**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**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 – 1**

**Студент гр. 23Б15-пу**

**Абрахин Е.Д.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc181398582)
2. [Описание задачи (формализация задачи) 3](#_Toc181398583)
3. [Теоретическая часть 4](#_Toc181398584)
4. [Основные шаги программы 5](#_Toc181398585)
5. [Блок схема программы 7](#_Toc181398586)
6. [Описание программы 8](#_Toc181398587)
7. [Рекомендации пользователя 9](#_Toc181398588)
8. [Рекомендации программиста 9](#_Toc181398589)
9. [Исходный код программы 9](#_Toc181398590)
10. [Контрольный пример 10](#_Toc181398591)
11. [Анализ результатов работы алгоритма и вводных условий 12](#_Toc181398592)
12. [Вывод 14](#_Toc181398593)
13. [Источники 15](#_Toc181398594)

# Цель работы

Целью данной лабораторной работы является исследование алгоритмов роевого интеллекта (в частности, метода роя частиц) для решения задач глобальной оптимизации, а также их сравнение с генетическими алгоритмами. В ходе работы будет разработана программа, реализующая алгоритм роя частиц (Particle Swarm Optimization, PSO) с возможностью применения модификации "Инерция веса". Программа будет протестирована на заданной тестовой функции для анализа эффективности PSO и влияния модификации " Инерция веса " на скорость нахождения оптимального решения.

# Описание задачи (формализация задачи)

1. Изучение особенностей алгоритмов роевого интеллекта:

Исследовать основные принципы работы алгоритмов роя частиц, их применение для поиска глобальных экстремумов, а также роль различных параметров в эффективности поиска. Ознакомиться с особенностями модификации "Инерция веса" и её влиянием на сходимость и результаты работы алгоритма.

1. Разработка программы для поиска минимума функции:

Написать программу, реализующую алгоритм роя частиц с возможностью использования "Инерция веса" модификации для оптимизации заданной функции.

1. Тестирование программы на выбранной тестовой функции:

Провести тестирование программы на функции x1^2+3⋅x2^2+2⋅x1⋅x2 из таблицы тестовых функций. Собрать результаты для анализа влияния модификации и параметров алгоритма на скорость нахождения оптимального решения.

1. Анализ результатов:

Сравнить полученные результаты для алгоритма роя частиц с "Инерция веса" модификацией и без неё, оценить влияние модификации и определить, какой подход демонстрирует лучшие результаты на тестовой функции.

# Теоретическая часть

Алгоритмы роя частиц

Алгоритм роя частиц (Particle Swarm Optimization, PSO) — это метод оптимизации, основанный на моделировании поведения роя частиц (таких как рой птиц или стая рыб). Алгоритм использует множество частиц, которые перемещаются по пространству поиска, стремясь к наиболее выгодным позициям. Каждая частица хранит своё текущее положение, скорость, а также лучшую позицию, в которой она когда-либо находилась (личный опыт). При этом частицы учитывают и лучшее глобальное положение, достигнутое всеми частицами, что позволяет алгоритму находить оптимальные или близкие к оптимальным решения для сложных задач.

Параметры алгоритма и их роль

1. Коэффициент текущей скорости (velocityFactor):

Умножается на текущую скорость частицы, регулируя её инерцию.

1. Коэффициент личного опыта (personalBestFactor):

Определяет, насколько сильно частица стремится к своему личному лучшему положению.

1. Коэффициент глобального опыта (globalBestFactor):

Определяет степень стремления частицы к глобальному лучшему положению, найденному всей популяцией.

Модификация "Инерция веса"

Модификация "Инерция веса" позволяет алгоритму PSO учитывать предыдущие скорости частиц при обновлении их положений. Это обеспечивает баланс между исследованием и использованием информации о текущем положении, что помогает улучшить сходимость и предотвращает преждевременное попадание в локальные минимумы. Инерция веса позволяет частицам сохранять свою скорость, адаптируясь к изменяющимся условиям оптимизации и повышая вероятность нахождения глобального минимума.

