Хартов А.Е. 8ПИЭ-41

Лабораторная работа № 1.
Оптимизация грузоперевозок малогабаритных грузовкак пример задачи о коммивояжере

Постановка задачи: Двумя автомобилями необходимо развезти товары с базы по 19 адресам. Матрица расстояний между адресами и базой задана в таблице. Составить оптимальный (рациональный) маршрут доставки грузов.

Таким образом нам нужно реализовать замкнутый алгоритм для двух коммивояжеров.

Листинг программы:

```
import numpy as np
from python_tsp.exact import solve_tsp_branch_and_bound
import datetime
from sklearn.cluster import KMeans

# Матрица расстояний
distance_matrix = np.array([
    [0, 3,  8,  4,  2,  9,  6,  9,  4,  7,  5,  9,  5,  7,  8,  5,  2,  5,  5,  8],
    [2, 0,  6,  5,  9,  2,  5,  6,  7,  9,  8,  2,  2,  2,  9,  4,  9,  6,  3,  3],
    [7, 6,  0,  8,  4,  3,  3,  4,  5,  4,  9,  7,  5,  4,  3,  2,  9,  4,  7,  8],
    [6, 7,  7,  0,  7,  9,  3,  6,  6,  8,  3,  4,  5,  6,  3,  9,  8,  9,  2,  4],
    [4, 4,  4,  6,  0,  9,  4,  9,  3,  8,  5,  4,  4,  6,  3,  5,  5,  6,  2,  9],
    [8, 2,  4,  6,  5,  0,  6,  2,  9,  5,  3,  6,  7,  8,  7,  8,  5,  8,  9,  6],
    [6, 2,  8,  3,  8,  6,  0,  7,  3,  8,  3,  3,  2,  7,  6,  5,  2,  2,  3,  8],
    [4, 4,  6,  4,  3,  2,  2,  0,  6,  7,  5,  6,  2,  7,  6,  3,  5,  2,  6,  6],
    [6, 5,  6,  9,  7,  9,  2,  6,  0,  8,  6,  6,  6,  8,  2,  8,  7,  9,  2,  8],
    [8, 7,  5,  2,  4,  3,  8,  3,  4,  0,  7,  7,  7,  3,  6,  6,  7,  4,  8,  6],
    [6, 3,  4,  7,  5,  2,  6,  3,  5,  8,  0,  8,  8,  5,  5,  2,  8,  3,  5,  2],
    [3, 4,  2,  6,  9,  8,  2,  5,  3,  5,  3,  0,  6,  9,  2,  7,  7,  8,  5,  4],
    [4, 3,  2,  6,  8,  3,  2,  6,  3,  7,  7,  7,  0,  8,  6,  8,  2,  8,  8,  3],
    [5, 8,  5,  8,  4,  4,  4,  8,  5,  4,  5,  4,  7,  0,  4,  6,  7,  4,  2,  6],
    [9, 6,  9,  9,  5,  9,  5,  8,  2,  6,  4,  3,  4,  4,  0,  7,  6,  4,  3,  8],
    [8, 7,  5,  2,  6,  3,  8,  3,  8,  3,  4,  7,  3,  7,  8,  0,  5,  6,  4,  3],
    [4, 4,  5,  4,  7,  9,  6,  5,  8,  6,  5,  6,  3,  7,  3,  5,  0,  6,  3,  6],
    [2, 2,  5,  6,  4,  9,  5,  8,  8,  4,  6,  3,  8,  9,  7,  9,  4,  0,  4,  7],
    [6, 3,  6,  9,  3,  2,  2,  4,  2,  7,  2,  3,  2,  8,  5,  4,  8,  7,  0,  6],
    [9, 7,  7,  7,  3,  9,  4,  8,  9,  5,  5,  8,  3,  6,  7,  4,  7,  6,  2,  0]
])

def solve_two_salesmen_tsp(distance_matrix):
    n_cities = len(distance_matrix)
    cities = np.arange(n_cities)

    # Шаг 1: Кластеризация городов на 2 группы (без учета города 0)
    kmeans = KMeans(n_clusters=2, random_state=0).fit(distance_matrix[1:, 1:])
    labels = kmeans.labels_

    # Разделяем города на две группы, добавляя город 0 в обе группы
    cities_group_1 = np.append([0], cities[1:][labels == 0])
    cities_group_2 = np.append([0], cities[1:][labels == 1])

    print(f"Города для коммивояжера 1: {cities_group_1}")
    print(f"Города для коммивояжера 2: {cities_group_2}")

    # Создаем матрицы расстояний для каждой группы
    distance_matrix_1 = distance_matrix[np.ix_(cities_group_1, cities_group_1)]
    distance_matrix_2 = distance_matrix[np.ix_(cities_group_2, cities_group_2)]

    # Шаг 2: Решаем задачу TSP для каждой группы с обязательным стартом и
финишем в 0
```

```
    st1 = datetime.datetime.now()
    perm1, dist1 = solve_tsp_branch_and_bound(distance_matrix_1)
    fin1 = datetime.datetime.now()

    st2 = datetime.datetime.now()
    perm2, dist2 = solve_tsp_branch_and_bound(distance_matrix_2)
    fin2 = datetime.datetime.now()

    # Восстанавливаем исходные индексы
    route1 = cities_group_1[perm1]
    route2 = cities_group_2[perm2]

    # Шаг 3: Выводим результаты
    print(f"Маршрут коммивояжера 1: {route1}, длина маршрута: {dist1}")
    print(f"Маршрут коммивояжера 2: {route2}, длина маршрута: {dist2}")
    print(f"Время решения для коммивояжера 1: {fin1 - st1}")
    print(f"Время решения для коммивояжера 2: {fin2 - st2}")

    total_distance = dist1 + dist2
    print(f"Общая длина маршрутов: {total_distance}")

# Вызов функции
solve_two_salesmen_tsp(distance_matrix)
```

