**实验报告**

――――遗传算法

姓名： 黄灿铭 学号：16340079 日期：2019/1/12

**摘要：**本实验将使用模拟退火试图求出一个大于100个城市数的TSP问题的最优解，从而掌握模拟退火的使用，总结出设计高效模拟退火的经验。

**1．导言**

TSP问题（Traveling Salesman Problem，旅行商问题），由威廉哈密顿爵士和英国数学家克克曼T.P.Kirkman于19世纪初提出。问题描述如下：

有若干个城市，任何两个城市之间的距离都是确定的，现要求一旅行商从某城市出发必须经过每一个城市且只在一个城市逗留一次，最后回到出发的城市，问如何事先确定一条最短的线路已保证其旅行的费用最少？

由于该问题的可行解是所有顶点的全排列，随着顶点数的增加，会产生组合爆炸，它是一个NP完全问题。由于其在交通运输、电路板线路设计以及物流配送等领域内有着广泛的应用，国内外学者对其进行了大量的研究。早期的研究者使用精确算法求解该问题，常用的方法包括：分枝定界法、线性规划法、动态规划法等。但是，随着问题规模的增大，精确算法将变得无能为力，因此，在后来的研究中，国内外学者重点使用近似算法或启发式算法，主要有遗传算法、模拟退火法、蚁群算法、禁忌搜索算法、贪婪算法和神经网络等。

本实验使用局部搜索和模拟退火来求得TSP问题的最优解

0. 局部搜索简介（LS）

局部搜索法 局部搜索是解决最优化问题的一种启发式算法。因为对于很多复杂的问题， 求解最优解的时间可能是极其长的。因此诞生了各种启发式算法来退而求其次寻 找次优解或近似最优解，局部搜索就是其中一种。它是一种近似算法 （Approximate algorithms）。

局部搜索算法是从爬山法改进而来的。简单来说，局部搜索算法是一种简单 的贪心搜索算法，该算法每次从当前解的邻域解空间中选择一个最好邻居作为下 次迭代的当前解，直到达到一个局部最优解(local optimal solution)。局部搜索从 一个初始解出发，然后搜索解的邻域，如有更优的解则移动至该解并继续执行搜 索，否则就停止算法获得局部最优解

1. 模拟退火简介（SA）

     在求解实际问题，我们可以采用搜索算法，比如爬山搜索等系列算法。但这些算法都是局部优化算法，在某些实际问题中还是有很多缺点。局部搜索算法(以爬山算法为代表)的缺点：仅适用于某类组合优化问题；所得到的近似解质量通常较差；时间复杂度高，且最坏情况下的时间复杂度未知；最致命的是无法跳离局部最优的“陷阱”。

    人们开始超越数学思维，从自然物理过程中寻找灵感。1982年,Kirkpatrick意识到固体退火算法与组合优化问题之间的类似性Metropolis等对孤立在恒定温度下达到热平衡的过程的模拟的启迪：把Metropolis准则引入优化过程中模拟退火算法（Simulated Annealing Algorithm，简称SAA），源于对固体退火过程的模拟，采用Metropolis接受准则，并用一组称为冷却表的参数控制算法进程，使算法在多项式时间里给出一个近似最优解。

    爬山搜索为代表的局部搜索算法都是仅适用于某类组合优化问题，所得到的近似解的质量通常较差。这类方法最致命的缺点是无法跳离局部最优的“陷阱”，最终停留在某个局部最优解上。为了克服这些弱点，人们开始超脱纯数学思维，到一些自然物理过程中寻找灵感。模拟退火算法就是一个成功的典范，其思想比方法本身更为重要。

2. 模拟退火核心思想

    模拟退火算法在处理全局优化、离散变量优化等困难问题中，具有传统优化算法无可比拟的优势。模拟退火算法的思想最早由Metorpolis等提出的。其出发点是基于物理中固体物质的退火过程与一般的组合优化问题之间的相似性。模拟退火法是一种通用的优化算法，其物理退火过程由以下三部分组成：

