网络空间先进技术研究院

机器学习与人工智能

实训报告

项目名称 遗传算法+TSP

学　　院 计算机科学与教育技术学院

专业班级 网络空间先进技术研究院

学生姓名 徐丹

学生学号 2111806061

任课教师 仇晶

2018 年 11 月

实训报告评定表

学生姓名	徐丹	学　号	2111806061	成绩	
专业班级	网络空间先进技术研究院				
项目名称	遗传算法+TSP				
指导教师评语	指导教师： 年　月　日				

目录

1 项目目的 4
2 项目环境和条件 4
3 项目原理 4
3.1 遗传算法有关的生物学概念 5
3.2 遗传算法的流程 6
3.3 适应度函数分析 9
4 项目内容 10
5 项目过程与内容 11
5.1 任务分析 11
5.2 数据分析 11
5.3 项目开发 12
5.4 关键问题 17
5.5 实验结果分析 17
6 项目总结与心得体会 19
7 参考文献 20

1 项目目的

熟悉和掌握遗传算法的运行机制和求解的基本方法。

遗传算法是一种基于空间搜索的算法，它通过自然选择、遗传、变异等操作以及达尔文的适者生存的理论，模拟自然进化过程来寻找所求问题的答案。其求解过程是个最优化的过程。一般遗传算法的主要步骤如下：

1. 随机产生一个确定长度的特征字符串组成的初始种群。
2. 对该字符串种群迭代地执行下面的步骤 a 和步骤 b，直到满足停止准则为止。

   （a）计算种群中每个个体字符串的适应值；

   （b）应用复制、交叉和变异等遗传算子产生下一代种群。
3. 把在后代中表现的最好的个体字符串指定为遗传算法的执行结果，即为问题的一个解。

2 项目环境和条件

笔记本电脑　Windows7 64bit　PyCharm　Python 3.6

3 项目原理

遗传算法（GA）是一种元启发式自然选择的过程，属于进化算法（EA）大类。遗传算法通常是利用生物启发算子，如变异、交叉和选择来生成高质量的优化和搜索问题的解决方案。

遗传算法本质上是一种搜索算法，搜索算法的共同特征为：

1. 首先组成一组候选解。
2. 依据某些适应性条件测算这些候选解的适应度。
3. 根据适应度保留某些候选解，放弃其他候选解。
4. 对保留的候选解进行某些操作，生成新的候选解。

借鉴生物进化理论，遗传算法将问题模拟成一个生物进化过程，通过遗传、交叉、突变、自然选择等操作产生下一代的解，并逐步淘汰适应度函数值低的解，增加适应度函数高的解。这样进化 N 代后就很有可能会进化出适应度函数值很高的个体。

3.1 遗传算法有关的生物学概念

（1）染色体（Chromosome）

生物是由细胞组成，每一个细胞中都有一套相同的染色体。一条染色体由若干基因(gene)组成，每个基因控制一种特定的蛋白质，从而决定生物的某种特征。所有染色体合称为基因组(genome)。基因组完全决定了一个生物个体。该个体在微观（基因）层次的表现称为基因型(genotype)，在宏观（特征）层次的表现称为显型 (phenotype)。在简单的遗传算法中，将基因组中的若干条染色体看作一整条染色体。

（2） 个体复制

在复制的过程中，父母的染色体通过交叉（Crossover)）产生子女的染色体。染色体还可以以一定的小概率变异（Mutation）。

（3）交叉(Crossover)

2 条染色体交换部分基因，来构造下一代的 2 条新的染色体。染色体交叉是以一定的概率发生的，这个概率记为 Pc。

交叉前：

00000|011100000000|10000

11100|000001111110|00101

交叉后：

00000|000001111110|10000

11100|011100000000|00101

（4）变异(Mutation)

在繁殖过程，新产生的染色体中的基因会以一定的概率出错，称为变异。变异发生的概率记为 Pm 。

变异前：

0000011100000000**10000**

变异后：

0000011100001000**10000**

（5）适应度函数 (Fitness Function)

用于评价某个染色体的适应度，用 f(x)表示。有时需要区分染色体的适应度函数与问题的目标函数。例如：0-1 背包问题的目标函数是所取得物品价值，但将物品价值作为染色体的适应度函数可能并不一定适合。适应度函数与目标函数是正相关的，可对目标函数作一些变

形来得到适应度函数。

3.2 遗传算法的流程

基本的遗传算法通常包括选择、交叉和变异这些基本遗传算子**错误!未找到引用源。**。其数学模型可表示为：

SAG=（C，E，P0，N，Φ，Γ，Ψ，T） (0.1)

其中的 C 为个体的编码方法；E 代表个体适应度评价函数；P0 是初始种群；N 为种群大小；Φ为选择算子；Γ 为交叉算子；Ψ 为变异算子；T 为遗传运算终止条件。遗传算法的流程如图 3.1 所示。

