华东师范大学计算机科学技术系上机实践报告

课程名称：人工智能	**年级**：2016 级	**上机实践成绩**：
指导教师：周爱民	**姓名**：汪春雨	**创新实践成绩**：
上机实践名称：旅行商问题求解	**学号**：10152150127	**上机实践日期**：2018/5/15
上机实践编号：No.4	**组号**：	**上机实践时间**：

一、 问题介绍

（本节介绍需要求解的问题是什么，为什么要采用我们介绍的方法求解）

1、旅行商问题描述

旅行商问题(TravelingSalesmanProblem，TSP)是一个经典的组合优化问题。经典的TSP可以描述为：一个商品推销员要去若干个城市推销商品，该推销员从一个城市出发，需要经过所有城市后，回到出发地。应如何选择行进路线，以使总的行程最短。

从图论的角度来看，该问题实质是在一个带权完全无向图中，找一个权值最小的Hamilton回路。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。

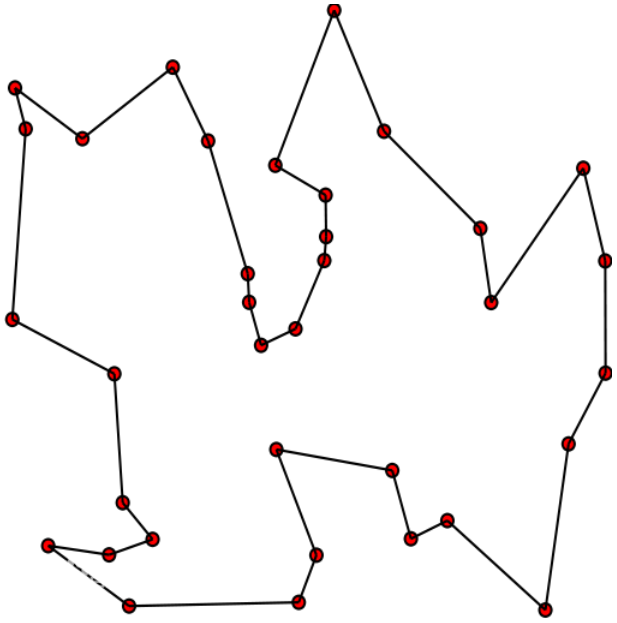

图1 旅行商问题解示例

2、本实验相关描述

输入：

从文件中国144个城市数据，其中cn144_location.txt第一列表示城市编号，第二列表示城市X坐标值，第三列表示城市Y坐标值，文件公144行；cn144_link表示城市之间连接关系，每一行表示一个允许的城市间连接，两列表示两个可以连接的城市编号。由于第二个文件出现些许问题，我们认为每两个城市之间均存在连接关系。

