模拟退火算法实现旅行商问题

模拟退火算法的步骤

1. 初始化参数：设置初始温度 T_S，终止温度 T_E，链长 L，及降温速率 $rating$ 。
2. 初始化路径：随机产生一个初始路径。
3. 变换：对当前个体 S_1 随机交换两个点的位置，得到新的个体 S_2。
4. Metropolis 准则：设置路径差：$dis = fit(S_2) - fit(S_1)$。如果 $dis < 0$，接受，否则以概率 e^{-dif/T_S} 接受新的路径。

   Metropolis准则公式：$P = \begin{cases} 1 & dif < 0 \\ exp(-dif/T) & dif >= 0 \end{cases}$
5. 降温：每次迭代，$T = rating * T$，直到 $T < T_E$。

代码和注释

- 引入相关的库，初始化参数

```
import copy
import math
from operator import ne
import random
from turtle import distance
import numpy as np
import matplotlib.pyplot as plt

T_start = 10000
T_end = 1e-3
L = 1000
rating = 0.9
cities = [
    (0.6606, 0.9500), (0.9695, 0.6470), (0.5906, 0.5029), (0.2124, 0.8274), (0.0398, 0.9697), (0.1367, 0.5979),
    (0.9536, 0.2184), (0.6091, 0.7148), (0.8767, 0.2395), (0.8148, 0.2867), (0.3876, 0.8200), (0.7041, 0.3296),
    (0.0213, 0.1649), (0.3429, 0.3025), (0.7471, 0.8192), (0.5449, 0.9392), (0.9464, 0.8191), (0.1247, 0.4351),
    (0.1636, 0.8646), (0.8668, 0.6768)
]
plt.rcParams['font.sans-serif'] = ['SimHei']  # 中文
plt.rcParams['axes.unicode_minus'] = False
```

- 设置计算每个点之间的函数

```
def count_dis():
    n = len(cities)
    res = [[0 for _ in range(n)] for _ in range(n)]
    for i in range(len(cities)):
        posi = cities[i]
        for j in range(i+1, len(cities)):
            posj = cities[j]
            res[i][j] = pow((pow(posi[0]-posj[0], 2) + pow(posi[1] - posj[1], 2)), 0.5)
            res[j][i] = res[i][j]  # 对称
    return res
```

- 设置计算所有点的距离总和

```
def count_dis_all(path, dis):
    all_dis = 0.0000
    for i in range(len(path)):
        if i == len(path)-1:
            all_dis += dis[path[i]][path[0]]
        else:
            all_dis += dis[path[i]][path[i+1]]
    return all_dis
```

- 计算下一代，即步骤3

```
def cnt_new_path(path):
    new_path = copy.copy(path)
    idx1 = random.randint(0, len(path) - 1)  # 产生两个索引
    idx2 = random.randint(0, len(path) - 1)  # 产生两个索引
    new_path[idx1], new_path[idx2] = new_path[idx2], new_path[idx1]  # 交换
    return new_path
```

- 实现 Metropolis 准则

```
def metropolis(old_path, new_path, dis, t):
    delta = count_dis_all(new_path, dis)-count_dis_all(old_path, dis)
    if delta < 0:  # 低于旧路径
        return copy.copy(new_path), count_dis_all(new_path, dis)
    if math.exp(-delta/t) >= random.uniform(0, 1):  # 高于旧路径，按exp(-delta/t)
        return copy.copy(new_path), count_dis_all(new_path, dis)
    return copy.copy(old_path), count_dis_all(old_path, dis)
```