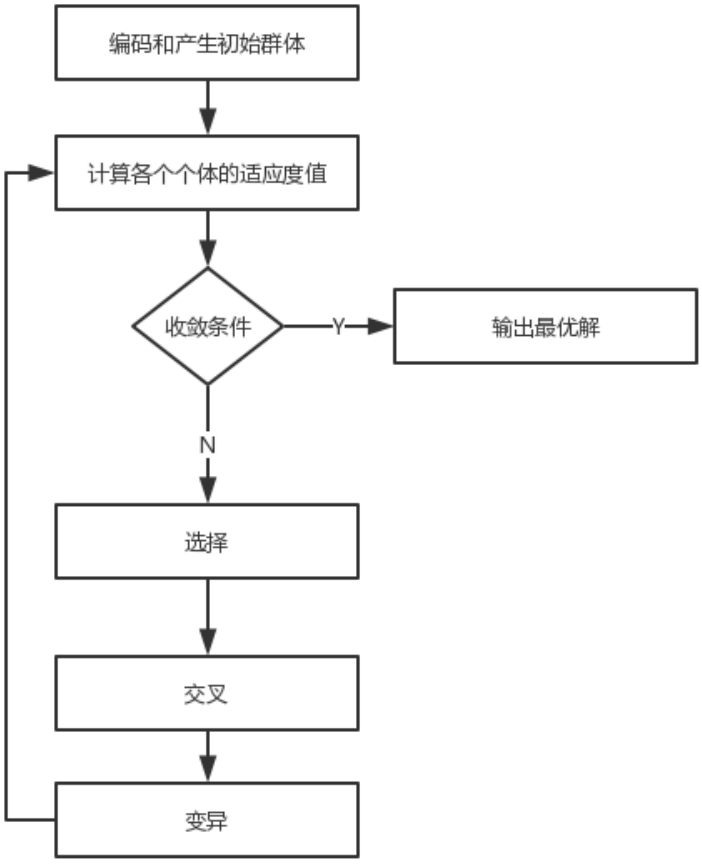

图 3.1 遗传算法流程

1 遗传算法第一步是将要运算的数据转化为可进行遗传运算的编码。编码方式直接决定了个体的染色体排列形式，同时直接影响遗传算法中的选择算子、交叉算子和变异算子的运算，当然也决定了解码方式。通常采用二进制编码。二进制编码使用字符号{0，1}作为编码符号，即是用一个{0，1}所组成的二进制符号串构成个体基因型。将二进制编码方法应用于

遗传算法中有如下优点：

1）遗传算法中的遗传操作如交叉、变异较容易实现，且容易用生物遗传理论来解释；

2）算法可处理的模式多，增强了全局搜索能力；

3）便于编码、解码操作；

4）符合最小字符集编码原则；

5）并行处理能力较强。

与此同时，二进制编码也有它的一些局限性和不足：二进制编码在存着连续函数离散化的映射误差，不能直接反应出所求问题的本身结构特征，不便于开发专门针对某类问题的遗传运算算子。

2　完成了编码后，就可以进行初始种群的设定。基本遗传算法的基本思想即是按随机方法（事先设置好的方式）在可能解空间内产生一个一定规模的初始群体，然后从这个初始群体开始遗传操作，同时为了避免产生局部最优解也要有变异的产生，然后搜索最优解，经过几十甚至更多代的进化变异，根据达尔文的生物进化论“物竞天择，适者生存”。最优解即是最后一代的种群，然后进行解码，最后得到最优解。

初始种群的设定一般服从下列规则：

1）根据优化问题的要求，把握最优解所占空间在整个问题空间的分布范围，然后在此分布范围内设定合适的初始群体。

2）首先随机生成一定数目的个体，然后从中根据约束条件和最优化要求挑出最好的个体加入到初始群体中。同时随机产生一些子个体，避免出现局部最优解。该过程不断迭代，直到初始群体中个体数目达到了预先确定的种群大小。

3　设置初始种群后，就进行基本遗传操作的选择阶段，即选择算子。选择算子的作用是选择合适基因参与遗传运算，目的为防止有用的遗传信息遗失失，从而提高全局收敛效率。常用的遗传算子有：

（1）轮盘赌选择机制

轮盘赌选择也称适应度比例选择，是遗传算法中最基本的选择机制，每个个体被选择进入下一代的概率为这个个体的适应度值占全部个体适应度值之和的比例。但是轮盘赌选择机制的缺点在于选择误差较大，而且不是所有高适应度值的个体都能被选中，适应度值较低但具有优良基因模式的个体被选择的概率也很低，这样就会导致“早熟”现象的产生，即还没有达到最优解就结束了迭代。

（2）最优保存选择机制

最优保存选择机制的基本思想：直接把群体中适应度最高的个体复制到下一代，而不进

行配对交叉等遗传操作。具体步骤如下：

1）找出当前群体中适应度值最高和最低的个体的集合；

2）若当代群体中存在适应度值比迄今为止最好个体的适应度高的个体，则用此个体作为新的迄今为止的最好个体（替代）；

3）用迄今为止的最好个体将当代群体中的最差个体替换掉；

最优保存选择机制的缺点：全局搜索能力不强，虽然对单峰性质优化问题的空间搜索具有较高的效率，但是对多峰性质空间的搜索效率很差，因此该方法只能作为辅助方法使用。

4　完成选择，类似于自然界中的繁殖下一代，即进行交叉。交叉算子在遗传算法中起着核心的作用，是产生新个体的主要方法。在设计交叉算子过程中，既要尽量保护具有优良性状，又要能够有效地产生出一些新的优良模式，主要包括：确定交叉点位置；确定基因交换的方式。二进制编码下的交叉算子分析：

