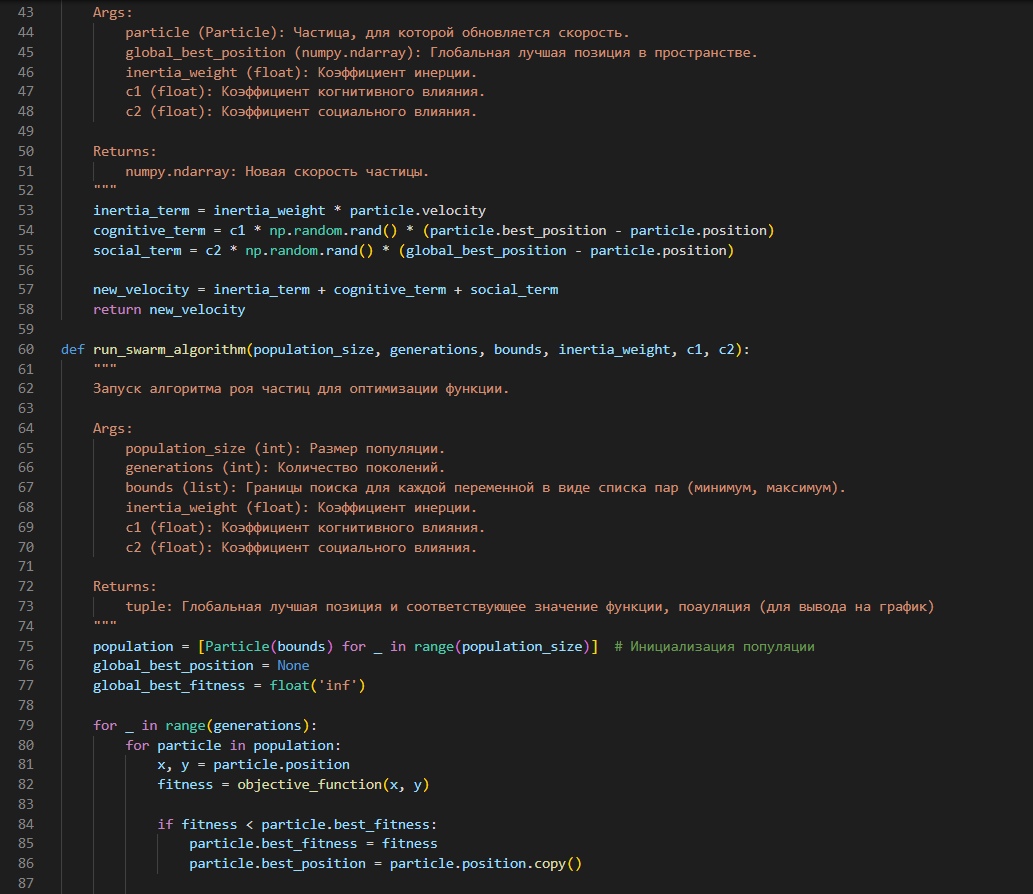
<https://numpy.org/doc/stable/>

<https://matplotlib.org/stable/users/index>

# **Листинг программы с детальными комментариями:**



*Рис 2: листинг*