    1）加温过程：其目的是增强粒子的热运动，使其偏离平衡位置。当温度足够高时，固体将熔为液体，从而消除系统原先存在的非均匀状态。

    2）等温过程：对于与周围环境交换热量而温度不变的封闭系统，系统状态的自发变化总是朝自由能减少的方向进行，当自由能达到最小时，系统达到平衡状态。

    3）冷却过程：使粒子热运动减弱，系统能量下降，得到晶体结构。其中，加温过程对应算法的设定初温，等温过程对应算法的Metropolis抽样过程，冷却过程对应控制参数的下降。这里能量的变化就是目标函数，我们要得到的最优解就是能量最低态。其中Metropolis准则是SA算法收敛于全局最优解的关键所在，Metropoli、准则以一定的概率接受恶化解，这样就使算法跳离局部最优的陷阱。

3. 遗传算法总体设计

（1）选S\_0作为初始状态，令S (0)=S\_0，同时设初始温度T，令i=0。

（2）令 T=T\_i，以T和S\_i调用Metorpolis抽样算法，返回状态S作为本算法的当前解，S\_i=S\_0。

（3）按照一定方式降温，即T =T\_(i+1)，其中T\_(i+1)<T\_i，i=i+1。

（4）检查终止条件，如果满足则转步骤 (5),否则转步骤(2)

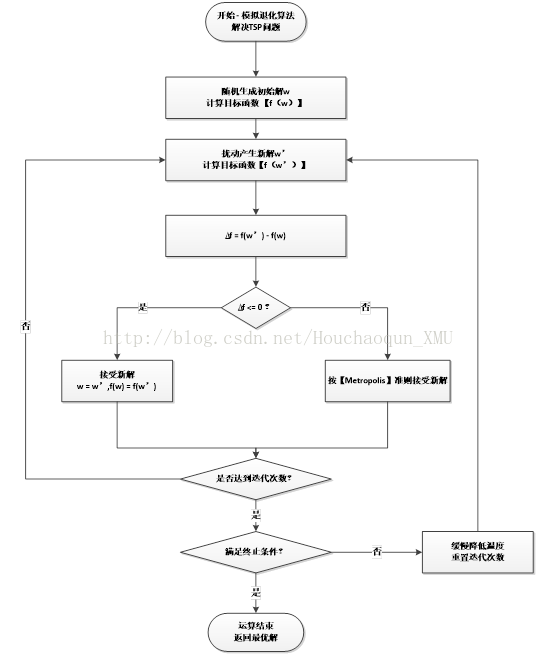
（5）当前解S\_i为最优解，输出结果，停止。Metropolis抽样算法描述如下：

* 令k=0时，当前解 S (0)=S\_0，在温度T下，进行以下各步操作。
* 按某个规定的方式根据当前解 S（k）所处的状态S产生一个近邻子集N (S（k))+S，从N(S (k))中随机得到一个新状态 S' 作为下一个候选解，计算能量之差△C'= C（S'） - C(S(k)）。
* 如果△C' <  0 ，则接受 S' 作为下一个当前解，否则，以概率exp(一△C'/ T)接受 S' 作为下一个当前解。若 S' 被接受，则令S (k十1) = S' ，否则S(k+1)=S（k）。
* k =k +1，检查算法是否满足终止条件，若满足，则转步骤(5)，否则转步骤(2)。
* 返回S(k)，结束。

4. Metropolis算法解释

模拟退火算法用Metropolis算法产生组合优化问题解的序列。并由Metropolis准则对应的转移概率P：

确定是否接受从当前解i 到新解j 的转移。式中t ∈R+ 表示控制参数。开始让t 取较大的值，在进行足够多的转移后，缓慢减小t 的值（初始温度乘以退火系数，如 0.98 等），如此重复直至满足某个停止准则时算法终止。模拟退火算法依据Metropolis准则接受新解，除接受优化解外，还在一个限定范围内接受恶化解。开始时t值较大，可能接受较差的恶化解，随着t值的减小，只能接受较好的恶化解；当t值趋于零值时，就不再接受任何恶化解。这就使得算法可以跳出局部最优陷阱。在算法执行期间，随着控制参数t值的减小，算法返回某个整体最优解得概率单调增大，返回某个非最优解的概率单调减小。