点式交叉算子：

在已经两两配对好的个体中随机选取一个或多个交叉点，然后交换对位的字串。其具体操作步骤如下：

1）采用随机的方法对个体进行两两配对；

2）在配对的个体中，采用随机的方法设置一个或者多个交叉点；

3）依据设定的原则进行染色体交换，形成新的个体。

一致交叉算子：

一致交叉算子通过设定屏蔽字（mask）的方式来决定两个配对个体的某些基因被继承。其具体操作步骤如下：

1）随机生成一个屏蔽字 W，使其与个体编码长度相等。设 W=w1w2…wi…wL，其中 L 为个体编码的长度；

2）当 wi =0 时，参与交换的父代个体在第 i 个基因座上保持不变；

3）当 wi =1 时，参与交换的父代个体在第 i 个基因座上相互交换基因。

5　为防止产生局部最优解，增加算法的局部随机搜索能力，从而可以维持种群的多样性，需要在算法中加入变异阶段，即变异算子。

变异算子模拟基因突变而得到新个体的现象。变异算子作为遗传算法的辅助性算子，其主要功能是使种群在进化过程中维持多样性、防止早熟。变异算子可以加强遗传算法解的局部随机搜索能力，与交叉算子结合共同完成对搜索空间搜索，使遗传算法能够快速完成寻优过程，最终收敛于最优解。

（1）二进制编码下的变异算子分析：

基本变异算子：

基本变异算子是指随机生成一个或多个变异位置，然后对其对应码值取反。具体操作过程：先指定一个变异概率 Pm，然后在（0，1）之间取一组随机数，其长度与编码长度相同。然后将随机数小于变异概率 Pm 的位置上的个体基因值取反。

（2）实数编码下的变异算子分析

当个体的染色体采用实数编码表示时，其变异操作应采用实值变异方法。该方法是用另外一个在规定范围内的随机实数取替换原变异未知上的基因值，产生一个新的个体，最常用的实值变异操作有：

1) 基于位置的变异方法:

该方法是先随机地产生两个变异位置，然后将第二个变异位置上的基因移动到第一个变异位置的前面。

2) 基于次序的变异

该方法是先随机地产生两个变异位置，然后交换着两个变异位置上的基因。

经过一次选择、交叉、变异就完成一次迭代，每一次迭代都要进行一次选择、交叉、变异，然后再进行适应度评估，选取最优个体，更新种群，然后，经过一定的迭代演变，得到最优的种群。

3.3 适应度函数分析

（1）基本的适应度函数

根据适应度值为非负的条件，直接以实际问题的目标函数转化为适应度函数。目标函数的优化方向应与适应度方向一致。这种表达方式会使得某些待求解的函数在函数值的分布上相差很大，种群的平均性能不能被这种情况下得到的平均适应度值所体现，影响算法性能。

（2）适应度函数的变换

1) 线性变换法

线性变换可用下式表示：

$$f' = \alpha * f + \beta \tag{0.2}$$

系数的确定满足如下条件：

$$f'_{avg} = f_{avg}$$

$$f'_{\max} = c_{mult} f'_{avg} \qquad C_{mult} = 1.0 \sim 2.0$$

(0.3)

式中，f 为原来的适应度函数，f’为经过线性拉伸变换后的适应度函数。系数α 和β 的值的设定需要满足以下条件：保持变换前后的适应度的平均值不变；为控制适应度值最大的个体在下一代中的复制，应该使得变换后适应度最大值应与原适应度平均值是一个指定倍数 c 的关系。

式中，favg 为平均适应度，F’max 为最大适应度，c 为最佳个体的期望复制数，一般为 1.0～2.0，当群体规模大小为 50～100 时，一般取值 1.2～2.0。为了避免种群内某些个体适应度远低于平均值而出现变换后适应度值为负的情况，可以进行另一种变换：

2) 幂函数变换

$$f'=f^{k} \tag{0.4}$$

k 与所求优化有关。

3) 指数变换法

$$f'=e^{-af} \tag{0.5}$$

a 决定了复制的强制性，其值越小，复制的强制性就趋向于那些具有最大适应度的个体。

4 项目内容

（1）初始化阶段

初始化对象：种群规模、城市数量、运行代数、交叉概率、变异概率

初始化数据：读入数据源，将坐标转换为距离矩阵（标准化欧式距离）

初始化种群：随机生成 n 个路径序列，n 表示种群规模。

（2）计算种群适应度

这里表示每条路径求和。

（3）计算累计概率

计算初始化种群中各个个体的累积概率

（4）迭代

选择算子：赌轮选择策略挑选下一代个体。

交叉运算：第 k 个算子和 k+1 个算子有一定的概率交叉变换，k=0、2、4、...、2n

变异运算：每个算子有一定概率基因多次对换，概率处决与变异概率

计算新的种群适应度以及个体累积概率，并更新最优解。

将新种群 newGroup 复制到旧种群 oldGroup 中，准备下一代进化（迭代）

（5）输出

输出迭代过程中产生的最短路径长度、最短路径出现代数、以及最短路径

5 项目过程与内容

5.1 任务分析

旅行商问题的描述是：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。旅行商问题一个典型的组合优化问题，并且是一个 NP 难题 ，其可能的路径总数与城市数目 n 是成指数型增长的，所以一般很难精确地求出其最优解，因而寻找出有效的近似求解算法就具有重要的意义。

