

|  |
| --- |
| 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](#_Toc184211326)

[1 Алгоритм пчелиной колонии 4](#_Toc184211327)

[1.1 Описание алгоритма 4](#_Toc184211328)

[1.2 Постановка задачи 5](#_Toc184211329)

[1.3 Ручной расчёт алгоритма 5](#_Toc184211330)

[1.4 Программная реализация 10](#_Toc184211331)

[Заключение 11](#_Toc184211332)

[Список информационных источников 12](#_Toc184211333)

[Приложения 13](#_Toc184211334)

# ВВЕДЕНИЕ

Алгоритм пчелиной колонии - это эвристический метод оптимизации, разработанный Марко Дориго и Дино Д'Агостино в 2005 году. Этот алгоритм вдохновлен поведением медоносных пчел, которые демонстрируют удивительную способность находить наилучшие источники нектара для сбора меда.

Основной целью работы пчелиной колонии в природе является разведка пространства вокруг улья с целью поиска нектара с последующим его сбором. Для этого в составе колонии существуют различные типы пчел: пчелы-разведчики и рабочие пчелы-фуражиры (кроме них, в колонии существуют трутни и матка, не участвующие в процессе сбора нектара). Разведчики ведут исследование окружающего улей пространства и сообщают информацию о перспективных местах, в которых было обнаружено наибольшее количество нектара (для обмена информацией в улье существует специальный механизм, именуемый танцем пчелы) [3].

Алгоритм пчелиной колонии моделирует это поведение. Вместо реальных пчел и танцев, алгоритм использует "искусственных пчел" и "искусственные танцы". Искусственные пчелы перемещаются по пространству поиска, представленному в виде графа или сетки, и оценивают качество каждой позиции. Затем они возвращаются в "улей" и передают информацию о найденных позициях другим пчелам. Вероятность выбора пчелой определенной позиции зависит от ее качества и количества информации, полученной от других пчел. Со временем, пчелы концентрируют свои усилия на наиболее перспективных позициях.

Алгоритм пчелиной колонии широко используется для решения различных задач оптимизации [2], таких как:

* задача календарного планирования;
* задача коммивояжёра;
* транспортная задача.

# 1 Алгоритм пчелиной колонии

## 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 [1].

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 [5]. 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:

Среди оставшихся точек лучшее значение имеется у пчелы : значение функции у неё равно 22.833.

Далее рассчитывается Евклидово расстояние между точкой и оставшимися точками по Формуле 1.1.2.

Следовательно, в область точки вошла точка .

Среди оставшихся точек лучшее значение имеется у пчелы : значение функции у неё равно 28.532.

Далее рассчитывается Евклидово расстояние между точкой и оставшимися точками по Формуле 1.1.2.

Следовательно, в область точки вошла точка .

Среди оставшихся точек лучшее значение имеется у пчелы : значение функции у неё равно 28.561.

Далее рассчитывается Евклидово расстояние между точкой и оставшимися точками по Формуле 1.1.2.

Следовательно, точка образует область сама с собой.

Среди оставшихся точек лучшее значение имеется у пчелы : значение функции у неё равно 33.228.

Далее рассчитывается Евклидово расстояние между точкой и оставшимися точками по Формуле 1.1.2.

Следовательно, в область точки вошла точка .

Среди оставшихся точек лучшее значение имеется у пчелы : значение функции у неё равно 43.799.

Далее рассчитывается Евклидово расстояние между точкой и оставшимися точками по Формуле 1.1.2.

Следовательно, в область точки вошла точка .

Среди оставшихся точек лучшее значение имеется у пчелы : значение функции у неё равно 49.092.

Поскольку точек больше не осталось, то точка образует область сама с собой.

Рассмотрим поиск в первой подобласти. Лучшая точка: (0.795, -1.717) со значением функции 22.833.

Новые сгенерированные точки имеют следующие координаты (точка является текущим центром области):

Минимальное значение среди достигнуто точкой (значение функции равно 7.640). Следовательно, эта точка становится центром области, и происходит переход к новой итерации.

## 1.4 Программная реализация

Для реализации расчётов алгоритма пчелиной колонии написан программный код на языке Python.

В программной реализации зафиксированы следующие параметры:

* количество пчёл: 100;
* количество итераций: 30;
* ;
* ;

Код реализации алгоритма пчелиной колонии для нахождения оптимального значения функции представлен в Листинге А.1.

На Рисунке 1.4.1 представлен результат выполнения программы для нахождения оптимального значения функции – консольный вывод результатов поиска в нескольких областях.

