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 – 2

Студент гр. 23Б15-пу
Антонян А. А.

Преподаватель
Дик А.Г.

Санкт-Петербург
2024 г.

Оглавление

1. **Цель работы** 3
2. **Теоретическая часть** 3
3. **Описание задачи** 5
4. **Основные шаги программы** 5
5. **Описание программы** 7
6. **Рекомендации пользователя** 8
7. **Рекомендации программиста** 8
8. **Исходный код программы** 8
9. **Контрольный пример** 8
10. **Исследование** 10
11. **Вывод** 11
12. **Источники** 12

Цель работы

Целью лабораторной работы является исследование особенностей алгоритмов роевого интеллекта для решения задач глобальной оптимизации и сравнение с генетическим алгоритмом.

Теоретическая часть

Алгоритмы роевого интеллекта — это семейство оптимизационных алгоритмов, которые моделируют коллективное поведение социальных организмов, таких как муравьи, пчёлы и т.д. для решения NP сложных и NP полных задач. Эти алгоритмы основываются на децентрализованном управлении и взаимодействиях множества простых агентов (частиц, муравьев и т.д.), которые координируются через поведенческие правила, что позволяет находить решения без центрального контроля. Основные алгоритмы роевого интеллекта:

1. **Алгоритм роя частиц**: Агенты (частицы) перемещаются по пространству поиска, стремясь к своим лучшим позициям и лучшей позиции всего роя.
2. **Муравьиный алгоритм**: Моделирует поведение муравьев в поиске кратчайшего пути к пище, оставляющих феромоны для привлечения других муравьев.
3. **Алгоритм пчелиного роя**: Пчёлы исследуют пространство поиска вокруг "цветков" (решений), передавая информацию о качестве решений.

Для решения задачи нахождения глобального минимума был реализован алгоритм роя частиц, так как по определению он больше всего подходит для этого.

Алгоритм роя частиц — это метод оптимизации, вдохновленный поведением стайных животных, таких как птицы или рыбы, которые объединяются для поиска пищи или избегания хищников.

Основные принципы:

Каждая частица роя представляет возможное решение задачи, и каждая частица движется по пространству поиска, ориентируясь на:

- свою личную лучшую позицию (где была найдена наилучшая оценка),
- глобальную лучшую позицию, найденную всем роем.

Алгоритм контролируется тремя параметрами:

1. **Инерция**, которая задает тенденцию частицы сохранять свою текущую скорость, что помогает исследовать пространство решений.
2. **Коэффициент когнитивного компонента**, который указывает, насколько сильно частица стремится вернуться к своему лучшему решению.
3. **Коэффициент социального компонента**, отвечающий за влияние глобального лучшего решения на движение частицы.

Также в алгоритме был применена модификация посредством применения **коэффициента сжатия**. Для улучшения поиска глобального минимума, инерция часто уменьшается в ходе работы алгоритма, что помогает рою сосредотачиваться вокруг лучших решений на поздних этапах и избегать застревания в локальных минимумах.

Описание задачи

Задача состоит в нахождении глобального минимума функции с помощью алгоритма роевого интеллекта. Алгоритм направлен на нахождение такой пары значений (x, y), при которой значение функции Розенброка: $f(x,y)=100\times(y-x^2)^2+(1-x)^2$ становится минимально возможным. Глобальный минимум для этой функции находится в точке (1,1), где значение функции равно 0.

Основные шаги программы

1. **Инициализация**: Каждая частица получает случайное начальное положение и скорость.
2. **Обновление скорости**: Скорость каждой частицы обновляется на основе инерции, личного лучшего положения и глобального лучшего положения.
3. **Обновление положения**: Положение частицы изменяется в соответствии с новой скоростью.
4. **Оценка решений**: Вычисляется значение целевой функции для текущих положений частиц, после чего личные и глобальные лучшие положения обновляются.
5. **Повтор**: Алгоритм выполняется заданное число итераций или до достижения критерия остановки.

