УДК 004

Применение алгоритма пчелиной колонии

Романов Даниил Алексеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Цель данного исследования заключается в применении алгоритма пчелиной колонии для решения задач оптимизации. Для этого был написан код на языке программирования Python в среде google colab. В результате исследования было установлено, что алгоритм пчелиной колонии является эффективным инструментом для решения задач оптимизации.
Ключевые слова: алгоритм пчелиной колонии, задача оптимизации, Python, Google Colab

Application of the bee colony algorithm

Romanov Daniil Alekseevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this study is to apply the bee colony algorithm to solve optimization problems. To do this, the code was written in the Python programming language in the google colab environment. As a result of the study, it was found that the bee colony algorithm is an effective tool for solving optimization problems.
Keywords: bee colony algorithm, optimization problem, Python, Google Colab

1 Введение

1.1 Актуальность

Оптимизация является важной задачей во многих областях, включая производство, науку, финансы и технологии. Алгоритм пчелиной колонии является одним из методов оптимизации, который показывает хорошие результаты в различных областях. Исследование данного метода является актуальным для улучшения его эффективности и расширения его применения.

1.2Обзор исследований

И. А. Ходашинский, И. В. Горбунов и П. А. Дудин посвятили свою статью применению муравьиного и пчелиного алгоритмов для обучения нечетких систем. В статье рассматриваются основные принципы работы алгоритмов и приводятся примеры их использования для оптимизации параметров нечетких моделей [1].

В статье "Оптимизация параметров нечетких систем на основе модифицированного алгоритма пчелиной колонии" авторства И.А.Ходашинского и И.В. Горбунова, опубликованной в журнале Мехатроника, автоматизация, управление, рассматривается использование модифицированного алгоритма пчелиной колонии для оптимизации параметров нечетких систем. В статье приводятся примеры применения алгоритма и описываются основные принципы его работы [2].

В. М. Курейчик и А. А. Кажаров описали применение пчелиных алгоритмов для решения задач комбинаторной оптимизации. В статье описываются основные принципы работы алгоритма и приводятся примеры его применения для решения задач нахождения оптимальных маршрутов и размещения объектов [3].

1.3 Цель исследования

Цель данного исследования состоит в применении алгоритма пчелиной колонии для решения задач оптимизации и анализе его результатов. В работе будет использоваться язык программирования Python и среда программирования google colab.

2 Материалы и методы

Для работы понадобится онлайн среда программирования Google Colab [4].

3 Результаты и обсуждение

Импортируем необходимые библиотеки. Библиотека random используется для генерации случайных чисел, а библиотека math - для выполнения математических операций (рис.1).

```
import random
import math
```

Рисунок 1 - Импорт библиотек

Определяем класс для представления пчел. Класс Bee представляет пчелу и содержит два атрибута: position - вектор, представляющий позицию пчелы в пространстве решений, и value - значение целевой функции для этой позиции (рис.2).

```
class Bee:
    def __init__(self, position, value):
        self.position = position
        self.value = value
```

Рисунок 2 - Класс Bee

Определяем функции для вычисления значения целевой функции. Функция objective_function(x) вычисляет значение целевой функции для заданного вектора x. В данном случае она просто суммирует квадраты каждого элемента вектора (рис.3).

```
def objective_function(x):
    return sum([i**2 for i in x])
```

Рисунок 3 - Функция objective_function

Определяем функцию для генерации начального роя пчел. Функция generate_initial_bee_hive() генерирует начальный рой пчел. Она принимает следующие аргументы:

- num_scout_bees: количество разведчиков (scout bees);
- num_best_bees: количество лучших пчел (best bees);
- num_best_sites: количество лучших мест (best sites);
- num_selected_bees: количество выбранных пчел (selected bees);
- num_elite_sites: количество элитных мест (elite sites);
- lower_bounds: список нижних границ для каждой переменной вектора решения;
- upper_bounds: список верхних границ для каждой переменной вектора решения.