**2．实验过程**

初始化参数

def initpara():

alpha = 0.99

t = (1,200)

markovlen = 1000

return alpha,t,markovlen

使用随机序列作为初始城市序列

solutionnew = np.arange(num)

计算距离的辅助矩阵

#得到距离矩阵的函数

def getdistmat(coordinates):

num = coordinates.shape[0] #194个坐标点

distmat = np.zeros((194,194)) #194X194距离矩阵

for i in range(num):

for j in range(i,num):

distmat[i][j] = distmat[j][i]=np.linalg.norm(coordinates[i]-coordinates[j])

return distmat

邻域动作（三交换和两交换）

#下面的两交换和三角换是两种扰动方式，用于产生新解

if np.random.rand() > 0.5:# 交换路径中的这2个节点的顺序

# np.random.rand()产生[0, 1)区间的均匀随机数

while True:#产生两个不同的随机数

loc1 = np.int(np.ceil(np.random.rand()\*(num-1)))

loc2 = np.int(np.ceil(np.random.rand()\*(num-1)))

## print(loc1,loc2)

if loc1 != loc2:

break

solutionnew[loc1],solutionnew[loc2] = solutionnew[loc2],

solutionnew[loc1]

else: #三交换

while True:

loc1 = np.int(np.ceil(np.random.rand()\*(num-1)))

loc2 = np.int(np.ceil(np.random.rand()\*(num-1)))

loc3 = np.int(np.ceil(np.random.rand()\*(num-1)))

if((loc1 != loc2)&(loc2 != loc3)&(loc1 != loc3)):

break

# 下面的三个判断语句使得loc1<loc2<loc3

if loc1 > loc2:

loc1,loc2 = loc2,loc1

if loc2 > loc3:

loc2,loc3 = loc3,loc2

if loc1 > loc2:

loc1,loc2 = loc2,loc1

#下面的三行代码将[loc1,loc2)区间的数据插入到loc3之后

tmplist = solutionnew[loc1:loc2].copy()

solutionnew[loc1:loc3-loc2+1+loc1] = solutionnew[loc2:loc3+1].copy()

solutionnew[loc3-loc2+1+loc1:loc3+1] = tmplist.copy()

是否接受新解

valuenew = 0

for i in range(num-1):

valuenew += distmat[solutionnew[i]][solutionnew[i+1]]

valuenew += distmat[solutionnew[0]][solutionnew[193]]

# print (valuenew)

if valuenew<valuecurrent: #接受该解

#更新solutioncurrent 和solutionbest

valuecurrent = valuenew

solutioncurrent = solutionnew.copy()

if valuenew < valuebest:

valuebest = valuenew

solutionbest = solutionnew.copy()

else:#按一定的概率接受该解

if np.random.rand() < np.exp(-(valuenew-valuecurrent)/t):

valuecurrent = valuenew

solutioncurrent = solutionnew.copy()

else:

solutionnew = solutioncurrent.copy()

内外循环：

while t > t2[0]:

for i in np.arange(markovlen):