- 绘制结果图像和迭代距离图像

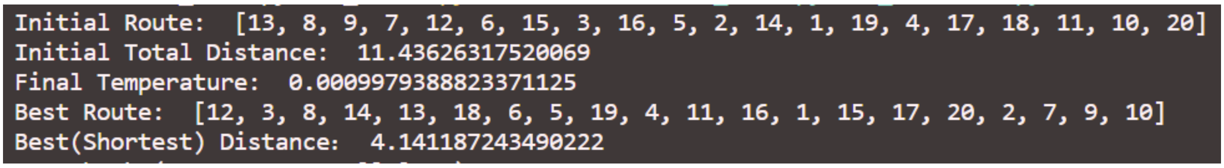
```
def draw_evolution(evolution):
    plt.clf()
    plt.plot([i for i in range(len(evolution))], evolution)
    plt.savefig('TSP_SAA_Dist.png', dpi=800)


def draw_res(best, file_name):
    x, y = [pos[0] for pos in cities], [pos[1] for pos in cities]
    plt.clf()  # 清空画布

    for i in range(len(cities)):
        st = cities[best[i]]
        e = cities[best[i + 1]] if i < len(best) - 1 else cities[best[0]]
        plt.arrow(st[0], st[1], e[0] - st[0], e[1] - st[1])

    for i in range(len(cities)):
        plt.text(x[best[i]], y[best[i]], "{}".format(
            (best[i] + 1)), size=15, color="r")
    plt.xlabel("x轴")
    plt.ylabel("y轴")
    plt.savefig(file_name+'.png', dpi=800)
```

- 主函数

```
def main():
    dist = count_dis()  # 距离矩阵
    path = random.sample(range(0, len(cities)), len(cities))  # 初始化路径
    draw_res(path, 'init_path')

    all_dist = count_dis_all(path, dist)  # 计算总路程
    print("Initial Route: ", [p+1 for p in path])
    print("Initial Total Distance: ", all_dist)

    t = T_start
    each_dist = []

    while t > T_end:
        for _ in range(L):
            new_path = cnt_new_path(path)  # 新路径
            path, all_dist = metropolis(path, new_path, dist, t)
            each_dist.append(all_dist)
        t *= rating

    # 打印退火后信息
    print("Final Temperature: ", t)
    print("Best Route: ", [p + 1 for p in path])
    print("Best(Shortest) Distance: ", all_dist)

    draw_res(path, "TSP_SAA_BEST")
    draw_evolution(each_dist)
```

