《人工智能基础》课程实验二报告

一、Hopfield网络

1.算法原理

1982年，Hopfield提出了Hopfield网络模型。Hopfield网络模型是一种循环神经网络，从输出到输入有反馈连接。反馈神经网络由于其输出端有反馈到其输入端；所以在输入的激励下，会产生不断的状态变化。当有输入之后，可以求取出相应的输出，这个输出反馈到输入从而产生新的输出，这个反馈过程一直进行下去。如果反馈神经网络是一个能收敛的稳定网络，则这个反馈与迭代的计算过程所产生的变化越来越小，一旦到达了稳定平衡状态；那么网络就会输出一个稳定的恒值。

而Hopfield网络的其核心思想是通过能量函数来实现网络的稳定性，将网络状态值与能量函数对应起来，能量函数的最小值即对应着网络的稳定输出，进而通过设计合适的有界能量函数 使其值能保证随着网络状态的变化而始终下降，从而实现了网络的稳定性。同时，到达稳定时的网络状态即是需要求解的结果。

Hopfield神经网络模型有离散型和连续性两种，离散型适用于联想记忆，连续性适合处理优化问题。

离散型Hopfield网络

每个神经元只取二元的离散值0、1。每个神经元的输入u_i和输出v_i的关系如下：

$$u_i(t+1)=\sum_j w_{ij}v_j(t)+I_i$$

$$v_i(t+1)=f(u_i)=\begin{cases}1, & u_i>0\\ 0, & u_i\leqslant 0\end{cases}$$

其中$ I_i 是神经元i的外部连续输入，f()$是激活函数。用于联想记忆时训练完成后权重是不变的，网络中可变的参数只有一直在更新的神经元状态和神经元输出。由于神经元随机更新，所以称此模型为离散随机型。当网络更新时，如果权重矩阵与非负对角线对称，则下面这个能量函数可以保证最小化，直到系统收敛到其稳定状态之一。

$$E=-\frac{1}{2}\sum_i\sum_j w_{ij}v_iv_j+\sum_{i=1}^{n}I_iv_i$$

连续型Hopfield网络

在生物系统中，由于神经元i的细胞膜输入电容 C_i、跨膜电阻R_i和确定阻抗$R_{ij}=W_{ij}^{-1}$，状态会滞后于其它神经元的瞬时输出v_i。神经元的输出将是0、1之间的连续值，而不是之前离散模型的二值。在这个模型中，使用如下的电阻电容微分方程来决定的更新速率:

$$v_i=f(u_i)=\frac{1+\tanh(u_i/u_0)}{2}$$

其中u_0为基准值，当$u_0\to\infty$时f_i成为硬限幅函数, 当$u_0\to 0$时f_i成为二值阈值函数。
整个电路系统的动态方程为：

$$C\frac{du_i}{dt}=-\frac{u_i}{R_i}+\sum_{j=1}^{n}W_{ij}V_j+I_i$$

若这个方程有解，则表示系统状态变化最终会趋于稳定。在对称连接和无自反馈的情况下，定义系统的能量函数为

$$E=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}W_{ij}V_iV_j-\sum_{i=1}^{N}V_iI_i$$

利用Hopfield网络这样的性质可以用来解决一些组合优化问题，把问题的目标函数和约束转化为网络的能量函数，问题的变量对应网络的状态，这样当网络的能量函数收敛于极小值时，网络的状态对应问题的一个最优解。

2.tsp问题

TSP问题就是在一城市集合$A,B,C,...$中找出一个最短且经过每个城市各一次并回到起点的路径。将TSP问题映射为神经网络动力系统可用以下步骤完成：

1) 将TSP问题的每一条可能路径用一换位矩阵表示，并给出相应的距离表示公式；

设问题中含有N个城市 ,用$N\times N$个神经元构成网络，称为换位矩阵。
d_{xy}为城市x与城市y之间的距离。
V_{x_i}为城市x的第i个神经元的状态。
w_{x_i,y_j} 为城市x的第i个神经元到城市y的第j个神经 元的连接权。

2) 找出一反应TSP约束优化问题的能量函数E；

每个城市只能被走过一次，因此换位矩阵每行能有一个1，每列只能有一个1；一共N个城市都要被访问，所以换位矩阵中元素1之和应为N。并且要求构建路径长度最短。按以上要求构建能量函数如下：

$$E=\frac{A}{2}\sum_{x}\sum_{i}\sum_{j}V_{xi}V_{xj}+\frac{B}{2}\sum_{i}\sum_{x}\sum_{y}V_{xi}V_{yi}+\frac{C}{2}\Big(\sum_{x}\sum_{i}V_{xi}-n\Big)^2+\frac{D}{2}\sum_{x}\sum_{y}\sum_{i}d_{xy}V_{xi}(V_{y,\ i+1}$$

