

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА** - **Российский технологический университет"**  **РТУ МИРЭА** |

**Институт** Информационных Технологий

**Кафедра** Вычислительной Техники

**ПРАКТИЧЕСКАЯ РАБОТА №5**

**«Алгоритм пчелиной колонии»**

**по дисциплине**

**«Системный анализ данных СППР»**

Студент группы: ИКБО-04-22 Егоров Л.А. *(Ф. И.О.студента)*

Преподаватель \_\_Железняк Л.М.\_\_ *(Ф.И.О. преподавателя)*

|  |  |  |
| --- | --- | --- |
|  |  |  |

Москва 2024

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc183614493)

[1 АЛГОРИТМ РОЯ ЧАСТИЦ 4](#_Toc183614494)

[1.1 Описание алгоритма 4](#_Toc183614495)

[1.2 Постановка задачи 5](#_Toc183614496)

[1.3 Ручной расчёт алгоритма 6](#_Toc183614497)

[1.4 Программная реализация 9](#_Toc183614498)

[ЗАКЛЮЧЕНИЕ 11](#_Toc183614499)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 12](#_Toc183614500)

[ПРИЛОЖЕНИЯ 13](#_Toc183614501)

[Приложение А 14](#_Toc183614502)

# ВВЕДЕНИЕ

В основу алгоритма оптимизации роем частиц положена социально-психологическая поведенческая модель толпы. Развитие алгоритма инспирировали такие задачи, как моделирование поведения птиц в стае и рыб в косяке. Целью было обнаружить базовые принципы, благодаря которым, например, птицы в стае ведут себя удивительно синхронно, меняя как по команде направления своего движения, так что стая движется как единое целое. К современному времени концепция алгоритма роя частиц развилась в высокоэффективный алгоритм оптимизации, часто составляющий конкуренцию лучшим модификациям генетического алгоритма.

В настоящее время роевой алгоритм применяются при решении задач численной и комбинаторной оптимизации, обучении искусственных нейронных сетей, построении нечетких контроллеров и т.д. в различных областях науки техники:

* управление энергетическими системами;
* решение NP-трудных комбинаторных проблем;
* задачи календарного планирования;
* оптимизация в мобильной связи;
* оптимизация процессов пакетной обработки;
* оптимизация многокритериальных задач;
* обработка изображений; распознавание образов;
* кластеризация данных;
* биоинформатика;
* проектирование сложных технических систем и т.д.

# 1 Алгоритм пчелиной колонии

Алгоритм роя частиц— это метод поиска оптимального решения, который позволяет оперирует популяцией агентов (частиц), которые являются возможными решениями. Эти частицы стремятся перемещаться к наилучшему решению, найденному одной из частиц роя, но также помнят свою оптимальную позицию – за счёт этого достигается то, что алгоритм менее склонен застревать в локальных минимумах.

## 1.1 Описание алгоритма

Сначала происходит инициализация начальных параметров и пчёл – генерация точек в области поиска (количество точек задано и равно S), а также свободных параметров алгоритма. Каждая точка имеет координаты (1.1.1).

(1.1.1)

где — номер частицы;

— размерность векторов в задаче.

Формирование подобластей происходит на основе Евклидова расстояния между пчёлами (1.1.2).

(1.1.2)

Сначала выбирается точка с наименьшим значением функции – она становится центром новой подобласти. Вокруг неё собираются все пчёлы, расстояние до которых от центральной пчелы меньше заданного числа . После проверки всех пчёл убираются те пчёлы, которые вошли в подобласть, и данные действия повторяются для оставшихся пчёл.

После формирования подобластей начинается поиск оптимального значения в каждой из них. В каждой области выбирается точка с наилучшим значение функции, вокруг неё в квадрате со стороной генерируются случайным образом S – 1 пчёл, а затем среди сгенерированных пчёл и центральной пчелы выбирается та, которая имеет наименьшее значение функции. Теперь эта точка становится центром новой области, и процесс повторяется до тех пор, пока не наилучшая точка не останется статичной в течение заданного числа итераций.