运用遗传算法（GA，Genetic Algorithm）求解旅行商问题（TSP，Travelling Salesman Problem）。

依照遗传算法的思想，将城市编码为“基因”（即所有城市从 0 开始顺序编号），然后生成若干个基因不同的个体（即城市编号的一个排列），让这些个体相互竞争（即采用交叉、变异的方法改变城市的排列），并使用一种评估机制让它们“优胜劣汰”（即取总路程的倒数这个评估函数），最终“进化”出足够优秀的解（即最优的路线和路程）。

5.2 数据分析

1 城市规模

选取 50 个城市作为一个种群，对城市进行实数编码，用遍历城市的顺序作为编码方式，比如：0，1，2，3，4，5，6，…，47，48，49，城市的坐标分别存储在 distance_x[]和 distance_y[]两个数组，然后用 random 函数对初始值进行随机化处理，从而可以更改初始城市坐标位置。

2 总路径计算

通过两点间距离公式计算两个城市之间的距离，然后进行累加，得到总路径长度。

3　评估总路径

总路径越短越好。为了处理方便，选取总路程的倒数作为评估标准。个体的分数越大，则总路程越小。 通过这个评估函数，我们便能给所有“基因”个体打分，并基于这个打分产生下一代。

5.3　项目开发

根据数据分析和任务分析，通过编写 GA.py，Life.py，TSPGA.py 完成不同功能，实现遗传算法求解旅行商问题。

5.3.1 GA 实现--GA.py

GA.py 中实现遗传算法类，流程如图 5.1 所示。

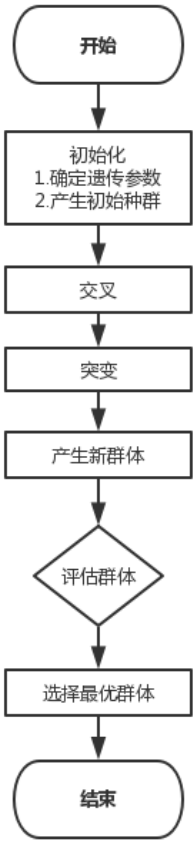

图 5.1 遗传算法类的流程

初始化参数如表 5.1 所示，具体函数如表 5.2 所示。

表 5.1 初始化参数

参数	含义
self.xKate =0.7	突变率

self.mutationRate=0.005	交叉率
self.mutationCount=0	突变次数
self.generation=0	进化次数
self.lives=[]	生命集合
self.bounds=0.0	得分总数
self.best=None	最优解
self.lifeCount=50	生命个数
self.geneLength = 50	基因长度

表 5.2 主要函数表

函数名称	实现功能
__xFunc(self, p1, p2)	默认交叉函数，产生新个体
__mFunc(self, gene)	默认变异函数，避免选择陷入局部最优解
__bear(self, p1, p2)	产生后代，进行交叉，突变
__getOne(self)	根据得分情况，随机取得一个个体，机率正比于个体的 score 属性
__newChild(self)	产生新的后代，调用__bear(self, p1, p2)
judge(self, f = lambda lf, av: 1)	根据传入的方法 f，求得最优生命体和生命集总分
next(self, n = 1)	演化至下 n 代，每一代进行如下操作：评估群体，新生命集，产生新的生命集个体，更新新的生命集

5.3.2 创建生命体--Life.py

创建 Life 类，用于创造生命集，在 GA 类中被调用。主要的函数如表 5.3 所示。

表 5.3 主要函数表

函数名称	实现功能
__init__(self, env, gene = None)	初始化，用于创建生命体基因
__rndGene(self)	随机初始化基因
setScore(self, v)	设置评估分数
addScore(self, v)	增加评价分数