3) 取极小值的神经网络连接权重矩阵和偏置参数。

反推出网络的权重矩阵为

$$W_{xi,yj}=-A\delta_{xy}(1-\delta_{ij})\ -B\delta_{ij}(1-\delta_{xy})\ -C\ -Dd_{xy}(\delta_{j,i+1}+\delta_{j,i-1})$$

其中$\delta_{ij}=1$当且仅当$i=j$

改进算法

能量函数的第三项仅仅在网络输出全为0时起作用，否则前两项已经保证了第三项成立，如果对能量函数的前两项作这样的修改：

$$\frac{A}{2}\sum_{x}(\sum_{i}V_{xi}-1)^2+\frac{B}{2}\sum_{i}(\sum_{x}V_{xi}-1)^2$$

那么第三项完全可以省去，于是Tsp的能量函数简化为：

$$E=\frac{A}{2}\sum_{x}(\sum_{i}V_{xi}-1)^2+\frac{B}{2}\sum_{i}(\sum_{x}V_{xi}-1)^2+\frac{D}{2}\sum_{x}\sum_{y}\sum_{i}d_{xy}V_{xi}V_{y,\ i+1}$$

由于行列的对称性取$A=B$，D项简化为$V_{y,i+1}$一项，方程式仍满足优化目标约束的要求。

推导出神经元（x，i）的动态方程如下：

$$\frac{du_{xi}}{dt} = -A(\sum_{j} V_{xj} - 1) - A(\sum_{y} V_{yi} - 1)^2 - D\sum_{y} d_{xy}V_{y,\ i+1}$$

网络权重为：

$$W_{xi,yj} = -A\delta_{xy} - A - \delta_{ij} - D_{xy} - \delta_{j,i+1}$$

3.实验结果

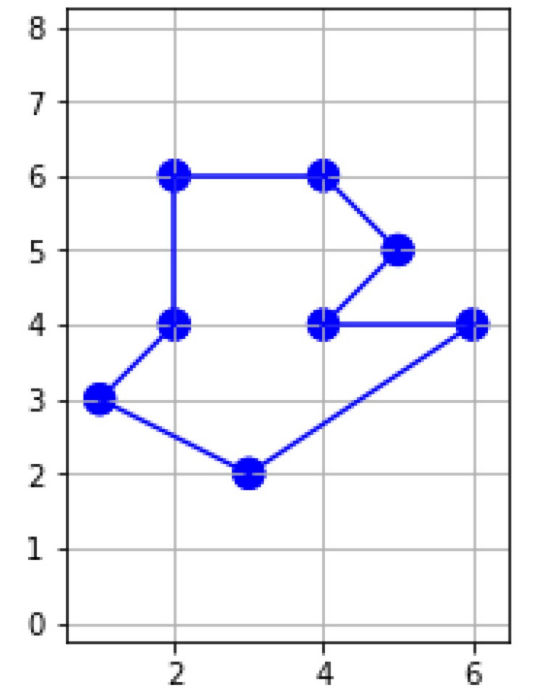
分别使用两个数据集进行测试。
迭代次数设置为10000
A设置为64
D设置为4
U_0 设置为0.009
得到的结果如下：

使用8个城市的数据集进行测试

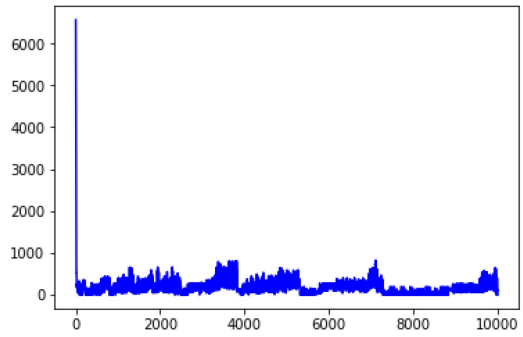
最终得到的最优路径为：

```
第75次迭代找到的次优解距离为：19.823028822792917，能量为：100.12970467723949，路径为：
0,1,3,4,6,7,5,2,0
第1148次迭代找到的次优解距离为：19.5121560482004，能量为：112.66306376935336，路径为：
4,0,3,7,1,2,5,6,4
第2483次迭代找到的次优解距离为：18.399582593338558，能量为：219.93872121701412，路径为：
0,1,7,2,6,4,5,3,0
第2677次迭代找到的次优解距离为：18.399582593338554，能量为：205.37276748769295，路径为：
0,1,7,2,6,5,4,3,0
第3357次迭代找到的次优解距离为：16.084259940083065，能量为：305.0646164462375，路径为：
0,1,2,7,6,5,4,3,0
```