**3．结果分析**

实验环境：python2.7

参数设置：

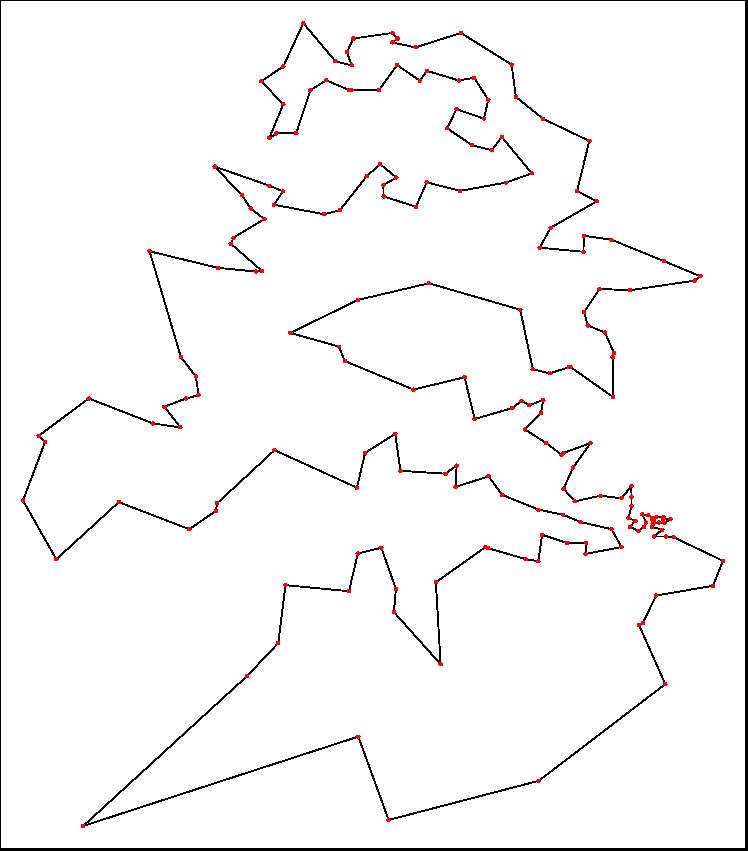
内循环迭代次次数：1000

初温设置：200

