**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 – 19**

**Студент гр. 23Б15-пу**

**Сурин И.С.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г**

Оглавление

[Цель работы 3](#_Toc181959962)

[Описание алгоритма роя частиц 3](#_Toc181959963)

[Описание задачи 4](#_Toc181959964)

[Основные шаги программы 4](#_Toc181959965)

[Описание программы 7](#_Toc181959975)

[Рекомендации для пользователя 7](#_Toc181959976)

**[Рекомендации для разработчика](#_Toc181959977)** [8](#_Toc181959977)

[Исходный код программы 8](#_Toc181959978)

[Контрольный пример 8](#_Toc181959979)

[Исследование 9](#_Toc181959980)

[Вывод 11](#_Toc181959981)

[Источники 11](#_Toc181959982)

**Цель работы**

Исследование алгоритмов роевого интеллекта для решения задач глобальной оптимизации и их сравнительная оценка с генетическим алгоритмом. Будет рассмотрен алгоритм частиц роя (PSO) и генетический алгоритм (GA), их параметры и влияние настроек на эффективность. Проведется эксперимент с несколькими задачами оптимизации для анализа преимуществ и недостатков каждого метода.

# **Описание алгоритма роя частиц**

Алгоритм роя частиц (PSO — Particle Swarm Optimization) был предложен в 1995 году. Он основан на наблюдениях за поведением роя птиц или стаи рыб, где каждая частица (птица) является потенциальным решением задачи, а взаимодействие между частицами позволяет улучшать поиск.

1. **Частицы** — каждое решение задачи представлено частицей, которая имеет: **Позицию** (текущее значение решения), **скорость** (вектор изменения позиции), **Личное лучшее положение** — лучшее найденное значение**, Глобальное лучшее положение —** лучшее значение среди всех частиц.
2. **Процесс работы алгоритма**
   * Инициализация: позиции и скорости частиц случайным образом.
   * Каждая частица оценивает свою позицию с помощью целевой функции.
   * Обновление позиции и скорости частицы:
     + **Скорость** частиц зависит от трех факторов:
       1. Инерция (сохранение текущей скорости).
       2. Когнитивный компонент (стремление к личному лучшему положению).
       3. Социальный компонент (стремление к глобальному лучшему положению).
   * Оценка новых позиций и обновление личных и глобальных лучших положений.
3. **Параметры**:
   * **Число частиц** — количество решений в поисковом процессе.
   * **Инерция (w)** — коэффициент, регулирующий влияние предыдущей скорости.
   * **Когнитивный (c₁) и социальный (c₂) коэффициенты** — определяют влияние личного и глобального лучшего положения.

Из преимуществ можно выделить простоту и гибкость, а также эффективность в решении сложных задач оптимизации.

К ограничениям можно отнести застреварние в локальных минимумах, чувствительность к настройке параметров.

# **Описание задачи**

Целью является нахождение минимального значения функции с помощью алгоритма роевого интеллекта. Задача состоит в том, чтобы найти такие значения переменных x1 и x2, при которых функция

F(x1,x2) = 

принимает минимальное значение. Эта функция не имеет глобального минимума, что можно доказать, проанализировав её стационарные точки и вычислив производные в этих точках. Тем не менее, алгоритм роевого интеллекта способен с высокой точностью находить локальные минимумы, что делает его эффективным инструментом для решения этой задачи.

# **Основные шаги программы**

### 1. **Инициализация программы**

### 1) Определение целевой функции F(x1,x2) =

### 2) Установка начальных значений для параметров алгоритма: количество частиц, количество итераций, коэффициенты инерции, когнитивного и социального компонентов.

1. Инициализация объектов.

### 2. **Инициализация роя частиц**

1) Создание случайных начальных позиций для каждой из частиц в пределах заданного диапазона (например, x,y∈[−10,10]) Установка случайных начальных скоростей для частиц.

2) Оценка начальных значений целевой функции для каждой частицы.

### 3. **Оценка позиций частиц**

1) Для каждой частицы вычисляется значение целевой функции.

2) Определение, является ли текущее положение частицы её личным лучшим положением (если значение целевой функции для текущей позиции меньше предыдущего лучшего).

3) Обновление личных лучших позиций (pbest) и их значений.

