实验二：用SA算法求解旅行商问题

1. 问题描述：

旅行商问题（Travelling Salesman Problem, 简记TSP，亦称货郎担问题)：设有N个城市和距离矩阵D=[dij]，其中dij表示城市i到城市j的距离，i，j=1，2 … n，则问题是要找出遍访每个城市恰好一次的一条回路并使其路径长度为最短。

1. 问题分析

1、存储结构

在TSP问题中每个城市都是连通的，N个城市之间均有道路，但距离不等，因此存储的数据结构我们可以用一个N\*N的矩阵D来表示各个城市之间的距离，其中，这个矩阵是一个对角线矩阵，因为D[i][j]=D[j][i]，而且对角线上的值都为0。然后每个解用一个长度为N的一维数组存储N个数的排列。

2、算法分析

本实验中使用模拟退火算法，模拟退火其实也是一种贪心算法，但是它的搜索过程引入了随机因素。模拟退火算法以一定的概率来接受一个比当前解要差的解，因此有可能会跳出这个局部的最优解，达到全局的最优解。

在求取最小值解f(x)问题中，模拟退火算法描述为

若f(i+1) <= f(i) (即移动后得到更优解)，则总是接受该移动

若f(i+1) > f(i) (即移动后的解 比当前解要差)，则以一定的概率接受移动，而且这个概率随着时间推移逐渐降低（逐渐降低才能趋向稳定）这里的概率P为

P = exp(f(i)-f(i+1)/(t))

3、算法参数

模拟退火算法包含以下几个参数，在此实验中分别取为下列值：

1. 控制参数 t 的初值 t0 -> 100
2. 控制参数 t 的 衰减函数 -> t = α \* t (α=0.99)
3. 控制参数 t 的终值 tf -> 1
4. Mapkob 链长 Lk -> 20000
5. 停止准则：连续2个Mapkob链中对路径无任何变动（优化或恶化的）时即停止算法运行。
6. 新解的产生：采用2变换法。任意产生两个随机数u和v (u<v且u !=v)，将u和v及其之间的顺序逆转。即由（π1 … πu-1πuπu+1 … πv-1πvπv+1 … πn）变为（π1 … πu-1πvπv-1 … πu-1πuπv+1 … πn）。
7. 编程环境

系统：Windows10

IDE：Jupyter notebook

Language：Python

1. 实验数据

本实验以中国的主要的28个城市为数据，在city\_location.csv文件中，存储了每个城市的经度和纬度（精确到一位小数），具体如图1所示。



图1 城市坐标信息

1. 实验结果

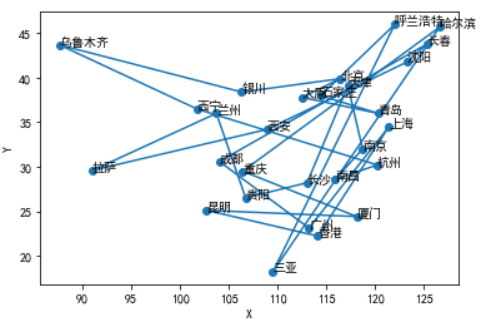


图2 最初随机初始化路径图

如图2所示，程序最开始随机初始化一个序列，此时城市之间的总距离为415.01。

程序运行完成后，系统输出的解为：

Array:[1, 12, 5, 13, 11, 8, 4, 3, 26, 10, 6, 16, 19, 17, 18, 21, 9,22, 20, 15, 27, 2, 0, 23, 24, 25, 14, 7]

Distance: 143.36626296229673

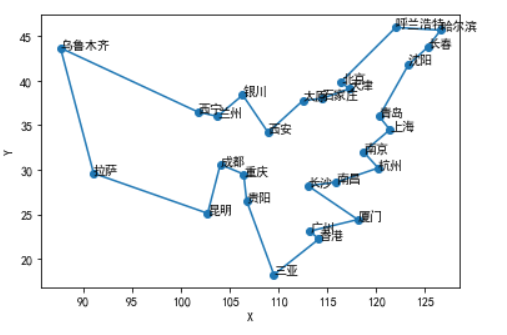


图3 城市路径最优解

如图3所示。最后找到了一条路径是上海->南京->杭州->南昌->长沙->厦门->广州->香港->三亚->贵阳->重庆->成都->昆明->拉萨->乌鲁木齐->西宁->兰州->银川->西安->太原->石家庄->天津->北京->呼兰浩特->哈尔滨->长春->沈阳->青岛->上海，其中总距离为143.37。

1. 实验小结

模拟退火算法由于设置了一个接受不好的解的概率，所以在优化过程中有可能会跳出局部最优解，但有一个问题是在求解过程中，解不是一直下降的，如图4所示。最开始温度t可能还很大，所以有更大的概率接受不好的值，而且接受值的范围也大，体现在波动的幅度很大并且频繁。后来随着温度t降低，波动幅度逐渐减小，此时接受不好值的概率大大降低。程序也逐渐找到最优解。

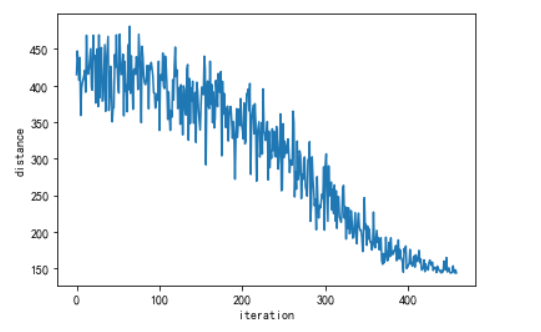


图4 距离曲线图