Оценка эффективности

Эффективность алгоритма роя частиц оценивается его способностью достигать оптимума за минимальное время. Критерии включают:

* Скорость сходимости: Время или число итераций, необходимых для достижения заданного уровня точности.
* Качество решения: Точность полученного решения относительно истинного глобального минимума.

# Основные шаги программы

PSO-Algorithm.py

1. Запуск программы:

Инициализация графического интерфейса с настройками и элементами управления.

1. Настройка параметров:

Отображение целевой функции: x1^2+3⋅x2^2+2⋅x1⋅x2.

* Пользователь вводит параметры:
* Коэффициент текущей скорости.
* Коэффициент личного лучшего значения.
* Коэффициент глобального лучшего значения.
* Количество частиц.
* Количество итераций.

Возможность выбора использования модификации "Инерция веса".

1. Запуск алгоритма:

При нажатии кнопки "Рассчитать частицы" запускается алгоритм оптимизации роя частиц (PSO).

1. Инициализация частиц:

Создание начальной популяции частиц в случайных позициях в заданном диапазоне значений.

Назначение начальных скоростей частиц.

Определение начального личного и глобального лучшего значения.

1. Вычисление функции приспособленности:

Определение фитнес-функции, которая соответствует целевой функции.

Использование текущих значений частиц для вычисления значений фитнес-функции.

1. Обновление частиц:

Применение формулы PSO для обновления скорости и позиции каждой частицы.

Использование коэффициентов для влияния текущей скорости, личного и глобального лучшего положения.

Опциональное обновление глобального лучшего значения с учетом модификации "Инерция веса" (если она включена).

1. Обновление интерфейса:

Обновление визуального отображения текущего положения частиц и глобального лучшего значения на графике.

Обновление информации о количестве пройденных итераций и текущих значениях глобально оптимальных x1, x2 и фитнес-функции.

1. Отображение результатов:

Отображение в интерфейсе текущего положения глобально лучшей частицы и значений её координат (x1 и x2), а также соответствующего значения целевой функции.

# Блок схема программы

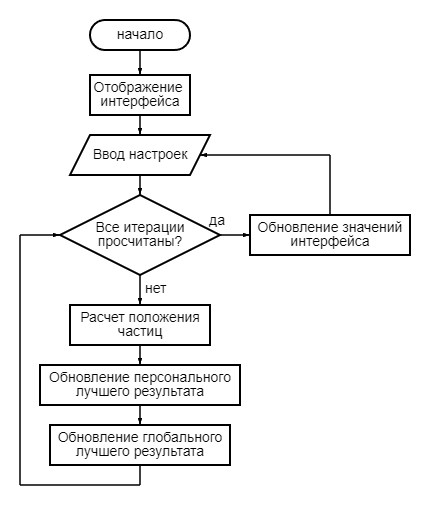


Рис 1. Блок-схема PSO-Algorithm.py

# Описание программы

Программа написана на языке Python 3.12.6 и использует библиотеки numpy [[1]](#_Источники), tkinter [[2]](#_Источники), random [[3]](#_Источники) и matplotlib [[4]](#_Источники) для реализации графического интерфейса решения оптимизационной задачи с использованием алгоритма роя частиц (PSO). Основной класс программы ParticleSwarmOptimizationApp включает в себя 7 методов, каждый из которых отвечает за определенный этап работы алгоритма:

Таблица 1. PSO-Algorithm.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| \_\_init\_\_ | Инициализация интерфейса и параметров приложения. | None |
| fitnessFunction | Оценка приспособленности (фитнесс-функция). | float |
| initializeParticles | Инициализация положения и скорости частиц, определение личных и глобальных лучших позиций. | numpy.ndarray |
| updateParticles | Обновление положения и скорости частиц на основе личного и глобального лучших значений. | None |
| plotParticles | Отображение частиц и их текущих положений на графике в окне интерфейса. | None |
| runAlgorithm | Запуск основного цикла оптимизации, обновление результатов и интерфейса с текущим лучшим решением. | None |

# Рекомендации пользователя

Программа позволяет запустить алгоритм оптимизации роя частиц для минимизации функции x1^2+3⋅x2^2+2⋅x1⋅x2 с помощью графического интерфейса.

