УДК 004

Применение роевого интеллекта, генерируемого искусственным интеллектом в оптимизации транспортных потоков

Екимова Яна Сергеевна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Статья посвящена исследованию возможностей применения роевого интеллекта, генерируемого искусственным интеллектом в оптимизации транспортных потоков. Роевой интеллект представляет собой класс методов, основанных на моделировании коллективного поведения систем, таких как муравьиные колонии или стаи птиц. В статье рассматриваются принципы работы алгоритмов роевого интеллекта и их потенциал для решения задач управления дорожным движением. Описаны основные модели, такие как алгоритм муравьиных колоний и алгоритм роя частиц, и особенности их применения в задачах маршрутизации, распределения транспортных средств и управления светофорами.
Ключевые слова: роевой интеллект, оптимизация транспортных потоков, алгоритм муравьиных колоний, алгоритм роя частиц, искусственный интеллект.

The use of swarm intelligence generated by artificial intelligence in optimizing traffic flows

Ekimova Yana Sergeevna
Sholom Aleichem Priamurskiy State University
Student

Abstract
The article is devoted to the study of the possibilities of using swarm intelligence generated by artificial intelligence in optimizing traffic flows. Swarm intelligence is a class of methods based on modeling the collective behavior of systems such as ant colonies or flocks of birds. The article discusses the principles of swarm intelligence algorithms and their potential for solving traffic management problems. The main models, such as the ant colony algorithm and the particle swarm algorithm, and the features of their application in routing, vehicle distribution, and traffic light control tasks are described.
Keywords: swarm intelligence, optimization of traffic flows, ant colony algorithm, particle swarm algorithm, artificial intelligence.

1. Введение

1.1. Актуальность

Оптимизация транспортных потоков является одной из ключевых проблем современных городов, характеризующихся высокой плотностью населения и возрастающей нагрузкой на дорожную инфраструктуру. Традиционные подходы к управлению дорожным движением, такие как координация светофоров и регулирование скорости, зачастую не справляются с задачей эффективной организации транспортных потоков, что приводит к образованию заторов, простоям автомобилей и росту загрязнения окружающей среды. В этой связи возрастает интерес к новым методам оптимизации транспортных процессов, в частности, к использованию принципов роевого интеллекта. Данный подход основан на моделировании коллективного поведения систем, характеризующихся децентрализованным управлением и способностью к адаптации в динамически изменяющихся условиях. Методы роевого интеллекта демонстрируют значительный потенциал для решения задач маршрутизации, распределения ресурсов и координации транспортных средств, что определяет их актуальность для совершенствования функционирования интеллектуальных транспортных систем. Комплексное исследование возможностей применения роевого интеллекта в оптимизации транспортных потоков позволит выявить ключевые преимущества данного подхода, определить оптимальные алгоритмы и области их применения, а также разработать рекомендации по внедрению методов роевого интеллекта в практику управления дорожным движением.

Обзор исследований

М.Н. Морозова, Е.И. Егоров применили метод роя частиц с использованием нескольких агентов для нахождения минимума функции Растригина [1]. А.В. Горячев и Н.Е. Новакова выполнили анализ способов решения задач дискретной оптимизации в САПР. Рассмотрели алгоритмы роевого интеллекта и их модификации. Представили один из вариантов решения задачи оптимизации маршрута проектирования [2].

В данной статье М.В. Тюхин, В.А. Ломазов дают определение биоинспирированным алгоритмам, используемым для решения оптимизационных задач. Приводится описание известных алгоритмов роевого интеллекта: муравьиного алгоритма, генетического алгоритма пчелиной колонии, алгоритма кошачьей стаи, алгоритма серых волков. Рассматриваются принципы работы данных алгоритмов и спектр их возможного применения [3]. В.З. Манусов, П.В. Матренин, Д.В. Орлов рассмотрели задачу оптимизации положений анцапф трансформаторов системы электроснабжения для снижения потерь мощности. В настоящее время применяются методы, основанные на эвристических правилах и нечеткой логике или методы, оптимизирующие части системы по-отдельности [4].

1.2. Цель исследования

Целью данного исследования является исследование возможностей применения методов роевого интеллекта сгенерированных при помощи искусственного интеллекта (ИИ), для оптимизации транспортных потоков в городских условиях.