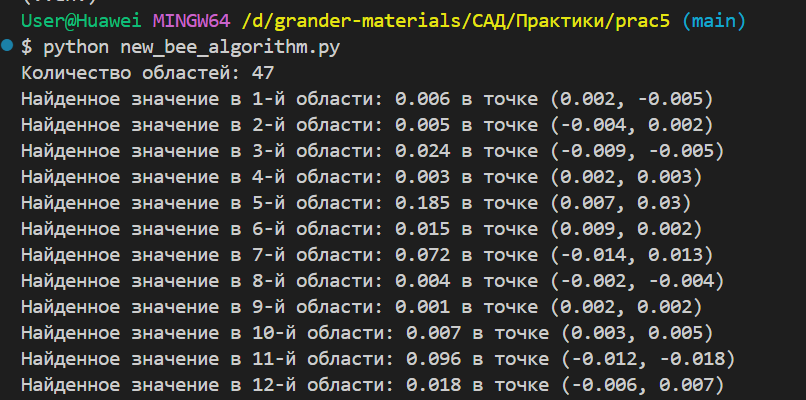


Рисунок 1.4.1 – Результаты поиска в первых 12 областях

# Заключение

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

В заключение можно отметить, что алгоритм пчелиной колонии является мощным инструментом для решения задач оптимизации (в том числе, задач нахождения глобального минимума функции), в которых стандартные методы недостаточно эффективны из-за наличия множества локальных минимумов. Алгоритм имеет высокую сходимость, однако его результативность сильно зависит от настройки большого количества свободных параметров.

# 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. Алгоритм пчелиной колонии. [Электронный ресурс]: Википедия. – URL: https://ru.wikipedia.org/wiki/Алгоритм\_пчелиной\_колонии (Дата обращения: 25.11.2024).
4. Сорокин, А. Б. Безусловная оптимизация : учебно-методическое пособие / А. Б. Сорокин, О. В. Платонова, Л. М. Железняк ; — Москва : МИРЭА - Российский технологический университет, 2020.
5. Функция Растригина. [Электронный ресурс]: Википедия. – URL: https://ru.wikipedia.org/wiki/Функция\_Растригина (Дата обращения: 04.12.2024).

# приложения

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

### Приложение А

Реализация пчелиного алгоритма в задаче оптимизации на языке Python.

Листинг А.1 – Реализация алгоритма пчелиной колонии

import numpy as np

def rastrigin(x: np.ndarray):

    return np.sum(np.square(x) - 10 \* np.cos(2 \* np.pi \* x) + 10, axis=-1)

class BeeColony:

    def \_\_init\_\_(self, bee\_count: int = 100,

                       max\_iter: int = 5,

                       max\_distance: float = 1,

                       search\_size: float = 1,

                       n: int = 2):

        self.bee\_count = bee\_count

        self.max\_iter = max\_iter

        self.max\_distance = max\_distance

        self.search\_size = search\_size

        self.n = n

        self.\_max = 5.12

        self.\_min = -self.\_max

    def init\_population(self):

        self.bees = (self.\_max - self.\_min) \* np.random.random(size=(self.bee\_count, self.n)) + self.\_min

        self.values = rastrigin(self.bees)

    def create\_fields(self):

        self.fields = []

        visited = []

        while len(visited) < self.bee\_count:

            min\_index = None

            best\_bee = None

            for i, bee in enumerate(self.bees):

                if i not in visited:

                    if min\_index is None and best\_bee is None \

                       or rastrigin(bee) < rastrigin(best\_bee):

                        best\_bee = bee

                        min\_index = i

            current\_field = []

            current\_field.append(min\_index)

            for i, bee in enumerate(self.bees):

                if i not in visited and i != min\_index:

                    if np.linalg.norm(bee - best\_bee) < self.max\_distance:

                        current\_field.append(i)

            self.fields.append(current\_field)

            visited.extend(current\_field)

        print(f'Количество областей: {len(self.fields)}')

    def find\_field\_best(self, k: int):

        cnt = 0

        new\_best\_bee = None

        while cnt < self.max\_iter:

            if new\_best\_bee is None:

                current\_best\_bee = self.bees[self.fields[k][0]]

            else:

                current\_best\_bee = new\_best\_bee

            min\_search = np.clip(current\_best\_bee - self.search\_size, self.\_min, self.\_max)

Окончание Листинга А.1

            max\_search = np.clip(current\_best\_bee + self.search\_size, self.\_min, self.\_max)

            new\_bees = (max\_search - min\_search) \* np.random.random(size=(self.bee\_count - 1, self.n)) + min\_search

            new\_bees = np.vstack([new\_bees, current\_best\_bee])

            values = rastrigin(new\_bees)

            new\_best\_bee = new\_bees[np.where(np.abs(np.min(values) - values) < 1e-10)][0]

            if np.linalg.norm(current\_best\_bee - new\_best\_bee) < 1e-5:

                cnt += 1

            else:

                cnt = 0

        print(f'Найденное значение в {k + 1}-й области: {rastrigin(new\_best\_bee):.3f}'

              f' в точке ({", ".join(map(lambda x: str(round(x, 3)), new\_best\_bee))})')

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

    bee\_colony = BeeColony(search\_size=1, max\_distance=0.8, max\_iter=30,

                           bee\_count=100)

    bee\_colony.init\_population()

    bee\_colony.create\_fields()

    for i in range(len(bee\_colony.fields)):

        bee\_colony.find\_field\_best(i)