输出：

最短路径序列及路径长度。

城市散点图：

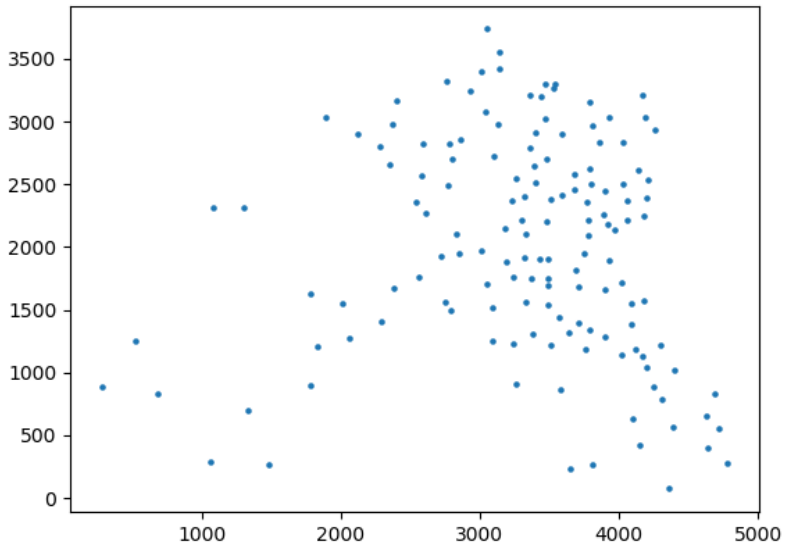

图2 原始数据散点图

二、 程序设计与算法分析

（本节介绍程序设计的分析过程，如数据结构定义，算法描述，算法框图等）

1、数据结构定义

1.1 染色体序列(基因序列)编码方式

使用城市序号列表标记染色体序列，如 [0,2,3,…143]

1.2 个体定义

即种群中的每个个体使用的数据结构（类）如下：

```
class Life(object):
    def __init__(self, chromosome,data):
        self.chromosome = [] #染色体序列
        self.adaptive_score = 0   #适应值
        self.update(chromosome, data)

    def update(self, chromosome, data):
        self.chromosome = chromosome
        self.adaptive_score = utils.get_adaptive_score(chromosome, data)
```

该类含有两个属性，分别是染色体序列和适应值，该对象实例化时其适应值即会算出赋值给该对象。适应值函数将在下面给出。

2、 策略选择

我们把种群规模（种群个体数）定义为N，基因长度定义为L，本实验中L=144。

2.1适应值函数

由于在遗传算法中我们追求适应值的最大化，而我们所求解的最优解的情况是距离和最小，所以我们把适应值函数定义为距离和的倒数：

$$F(\text{chromosome}) = \frac{1}{\sum \boldsymbol{dis}(\mathbf{chromosome[i],chromosome[i+1]})}$$

其中，分母为路径上的距离和。

函数定义如下：

```
import math
def distance(pos1, pos2):#求两点间距离
    #print(pos1, pos2)
    return math.sqrt((pos1[0]-pos2[0])**2 + (pos1[1]-pos2[1])**2)

def get_adaptive_score(sequence, data):#求适应值函数
```

```
    return 1/get_distance_all(sequence, data)


def get_distance_all(sequence, data):#路径上的距离和
    total = 0
    for i in range(len(sequence) - 1):
        total += distance(data[sequence[i]], data[sequence[i + 1]], )
    return total
```

2.2 个体被选择的概率计算：轮盘赌

$$p(x_i) = \frac{F(x_i)}{\sum_{j=1}^{N} F(x_j)}$$

其中，xi为每一个个体， i=1,2,3..,N, F(x)为适配值函数。

2.3 交配策略(交配概率为Pm)