1. Запустите программу и задайте параметры алгоритма:

* Коэффициент текущей скорости
* Коэффициент личного лучшего значения
* Коэффициент глобального лучшего значения
* Количество частиц
* Число итераций для выполнения алгоритма
* Параметр включения/выключения "Инерция веса" модификации

1. Нажмите "Рассчитать частицы" для запуска алгоритма оптимизации. Положение частиц будет обновляться в зависимости от их личных и глобальных лучших позиций, и все частицы отобразятся на графике.
2. После завершения итераций:

В поле "Лучшие решения" отобразится оптимальное найденное значение функции, а также координаты x1 и x2 для наилучшего решения.

# Рекомендации программиста

Для корректной работы программы убедитесь, что установлены следующие компоненты:

* Python версии 3.12.0 или выше;
* Необходимые библиотеки: numpy, tkinter, random, matplotlib.

Шаги по установке:

1. Разместите все файлы программы в одной директории.
2. Выполните команду для запуска программы:

python PSO-Algorithm.py

1. Убедитесь, что интерфейс запускается корректно и результаты отображаются как ожидалось.

# Исходный код программы

<https://github.com/FasterXaos/Algorithms_and_Data_Structures>

# Контрольный пример

PSO-Algorithm.py

1. Запуск программы и ввод параметров

Для запуска программы откройте файл, содержащий код ParticleSwarmOptimizationApp. Программа откроет графический интерфейс для настройки и запуска алгоритма роя частиц для оптимизации функции x1^2+3⋅x2^2+2⋅x1⋅x2.

1. Ввод начальных параметров

При запуске программы отобразится окно с полями ввода параметров алгоритма (Рис. 2).

Введите значения для следующих параметров:

* Коэффициент текущей скорости — отвечает за влияние текущей скорости на перемещение частиц.
* Коэффициент собственного лучшего значения — определяет, насколько сильно частицы будут следовать за своими наилучшими предыдущими позициями.
* Коэффициент глобального лучшего значения — задает, как сильно частицы будут следовать за наилучшей позицией из всей группы.
* Количество частиц — размер популяции, участвующей в процессе оптимизации.
* Количество проходов — количество итераций, которые алгоритм выполнит для нахождения оптимального решения.
* Использовать "Инерция веса" модификацию — флажок, который позволяет использовать дополнительную модификацию алгоритма.

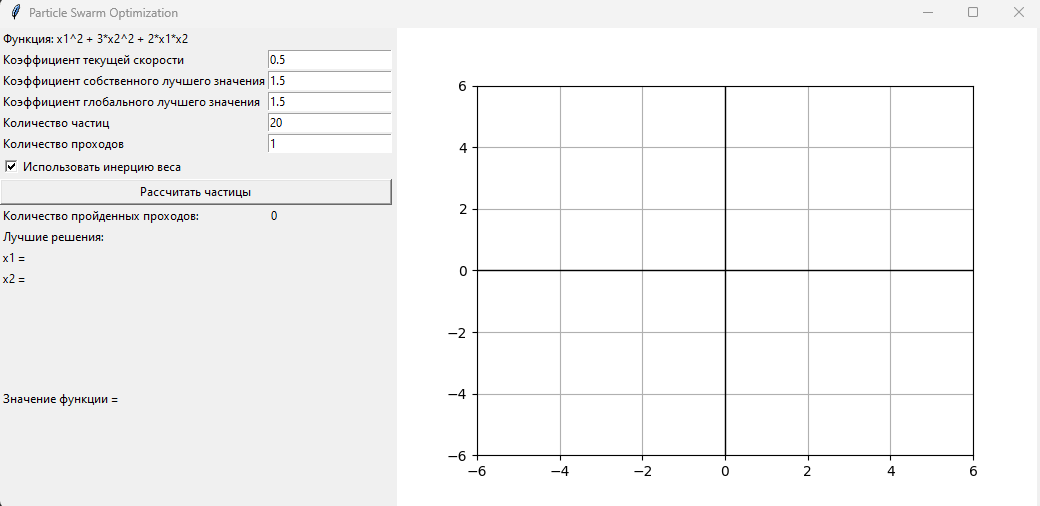


Рис 2. Пример интерфейса программы

1. Запуск алгоритма

После задания всех параметров нажмите кнопку «Рассчитать частицы» для запуска алгоритма. Программа выполнит заданное количество итераций, обновляя положение частиц на каждой итерации, и отобразит результаты (Рис. 3).

