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 计算机科学与教育软件学院

专业班级 计算机技术方班

学生姓名&学号：

张　宇 2111806072、王朝斌 2111806049

王为国 2111806051、徐　丹 2111806061

冯　毅 2111806017

任课教师 仇晶

2018 年　11 月

实训报告评定表

学生姓名	张 宇 王朝斌 王为国 徐 丹 冯 毅	学　号	2111806072 2111806049 2111806051 2111806061 2111806017	成绩	
专业班级	计算机技术方班				
报告名称	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

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[2]。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)。[3]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[3]（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[3]。

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[3]。

变异前：

0000011100000000010000

变异后：

0000011100001000010000

（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[4]。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。

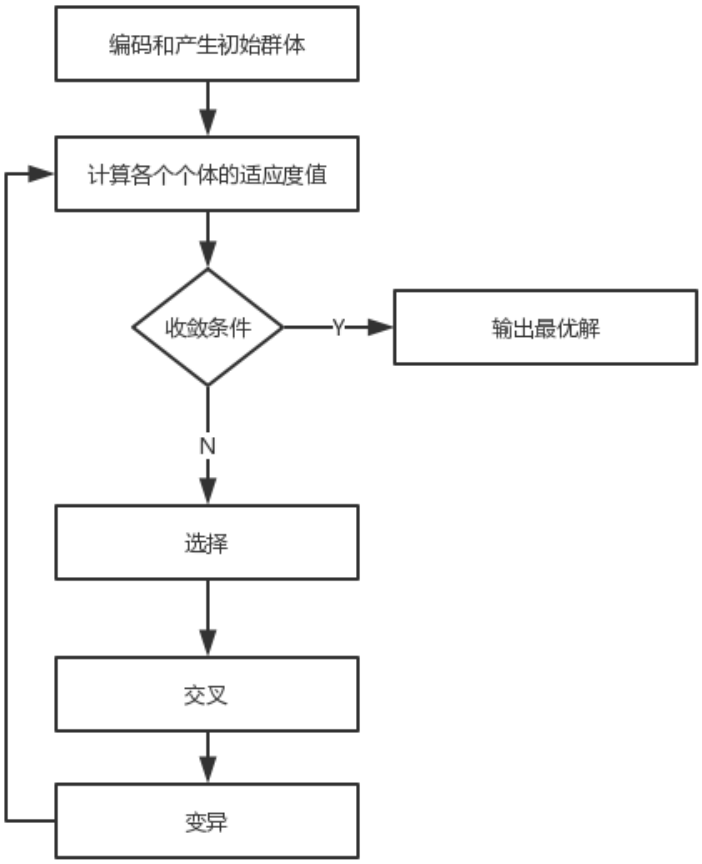

图 3.1 遗传算法流程

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[5]：

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[5]：

（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[5]：

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[5]。

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[6]