5.3.3 旅行商问题求解--TSPGA.py

TSPGA.Py 实现可视化界面，调用 GA 类，完成四大功能：随机初始、开始进化、停止进化、退出程序。流程如图 5.2 所示。

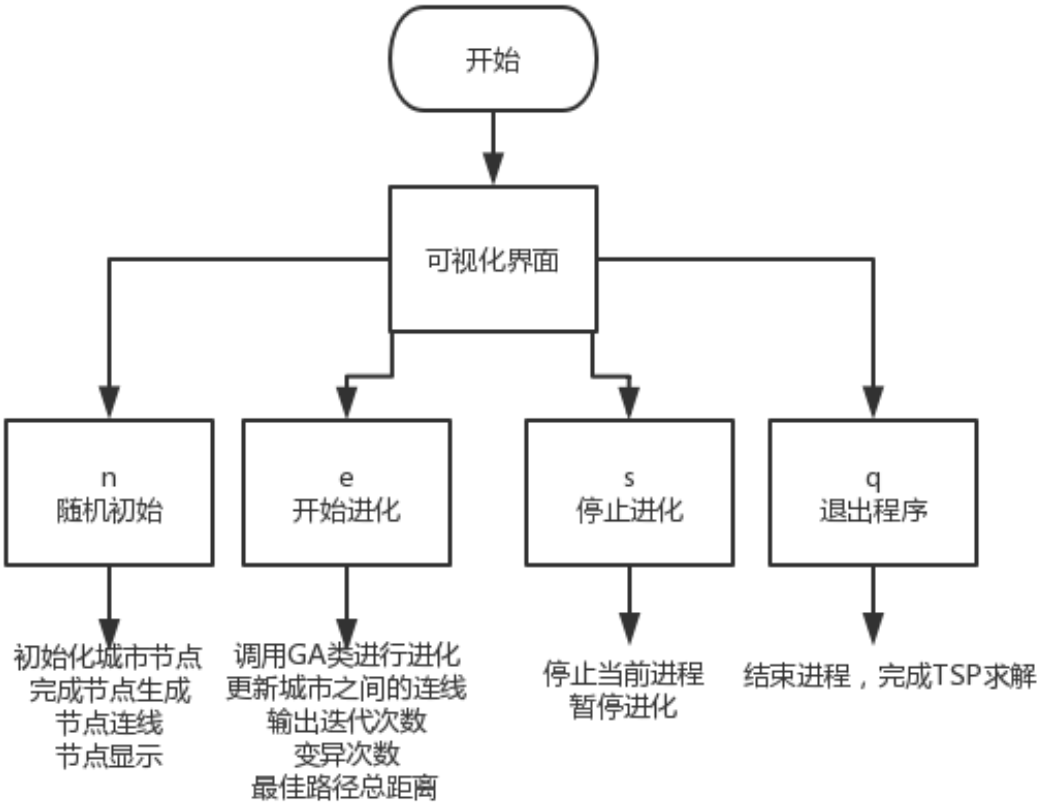

图 5.2TSPGA.py 流程

通过定义了四个按键响应程序 n，e，s，q，完成 TSP 的求解，其中关键的主要函数如表 5.4 所示。

表 5.4 主要函数表

函数	作用
__init__(self, root, width = 800, height = 600, n =50)	调用 Tkinter 类进行界面的初始化，城市数目初始化为 50
__bindEvents(self)	按键响应程序,完成四大功能： 随机初始；开始进化；停止进化；退出程序

distance(self, order)		得到当前顺序下连线总长度
new(self, evt = None)		随机初始，随机初始化城市坐标并显示，顺序连线城市
GA 类	mkLife(self) judge(self) xFunc(self) mFunc(self) save() lifeCount = 50, xRate = 0.7, mutationRate = 0.1,	创造新生命，随即顺序 评价函数 交叉函数 变异函数 保存 生命个数初始为 50 交叉率为 0.7 突变率为 0.1
evolve(self, evt = None)		调用 GA 类完成遗传进化，更新画布输出迭代次数，变异次数，最佳路径距离
line(self, order)		将节点按 order 顺序连线
clear(self)		清除画布
quite(self, evt)		退出程序
stop(self, evt)		停止进化
Threading 类		防止线程死锁

其中调用 Threading 类，防止线程死锁，然后对进程进行停止和启动。对应开始进化、停止进化、退出程序。在开始进化中开启线程，在停止进化和退出程序中停止线程，具体代码如图 5.3 所示。

```
self.__lock = threading.RLock()      # 线程锁

# 停止线程
self.__lock.acquire()
self.__running = False
self.__lock.release()
```

```
# 开启线程
self.__lock.acquire()
self.__running = True
self.__lock.release()
```

图 5.3 线程调度代码

其中在初始化界面使用了 Tkinter 类，用于绘制城市节点，节点连线，显示坐标，设置颜色，标题，Tkinter 类的作用如表 5.5 所示，具体实现代码如图 5.4 所示。

表 5.5Tkinter 类

名称	作用
Tkinter.Canvas	Canvas 为 Tkinter 提供了绘图功能. 其提供的图形组件包括 线形, 圆形, 图片, 甚至其他控件

```
# Tkinter.Canvas,Canvas 为 Tkinter 提供了绘图功能. 其提供的图形组件包括 线形, 圆形, 图片, 甚至其他控件
self.canvas = Tkinter.Canvas(
        root,
        width = self.width,
        height = self.height,
        bg = "#EBEBEB",              # 背景白色
        xscrollincrement = 1,        #用于滚动请求水平滚动的数量值
        yscrollincrement = 1         #类似 xscrollincrement, 但是垂直方向
    )
self.canvas.pack(expand = Tkinter.YES, fill = Tkinter.BOTH)
self.title("TSP遗传算法(n:随机初始 e:开始进化 s:停止进化 q:退出程序)")

# 初始化城市节点
for i in range(len(distance_x)):
    # 在画布上随机初始坐标
    x = int(random.random()*(distance_x[i]))+30
    y = int(random.random()*(distance_y[i]))+30
    #x = distance_x[i]
    #y = distance_y[i]
    self.nodes.append((x, y))
    # 生成节点椭圆，半径为self.__r
    node = self.canvas.create_oval(x - self.__r,
            y - self.__r, x + self.__r, y + self.__r,
            fill = "#ff0000",      # 填充红色
            outline = "#000000",   # 轮廓白色
            tags = "node",
        )
    self.nodes2.append(node)
    # 显示坐标
    self.canvas.create_text(x,y-10,                 # 使用create_text方法在坐标（302，77）处绘制文字
            text = '('+str(x)+','+str(y)+')',       # 所绘制文字的内容
            fill = 'black'                          # 所绘制文字的颜色为灰色
        )
```