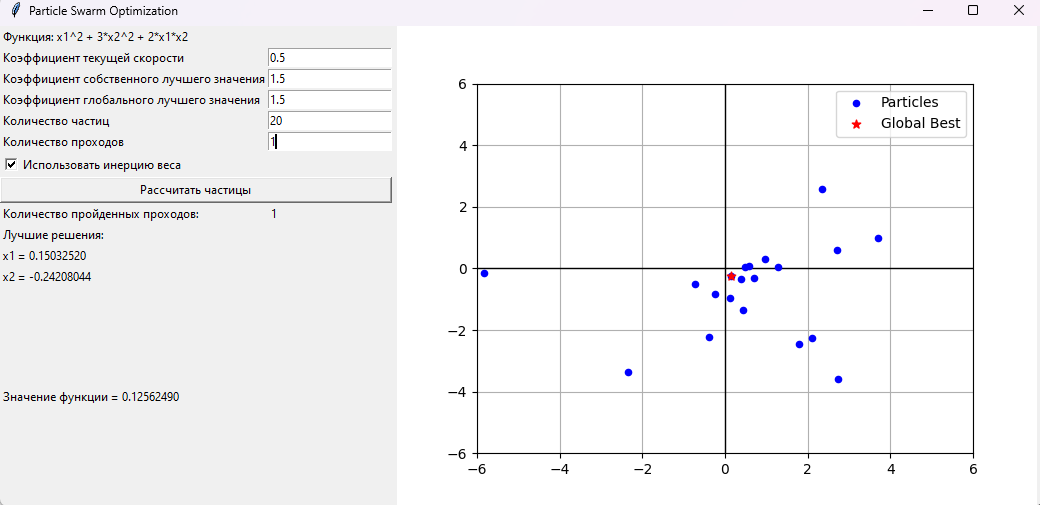


Рис 3. Пример результатов программы

1. Просмотр результатов

После выполнения расчетов на экране отобразятся:

* Количество пройденных проходов.
* Лучшие найденные значения для переменных x1 и x2.
* Наилучшее значение функции для этих переменных.

В правой части окна (Рис. 3) будет доступен график, в котором отображены текущие позиции частиц и наилучшее значение, достигнутое за время работы алгоритма.

# Анализ результатов работы алгоритма и вводных условий

Тесты проводились с коэффициентом текущей скорости 0.5, собственного лучшего решения 1.5 и глобального лучшего решения 1.5.

Таблица 2: тесты с разным количеством итераций

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 10 | 20 | (0.08185551, -0.04066367) | 0.00500384 |
| 20 | 20 | (-0.00001211, -0.00132893) | 0.00000533 |
| 30 | 20 | (0.00003821, 0.00020204) | 0.00000014 |
| 40 | 20 | (-0.00002656, -0.00000994) | 0.00000000 |

Таблица 3: тесты с разным размером популяции

|  |  |  |  |
| --- | --- | --- | --- |
| Количество итераций | Размер популяции | Наилучшее решение | Значение в лучшей точке |
| 20 | 10 | (-0.00715272, -0.00124696) | 0.00007366 |
| 20 | 20 | (-0.00051540, 0.00053527) | 0.00000057 |
| 20 | 30 | (0.00015793, 0.00122801) | 0.00000494 |
| 20 | 40 | (0.00089746, 0.00092712) | 0.00000172 |

Таблица 4: тесты разных алгоритмов с модификацией без

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество итераций | ГА | ГА с модификацией | РА | РА с модификацией |
| 10 | 0.31392038 | 0.05974174 | 0.00381069 | 0.00000001 |
| 20 | 0.01599560 | 0.04270814 | 0.00004024 | 0.00000000 |
| 30 | 0.00112951 | 0.04270814 | 0.00000004 | 0.00000000 |
| 40 | 0.00112951 | 0.01971530 | 0.00000000 | 0.00000000 |