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：

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

$$f'_{\max} = c_{mult} f'_{avg} \qquad C_{mult} = 1.0 : 2.0 \tag{0.3}$$

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）遗传算法的理解与分析

（2）遗传算法在旅行商问题的实现

（3）实验结果分析

（4）项目总结与心得体会

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。

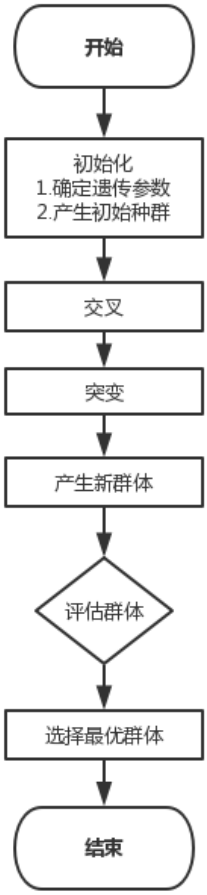

图 5.1 遗传算法类的流程

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。

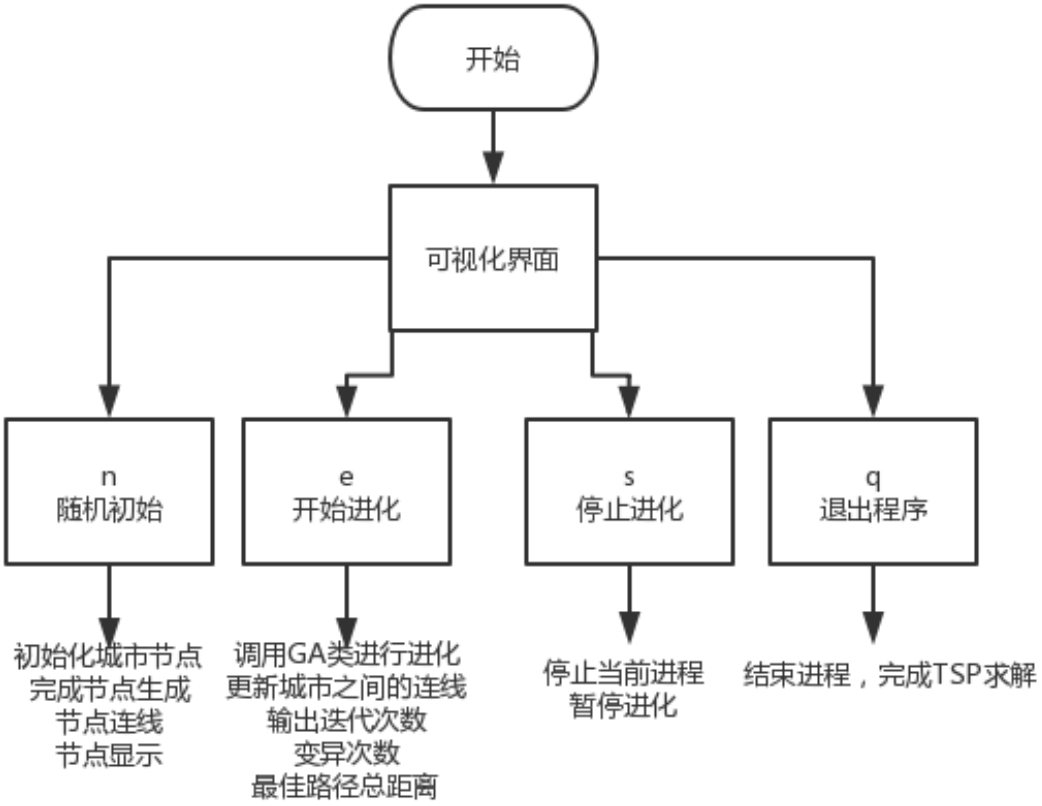

图 5.2TSPGA.py 流程

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if2
        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 3030。实验的迭代效果如图 5.4，5.5，5.6 所示。