2. Материалы и методы

Для исследования оптимизации транспортных потоков был применен метод роевого интеллекта (Particle Swarm Optimization, PSO), реализованный на языке Python с использованием библиотеки Matplotlib для визуализации результатов. Исследование проводилось в среде Google Colab, что обеспечило доступ к вычислительным ресурсам и удобство работы с кодом.

Ноутбук можно посмотреть по ссылке https://colab.research.google.com/drive/1REyu8IJWN8_z_FlslRhpTyPe8sSL9Hxz?usp=sharing.

3. Результаты

Анализ проводится на платформе Google Collab. Прежде чем приступить к работе, необходимо правильно сформулировать запрос искусственному интеллекту. Для этого воспользуемся открытым чатом с искусственным интеллектом ChatGPT (Рис.1).

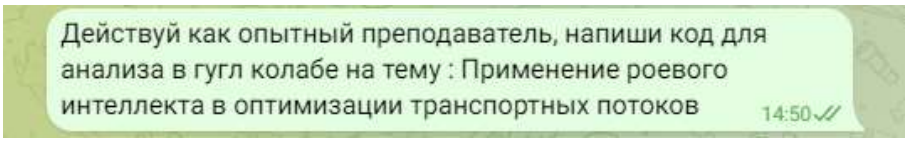


Рисунок 1. Запрос, введённый в ChatGPT

Затем он выдает следующий код:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.spatial.distance import cdist

# Определение параметров роевого интеллекта
num_particles = 50
num_iterations = 100
c1 = 2
c2 = 2
w = 0.5

# Определение области поиска
x_min, x_max = 0, 10
y_min, y_max = 0, 10

# Определение целевой функции (транспортный поток)
def objective_function(x, y):
    # Здесь необходимо определить реалистичную целевую функцию, отражающую транспортный поток
```

```
        return (x - 5)**2 + (y - 5)**2

    # Инициализация роевого интеллекта
    positions = np.random.uniform(low=[x_min, y_min], high=[x_max,
y_max], size=(num_particles, 2))
    velocities = np.zeros((num_particles, 2))
    personal_best_positions = positions.copy()
    personal_best_values = np.array([objective_function(p[0], p[1])
for p in positions])
    global_best_position =
personal_best_positions[np.argmin(personal_best_values)]
    global_best_value = np.min(personal_best_values)

    # Визуализация начальной позиции частиц
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(positions[:, 0], positions[:, 1],
[objective_function(p[0], p[1]) for p in positions], c='b',
marker='o')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Transportation Flow')
    ax.set_title('Initial Particle Positions')
    plt.show()

    # Оптимизация с помощью роевого интеллекта
    for i in range(num_iterations):
        for j in range(num_particles):
            # Обновление скорости и позиции частицы
            r1 = np.random.rand(2)
            r2 = np.random.rand(2)
            velocities[j] = w * velocities[j] + c1 * r1 *
(personal_best_positions[j] - positions[j]) + c2 * r2 *
(global_best_position - positions[j])
            positions[j] += velocities[j]

            # Обновление персональной и глобальной лучшей позиции
            new_value = objective_function(positions[j][0],
positions[j][1])
            if new_value < personal_best_values[j]:
                personal_best_positions[j] = positions[j]
                personal_best_values[j] = new_value
            if new_value < global_best_value:
                global_best_position = positions[j]
                global_best_value = new_value

    # Визуализация конечного результата
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(positions[:, 0], positions[:, 1],
[objective_function(p[0], p[1]) for p in positions], c='b',
marker='o')
    ax.scatter(global_best_position[0], global_best_position[1],
objective_function(global_best_position[0], global_best_position[1]),
c='r', marker='*')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Transportation Flow')
```