### 4. **Обновление глобального лучшего положения**

1) После оценки позиций всех частиц, выбирается наилучшее из личных лучших положений (pbest).

1. Это наилучшее положение становится глобальным лучшим (gbest), если оно дает минимальное значение функции среди всех частиц.

### 5. **Обновление скорости и позиции частиц**

1) Для каждой частицы обновляется скорость с использованием формулы, которая учитывает инерцию, когнитивный компонент, социальный компонент.

1. Обновление позиции частицы на основе новой скорости.

### 6. **Повторение процесса**

1) Повторение шагов 3-5 для заданного числа итераций (или до достижения критерия остановки, например, если улучшение минимально).

1. Каждая итерация улучшает положение частиц, приближая их к оптимальному решению.

### 7. **Вывод результата**

1) После завершения всех итераций выводится глобальное лучшее положение (gbest) — оптимальные значения x1 и x2, а также минимальное значение функции, найденное в процессе оптимизации.

1. При необходимости, можно отобразить график, показывающий распределение частиц на каждой итерации.

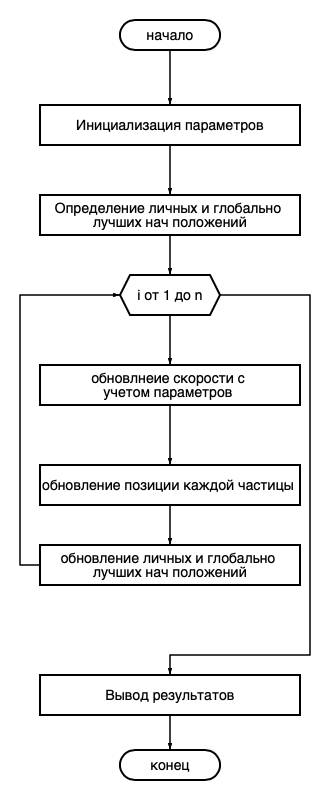


Рис. 1 блок-схема 1.py

# 

# **Описание программы**

Программная реализация написана на языке Python 3.12.2 с использованием библиотеки numpy и Tkinter. В процессе разработки программы использовался следующий модули:

Таблица 1 1.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| objective\_function | Вычисляет значение целевой функции для заданной позиции (x1, x2) | float |
| \_\_init\_\_ | Инициализирует параметры роя, такие как количество частиц, их скорости и позиции | None |
| step | Обновляет позиции и скорости частиц, а также находит лучшее положение для каждой частицы | None |
| run\_pso | |  | | --- | | Запускает алгоритм PSO, выполняя оптимизацию и обновляя график | | None |
| plot\_particles | |  | | --- | |  |  |  | | --- | | Отображает распределение частиц на графике | | None |

# **Рекомендации для пользователя**

Для того чтобы запустить программу, убедитесь, что на вашем компьютере установлен Python и все необходимые библиотеки. После запуска приложения вам будет предложено ввести параметры для настройки алгоритма. Заполните соответствующие поля, чтобы задать необходимые значения. По завершению работы алгоритма результаты будут отображены на графике.