Рис. 1. Блок-схема основной программы

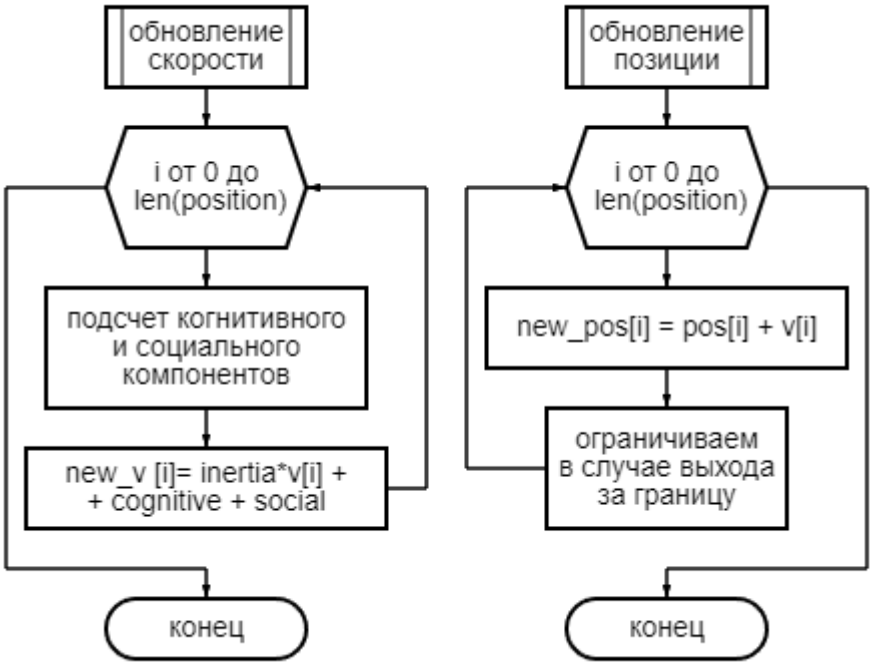

Рис. 2. Блок-схемы подпрограмм

Описание программы

Программная реализация написана на языке Python 3.12.2 с использованием библиотек PyQt5[1] и matplotlib[2]. В процессе разработки программы использовался следующий модуль:

Таблица1 swarm_intelligence.py

Функция	Описание	Возвращаемое значение
initialize_swarm()	Создает рой.	list
update_velocity()	Обновляет скорость частицы	None
update_position()	Обновляет позицию частицы	None
particle_swarm_optimization()	Главная функция, включает в себя все вышеупомянутые. Реализует весь алгоритм	tuple

Рекомендации пользователя

Для запуска программы убедитесь, что у вас установлен Python и необходимые библиотеки, такие как PyQt5[1] и matplotlib[2] Код можно запустить в среде разработки или через командную строку, используя консоль для настройки параметров и генерации данных. Запуск программы производится через файл swarm_intelligence.py.

При запуске программы вам будет предложено выбрать параметры запускаемого алгоритма. Вводите ответы соответствующие поля в GUI. Результаты работы будут представлены в таблице.

Рекомендации программиста

Для поддержания актуальности и работоспособности программы используйте последние версии библиотек, особенно PyQt5[1] и matplotlib[2]. Применяйте практики надлежащего именования переменных и функций для улучшения читаемости кода.

Исходный код программы:

https://github.com/ArseniiAntonian/spbu-algorithms-and-data-structures

Контрольный пример

1. **Запуск программы**
   Для запуска программы используйте файл swarm_intelligence.py. Программа загружает GUI и позволяет пользователю настроить параметры алгоритма.

2. **Выбор параметров**
   После запуска программы пользователю будет предложено выбрать параметры (Рис. 3).