**

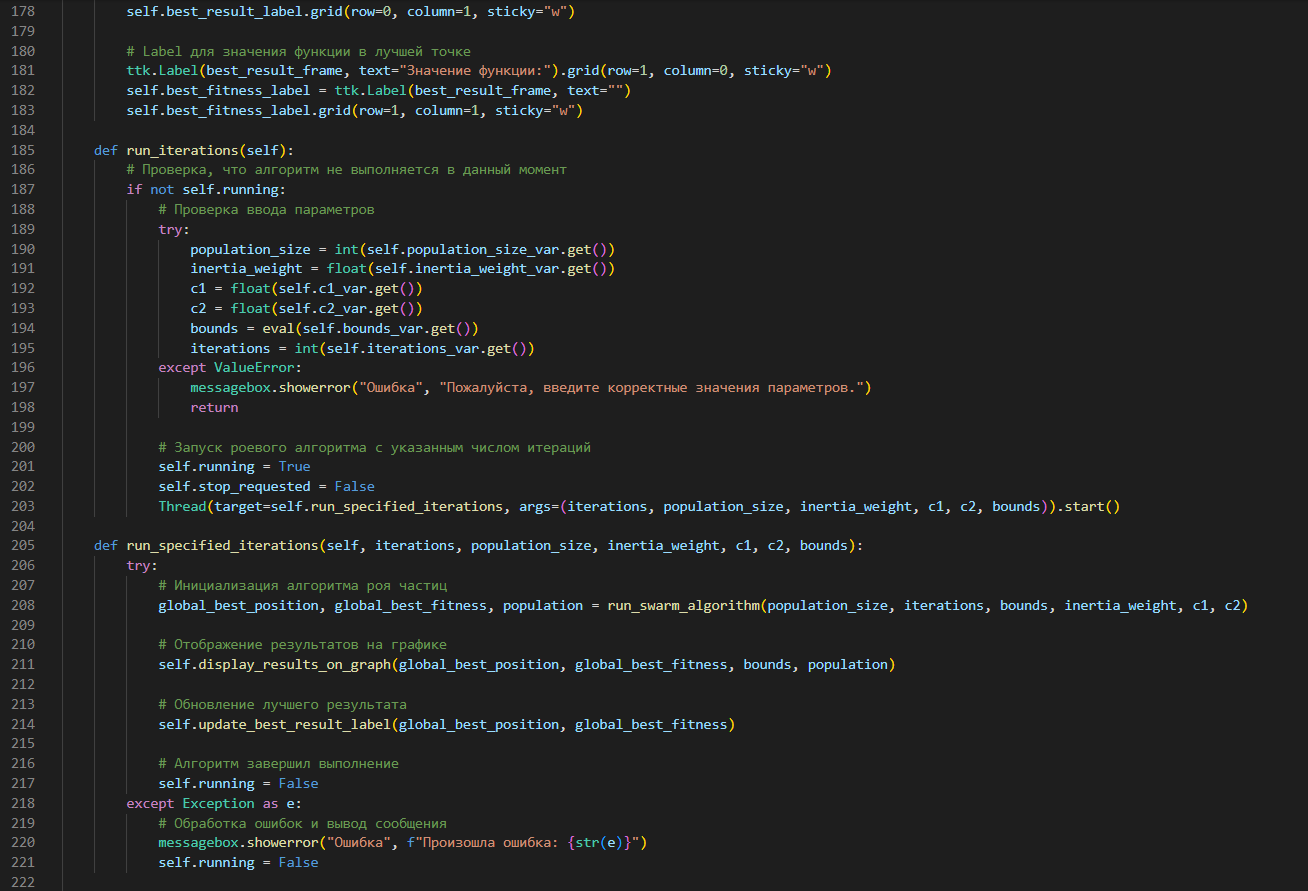
*Рис 3: листинг*

**

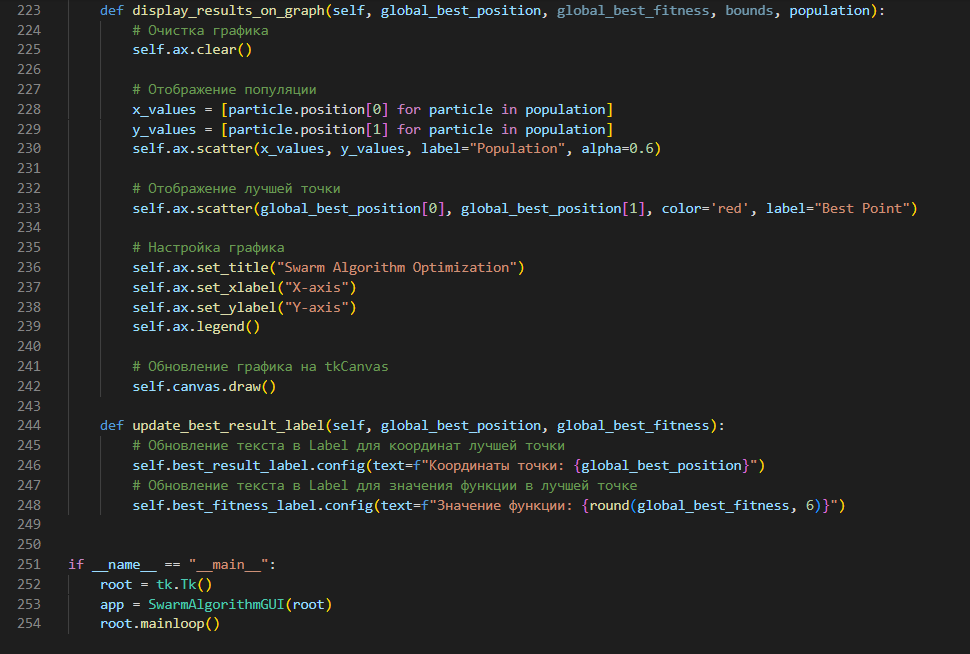
*Рис 4: листинг*

**

*Рис 5: листинг*

**

*Рис 6: листинг*

**

*Рис 7: листинг*