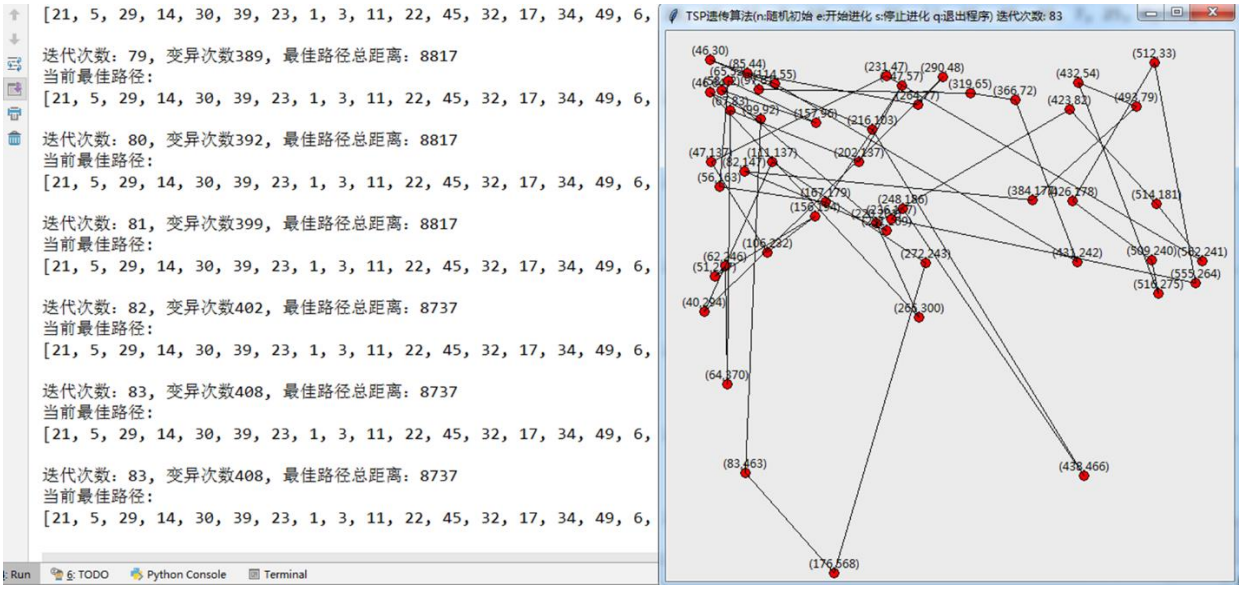

图 5.4 迭代 83 次

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，路径长度为 8737。

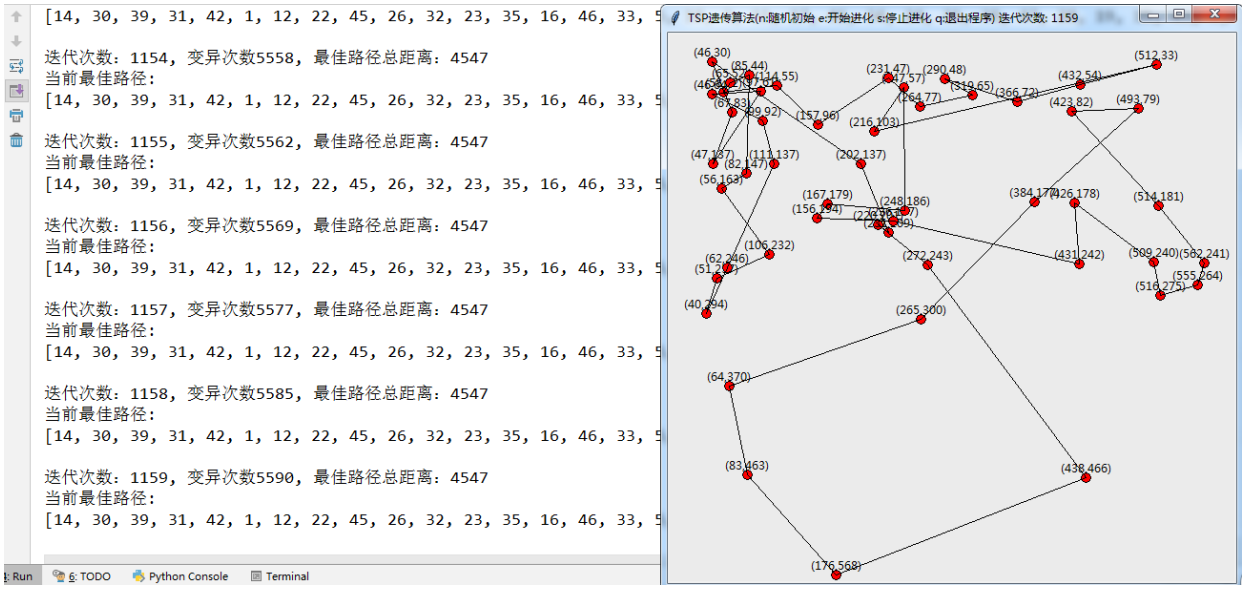

图 5.5 迭代 1159 次

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，路径长度为 4547。

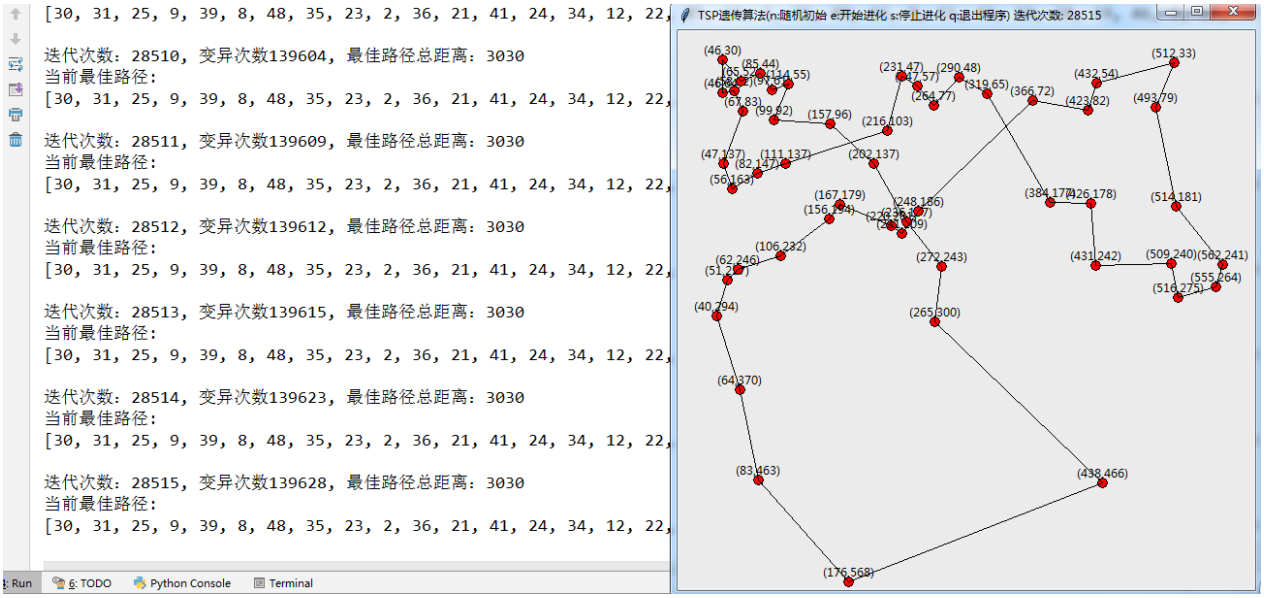

图 5.6 迭代 28515 次

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，路径长度为 3030。

选取其中 10 代(1、79、124、328、980、1675、3366、25288、30000)制作流程图，展现路径变化的趋势，如图 5.7 所示。

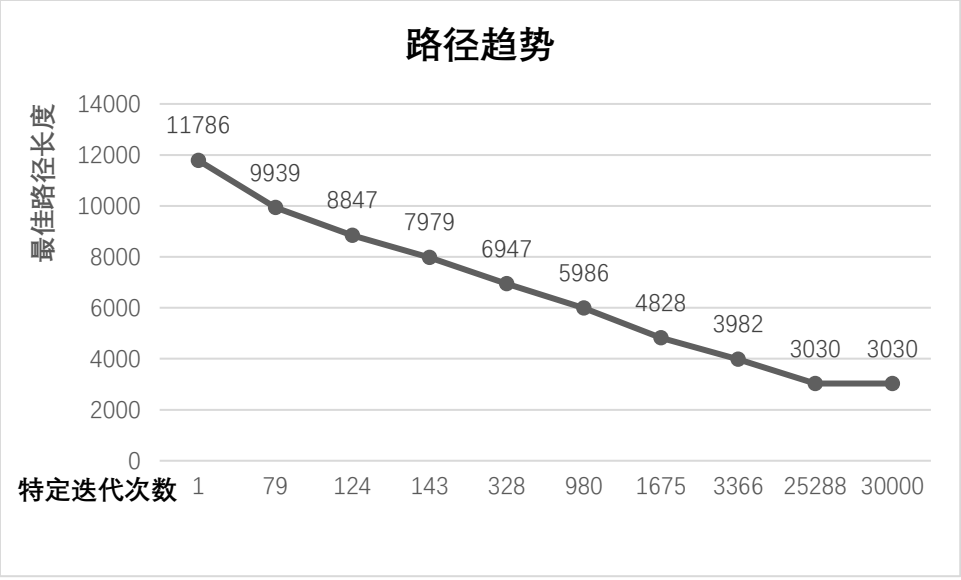

图 5.7 不同迭代下最佳路径长度

由图 5.7 可以得出，随着迭代次数的增大，城市的总路径不断递减，到 25288 代之后，总路径趋于稳定，得到总路径的最优解：3030。