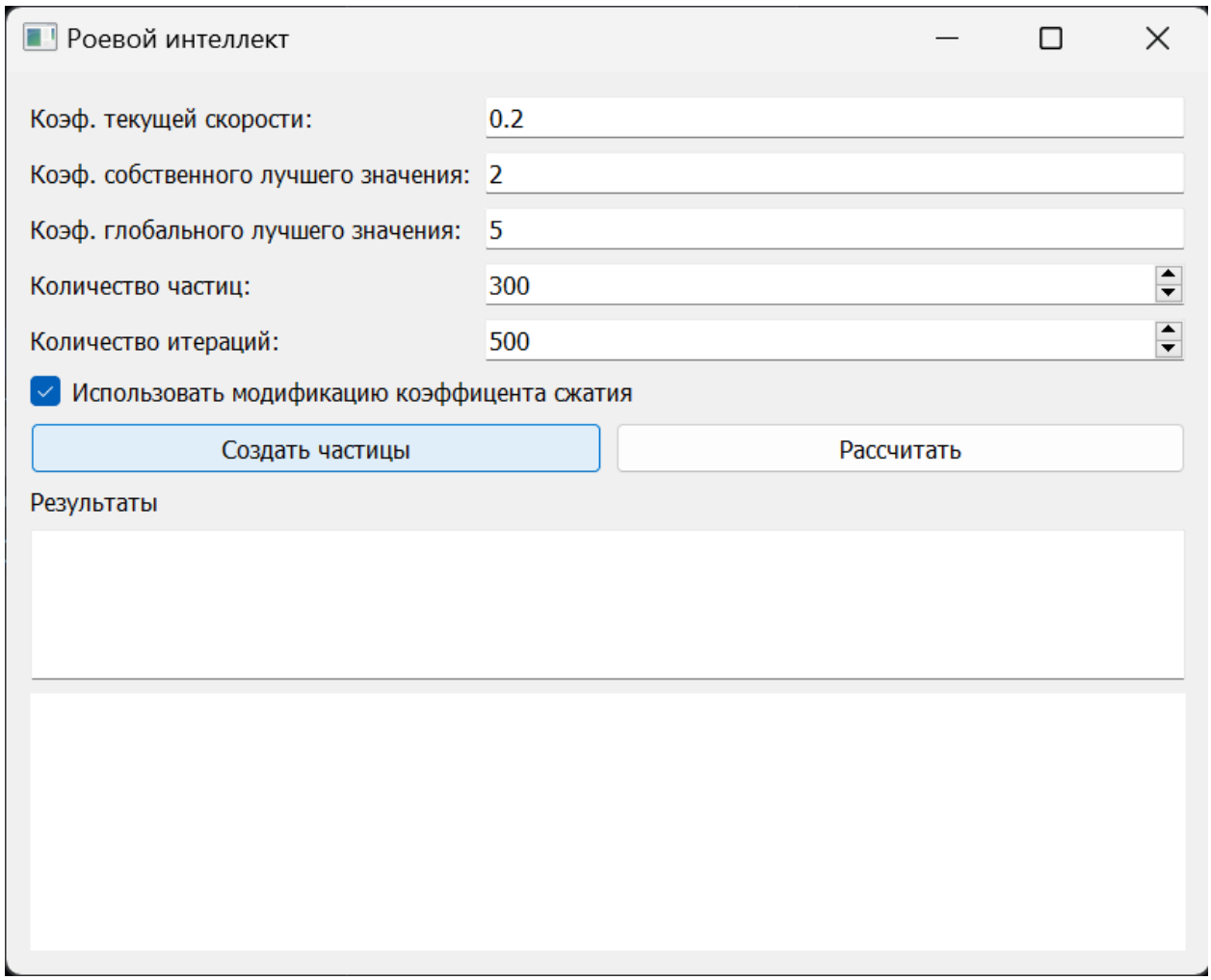

Рис 3. Пример выбора параметров

3. **Обработка данных и вывод результатов**

   После выбора параметров пользователю предложено самостоятельно создать частицы, а затем запустить алгоритм. Результат работы будет выводиться в окне ниже и на графике (Рис. 4).