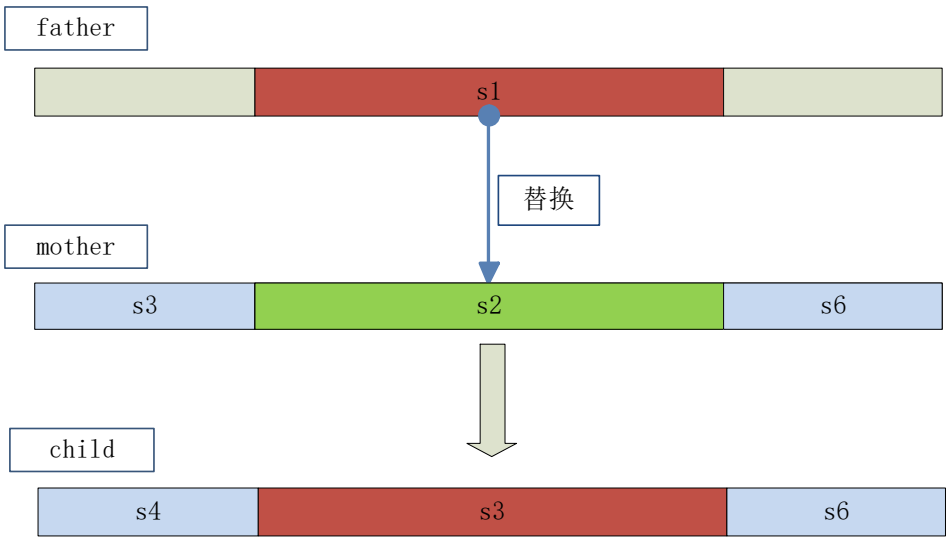

图3 交配示意图

用father的s1片段替换mother的s2片段得到子代的染色体序列，很容易想到在子代的s4, s3, s6中会有序列的重复，为了解决该问题，我们在mother的序列中去除有和s1中值相同的位置的值，

然后再把s1插入到mother的对应位置上生成子代染色体序列。

算法描述如下：

```
Input: father, mother (datastructure: Life)
a) index1 = random(0,N-1), index2 = random(index1,N-1)
b) chromosomePiece=father.chromosome[index1:index2],
   newchromosome=[], fatherIndex = 0
c) for g in mother.chromosome:
       i)    if g not in chromosomePiece: newchromosome.append(g),
                                          fatherIndex += 1
       ii)   if fatherIndex == index1:
                 newchromosome.extend(chromosomePiece)
                 fatherIndex += 1
d) return newchromosome
```

2.4 变异策略(变异概率为Pv)

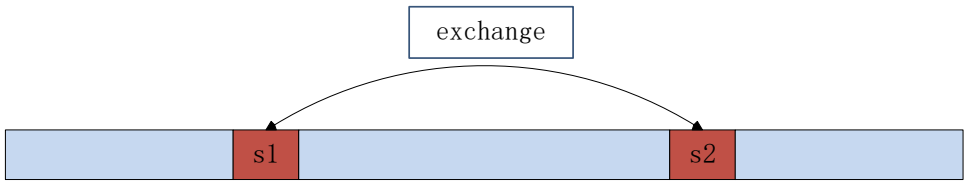

图4 变异示意图

a) 随机在序列中选择两个下标为s1，s2

b) 交换s1,s2

3、遗传算法

给定群体规模N，交配概率Pm，变异概率Pv，最大迭代次数n

1）生成具有N个个体的初始群体lives[]，下一代为newLives = []

2）是否迭代结束，结束则 goto 11

3）将群体中最优的个体（适配值最高）加入下一代 add(newLives[], best)

4）If len(newLives>=N) lives[] = newLives[], goto 3

5）采用轮盘赌算法从种群中选出一个个体作为father

6）按交配概率Pm决定father是否有交配权， 没有则 child=father, goto 8

7）采用轮盘赌算法从种群中选出一个个体作为mother

8）Child = mating(father, mother) 交配产生后代

9）以概率Pv产生突变，child=variation(child)