6 项目总结与心得体会

张宇

这次项目是开学以来第一次团队性质的合作完成一个较为完整的课题，无论是对我本人，还是我们的组员的团结合作，沟通方面和学术研究方面都是一个不小的锻炼，在这次项目的完成中，我们组 5 个人各有分工，同时又互相协作，共同高效的在一到两周的时间内完成了对‘遗传算法’这一课题的从发展到原理，再到最后的应用，以及代码的实现的整个过程，这过程中我们互相协作，同时提升了自己也暴露了自己的不足，对于这次的项目完成整个过程，我个人有如下总结和心得：

1. 合作是第一位，个人能力才是其次。由于我们组是第一个在课堂上汇报，在老师提出要汇报，到我们汇报，整个过程不到两周，在两周的时间内完成对算法的理解，代码编写，ppt 制作以及汇报的准备，再加上我们还有平常的上课和日常的其他事务，时间不可谓不紧，因此，无论个人能力再强大，也是不能完成的，因此我们一开始拿到这个题目，就开始分工，我本科接触过遗传算法，因此我选择讲原理，徐丹之前做过老师，因此他能够以一种活泼且严谨的态度和能力开场，讲解遗传算法的发展和历史，王朝斌和冯毅本科主修过计算机和编程，因此他们负责了应用和代码展示，而王为国文字功底很很好，因此对我们 ppt 中的语言和措辞，以及汇报的过程提出了很多建议，同时在报告方面也出了很大力，我们各有分工，才能高效高质的完成这个任务。