```
ax.set_title('Final Particle Positions and Global Best Solution')
plt.show()
```

Следующим шагом будет детальное рассмотрение сгенерированного кода, разделение его на части и подробное описание каждой из них.

Создадим блокнот в Google Colab, в блоке кода импортируем необходимые библиотеки для работы с числовыми данными, построения графиков и вычисления расстояний между точками в пространстве (Рис.2).

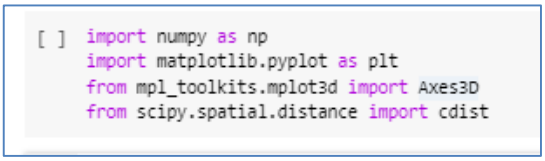


Рисунок 2. Импорт библиотек

Далее установим базовые параметры для алгоритма роевого интеллекта (Particle Swarm Optimization, PSO). Эти параметры определяют поведение частиц в процессе оптимизации и влияют на эффективность поиска оптимального решения.

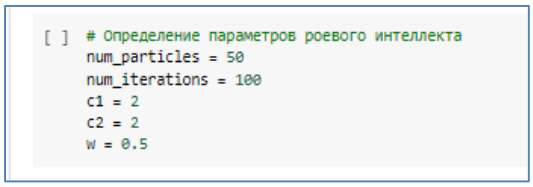


Рисунок 3. Установка базовых параметров

Теперь установим границы области поиска для частиц в алгоритме роевого интеллекта. Частицы будут случайным образом инициализироваться и перемещаться в пределах этой области. Значения x_min и x_max задают диапазон для оси X, а y_min и y_max — для оси Y. В данном случае, область поиска представляет собой квадрат со стороной 10 единиц, центрированный в начале координат.

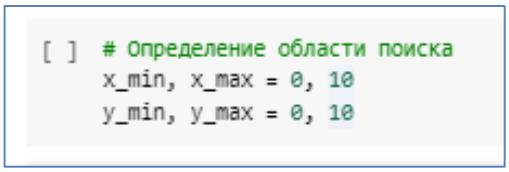
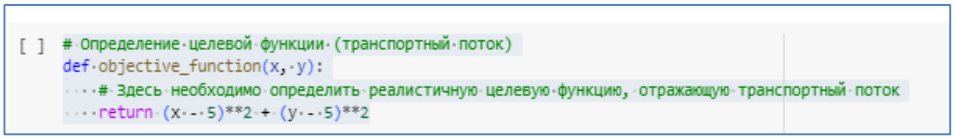


Рисунок 4. Установка границ области

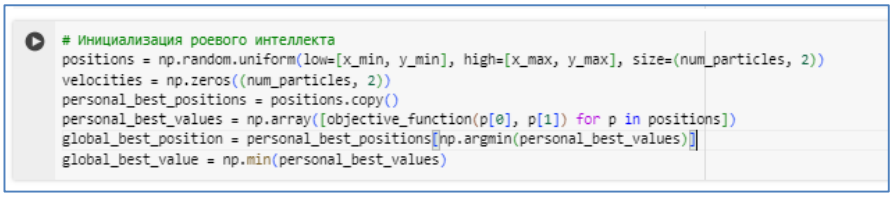
Далее определяем функцию, которая будет использоваться в качестве целевой функции для оптимизации в алгоритме роевого интеллекта. Целевая функция должна отражать критерий оптимизации, в данном случае, эффективность транспортного потока.



```
[ ]  # Определение целевой функции (транспортный поток)
     def objective_function(x, y):
         # Здесь необходимо определить реалистичную целевую функцию, отражающую транспортный поток
         return (x - 5)**2 + (y - 5)**2
```

Рисунок 5. Определение функции

Теперь проведем инициализацию начальных условий для алгоритма роевого интеллекта, включая начальные позиции и скорости частиц, а также установим личные и глобальные лучшие решения на основе начальных позиций.



