**任课教师：黄文清**

**《智能信息处理及应用》**

**（ 2020 -- 2021 学年 第一学期）**

**实**

**验**

**报**

**告**

**学号：2018329621262**

**2018329621277**

**姓名：刘恒玮**

**朱若晨**

**班级：智能科学与技术（2）班**

**实验：禁忌搜索算法解决TSP问题**

**一、实验目的**

1. 掌握禁忌搜索算法的基本原理和算法流程。
2. 能根据TSP问题的特点，给出合理的禁忌对象和禁忌长度、邻域搜索规则和藐视准则。
3. 复习python语言，能熟练使用python语言编写程序。

**二、实验环境与设备**

本次实验以小组形式进行。

实验环境：python语言。

实验设备：win10 Intel i7 2.21GHz。

**三、实验原理**

1）禁忌搜索算法的基本原理分析：

基本思想：采用邻域选优的搜索方法，为了逃离局部最优解，算法必须能够接受劣解，也就是每一次得到的解不一定优于原来的解。但是，一旦接受了劣解，算法迭代即可能陷入循环。为了避免循环，算法将最近接受的一些移动放在禁忌表中，在以后的迭代中加以禁止。即只有不再禁忌表中的较好解（可能比当前解差）才能接受作为下一代迭代的初始解。随着迭代的进行，禁忌表不断更新，经过一定的迭代次数后，最早进入禁忌表的移动就从禁忌表中解禁退出。

邻域：

对于组合优化问题，给定任意可行解x，x∈D，D是决策变量的定义域，对于D上的一个映射：N：x∈D→N(x)∈2(D) 其中2(D)表示D的所有子集组成的集合，N(x)成为x的一个邻域，y∈N(x)称为x的一个邻居。

候选集合：

候选集合一般由邻域中的邻居组成，可以将某解的所有邻居作为候选集合，也可以通过最优提取，也可以随机提取，例如某一问题的初始解是[1,2,3]，若通过两两交换法则生成候选集合，则可以是[1,3,2],[2,1,3],[3,2,1]中的一个或几个。

禁忌表：

禁忌表包括禁忌对象和禁忌长度。由于在每次对当前解的搜索中，需要避免一些重复的步骤，因此将某些元素放入禁忌表中，这些元素在下次搜索时将不会被考虑，这些被禁止搜索的元素就是禁忌对象；禁忌长度则是禁忌表所能接受的最多禁忌对象的数量，若设置的太多则可能会造成耗时较长或者算法停止，若太少则会造成重复搜索。

评价函数：

用来评价当前解的好坏，TSP问题中是总旅程距离。

特赦规则：

禁忌搜索算法中，迭代的某一步会出现候选集的某一个元素被禁止搜索，但是若解禁该元素，则会使评价函数有所改善，因此我们需要设置一个特赦规则，当满足该条件时该元素从禁忌表中跳出。

终止规则：

一般当两次迭代得到的局部最优解不再变化，或者两次最优解的评价函数差别不大，或者迭代n次之后停止迭代，通常选择第三种方法。

2）TSP问题的描述：

假设有一个旅行商人要拜访n个城市，他必须选择所要走的路径，路径的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市。路径的选择目标是要求得的路径路程为所有路径之中的最小值，用遗传算法，模拟染色体的遗传过程，进行求解。

**四、实验内容**

（1）采用贪婪算法生成初始解。从第一个城市出发找寻与其距离最短的城市并标记，然后继续找寻与第二个城市距离最短的城市并标记，直到所有城市被标记完。最后回到第一个城市(起点城市)  
def greedy\_initial\_route(remain\_cities):

    #采用贪婪算法生成初始解。从第一个城市出发找寻与其距离最短的城市并标记，

    #然后继续找寻与第二个城市距离最短的城市并标记，直到所有城市被标记完。

    #最后回到第一个城市(起点城市)

    cand\_cities = remain\_cities[:]

    current\_city = origin

    mile\_cost = 0

    initial\_route = []

    while len(cand\_cities) > 0:

        next\_city,distance = nearest\_city(current\_city,cand\_cities) #找寻最近的城市及其距离

        mile\_cost += distance

        initial\_route.append(next\_city) #将下一个城市添加到路径列表中

        current\_city = next\_city #更新当前城市

        cand\_cities.remove(next\_city) #更新未定序的城市

    mile\_cost += dis[initial\_route[-1]-1][0] #回到起点

    return initial\_route,mile\_cost  
（2）获取当前邻居城市中距离最短的1个  
def nearest\_city(current\_city,cand\_cities):#获取当前邻居城市中距离最短的1个

    temp\_min = float('inf')

    next\_city = None

    for i in range(len(cand\_cities)):

        distance = dis[current\_city-1][cand\_cities[i]-1]

        if distance < temp\_min:

            temp\_min = distance

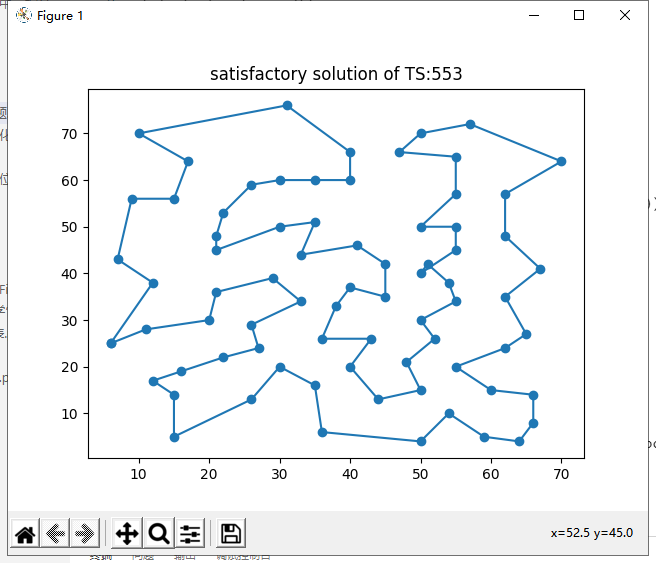
            next\_city = cand\_cities[i]

    return next\_city,temp\_min  
（3）对邻域规模、禁忌表、藐视法则等参数的设置在代码中体现。

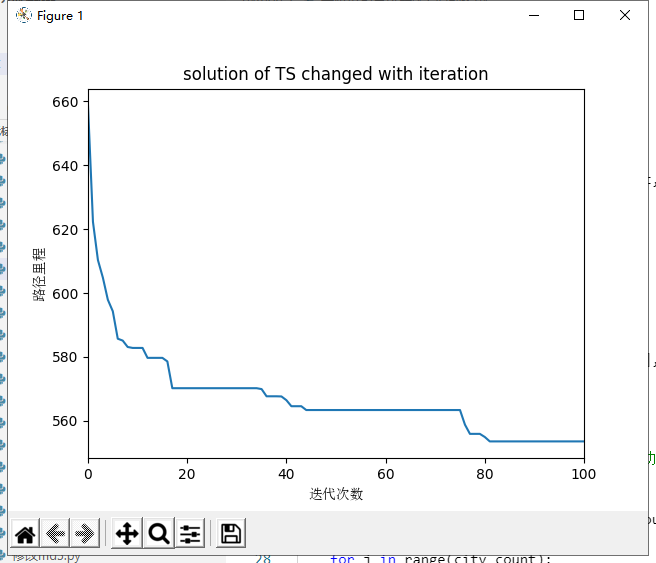
**五、实验结果**

实验参数设置：设置总的城市数city\_count，起点和终点城市origin，迭代过程中变动的城市remain\_cities，邻接矩阵dis，改良次数improve\_count。

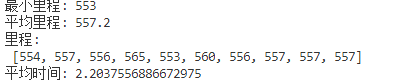
实验结果截图：



图一：迭代100次75个的最短路径图



图二 最优路程随迭代次数的变化图



图三 运行十次确定最优解

实验结论：禁忌搜索算法对比遗传算法和蚁群算法迭代次数少，程序运行时间短，由图三可以发现禁忌搜索算法每次都能得到较为理想的最优解，不会像蚁群算法和遗传算法一样容易陷入局部最优解。禁忌搜索算法和遗传算法对初始解的依赖都比较大，好的初始解都能使的算法收敛快；遗传算法收敛速度慢，局部搜索能力差，控制变量较多；蚁群算法对初始路线要求不高，蚁群算法的参数数目少，设置简单，易于蚁群算法应用到其它组合优化问题的求解，但是蚁群算法不适合大规模的TSP问题。

**六、实验总结**

通过本次实验，我理解了禁忌搜索算法的具体过程以及如何运用禁忌搜索算法去解决TSP问题。本次实验也不是一帆风顺的，实验过程中出现了一些问题，通过思考、查阅资料和上机实践，有些问题得以解决，有些问题还存有疑问。