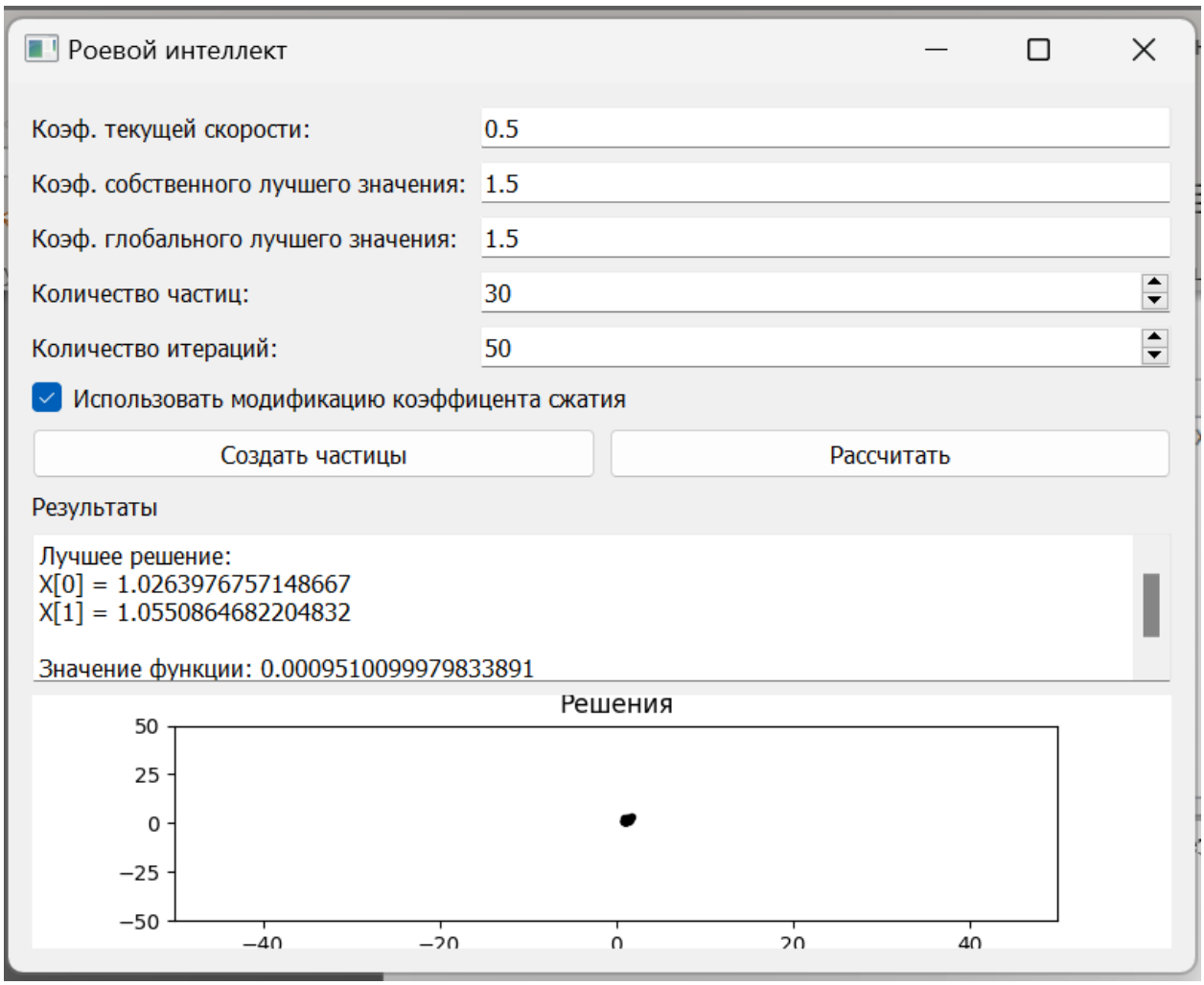

Рис 4. Результат работы алгоритма

Исследование

В рамках данной лабораторной работы было проведено сравнение работы алгоритма роя частиц с модификацией или без. с генетическим алгоритмом – это эвристические алгоритмы, поэтому идеальную сходимость гарантировать нельзя. На начальных этапах роевой алгоритм быстрее покрывает пространство решений благодаря высокой инерции. Это ускоряет начальное исследование и способствует более быстрому достижению оптимальных областей. Затем снижение инерции позволяет алгоритму точнее локализовать минимум, что повышает его эффективность. На основе проведенных тестов можно сделать вывод, что модификация коэффициентом сжатия значительно улучшает сходимость роевого алгоритма, а модифицированный алгоритм быстрее приходит к наиболее точному ответу. Имея следующие параметры (Рис.5), генетический алгоритм пришел к ответу с нужной точностью 5 раз.