Такой поиск проводится в каждой из полученных подобластей, и точкой останова алгоритма является окончание поиска в последней области.

Точкой останова алгоритма является выполнение заданного числа итераций.

## 1.2 Постановка задачи

Цель работы: реализовать алгоритм пчелиной колонии для нахождения оптимального значения функции.

Поставлены следующие задачи:

* изучить алгоритм пчелиной колонии;
* выбрать тестовую функцию для оптимизации (нахождение глобального минимума);
* произвести ручной расчёт одной итерации алгоритма;
* разработать программную реализацию алгоритма пчелиной колонии для задачи минимизации функции.

Выбранная функция для оптимизации: функция Растригина (1.2.1). Глобальный минимум функции достигается в точке (0;0) и равен 0, при этом, в остальных локальных минимумах значение функции больше нуля. Функция рассматривается на области .

(1.2.1)

## 1.3 Ручной расчёт алгоритма

Выбранная функция: функция Растригина от двух переменных. Её формула представлена Формулой 1.2.1.

На Рисунке 1.3.1 представлен график этой функции.



Рисунок 1.3.1 – График функции Растригина

Инициализированы свободные параметры алгоритма:

;

;

Количество пчёл разведчиков (S): 10.

Создано 10 пчёл со следующими координатами:

## 1.4 Программная реализация

Для реализации расчётов алгоритма роя частиц написан программный код на языке Python.

В программной реализации зафиксированы следующие параметры:

* количество частиц: 20;
* количество итераций: 30
* c1 и c2: 2.

Код реализации роевого алгоритма для нахождения оптимального значения функции представлен в Листинге Б.1.

На Рисунке 1.4.1 представлен результат выполнения программы для нахождения оптимального значения функции – график зависимости оптимального решения от номера итерации.

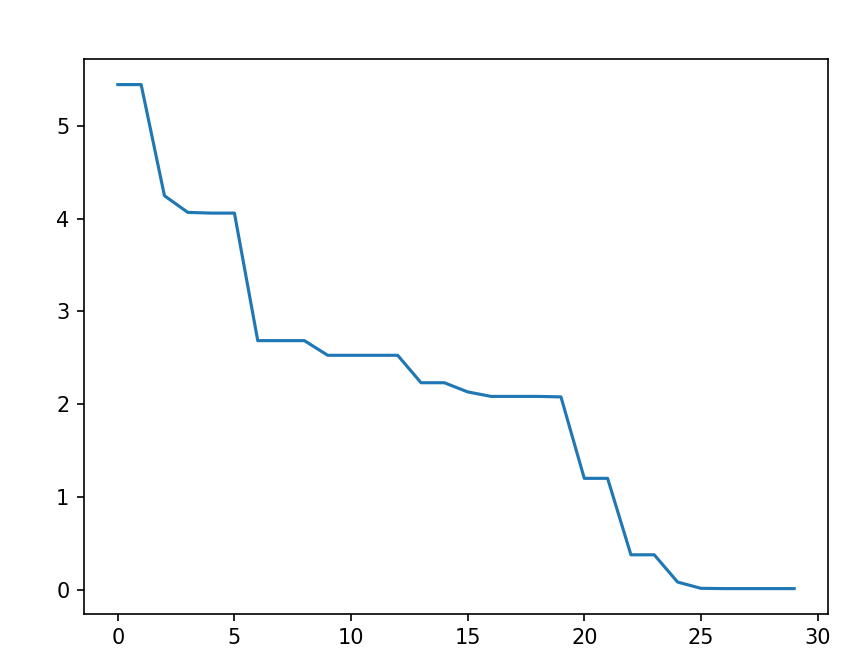


Рисунок 1.4.1 – График зависимости оптимального значения функции от номера итерации

# Заключение

В ходе выполнения данной работы выполнены поставленные задачи – изучен алгоритм роя частиц, произведён его ручной расчёт для решения задачи поиска глобального минимума функции, а также разработаны программы на языке Python для нахождения глобального минимума функции Растригина от двух переменных.