10）Add(newLifes, child), goto 4

11）输出best，最佳序列，以及距离和

三、 实验结果

（本节列出实验的结果，必要时加入一些自己的分析）

1、参数选择

1.1 由于交配概率和变异概率具有随机性，在我的测试下，选择为0.8， 0.1 较好

1.2 群体规模N的选择

mating_rate=0.8

variation_rate=0.1

N=10,15, 20,25,30，…200

n=100

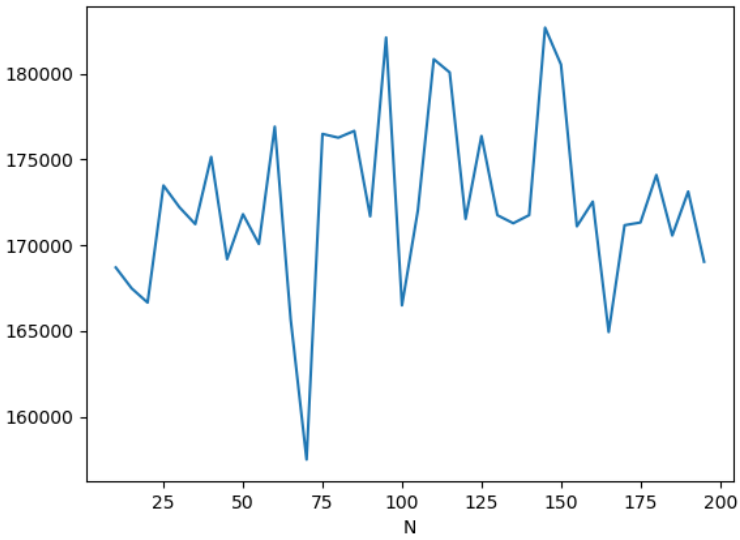

图 5 种群规模 N-距离和

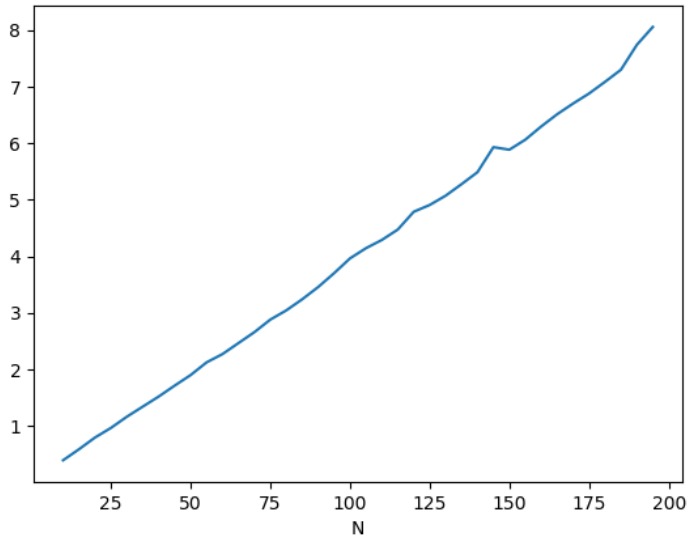

图 6 种群规模 N-时间（s）

由于随机性的存在，图五波动性较大，但是种群规模选择 N=25 是比较稳妥的。

1.3 迭代次数n的选择

mating_rate=0.8

variation_rate=0.1

N=10

n in [10,100,1000,10000,100000,200000]