可视化最优路径：



最优路径长度随迭代次数变化：

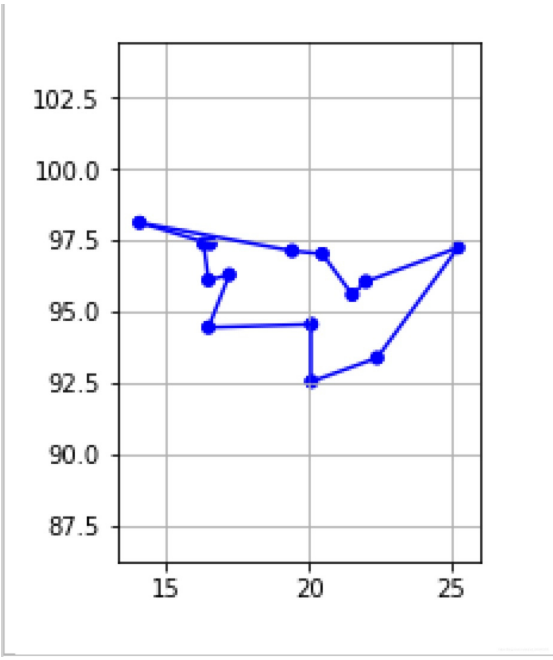


使用14个城市的数据集进行测试

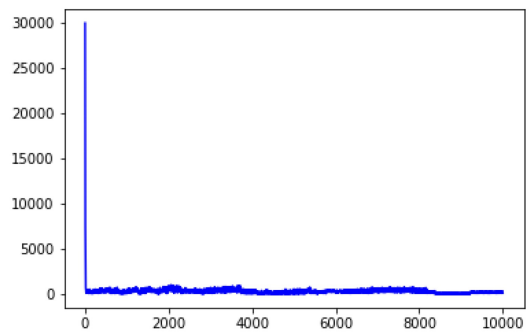
最终得到的最优路径为：

```
5,12,6,9,8,10,7,0,1,2,13,3,4,11,5
第4336次迭代找到的次优解距离为：33.1826580172479，能量为：189.85510382642255，路径为：
6,12,9,10,8,7,0,1,2,13,3,4,11,5,6
第5525次迭代找到的次优解距离为：33.1598560970073，能量为：258.0841686205351，路径为：
12,9,8,10,0,7,1,2,13,3,4,11,5,6,12
第5830次迭代找到的次优解距离为：32.504241715378974，能量为：281.6552715113886，路径为：
12,9,8,10,0,7,1,2,13,3,4,5,11,6,12
第5893次迭代找到的次优解距离为：32.173470500249095，能量为：286.75469812190613，路径为：
12,9,8,10,7,0,1,2,13,3,4,5,11,6,12
第9444次迭代找到的次优解距离为：32.12705156950196，能量为：253.80235837276402，路径为：
9,10,8,0,7,1,13,2,3,4,5,11,6,12,9
```

可视化最优路径：



最优路径长度随迭代次数变化：



二、遗传算法

1.算法原理

遗传算法是计算数学中用于解决最优化的搜索算法，是进化算法的一种。进化算法最初是借鉴了进化生物学中的一些现象而发展起来的，这些现象包括遗传、突变、自然选择以及杂交等。

遗传算法通常实现方式为一种计算机模拟。对于一个最优化问题，一定数量的候选解（称为个体）的抽象表示（称为染色体）的种群向更好的解进化。传统上，解用二进制表示（即0和1的串），但也可以用其他表示方法。进化从完全随机个体的种群开始，之后一代一代发生。在每一代中，整个种群的适应度被评价，从当前种群中随机地选择多个个体（基于它们的适应度），通过自然选择和突变产生新的生命种群，该种群在算法的下一次迭代中成为当前种群。

遗传算法的具体流程为：

1. 初始化
2. 循环:
3. 选择父种群
4. 交叉
5. 突变
6. 评价适应度
7. 选择子种群
8. 直到终止条件满足

2.tsp问题

基因表示

在tsp问题中，设有n个城市编号$0, 1, ..., n-1$，那么一个解即为$0, 1, ..., n-1$的一个全排列。

交叉/重组

首先设定交叉/重组可能性$recombination_{prob}$，然后以这个概率进行重组。
重组的具体实现为：挑选两个父代基因，随机选择他们的一部分进行交换。由于基因表示遵循一定规律，重组操作可能造成冲突，还要交换一部分重组后的基因，消去冲突。