图 5.4 画布创建代码

其中关键部分在于调用 GA 类，完成遗传进化，求解出城市之间的最短总路径。TSPGA.py 根据旅行商问题的具体情况，重新改写评价函数、交叉函数、变异函数，如表 5.6 所示，具体代码如图 5.5 所示。

表 5.6 主要函数表

函数	具体改写后的说明
xFunc(self)：交叉函数	选择 lf2 序列前子序列交叉到 lf1 前段，删除重复元素
mFunc(self)：变异函数	选择两个不同位置基因交换，第一个选择的基因重新加入到序列尾端
judge(self)：评价函数	lambda lf, av = 100: 1.0 / self.distance(lf.gene)

```
# 评价函数
def judge(self):

    return lambda lf, av = 100: 1.0 / self.distance(lf.gene)

# 交叉函数：选择lf2序列前子序列交叉到lf1前段，删除重复元素
def xFunc(self):

    def f(lf1, lf2):
        p2 = random.randint(1, self.n - 1)
        # 截取lf2
        g1 = lf2.gene[0:p2] + lf1.gene
        g11 = []
        for i in g1:
            if i not in g11:
                g11.append(i)
        return g11
    return f

# 变异函数:选择两个不同位置基因交换，第一个选择的基因重新加入到序列尾端
def mFunc(self):

    def f(gene):
        p1 = random.randint(0, self.n - 1)
        p2 = random.randint(0, self.n - 1)
        while p2 == p1:
            p2 = random.randint(0, self.n - 1)
        gene[p1], gene[p2] = gene[p2], gene[p1]
        gene.append(gene[p2])
        del gene[p2]
        return gene

    return f
```

图 5.5 交叉函数；变异函数；评价函数的代码

5.4 关键问题

首先是遗传算法的关键函数的实现，后代的产生以及竞争方式；然后是城市的编码方式；最后需要实现一个可视化的界面来动态地展示当前路线，这个动态变化要跟得上算法处理结果的产生。

5.5 实验结果分析

程序一开始运行会不断进行迭代计算当前一代的最佳路径，不会自动停止，需要点击”s”(停止进化)和”q”（退出程序）来停止计算。当迭代到 28515 代时，路径长度已稳定在