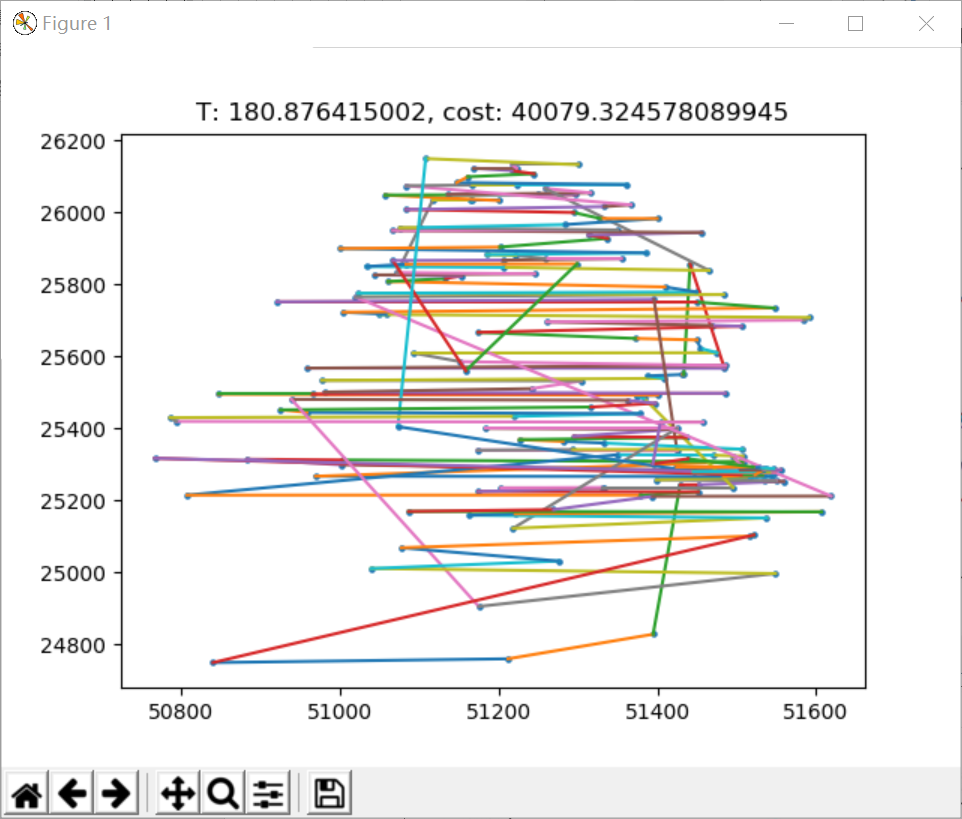
降温函数：t = 0.99 \* t

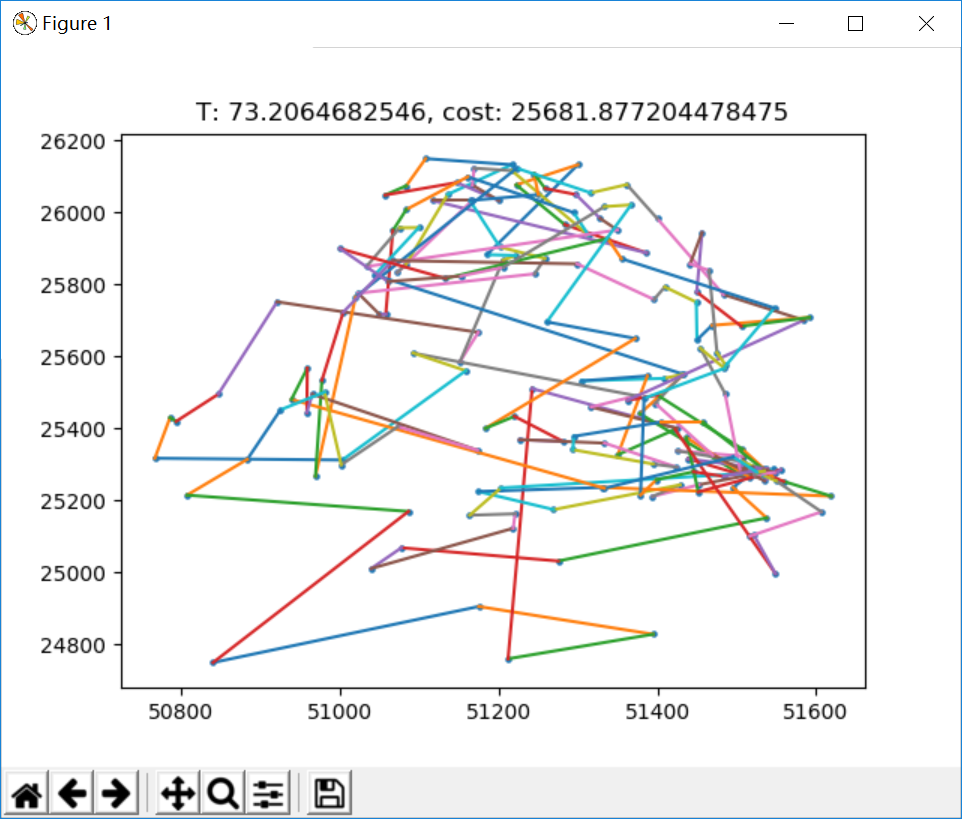
终温设置：1

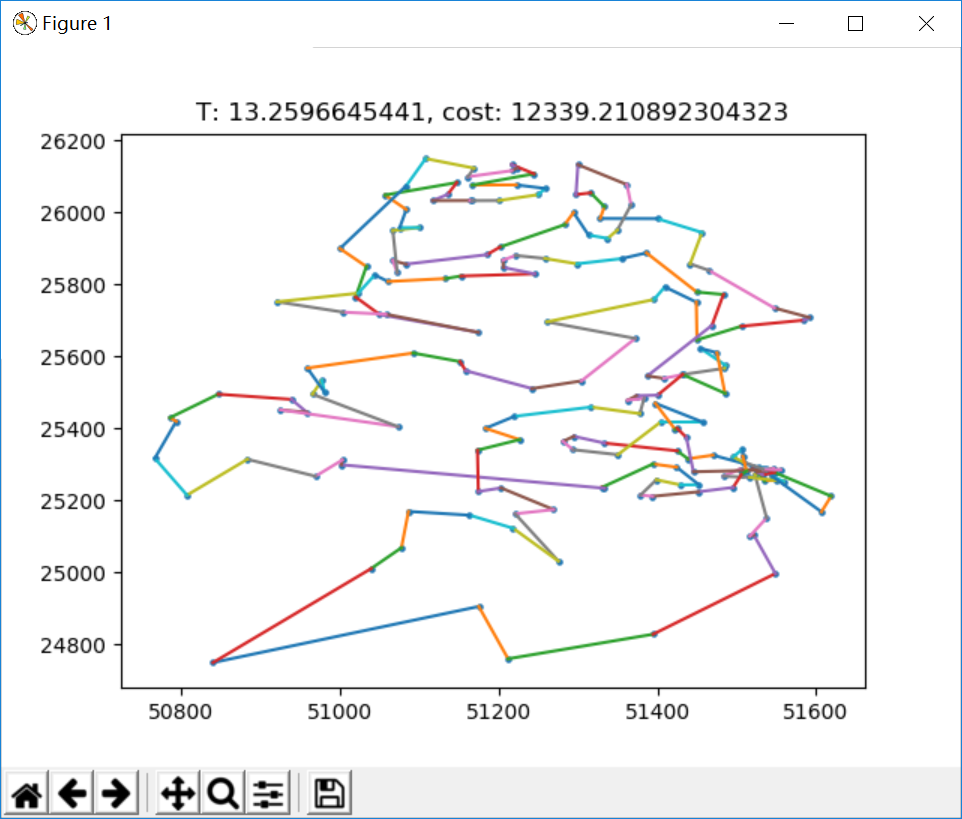