突变

首先设定突变可能性$mutation_{prob}$，然后以这个概率进行突变。
突变的具体实现为：随机挑选该基因的两个位，进行交换。

适应度评价

设两个城市之间i, j的距离为d_{ij}，那么一条路线$x_1, x_2, ..., x_n$所需的距离为：

$$length = \sum_{n} d_{x_i, x_i+1} + d_{x_n, x_1}$$

为了使距离最小，适应度函数可以设置为：

$$fitness = \frac{1}{length}$$

选择策略

选择采用轮盘赌选择：

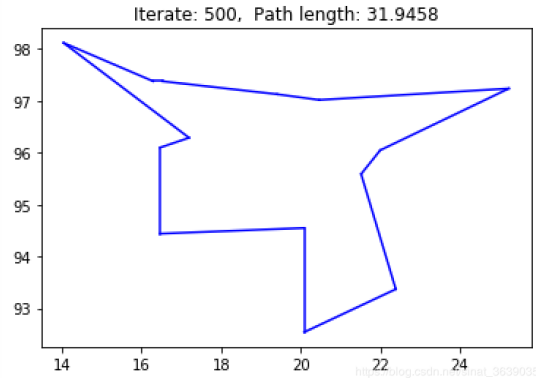
$$Prob(h_i) = \frac{fitness_{h_i}}{\sum_n fitness_{h_i}}$$

3.实验结果

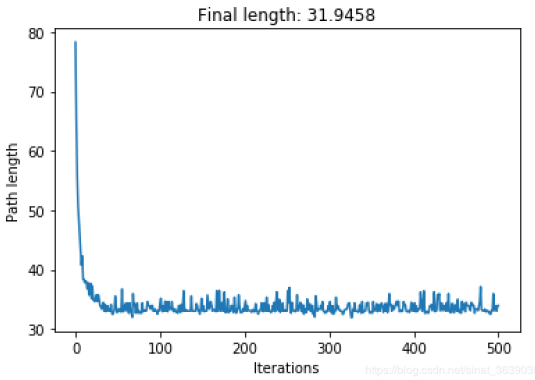
最大进化代数设置为500
种群数目500
重组概率0.9
变异概率1.0
在14、29、38、52个城市的数据集进行测试：