### **Рекомендации для разработчика**

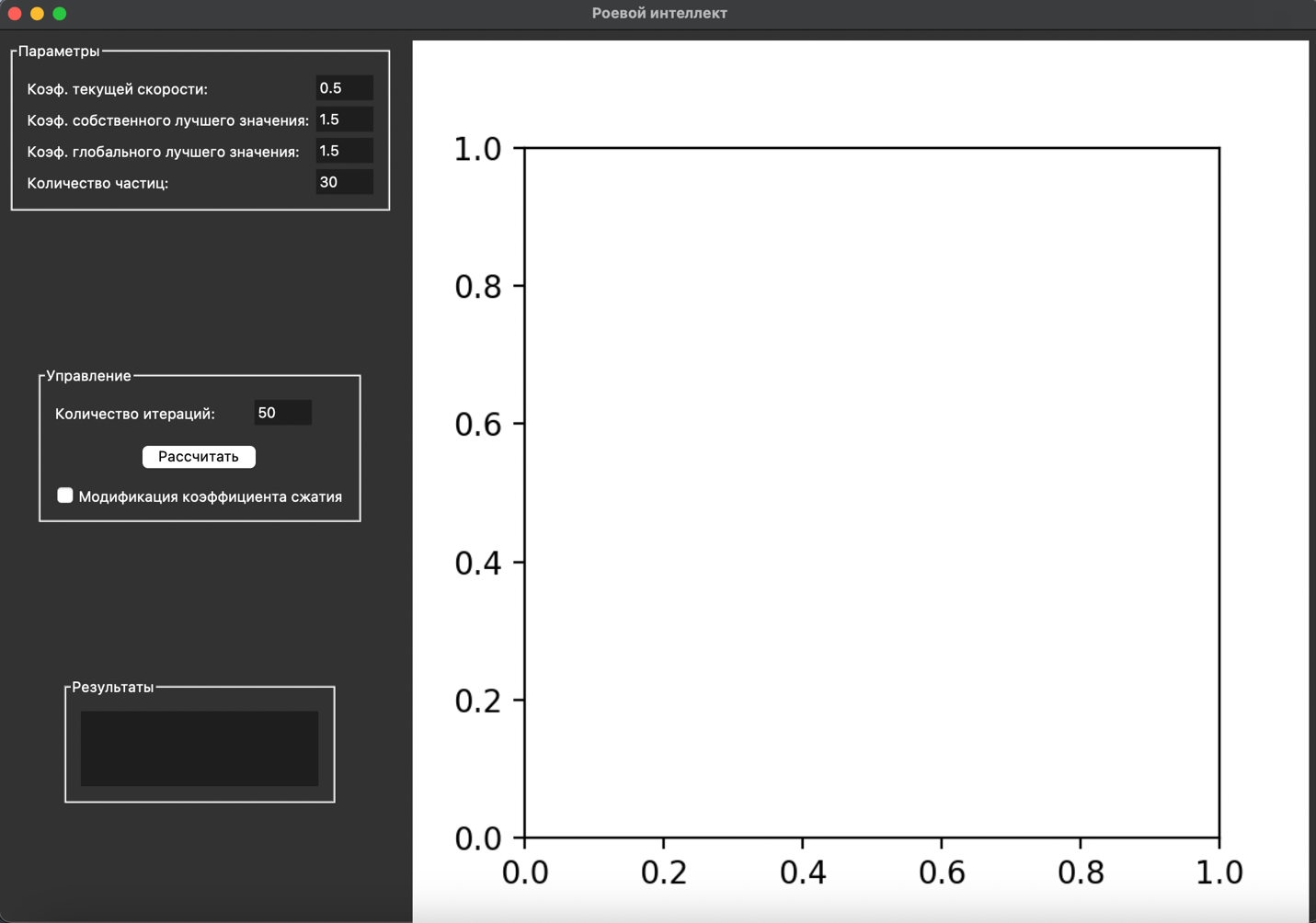
Для поддержания корректной работы программы рекомендуется использовать актуальные версии библиотек, таких как numpy и Tkinter. Также важно соблюдать лучшие практики кодирования, включая правильное именование переменных и функций, чтобы улучшить читаемость и поддержку кода в будущем.

# **Исходный код программы**

<https://github.com/Ignatio27/spbu-algorithms-and-data-structures->

# **Контрольный пример**

1. **Запуск программы**  
   Для запуска программы используйте файл 1.py.
2. **Выбор параметров роевого алгоритма**  
   После запуска программы пользователю будет предложено выбрать параметры (Рис. 2).

  
 Рис 2. Пример выбора параметров

1. **Обработка данных и вывод результатов**  
   После нажатия кнопки рассчитать, программа выводит результаты (Рис.3)



Рис .3 Результат работы алгоритма

# **Исследование**

Коэффициент сжатия играет ключевую роль в поведении алгоритма роевого интеллекта, влияя на скорость и точность поиска. Когда коэффициент сжатия включен, он способствует уменьшению скорости движения частиц, что позволяет алгоритму более точно исследовать пространство решений.