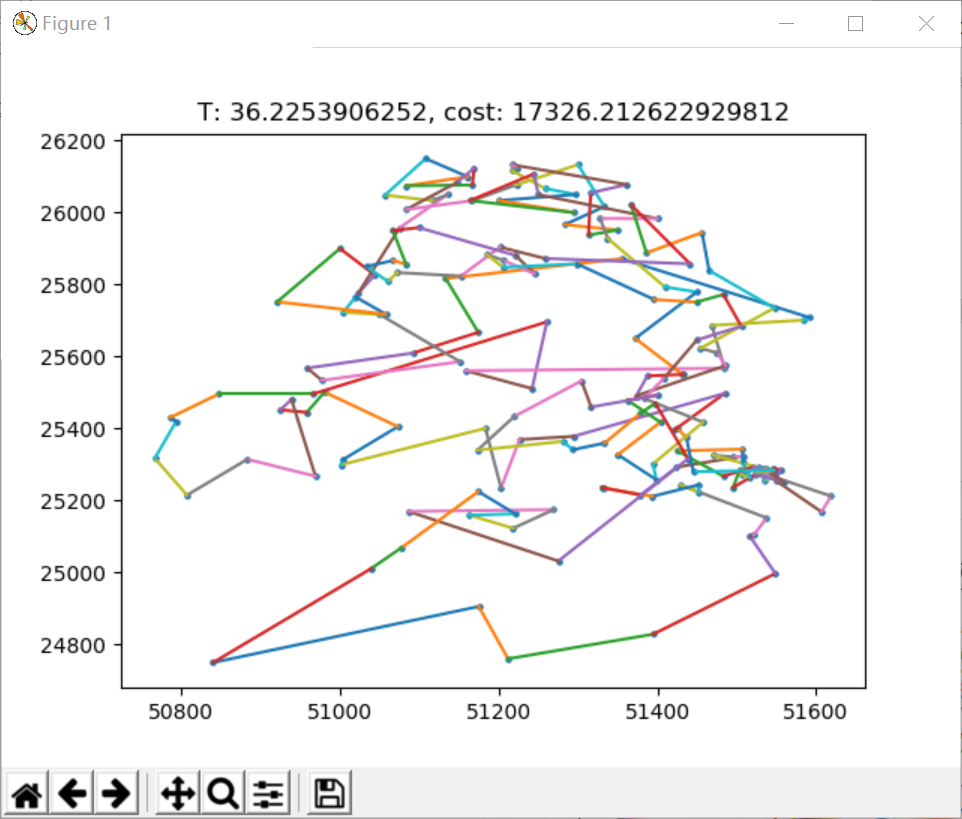
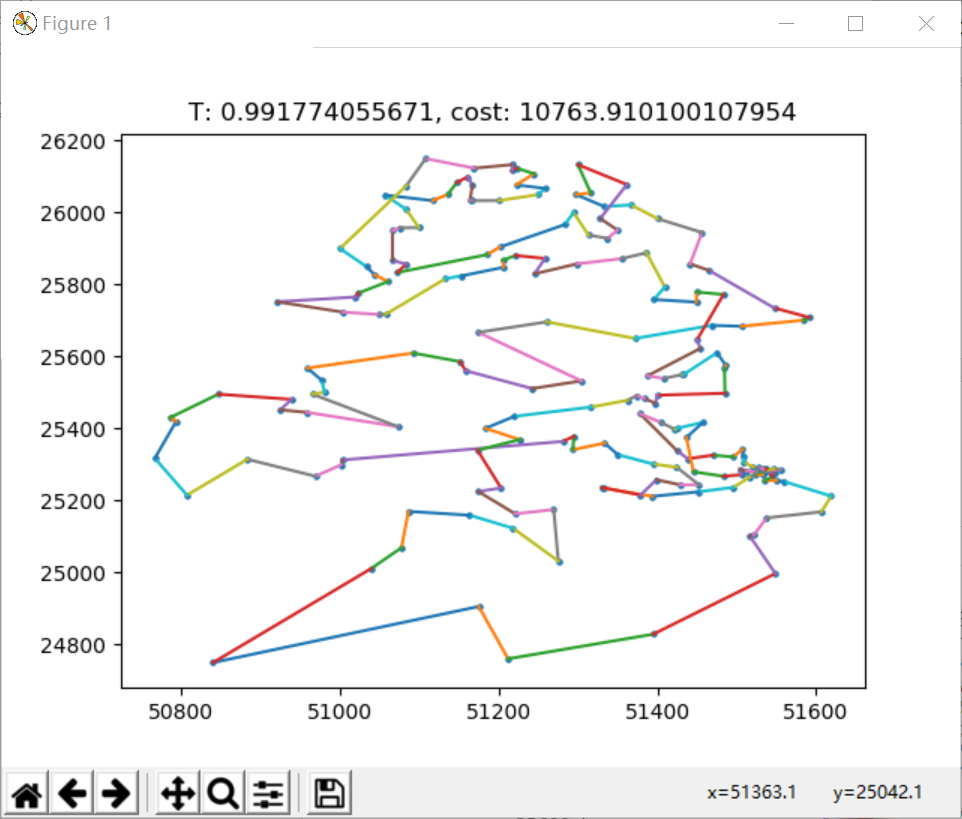
这是前人算出的最优解：9352