2. 沟通很关键，集思广益。闭门造车是不可取的，我们虽然个子有分工，但是交流也很多，这样对我们各自的发挥都有很大的帮助，同时，我们咨询了一些班上的较为精通这方面的同学看，也得到了很多建议，通过这种方式，我们就能节省很多时间。
3. 实践是检验真理的唯一标准。实践，尤其是在计算实际这一领域，是非常重要的，之前我也了解过遗传算法，但是只是浮于表面，并没有深入了解，但是这次的项目，我参与了整个过程，不仅是原理，也了解了遗传算法的发展和历史，同时，解除了代码，对怎么做一个相对完成度较高的程序也有了了解和自己的理解，同时与应用接地，更深入了解了这样一个很有创意的算法。

实验部分：

在这次项目中，理解了遗传算法的原理和发展，并通过 Python 语言实现了遗传算法的实现，并解决了一个经典的最优化问题——旅行商问题，最终实现了最优解的求解，实现了实

验的要求，但仍存在一些问题：

1. PPT 讲解过程中一些例子取得太过深入，在讲解最基本的问题时忽略了观众的认知程度，直接举出一些进阶的例子，观众难以理解。
2. 代码实现过程中出现了一些小 bug，最后得到处理，但是暴露了事先准备疏忽。
3. PPT 制作内容较多，由于准备很充足，因此想给观众讲很多内容，忽略了观众在有限的时间内的接受程度。