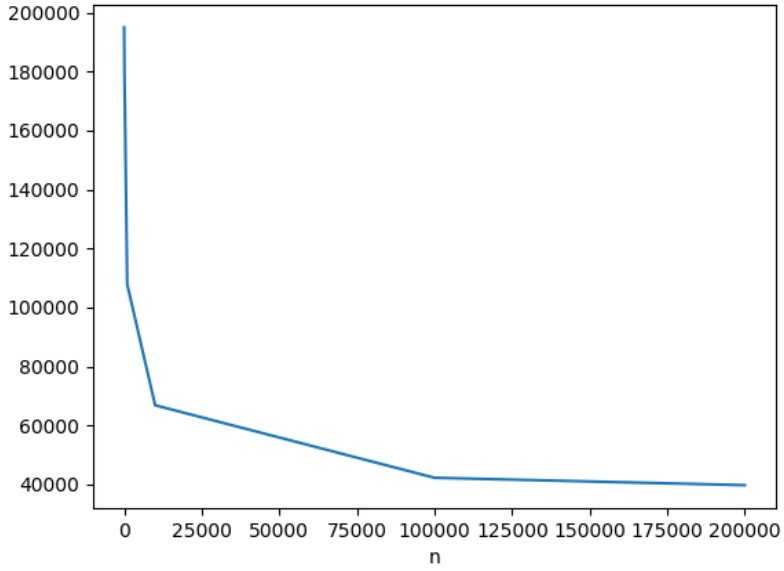

图7 迭代次数-距离和

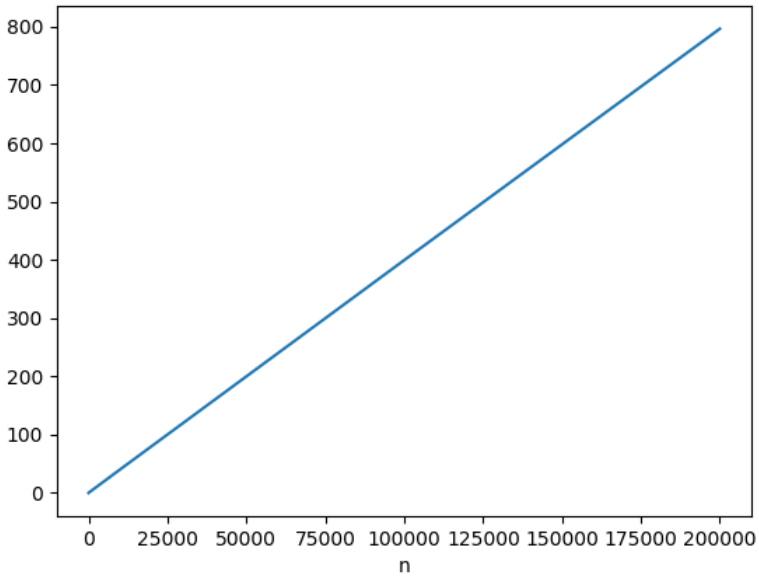

图8 迭代次数-时间（s）

2、最佳结果如下：

参数;

#群体规模N N=25

#交配概率 mating_rate = 0.8

#变异概率 variation_rate = 0.1

#迭代次数 n=200,000

结果：城市序列

[59, 55, 60, 56, 44, 47, 48, 45, 46, 77, 80, 81, 79, 78, 99, 104, 101, 84, 85, 87, 120, 121, 132, 133, 134, 130, 140, 141, 139, 136, 128, 138, 137, 123, 122, 119, 117, 124, 125, 143, 142, 116, 113, 118, 108, 107, 105, 98, 13, 7, 3, 2, 8, 26, 23, 30, 31, 36, 34, 35, 38, 33, 32, 28, 29, 27, 37, 17, 19, 20, 16, 15, 39, 40, 129, 126, 131, 127, 135, 111, 110, 109, 115, 114, 89, 90, 88, 76, 73, 75, 72, 1, 69, 68, 71, 67, 70, 62, 96, 100, 97, 12, 43, 18, 11, 9, 0, 4, 6, 24, 25, 21, 22, 66, 64, 63, 65, 61, 5, 95, 10, 14, 103, 106, 102, 112, 86, 82, 91, 92, 94, 93, 74, 83, 41, 42, 50, 49, 53, 51, 52, 57, 58, 54]