28515。实验的迭代效果如图 5.4，5.5，5.6 所示。选取其中 10 代制作流程图，展现路径变化的趋势，如图 5.7 所示。

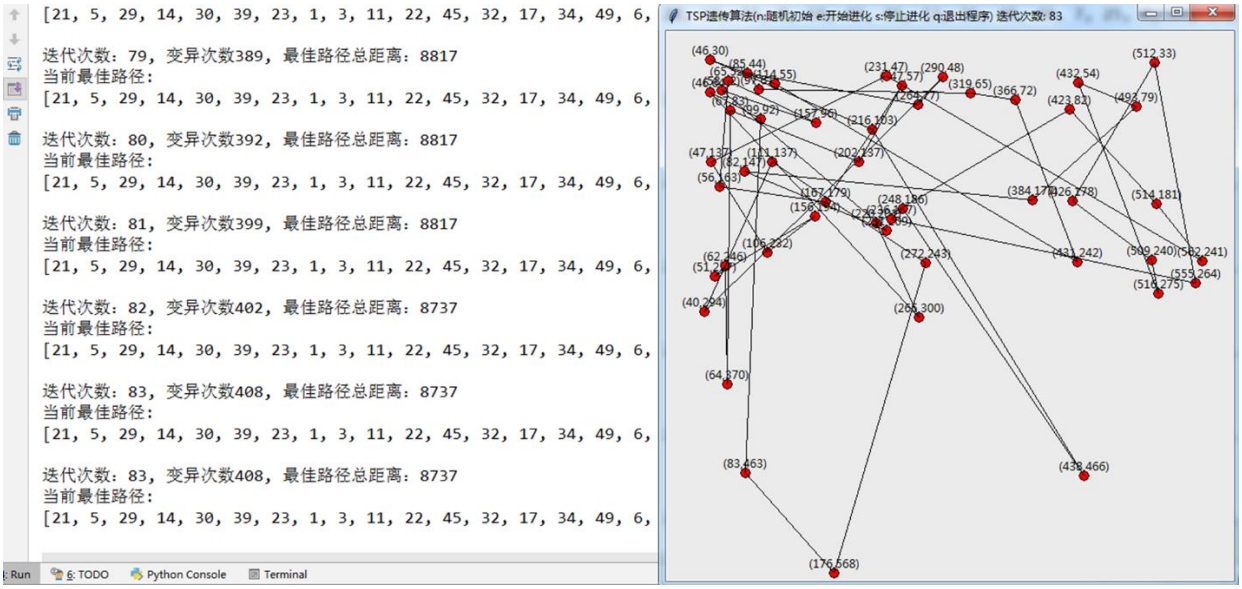

图 5.4 迭代 83 次

迭代次数为 83 次的结果如下：从中可以看出城市各点之前的路线比较杂乱，总距离比较大。

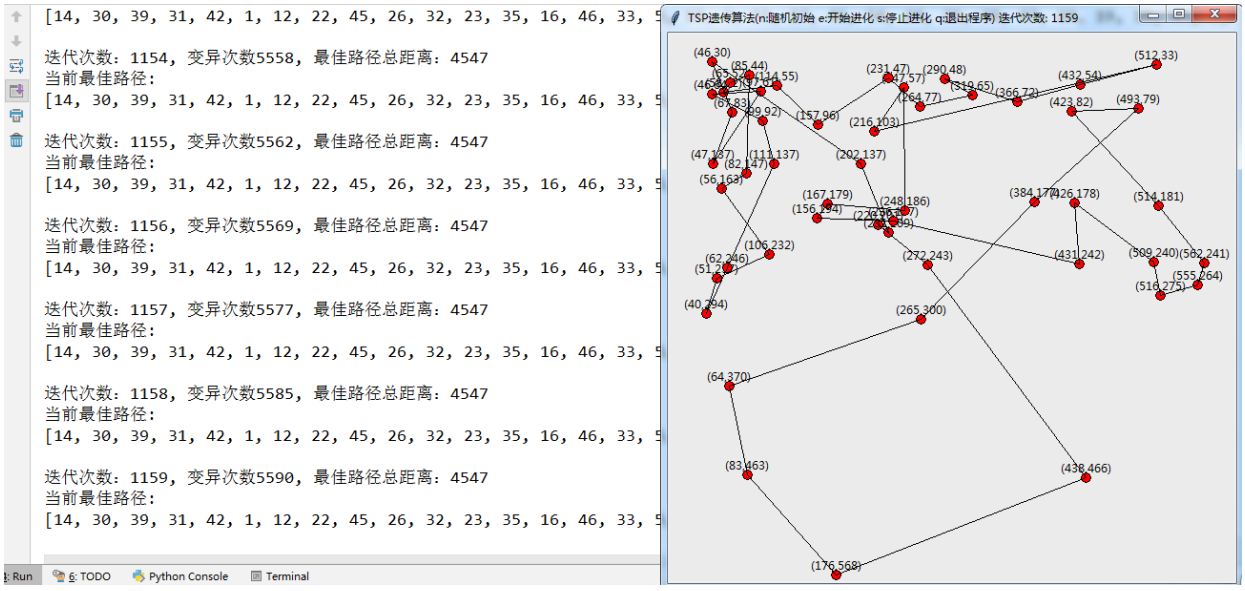

图 5.5 迭代 1159 次

迭代次数为 1159 次的结果分析：经过多次的迭代处理后，可以看出城市路线变得清晰，总距离较之前的缩短了一倍左右。

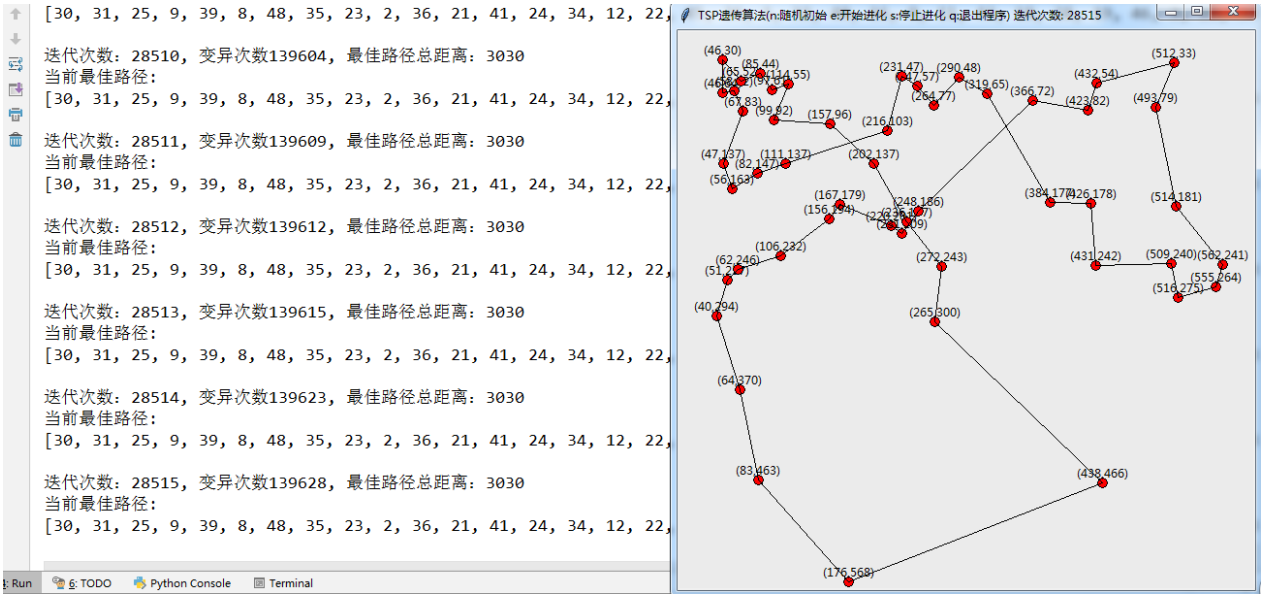

图 5.6 迭代 28515 次

迭代次数为 28515 次的结果分析：在更多次的迭代后，我们发现路线一直没有发生变化，基本达到了理想的效果，此时的路线和总距离在本次的算法处理中达到了最优。

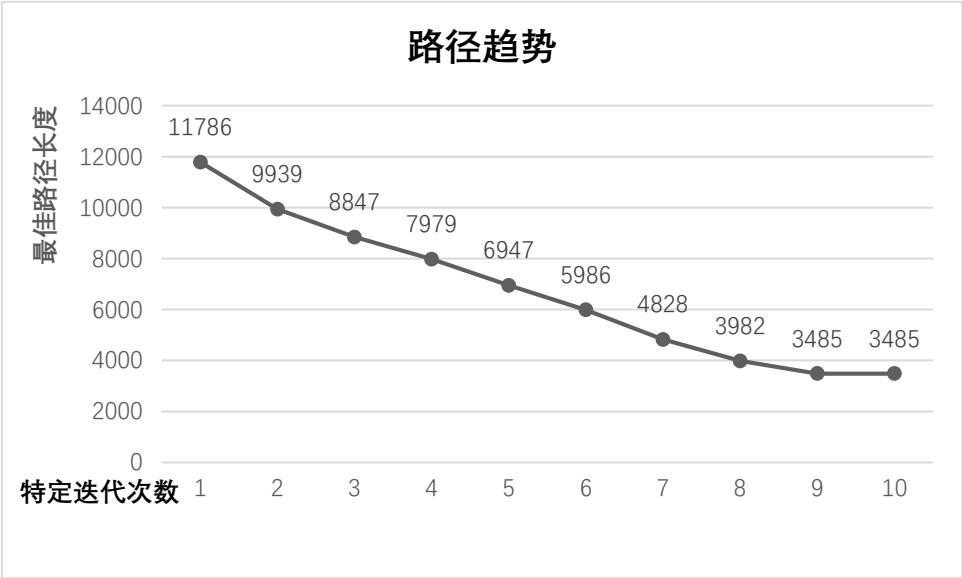

图 5.7 不同迭代下最佳路径长度

6 项目总结与心得体会

通过本次实验，我对计算机和人工智能有了一定的了解，同时也对遗传算法的优点和缺点总结出了一些心得体会。

遗传算法的优点:

（1）群体搜索，易于并行化处理；

（2）不是盲目穷举，而是启发式搜索；

（3）适应度函数不受连续、可微等条件的约束，适用范围很广。

（4）容易实现。一旦有了一个遗传算法的程序，如果想解决一个新的问题，只需针对新的问题重新进行基因编码就行；如果编码方法也相同，那只需要改变一下适应度函数就可以了。

遗传算法的缺点:

（1）全局搜索能力不强,很容易陷入局部最优解跳不出来；(可结合 SA 进行改进,因为 SA 在理率上是 100%得到全局最优的,但搜索代价高)

（2）计算时间长。

在之后的学习中，我计划尽快完成 tensorflow 等的使用，来进一步掌握深度学习等技术。

7 参考文献

[1] 李和壁. 遗传算法(GA)在旅行商问题(TSP)中的应用[J]. 科技创新与应用, 2015(10):48-49.

[2] 王煦法. 遗传算法及其应用[J]. 小型微型计算机系统, 1995, 23(2):9-10.

[3] 吉根林. 遗传算法研究综述[J]. 计算机应用与软件, 2004, 21(2):69-73.

[4] 李飞, 白艳萍. 用遗传算法求解旅行商问题[J]. 中北大学学报(自然科学版), 2007, 28(1):49-52.

[5] 孙惠文. 遗传算法求解旅行商问题[J]. 西南交通大学学报, 1996, 31(5):550-554.

[6] 陈江华, 林爱文, 杨明,等. 遗传算法求解 TSP 问题的研究进展[J]. 昆明理工大学学报(自然科学版), 2003, 28(4):9-13.

[7] 廖晓明, 罗四维. 遗传算法用于 TSP 问题的研究[J]. 北京交通大学学报,

1995(4):563-566.

[8] 代桂平，王勇，侯亚荣．基于遗传算法的TSP问题求解算法及其系统[J]．微计算机信息，2010，26(4):15-16.

[9] 易敬，王平，李哲．基于遗传算法的TSP问题研究[J]．信息技术，2006，30(7):110-112.

[10] 余一娇．用简单遗传算法求解TSP问题的参数组合研究[J]．华中师范大学学报(自然科学版)，2002，36(1):25-29.