最后，这次的项目的经历对我来说，不仅是技术的提升，也是计算机思维的培养和团队合作能力的一个提升和培养。

王朝斌

6.1 项目总结

通过这次“遗传算法+TSP”的实验，主要完成了两大任务：遗传算法理解和遗传算法求解旅行商问题。

在遗传算法的理解中，从遗传算法的定义、相关生物学概念、遗传算法过程三个方面来分析。其中，在遗传算法的过程中，对编码、选择、交叉、变异四个步骤进行了具体分析。

1　编码是运用遗传算法时要解决的首要问题，不同问题要选择不同的编码方式，是设计遗传算法时的关键步骤。编码方法影响到交叉算子、变异算子等遗传算子的运算方法，一定程度上决定了遗传进化的效率。

2　选择是用来确定如何从父代群体中按某种方法选取那些个体，以便遗传到下一代群体，常见的选择算子有：轮盘赌选择、随机竞争选择、最佳保留选择。

3　交叉操作，是指对两个相互配对的染色体按某种方式相互交换其部分基因，从而形成两个新的个体，交叉算子：两点交叉、多点交叉、均匀交叉。通过交叉操作可以获得新的个体。

4　变异运算，是指将个体染色体编码串中的某些基因座上的基因值用该基因座上的其它等位基因来替换，从而形成新的个体。通过变异操作可以避免选择陷入局部最优解。

在遗传算法求解旅行商问题中，基于遗传算法的流程，编写了 GA.py，Life.py，TSPGA.py 来完成实验。首先， GA.py 实现 GA 类，完成选择、评估、交叉、变异等功能，Life.py 创建新的生命集，TSPGA.py 实现可视化界面，调用 GA 类，完成四大功能：随机初始（n）、开始进化（e）、停止进化（s）、退出程序（q）。

1　点击“n”将会随机改变画布上初始的城市节点和连线，生成新的城市序列。

2　点击“e”则开始进行遗传演变，调用 GA 类，每一代进行选择、交叉、变异，然后进行评估，将当前最优个体加入种群，生成新的种群。每一次进化，城市的总路径不断减少，然后不停迭代，向城市的总路径不断递减的方向发展。在程序实现中，没有设置最大迭代次数，需要点击“s”或者“q”来终止迭代运算。

6.2　心得体会

在这次实验中，主要负责遗传算法的流程分析和部分 TSPGA.py 的代码编写。通过这次实验，理解了遗传算法的流程，遗传算法虽然不一定能得到最优解，但是能够为我们在一个巨大的解空间中，通过交叉和变异，生成新个体和避免陷入局部最优解，从而很快速的搜索解空间，搜索到一个接近最优的次优解，或者就是最优解。在编程中，调用了 Tkinter 类，实现可视化的界面，编写四个按键响应程序：随机初始（n）、开始进化（e）、停止进化（s）、退出程序（q），完成 TSP 的求解。

在程序运行中，还存在着一些问题，比如城市的路径长度已经趋于一个稳定值是时，画布的城市连线还会发生一些轻微变化，这还待后续的代码优化。

王为国

遗传算法是模仿自然界生物进化机制发展起来的随机全局搜索和优化方法，它借鉴了达尔文的进化论和孟德尔的遗传学说。其本质是一种高效、并行、全局搜索的方法，它既能在搜索中自动获取和积累有关空间知识，并自适应地控制搜索过程以求得最优解遗传算法操作使用适者生存的原则，在潜在的解决方案种群中逐次产生一个近视最优方案。在遗传算法的每一代中，根据个体在问题域中的适应度值和从自然遗传学中借鉴来的再造方法进行个体选择，产生一个新的近视解。这个过程导致种群中个体的进化，得到的新个体比原个体更适应环境，就像自然界中的改造一样。