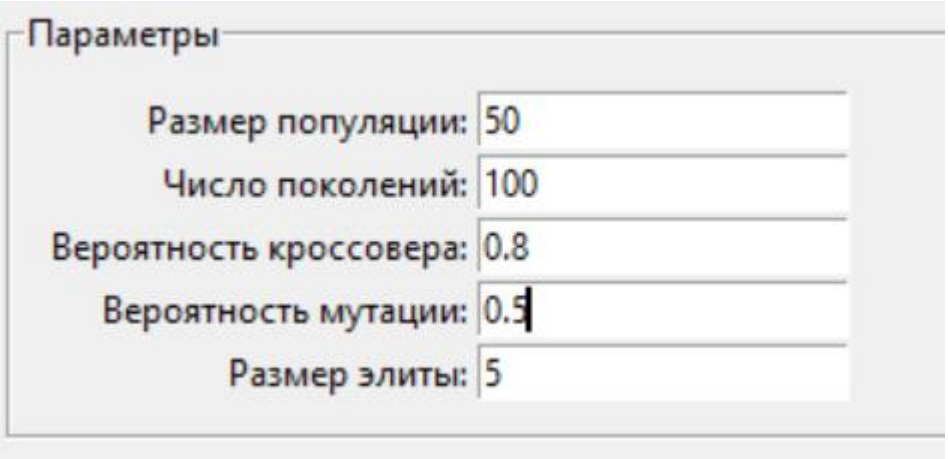

Рис. 5 Экспериментальны параметры ГА

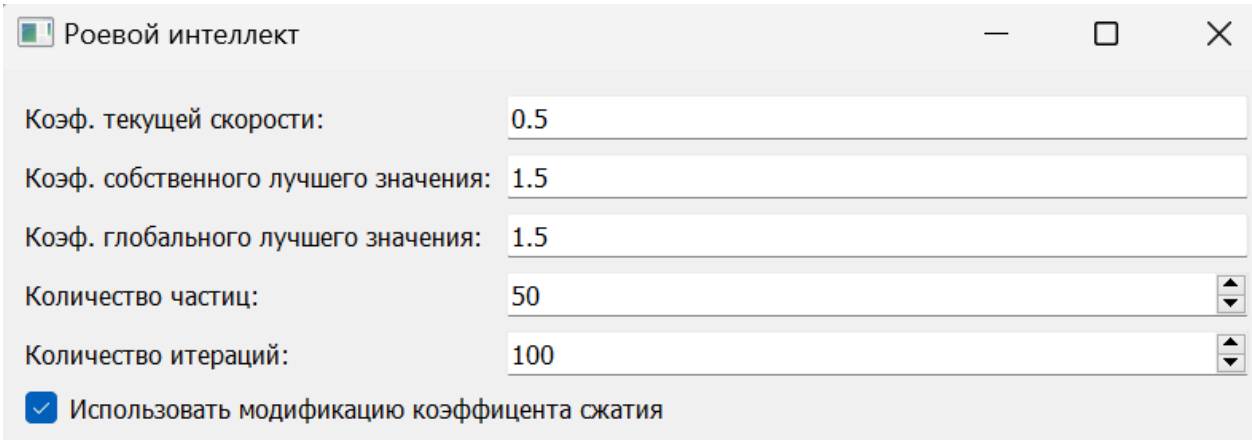

Рис. 6. Экспериментальные параметры РА

Модифицированный роевой алгоритм с похожими параметрами (Рис. 6) сошелся 9 раз из 10 на случайных запусках, показав высокую точность (минимум до 10^(-4)). Можно сделать вывод о большом разрыве в эффективности между этими семействами алгоритмов.

Вывод

В рамках данной работы был разработан алгоритм для нахождения глобального минимума функции. Программа была протестирована на функции Розенбаума. Реализованный алгоритм обеспечивает возможность оптимизации различных функций, например поиск экстремумов или решение NP полных задач. Была реализована модификация с помощью коэффициента

сжатия, что помогло избегать попадания частиц в локальные минимумы и ускорить сходимость алгоритма к глобальному минимуму.

Источники

1. PyQt5 documentation // PyQt5 URL: https://pypi.org/project/PyQt5/ (дата обращения: 5.11.2024).
2. Matplotlib documentation // Matplotlib URL: https://matplotlib.org/stable/index.html (дата обращения: 5.11.2024)