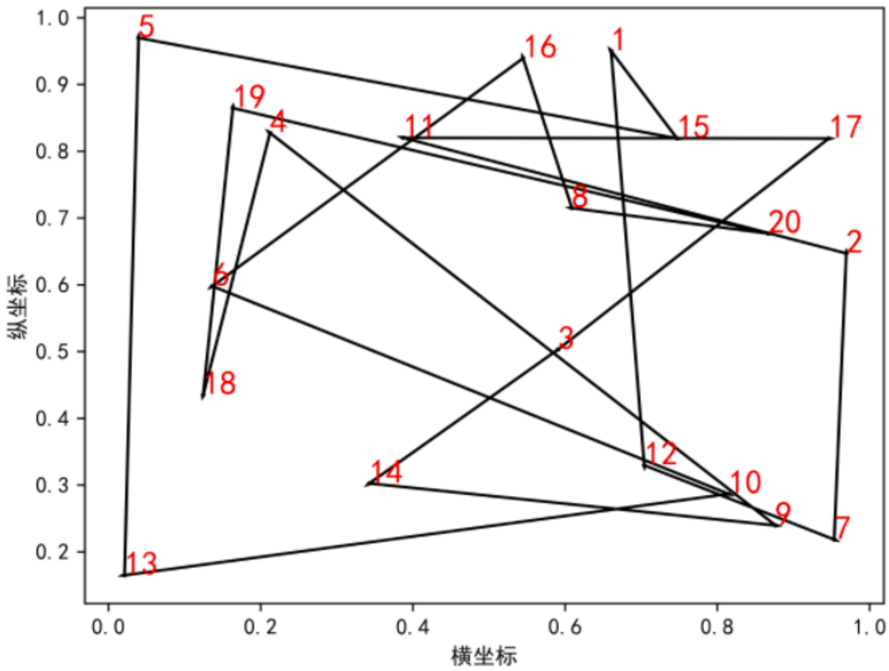
运行截图

- 初始路径和结果路径

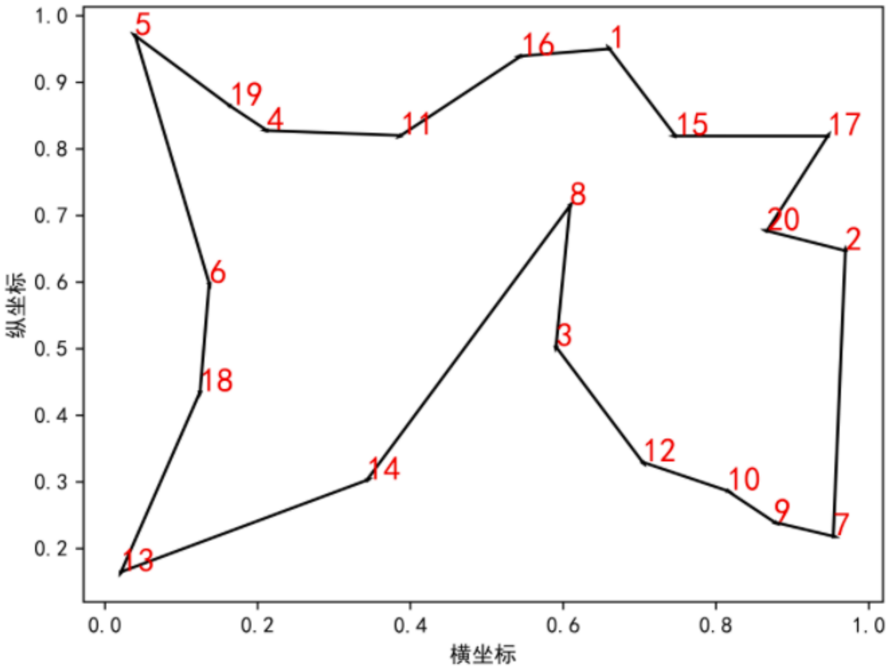


```
Initial Route:  [13, 8, 9, 7, 12, 6, 15, 3, 16, 5, 2, 14, 1, 19, 4, 17, 18, 11, 10, 20]
Initial Total Distance:  11.43626317520069
Final Temperature:  0.0009979388823371125
Best Route:  [12, 3, 8, 14, 13, 18, 6, 5, 19, 4, 11, 16, 1, 15, 17, 20, 2, 7, 9, 10]
Best(Shortest) Distance:  4.141187243490222
```

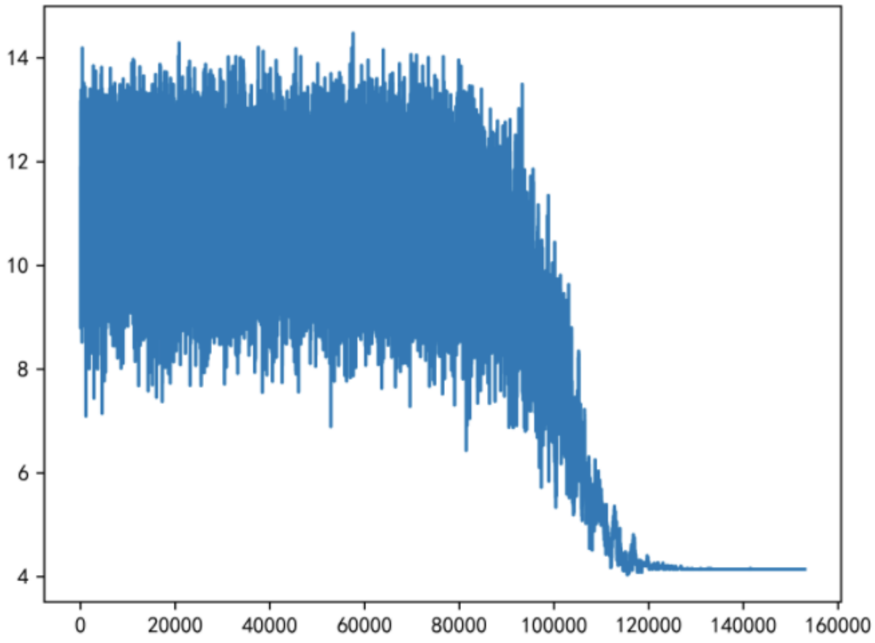
- 初始化路径的路线图



- 最佳路线图



- 迭代的路程图



14
12
10
8
6
4
0
20000
40000
60000
80000
100000
120000
140000
160000