Python | 动态规划求解TSP

cathy1997 关注

2019.12.25 14:04:41 字数 253 阅读 2,009

主要参考：Python求解tsp问题（动态规划，简单易懂）CSDN博客

解题思路主要有两部分：

1. i为当前节点（城市），S为还没有遍历的节点（城市集合），表示从第i个节点起，经历S集合中所有的点，到达终点的最短路径长度。

$$\varepsilon(i,S)=min\left(d(i,j)_{j\in S}+\varepsilon\left(j,S_{/j}\right)\right)$$

2. 回溯找到最优的路径，需要将S集合一一对应一个数字（类似于编码，一般用二进制），然后比如从节点i等于0开始，未经历集合为 `S={1, 2, 3}`，而下一步最优的节点 j 等于2，那么 `M[i][s]=j`，回溯时只用从 `M[0][S]` 向后推即可。

```
import numpy as np
import itertools
import random
import matplotlib.pyplot as plt
# 设置中文识别
plt.rcParams['font.sans-serif'] = ['SimHei']

# tsp问题
class Solution:
    def __init__(self,X,start_node):
        self.X = X #距离矩阵
        self.start_node = start_node #开始的节点
        self.array = [[0]*(2**(len(self.X)-1)) for i in range(len(self.X))] #记录处于x节点，未经

    def transfer(self, sets):
        su = 0
        for s in sets:
            su = su + 2**(s-1) # 二进制转换
        return su

    # tsp总接口
    def tsp(self):
        s = self.start_node
        num = len(self.X)
        cities = list(range(num)) #形成节点的集合
        # past_sets = [s] #已遍历节点集合
        cities.pop(cities.index(s)) #构建未经历节点的集合
        node = s #初始节点
        return self.solve(node, cities) #求解函数

    def solve(self, node, future_sets):
        # 迭代终止条件，表示没有了未遍历节点，直接连接当前节点和起点即可
        if len(future_sets) == 0:
            return self.X[node][self.start_node]
        d = 99999
        # node如果经过future_sets中节点，最后回到原点的距离
        distance = []
        # 遍历未经历的节点
        for i in range(len(future_sets)):
            s_i = future_sets[i]
            copy = future_sets[:]
            copy.pop(i) # 删除第i个节点，认为已经完成对其的访问
            distance.append(self.X[node][s_i] + self.solve(s_i,copy))
        # 动态规划递推方程，利用递归
        d = min(distance)
        # node需要连接的下一个节点
```

推荐阅读

iOS排列组合算法
阅读 42

Leetcode笔记1
阅读 98

数组中只出现一次的数字
阅读 279

减小和重新排列数组后的最大元素
阅读 125

一起学算法-35. 搜索插入位置
阅读 368

网络投稿平台

```
        return d

# 计算两点间的欧式距离
def distance(vector1,vector2):
    d=0;
    for a,b in zip(vector1,vector2):
        d+=(a-b)**2;
    return d**0.5;
```

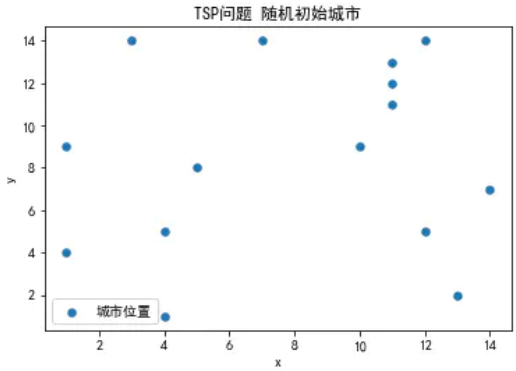
首先在（10，10）的坐标上，随机生成10个城市：

```
# 随机生成10个坐标点
n = 10
random_list = list(itertools.product(range(1, n), range(1, n)))
cities = random.sample(random_list, n)

x = []
y = []
for city in cities:
    x.append(city[0])
    y.append(city[1])

fig = plt.figure()
plt.scatter(x,y,label='城市位置',s=30)
plt.xlabel('x')
plt.ylabel('y')
plt.title('TSP问题 随机初始城市')

plt.legend()
plt.show()
```



使用欧氏距离计算两两城市之间的距离：

```
distence_matrix = np.zeros([n,n])
for i in range(0, n):
    for j in range(n):
        distence = distance(cities[i],cities[j])
        distence_matrix[i][j] = distence
```

使用动态规划求解TSP问题：

```
S = Solution(distence_matrix,0)
print("最短距离: " + str(S.tsp()))
# 开始回溯
M = S.array
lists = list(range(len(S.X)))
start = S.start_node
city_order = []
while len(lists) > 0:
    lists.pop(lists.index(start))
    m = S.transfer(lists)
```

推荐阅读

iOS排列组合算法
阅读 42

Leetcode笔记1
阅读 98

数组中只出现一次的数字
阅读 279

减小和重新排列数组后的最大元素
阅读 125

一起学算法-35. 搜索插入位置
阅读 368

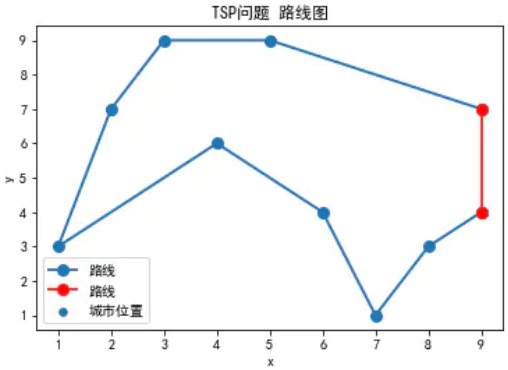
网络投稿平台

得到最终的访问路线：

```
x1 = []
y1 = []
for city in city_order:
    x1.append(city[0])
    y1.append(city[1])

x2 = []
y2 = []
x2.append(city_order[-1][0])
x2.append(city_order[0][0])
y2.append(city_order[-1][1])
y2.append(city_order[0][1])

plt.plot(x1,y1,label='路线',linewidth=2,marker='o',markersize=8)
plt.plot(x2,y2,label='路线',linewidth=2,color='r',marker='o',markersize=8)
plt.xlabel('x')
plt.ylabel('y')
plt.title('TSP问题 路线图')
plt.legend()
plt.show()
```



0人点赞 >

Python

更多精彩内容，就在简书APP

简书

"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下

cathy1997 别的小朋友都会写程序了
我怎么还不会呀
总资产45 共写了1.6W字 获得53个赞 共53个粉丝

关注

推荐阅读

iOS排列组合算法
阅读 42

Leetcode笔记1
阅读 98

数组中只出现一次的数字
阅读 279

减小和重新排列数组后的最大元素
阅读 125

一起学算法-35. 搜索插入位置
阅读 368

网络投稿平台

推荐阅读

iOS排列组合算法
阅读 42

Leetcode笔记1
阅读 98

数组中只出现一次的数字
阅读 279

减小和重新排列数组后的最大元素
阅读 125

一起学算法-35. 搜索插入位置
阅读 368

网络投稿平台