Функция генерирует начальный рой пчел, случайным образом распределяя пчел по всему пространству решений. Она также выбирает лучшие пчелы и лучшие места, а также выбранные пчелы и элитные места (рис.4).

```
def generate_initial_bee_hive(num_scout_bees, num_best_bees,\
                              num_best_sites, num_selected_bees, \
                              num_elite_sites, lower_bounds, \
                              upper_bounds):
    hive = []
    for i in range(num_scout_bees):
        position = [random.uniform(lower_bounds[j], upper_bounds[j]) \
                    for j in range(len(lower_bounds))]
        value = objective_function(position)
        hive.append(Bee(position, value))
    best_bees = sorted(hive, key=lambda bee: bee.value)[:num_best_bees]
    best_sites = [bee.position for bee in best_bees]
    selected_bees = best_bees[:num_selected_bees]
    elite_sites = best_sites[:num_elite_sites]
    for i in range(num_best_bees, num_best_sites):
        position = [random.uniform(lower_bounds[j], upper_bounds[j]) \
                    for j in range(len(lower_bounds))]
        value = objective_function(position)
        hive.append(Bee(position, value))
    return hive, best_bees, best_sites, selected_bees, elite_sites
```

Рисунок 4 - Функция generate_initial_bee_hive

Определяем функцию для выбора следующей позиции для пчелы. Функция select_new_position() выбирает новую позицию для пчелы. Она принимает следующие аргументы: position: текущая позиция пчелы; best_site: лучшее место в рое; elite_site: элитное место в рое; lower_bounds: список нижних границ для каждой переменной вектора решения; upper_bounds: список верхних границ для каждой переменной вектора решения.

Функция использует формулу, основанную на поведении пчел в поиске пищи, чтобы выбрать новую позицию для пчелы. Она также гарантирует, что новая позиция не выходит за пределы нижних и верхних границ (рис.5).

```
def select_new_position(position, best_site, elite_site, \
                        lower_bounds, upper_bounds):
    phi = random.uniform(-1, 1)
    new_position = [position[i] + phi * (position[i] - best_site[i]) \
                    + random.uniform(-1, 1) * \
                     (position[i] - elite_site[i]) \
                    for i in range(len(position))]
    new_position = [min(max(new_position[i], lower_bounds[i]), \
                        upper_bounds[i]) for i in range(len(position))]
    return new_position
```

Рисунок 5 - Функция select_new_position()

Определяем функцию для обновления роя пчел. Функция update_bee_hive() обновляет рой пчел. Она принимает следующие аргументы:

- hive: текущий рой пчел;
- best_bees: лучшие пчелы в рое;
- best_sites: лучшие места в рое;
- selected_bees: выбранные пчелы в рое;
- elite_sites: элитные места в рое;
- lower_bounds: список нижних границ для каждой переменной вектора решения;
- upper_bounds: список верхних границ для каждой переменной вектора решения.

Функция для каждой пчелы в рое выбирает новую позицию и проверяет, улучшает ли это значение целевой функции. Если новое значение лучше, чем старое, то пчела перемещается на новую позицию. Затем функция выбирает новый набор лучших пчел, выбранных пчел и элитных мест (рис.6).

```
def update_bee_hive(hive, best_bees, best_sites, selected_bees, \
                    elite_sites, lower_bounds, upper_bounds):
    for i in range(len(hive)):
        if hive[i] not in best_bees:
            if random.random() < 0.5:
                best_site = random.choice(best_sites)
                elite_site = random.choice(elite_sites)
                new_position = select_new_position(hive[i].position, \
                                                   best_site, elite_site, \
                                                   lower_bounds, upper_bounds)
            else:
                selected_bee = random.choice(selected_bees)
                new_position = select_new_position(hive[i].position, \
                                                   selected_bee.position, \
                                                   selected_bee.position, \
                                                   lower_bounds, upper_bounds)
            new_value = objective_function(new_position)
            if new_value < hive[i].value:
                hive[i] = Bee(new_position, new_value)
    best_bees = sorted(hive, key=lambda bee: bee.value)[:len(best_bees)]
    best_sites = [bee.position for bee in best_bees]
    selected_bees = best_bees[:len(selected_bees)]
    elite_sites = best_sites[:len(elite_sites)]
    return hive, best_bees, best_sites, selected_bees, elite_sites
```

Рисунок 6 - Функция update_bee_hive

Определяем функцию для выполнения алгоритма пчелиной колонии. Функция run_bee_algorithm() выполняет алгоритм пчелиной колонии. Она принимает следующие аргументы:

- num_iterations: количество итераций, которое нужно выполнить;
- num_scout_bees: количество разведчиков (scout bees);
- num_best_bees: количество лучших пчел (best bees);
- num_best_sites: количество лучших мест (best sites);

- num_selected_bees: количество выбранных пчел (selected bees);
- num_elite_sites: количество элитных мест (elite sites);
- lower_bounds: список нижних границ для каждой переменной вектора решения;
- upper_bounds: список верхних границ для каждой переменной вектора решения.