```
# Инициализация роевого интеллекта
positions = np.random.uniform(low=[x_min, y_min], high=[x_max, y_max], size=(num_particles, 2))
velocities = np.zeros((num_particles, 2))
personal_best_positions = positions.copy()
personal_best_values = np.array([objective_function(p[0], p[1]) for p in positions])
global_best_position = personal_best_positions[np.argmin(personal_best_values)]
global_best_value = np.min(personal_best_values)
```

Рисунок 6. Инициализация начальных условий

Далее пропишем код, который отобразит трехмерный график, на котором будут визуализироваться начальные позиции частиц в пространстве поиска. Каждая точка на графике соответствует начальной позиции частицы, а ее высота (z-координата) определяется значением целевой функции в этой точке.

Рисунок 7. Код

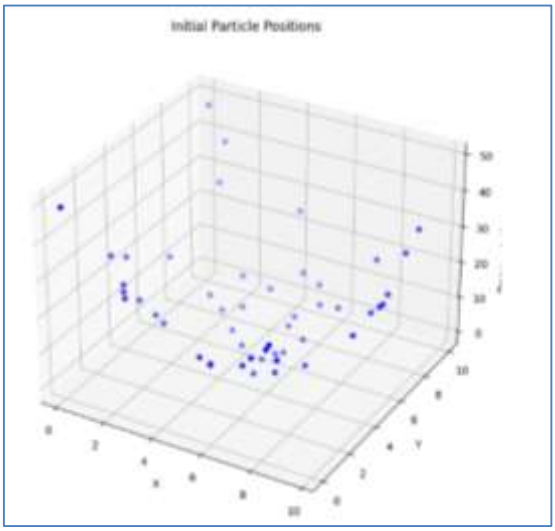


Рисунок 8. Визуализация начальных позиций

Теперь пропишем код, для итеративного процесса оптимизации, где для каждой частицы в рое обновятся ее скорость и позиция в соответствии с

формулой роевого интеллекта. Случайные числа r1 и r2 используются для внесения случайности в обновление скорости, что способствует более широкому поиску в пространстве решений.

Рисунок 9. Итеративный процесс оптимизации

Далее пропишем код, для итеративного процесса оптимизации, где для каждой частицы в рое обновляются ее скорость и позиция, а также проверяется, улучшилось ли ее личное лучшее решение и глобальное лучшее решение. Если новое значение целевой функции для частицы лучше ее предыдущего личного лучшего значения, то обновляются и личная лучшая позиция, и значение. Если же новое значение лучше текущего глобального лучшего значения, то обновляются и глобальная лучшая позиция, и значение.

Рисунок 10. Итеративный процесс оптимизации

Теперь создадим и отобразим трехмерный график, на котором будут визуализироваться конечные позиции частиц после завершения оптимизации. Каждая точка на графике соответствует конечной позиции частицы, а ее высота (z-координата) определяется значением целевой функции в этой точке. Глобальная лучшая позиция отмечается красным цветом, что позволяет визуально определить оптимальное решение.

Рисунок 11. Код

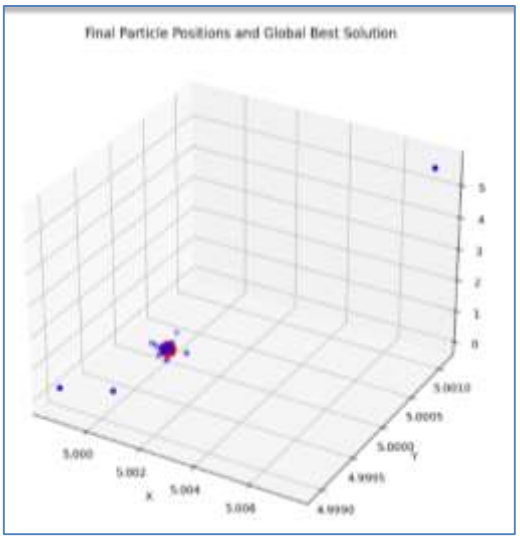


Рисунок 12. График

4. Выводы

Алгоритм роевой оптимизации, сгенерированный при помощи ИИ, основанный на поведении роев птиц или косяков рыб, позволяет эффективно находить решения, минимизирующие заданную целевую функцию, которая в данном случае отражает транспортный поток. Результаты визуализации показывают, что в процессе итерационной оптимизации частицы (представляющие транспортные средства) сходятся к глобальному оптимальному решению, обеспечивающему минимальное значение транспортного потока. Это свидетельствует о высокой эффективности применения роевого интеллекта для решения задач управления дорожным движением.

Библиографический список

1. Морозова М.Н., Егоров Е.И. Применение роевого интеллекта для оптимизации сложных процессов // Автоматизация и управление в машино- и приборостроении: сборник научных трудов. Саратов: Саратовский государственный технический университет имени Гагарина Ю.А., 2021. С. 72-78.
2. Горячев А.В., Новакова Н.Е. Применение алгоритмов роевого интеллекта для решения задач дискретной оптимизации в САПР // Информационные системы и технологии в моделировании и управлении. 2020. С. 15-19.
3. Тюхин М.В., Ломазов В.А. Применение биоинспирированных алгоритмов роевого интеллекта для решения задач оптимизации //Вопросы устойчивого развития общества. 2021. №. 6. С. 811-815.
4. Манусов В. З., Матренин П. В., Орлов Д. В. Оптимизация коэффициентов трансформации с применением алгоритмов направленного перебора и роевого интеллекта //Проблемы региональной энергетики. 2017. №. 1 (33). С. 15-23.