1. **На начальных этапах поиска**: При включении коэффициента сжатия частички движутся быстрее, что ускоряет начальное исследование и помогает алгоритму покрывать более широкие области пространства решений. Это способствует глобальному поиску, так как частицы имеют более высокие скорости и могут быстрее перемещаться между различными областями.
2. **По мере приближения к оптимуму**: Когда алгоритм начинает приближаться к потенциальным оптимальным решениям, коэффициент сжатия начинает уменьшать скорость движения частиц. Это помогает алгоритму сосредоточиться на более точном поиске вблизи найденных хороших позиций, таким образом, улучшая локальный поиск. Уменьшение скорости также помогает избежать чрезмерных колебаний вокруг минимума и способствует более точному нахождению оптимума.
3. **Общее влияние на эффективность**: Сжатие скорости позволяет алгоритму найти более точные решения, особенно в ситуациях, когда простое исследование пространства с высокой скоростью не дает нужных результатов. Это снижает риск застревания в локальных минимумах и увеличивает вероятность нахождения более качественного глобального оптимума, улучшая как точность, так и эффективность поиска.

**Сравнение генетического и роевого алгоритмов**

Таблица 2 сравнение ГА и РА

|  |  |  |
| --- | --- | --- |
| Критерий | Генетический алгоритм | Алгоритм роя частиц |
| Основной принцип | |  | | --- | | Эволюционный подход: основан на операторах селекции, кроссовера и мутаций |  |  | | --- | |  | | |  | | --- | | Моделирует коллективное поведение частиц, стремящихся к цели |  |  | | --- | |  | |
| |  | | --- | | **Структура** |  |  | | --- | |  | | |  | | --- | | Популяция решений (хромосом) |  |  | | --- | |  | | Рой частиц, каждая из которых — потенциальное решение |
| Подход к поиску | Локальный и глобальный поиск через кроссовер и мутацию | Обновление скоростей и позиций с учетом личного и глобального опыта |
| Области применения | |  | | --- | | Задачи комбинаторной оптимизации (например, маршрутизация) | | Непрерывные задачи оптимизации (например, регрессия) |
| Скорость сходимости | Медленная сходимость из-за генерации новых решений | Обычно быстрая сходимость, особенно на первых этапах поиска |
| Гибкость | Высокая, может адаптироваться под разные задачи | Более ограниченная гибкость, сложен для дискретных задач |
| Преимущества | Хорошо подходит для сложных и нелинейных задач | Прост в реализации, быстрый, эффективен в задачах с непрерывными переменными |
| Недостатки | Требует больших вычислительных ресурсов, может застрять в локальном минимуме | Может застревать в локальном оптимуме, труден для дискретных задач |

Таблица 2 сравнение запусков ГА и РА

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Алгоритм | | Количество попаданий из 10 | Процент попаданий | Время выполнения (сек) | Средняя ошибка | Минимальная ошибка | Максимальная ошибка |
| Генетический алгоритм с модификацией | 7 | 70% | 0.0021 | 0.5544 | 0.012 | 1.346 |
| Генетический алгоритм без модификации | 6 | 60% | 0.0019 | 0.6343 | 0.092 | 1.812 |
| Роевой алгоритм с модификацией | 9 | 90% | 0.0017 | 0.4232 | 0.034 | 1.203 |
| Роевой алгоритм без модификации | 8 | 80% | 0.0018 | 0.4985 | 0.045 | 1.458 |

**Эффективность поиска решений**

1. Наибольшее количество попаданий (9 из 10) достигнуто использованием роевого алгоритма с модифицированным коэффициентом сжатия. Этот результат говорит о том, что данный алгоритм наиболее стабильно и точно достигает целевого значения.

**Время выполнения**

1. Время работы у всех алгоритмов отличается незначительно, что делает их сравнительно равными по этому критерию. Однако роевой алгоритм с модификацией также оказывается чуть быстрее.

**Cредняя ошибка**

1. Роевой алгоритм с коэффициентом сжатия показал наименьшую среднюю ошибку, что подтверждает его высокую точность по сравнению с другими методами.

# **Вывод**

Алгоритм роевого интеллекта с модификацией коэффициента сжатия оказывается более эффективным по сравнению с вариантом без этой модификации. Включение коэффициента сжатия позволяет частицам сначала двигаться быстрее, что способствует более широкому исследованию пространства решений и ускоряет глобальный поиск. Коэффициент сжатия повышает гибкость алгоритма, обеспечивая баланс между исследованием большого пространства на начальных этапах и более детализированным поиском вблизи оптимальных точек на последних этапах

# **Источники**

1. Numpy documentation // Numpy URL:

https://numpy.org (дата обращения:8.11.2024).

1. Tkinter documentation // Tkinter URL:

https://docs.python.org/3/library/tkinter.html (дата обращения: 8.11.2024)