В заключение можно отметить, что роевой алгоритм является мощным инструментом для решения задач оптимизации, в которых стандартные методы недостаточно эффективны из-за наличия множества локальных минимумов. При этом, алгоритм является простым в реализации и имеет мало свободных параметров, из-за чего алгоритм не нуждается.

# Список информационных источников

1. Сорокин, А.Б. Введение в роевой интеллект: теория, расчеты и приложения [Электронный ресурс]: Учебно-методическое пособие / А.Б. Сорокин. — М.: Московский технологический университет (МИРЭА), 2019.
2. Карпенко, А.П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учебное пособие / А. П. Карпенко. – Москва: Издательство МГТУ им. Н. Э. Баумана, 2021. – 446 c.
3. Пряжников, А. А. Имитация отжига: простое объяснение метода и его применение [Электронный ресурс]. URL: <https://pryazhnikov-com.turbopages.org/pryazhnikov.com/s/notes/simulated-annealing/> (Дата обращения: 12.11.2024).
4. Сорокин, А. Б. Безусловная оптимизация : учебно-методическое пособие / А. Б. Сорокин, О. В. Платонова, Л. М. Железняк ; — Москва : МИРЭА - Российский технологический университет, 2020.

# приложения

Приложение А — Реализация роевого алгоритма в задаче оптимизации на языке Python.

### Приложение А

Реализация роевого алгоритма в задаче оптимизации на языке Python.

Листинг А.1 – Реализация роевого алгоритма

import numpy as np

import matplotlib.pyplot as plt

def rastrigin(x: np.ndarray):

    return 10 \* len(x) + np.sum(x\*\*2 - 10 \* np.cos(2 \* np.pi \* x))

class Particle:

    def \_\_init\_\_(self):

        self.\_max = 5.12

        self.\_min = -self.\_max

        self.n = 2

        self.x = np.random.random(size=self.n) \* (self.\_max - self.\_min) + self.\_min

        self.best\_x = self.x.copy()

        self.speed = np.zeros(shape=(self.n,))

    def correct\_speed(self, global\_best: np.ndarray):

        c1 = 2

        c2 = 2

        alpha = np.random.random()

        self.speed += c1 \* alpha \* (self.best\_x - self.x) + c2 \* (1 - alpha) \* (global\_best - self.x)

    def correct\_position(self):

        self.x += self.speed

    def \_\_str\_\_(self):

        return f"Текущая позиция: {self.x} -- лучшая позиция: {self.best\_x}"

class Swarm:

    def \_\_init\_\_(self, particle\_count: int = 10):

        self.particles = [Particle() for \_ in range(particle\_count)]

        self.best\_solution: np.ndarray | None = None

    def solution\_step(self):

        if self.best\_solution is None:

            self.best\_solution = self.particles[0].best\_x.copy()

        for particle in self.particles:

            if rastrigin(particle.x) < rastrigin(particle.best\_x):

                particle.best\_x = particle.x.copy()

            if rastrigin(particle.best\_x) < rastrigin(self.best\_solution):

                self.best\_solution = particle.best\_x.copy()

        for particle in self.particles:

            particle.correct\_speed(self.best\_solution)

            particle.correct\_position()

        return rastrigin(self.best\_solution), self.best\_solution

    def draw\_swarm(self):

        particle\_x = [particle.x[0] for particle in self.particles]

        particle\_y = [particle.x[1] for particle in self.particles]

        ax = plt.gca()

        ax.set\_xlim(-5.12, 5.12)

        ax.set\_ylim(-5.12, 5.12)

Окончание Листинга А.1

        plt.scatter(particle\_x, particle\_y)

        plt.show()

class Solution:

    def \_\_init\_\_(self):

        self.swarm = Swarm(particle\_count=20)

    def solve(self):

        steps = 30

        history = []

        for step in range(steps):

            value, x = self.swarm.solution\_step()

            print(f'Итерация: {step + 1}. Значение: {value} в точке {x}')

            history.append(value)

        plt.plot(history)

        plt.show()

def main():

    solution = Solution()

    solution.solve()

if \_\_name\_\_ == '\_\_main\_\_':

    main()