Функция генерирует начальный рой пчел, затем запускает цикл, в котором обновляет рой пчел на каждой итерации. По завершении цикла функция возвращает лучшую позицию и соответствующее значение целевой функции (рис.7).

```
def run_bee_algorithm(num_iterations, num_scout_bees, num_best_bees,\
                      num_best_sites, num_selected_bees, \
                      num_elite_sites, lower_bounds, upper_bounds):
    hive, best_bees, best_sites, selected_bees, \
    elite_sites = generate_initial_bee_hive(num_scout_bees, \
                                            num_best_bees, \
                                            num_best_sites, \
                                            num_selected_bees, \
                                            num_elite_sites, \
                                            lower_bounds, \
                                            upper_bounds)
    for i in range(num_iterations):
        hive, best_bees, best_sites, selected_bees,\
         elite_sites = update_bee_hive(hive, best_bees, \
                                       best_sites, selected_bees,\
                                       elite_sites, lower_bounds, \
                                       upper_bounds)
    return best_bees[0].position, best_bees[0].value
```

Рисунок 7 - Функция run_bee_algorithm

Вызовем функцию run bees algorithm(). Для примера запускаем алгоритм пчелиной колонии для задачи оптимизации с 3 переменными в промежутке от -5 до 5. Запускаем алгоритм на 100 итераций с параметрами, определенными в функции run_bee_algorithm(). Затем выводим на экран лучшую позицию и соответствующее значение целевой функции (рис.8).

```
lower_bounds = [-5, -5, -5]
upper_bounds = [5, 5, 5]
position, value = run_bee_algorithm(num_iterations=100, \
                                    num_scout_bees=50, \
                                    num_best_bees=20, \
                                    num_best_sites=10, \
                                    num_selected_bees=5, \
                                    num_elite_sites=2, \
                                    lower_bounds=lower_bounds, \
                                    upper_bounds=upper_bounds)
print(f'Best position: {position}, Best value: {value}')
```

Рисунок 8 - Пример вызова функции run bees algorithm

Результаты выполнения программы могут отличаться в зависимости от запуска, так как алгоритм пчелиной колонии является стохастическим методом оптимизации. Однако, в целом, программа должна выводить на экран лучшую найденную пчелами позицию и соответствующее значение целевой функции. Например, при одном из запусков программа может вывести следующий результат (рис.9).

```
Best position: [-1.1005875820263471e-06, 4.0945015922092673e-07, 8.583031080856098e-07], Best value: 2.1156266839460618e-12
```

Рисунок 9 - Результат выполнения программы

Это означает, что пчелы нашли оптимальную позицию в точке (-1.1005875820263471e-06, 4.0945015922092673e-07, 8.583031080856098e-07), где значение целевой функции равно 2.1156266839460618e-12. Это может быть полезно, если мы хотим минимизировать значение данной функции.

Выводы

В данной работе был исследован алгоритм пчелиной колонии и его применение в задачах оптимизации. Был написан код на языке Python, реализующий алгоритм пчелиной колонии. В ходе исследования было установлено, что алгоритм пчелиной колонии является эффективным инструментом для решения задач оптимизации. Программа может быть использована для оптимизации функций в различных областях:

- Машинное обучение: для настройки гипер-параметров моделей машинного обучения, таких как коэффициенты регуляризации, скорость обучения, количество слоев и нейронов в них и т.д.
- Финансы: для оптимизации портфелей инвестиций, распределения капитала между различными активами и т.д.
- Инженерия: для оптимизации параметров процессов и конструкций, например, для оптимизации геометрии лопастей вентилятора, настройки параметров регулирования температуры и давления в промышленных процессах и т.д.

Также программа может быть использована в обучающих целях, для изучения алгоритма пчелиной колонии и оптимизации функций в общем.

Ознакомиться с кодом и проверить его работоспособность можно по данной ссылке [5].

Библиографический список

1. Ходашинский И. А., Горбунов И. В., Дудин, П. А. Алгоритмы муравьиной и пчелиной колонии для обучения нечетких систем // Доклады Томского государственного университета систем управления и радиоэлектроники. 2009. Т. 2 (20). С. 157-161.
2. Ходашинский И. А., Горбунов И. В. Оптимизация параметров нечетких систем на основе модифицированного алгоритма пчелиной колонии // Мехатроника, автоматизация, управление. 2012. № 10. С. 15-20.
3. Курейчик В. М., Кажаров А. А. Использование пчелиных алгоритмов для решения комбинаторных задач // Штучний інтелект. 2010. № 1. С. 86-91.
4. Google Colab URL: https://colab.research.google.com
5. Код программы URL: https://colab.research.google.com/drive/149PJ_cWK8amtDMlQswMj3Iddp9lN2-9s?usp=sharing