Алгоритм для решения задачи двух коммивояжеров можно описать следующим образом:

1. Подготовка данных
   - Матрица расстояний (`distance_matrix`) задает расстояния между городами, где каждый элемент матрицы `[i, j]` — это расстояние между городами `i` и `j`.
   - Все города, включая стартовый город 0, индексируются в массиве `cities`.

2. Кластеризация городов
   - Город 0 является стартовой и конечной точкой для обоих коммивояжеров.
   - Все остальные города (без города 0) разделяются на две группы с помощью алгоритма KMeans. Это делается с целью распределить города между двумя коммивояжерами.
   - Алгоритм кластеризации KMeans делит города на две группы по географической близости (схожести расстояний).
   - Каждой группе добавляется город 0, который является общей стартовой и конечной точкой для обоих маршрутов.

3. Создание матриц расстояний для двух групп
   - На основе разбиения городов на две группы, создаются две подматрицы расстояний: одна для первой группы городов, вторая — для второй группы.
   - Эти подматрицы содержат расстояния только между городами, входящими в соответствующую группу (включая город 0).

4. Решение задачи TSP для каждой группы

- Для каждой из подматриц расстояний решается задача коммивояжера с помощью алгоритма ветвей и границ (`solve_tsp_branch_and_bound`).

- Этот алгоритм находит оптимальный маршрут для каждого коммивояжера, который минимизирует общее расстояние, начиная и заканчивая маршрут в городе 0.

5. Восстановление маршрутов

- После решения задачи для каждой подгруппы, результаты (маршрут и его длина) выводятся для каждого коммивояжера.

- Индексы городов восстанавливаются, чтобы отобразить их исходные номера.

6. Подсчет общей длины маршрутов

- Общая длина маршрутов обоих коммивояжеров суммируется и выводится как итоговое решение.

Таким образом, этот алгоритм эффективно решает задачу двух коммивояжеров, деля города на две группы и решая TSP для каждой группы отдельно.

Результаты работы:

```
Города для коммивояжера 1: [ 0  3  4  6  8 11 12 13 14 16 17 18]
Города для коммивояжера 2: [ 0  1  2  5  7  9 10 15 19]
Маршрут коммивояжера 1: [ 0  4  8 14 13 18 12 16  3 11  6 17  0], длина маршрута: 31.0
Маршрут коммивояжера 2: [ 0 10  2 15 19  9  7  5  1  0], длина маршрута: 28.0
Время решения для коммивояжера 1: 0:00:00.013718
Время решения для коммивояжера 2: 0:00:00.002813
Общая длина маршрутов: 59.0
```

Итог:

В ходе выполнения лабораторной работы был реализован алгоритм решения замкнутой задачи о коммивояжере для двух автомобилей