某次求解过程：





10次模拟退火结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10715 | 10763 | 10361 | 10756 | 10808 |
| 10227 | 10389 | 10673 | 10436 | 10439 |

局部搜索：

参数设置：

迭代次数：15000

邻域动作：两交换，三交换

10次局部搜索结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 40832 | 39213 | 39726 | 40339 | 39686 |
| 39935 | 41213 | 39718 | 39436 | 40391 |

局部搜索惨不忍睹，从最开始跳进一个局部最优解就出不来了，可能跟选择的邻域动作有关，两交换和三交换造成的扰动比较小。

模拟退火就好多了，基本上每次都能进到10%。

**4．结论**

模拟退火

优点：

    1）描述简单，使用灵活，运用广泛，运行效率高；

    2）需要较少的初始化条件约束，以概率 P(i) 收敛于全局最优；

    3）具有并行性；

缺点：

    1）收敛速度慢，执行时间长；

    2）参数依赖较大；

**主要参考文献**

冯剑, 岳琪. 模拟退火算法求解TSP问题[J]. 森林工程, 2008, 24(1):94-96.

王轩, 李元香. 一种结合局部搜索策略的求解 TSP 的演化算法[J]. 计算机工程, 2006, 32(9):16-18.

https://blog.csdn.net/Houchaoqun\_XMU/article/details/54578089