**问题1**：每次运行结果得到的最优解会有所不同，如何统计多次运行的最优解，并计算平均最优解，和多次对比得到的最优解，并统计程序运行所需的时间?

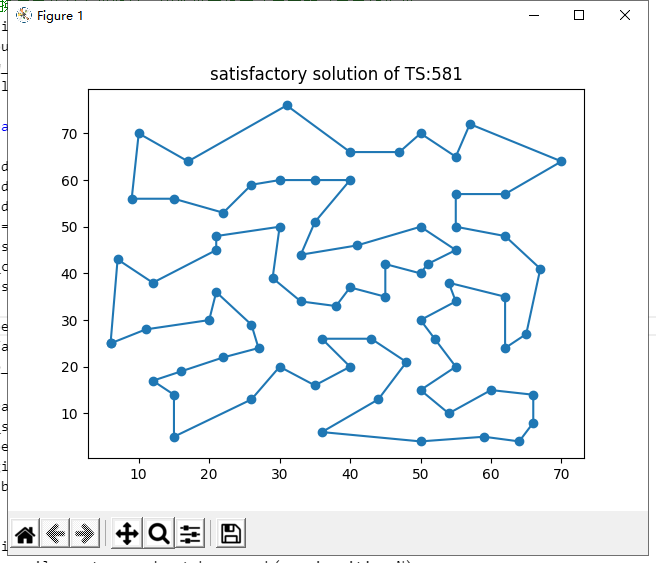
**解决方案**：在主函数中设置了一个记录平均里程的列表Mcost和记录平均时间的列表Tcost，分别用来记录多次运行得到的最优路径和程序运行所需要的时间，最后在Mcost取一个最小值，就可以得到多次运行的最优解了。

**问题2**：如何确定一个效果较为理想的终止规则?

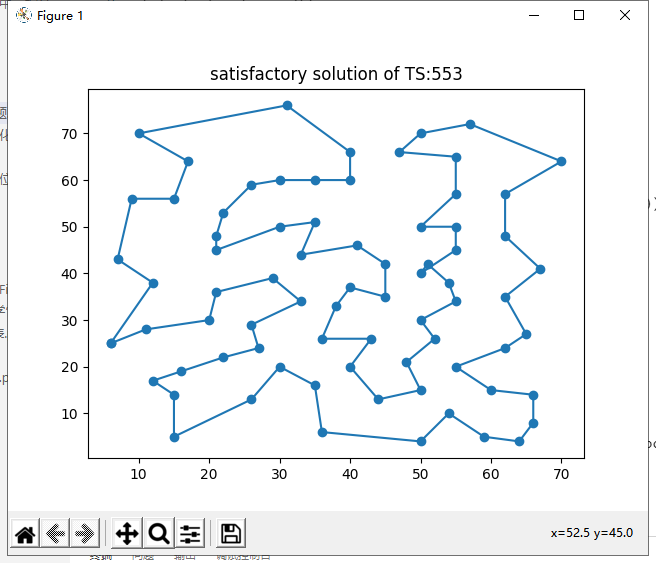
**解决方案**：一般当两次迭代得到的局部最优解不再变化，或者两次最优解的评价函数差别不大，或者迭代N次之后停止迭代，我们选用的是迭代100次后停止迭代。

**问题3：**如何解决对初始解依赖过强的问题?

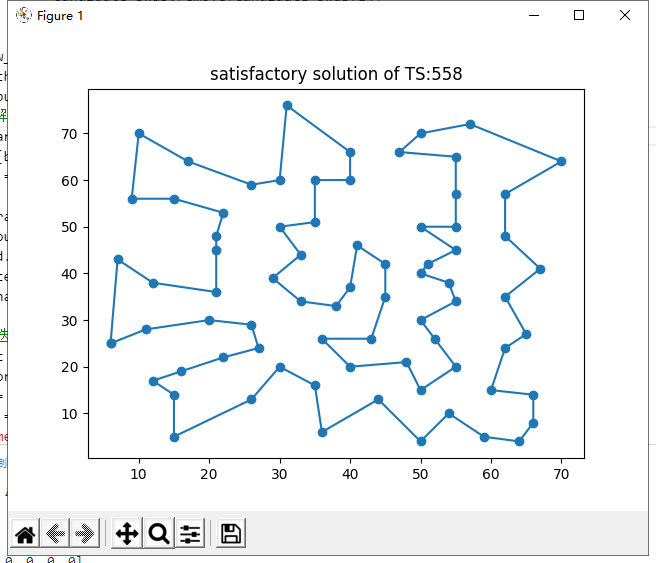
**解决方案**：采用贪婪算法生成初始解从第一个城市出发找寻与其距离最短的城市并标记，然后继续找寻与第二个城市距离最短的城市并标记，直到所有城市被标记完。 采取改良圈算法初始化路径，随机交换两个点（起点和终点除外）的位置，若交换后路径CR的里程小于交换前路径R的里程,则接受此次交换,令R = CR，否则重新交换，直到成功交换次数达到设置的改良次数。发现两种方法都能得到较为理想的结果。



图四：初始解随机的最优路径图



图五：采用贪婪算法生成初始解的最优路径图



图六：采用改良圈生成初始解的最优路径图

**参考资料：**

1. 《禁忌搜索算法解决TSP问题》实验报告指导书
2. 《智能信息处理技术原理与应用》
3. 《人工智能》（第3版）