#距离和

37466.531758304554

#所用时间

Total time elapsed: 00:32:24

#路径图

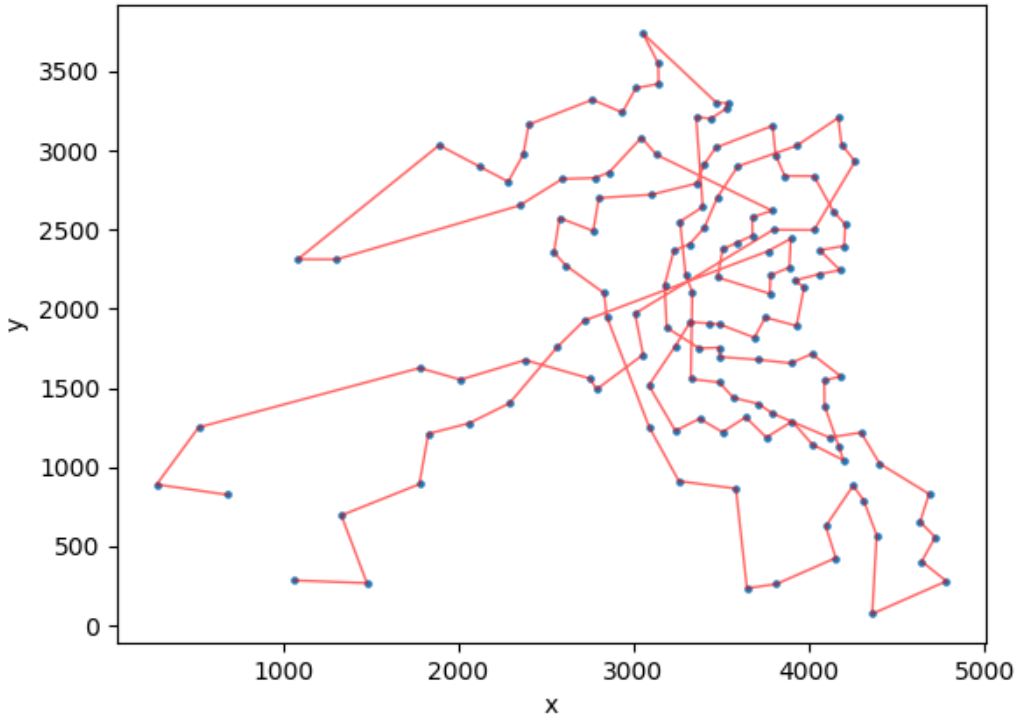

图9 路径图

3.结果分析与总结

3.1 首先迭代次数n越大得到的解越优，这是由遗传算法决定的，因为自然选择的代数越长，才能逐步进化出最优的品种基因，这就是所谓的‘物竞天择，适者生存’，由于计算机的性能有限，当前最大的n=200,000

3.2经过以上的测试，我们发现最优的种群规模N大概在100左右

3.3交配概率（0.7~0.9），变异概率（0~0.2）这样的参数选择是符合自然规律的。

3.4 求解时间与迭代次数成正比，距离和与迭代次数成指数下降趋势，最终收敛

3.5 求解时间与种群规模成正比

3.6 变异的作用：为了避免求解过程陷入局部最优解，给出向外走的机会

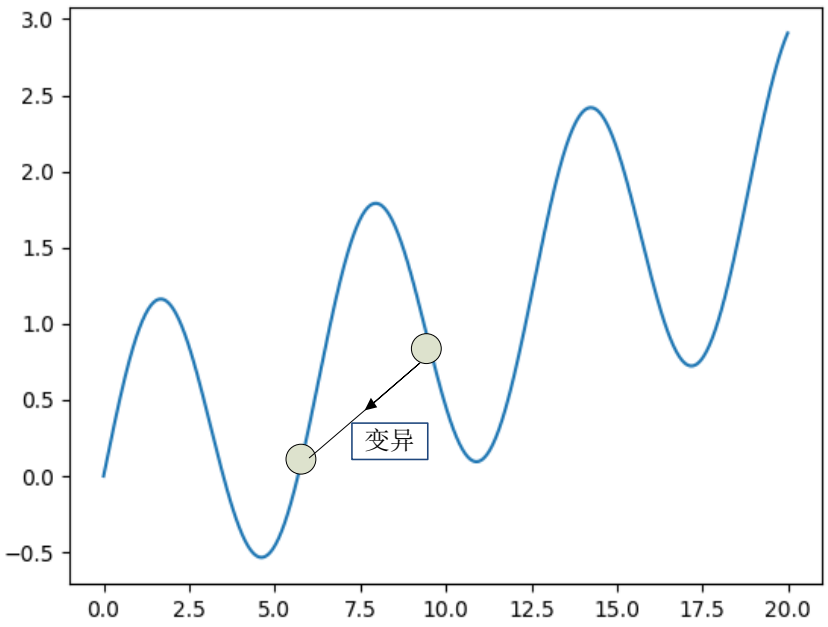

图10 变异过程导致最优解的迁移过程

四、 附件

（本节非必须的，可以列出源代码等，但是要把格式组织好）

Pycharm工程文件夹：TSA_GA