遗传算法在人工智能的众多领域具有广泛应用。例如，机器学习、聚类、控制（如煤气管道控制）、规划（如生产任务规划）、设计（如通信网络设计、布局设计）、调度（如作业车间调度、机器调度、运输问题）、配置（机器配置、分配问题）、组合优化（如 tsp、背包问题）、函数的最大值以及图像处理和信号处理等等。遗传算法多用应与复杂函数的优化问题中。

遗传算法模拟了自然选择和遗传中发生的复制、交叉、和变异等现象，从任一初始种群出发，通过随机选择、交叉、变异操作，产生一群更适合环境的个体，使群体进行到搜索空间中越来越好的区域，这样一代一代地不断繁衍进化，最后收敛到一群最适合环境的个体求得问题的最优解。

算法流程 1. 编码：解空间中的解数据 x，作为作为遗传算法的表现型形式。从表现型到基本型的映射称为编码。遗传算法在进行搜索之前先将解空间的解数据表示成遗传空间的基本型串结构数据，这些串结构数据的不同的组合就构成了不同的点。

2. 初始种群的形成：随机产生 n 个初始串数据，每个串数据称为一个个体，n 个串数据构成了一个群体。遗传算法以这 n 个串结构作为初始点开始迭代。设置进化代数计数器 t 0;设置最大进行代数 t；随机生成 m 个个体作为初始群体 p(0)。

3. 适应度检测：适应度就是借鉴生物个体对环境的适应程度,适应度函数就是对问题中的个体对象所设计的表征其优劣的一种测度。根据具体问题计算 p(t)的适应度。

4. 选择：将选择算子作用于群体。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。选择操作是建立在群体中个体的适应度评估基础上的。

5. 交叉：将交叉算子作用于群体。所谓交叉是指把两个父代个体的部分结构加以替换重组而生成新个体的操作。遗传算法中起核心作用的就是交叉算子。

6. 变异：将变异算子作用于群体。即是对群体中的个体串的某些基因座上的基因值作变动。群体 p(t)经过选择、交叉、变异运算之后得到下一代群体 p(t+1)。

7. 终止条件判断:若 t<=t,则 t=t+1，转到第 3 步，否则以进化过程中所得到的具有最大适应度个体作为最优解输出，终止计算。

在此次实验过程中，加强了对遗传算法的认识，提升了自己的代码能力，大大的提高团队合作的能力，为日后的学术打下坚实的基础。

徐丹

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。

冯毅

项目总结：

本次项目目的是为了了解遗传算法的运行机制和求解的基本方法，了解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。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。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。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。

基于遗传算法的思想，我们进行求解旅行商问题，旅行商问题是指商人要拜访 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。第一步是将要运算的数据转化为可进行遗传运算的编码，我们选取了 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，以随机的方式进行排列，在可视化界面上以点连线的方式呈现。设置好交叉、变异函数，在产生的后代中，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，求得最优生命体和生命集总分。我们在可视化的界面实现了开始进化、停止进化、退出程序的功能，实时展示每一次迭代所产生的结果，经过多次实验，发现用遗传算法处理旅行商问题能得到很好的路径和距离。

心得体会：

本次项目我主要参与了遗传算法的代码编写和调试，以及课堂上的演示。在学习遗传算法的编写过程，我对遗传算法的基本处理过程有了比较直观的认识，特别是处理旅行商这个问题很有意思，能在界面上看到点与点连线的变化，我发现在生活中通过利用这个遗传算法能比较高效地节省时间，比如在学校里面，当需要去多个地方处理事情的时候，我可以计算一下去各个地方的先后顺序，这样处理事情就不会频繁的来回跑。

其次，我对 Python 代码的编写能力也有了一定的提升，懂得了如何去找资料，以及处理各种出现的错误代码。在课堂上的代码演示中让我有了一次能与老师同学们交流的机会，从中也发现了一些代码中出现的问题，比如城市结点数不能按需设置，不够灵活，必须通过修改相关代码才能呈现所要的效果，这些问题在以后的编程过程中需要注意。

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.