Таблица 2 демонстрирует, как изменение числа итераций влияет на качество решения при фиксированном размере популяции. В условиях малого числа итераций (10) значение функции достигает 0.0050, что говорит о начальном приближении к оптимуму. С увеличением итераций до 20 результат существенно улучшается и достигает значения 0.00000533. Наилучший результат — 0.00000000 — наблюдается при 40 итерациях, что указывает на достижение глобального минимума функции. Таким образом, увеличение количества итераций позволяет алгоритму постепенно улучшать решение и стабилизироваться на оптимальном значении, показывая эффективность работы алгоритма при большом числе проходов.

Таблица 3 демонстрирует, что с увеличением размера популяции при фиксированном количестве итераций (20) алгоритм способен достигать более точных значений функции. При размере популяции 10 наилучшее найденное значение равно 0.00007366, что значительно улучшает качество решения по сравнению с меньшими размерами популяции. В ходе увеличения популяции до 20 точность решения возросла до 0.00000057. Однако при увеличении популяции до 30 и 40 не наблюдается явного улучшения, так как полученные значения (0.00000494 и 0.00000172) лишь немного уступают предыдущим результатам. Таким образом, увеличение размера популяции способствует нахождению более точного решения, однако после определенного порога возможен эффект насыщения, при котором дальнейшее увеличение не приводит к значительным улучшениям.

Таблица 4 сравнивает различные подходы: стандартный генетический алгоритм (ГА), генетический алгоритм с модификацией выбора родителя, рой частиц (РА) и рой частиц с инерцией веса. При меньшем числе итераций (10) ГА демонстрирует наихудший результат (0.3139) по сравнению с модифицированными версиями и РА, которые показывают более эффективные значения (0.0597 для модифицированного ГА, 0.0038 для РА и 0.00000001 для модифицированного РА). На 20 итерациях все методы достигают значительно лучших результатов, особенно РА (0.00004 для стандартного РА и 0.0000000 для модифицированного). На 40 итерации обе версии РА показывают нулевое значение, в то время как ГА достигает только 0.0011, что указывает на более медленное сходство генетического алгоритма по сравнению с роем частиц. Модификация алгоритмов способствует улучшению качества решения на начальных этапах, однако с увеличением итераций различия между методами становятся менее значимыми, поскольку алгоритмы с модификацией и без нее стабилизируются на одном уровне.

Анализ модификации для РА

Использование модификации инерции веса в алгоритме РА значительно повышает точность поиска на начальных итерациях. Однако с увеличением числа итераций эффект модификации становится менее выраженным, так как алгоритм стабилизируется на оптимальных значениях и достигает глобального минимума. Модификация инерции веса помогает сохранить движение частиц, что способствует более сбалансированному исследованию пространства решений на ранних этапах, прежде чем алгоритм начнет сходиться к оптимальным значениям.

# Вывод

В данной работе был разработан алгоритм роя частиц (РА) с реализацией графического интерфейса, который позволяет визуализировать процесс поиска оптимума функции. Результаты экспериментов продемонстрировали, что увеличение количества частиц и итераций способствует повышению точности найденного решения, особенно при использовании модификации инерции веса. Эта модификация позволяет алгоритму более эффективно управлять скоростью движения частиц, что ведет к более быстрому и точному сходу к глобальному оптимуму. Как показали результаты, применение инерции веса помогает алгоритму избежать попадания в локальные минимумы и значительно улучшает качество решений на начальных этапах оптимизации, прежде чем алгоритм стабилизируется на оптимальных значениях.

# Источники

1. numpy // Numpy URL: <https://numpy.org/> (дата обращения: 02.11.2024).
2. tkinter — Python interface to Tcl/Tk // Tkinter URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: 02.11.2024).
3. random // Random URL: <https://docs.python.org/3/library/random.html> (дата обращения: 02.11.2024).
4. Matplotlib: Visualization with Python // Matplotlib URL: <https://matplotlib.org/> (дата обращения: 02.11.2024).