使用14个城市的数据集测试

最终得到的最优路径为：

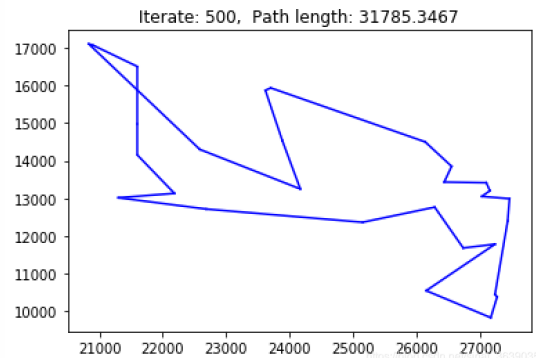


最优路径长度随迭代次数变化：

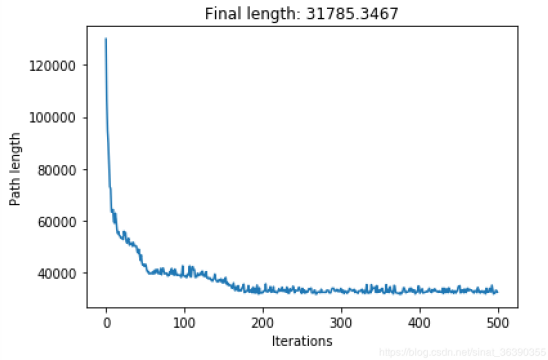


使用29个城市的数据集测试

最终得到的最优路径为：

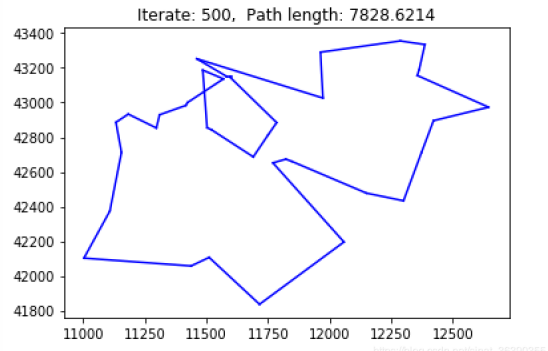


最优路径长度随迭代次数变化：

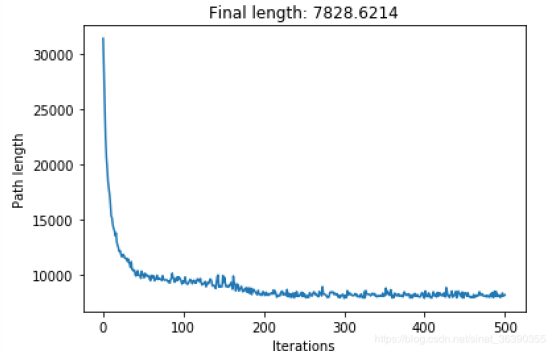


使用38个城市的数据集测试

最终得到的最优路径为：

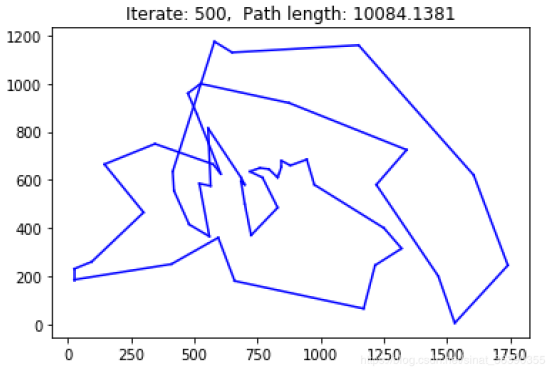


最优路径长度随迭代次数变化：

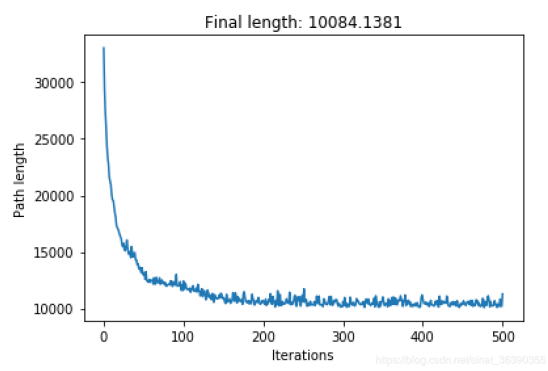


使用52个城市的数据集测试

最终得到的最优路径为：



最优路径长度随迭代次数变化：



三、蚁群算法

1.算法原理

蚁群算法是一种用来寻找优化路径的概率型算法。它由Marco Dorigo于1992年在他的博士论文中提出，其灵感来源于蚂蚁在寻找食物过程中发现路径的行为。
蚂蚁找到最短路径要归功于信息素和环境，假设有两条路可从蚁窝通向食物，开始时两条路上的蚂蚁数量差不多：当蚂蚁到达终点之后会立即返回，距离短的路上的蚂蚁往返一次时间短，重复频率快，在单位时间里往返蚂蚁的数目就多，留下的信息素也多，会吸引更多蚂蚁过来，会留下更多信息素。而距离长的路正相反，因此越来越多的蚂蚁聚集到最短路径上来。
蚁群算法就是根据这一特点，通过模仿蚂蚁的行为，从而实现寻优。这种算法有别于传统编程模式，其优势在于，避免了冗长的编程和筹划，程序本身是基于一定规则的随机运行来寻找最佳配置。也就是说，当程序最开始找到目标的时候，路径几乎不可能是最优的，甚至可能是包含了无数错误的选择而极度冗长的。但是，程序可以通过蚂蚁寻找食物的时候的信息素原理，不断地去修正原来的路线，使整个路线越来越短，也就是说，程序执行的时间越长，所获得的路径就越可能接近最优路径。

蚁群算法的具体流程为：

1. 初始化
2. 循环：

3.   每只蚂蚁根据选择策略选择最佳路径

4.   更新信息素

5. 直到终止条件满足

2.tsp问题

选择策略

每次迭代中，位于i城市的蚂蚁选择j城市为下一个城市的概率为

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in N}(\tau_{il})^{\alpha}(\eta_{il})^{\beta}}$$

其中τ_{ij}表示路径(i,j)上信息素的浓度，α为信息启发式因子，表示轨迹的相对重要性，反映了蚂蚁在运动过程中积累的信息在蚂蚁运动时所起的作用，其值越大，则该蚂蚁越倾向于选择其它蚂蚁经过的路径，蚂蚁之间的协作性越强，即算法的exploration能力更强。

η_{ij}表示路径(i,j)的长度分之一，即$\frac{1}{d_{ij}}$，β为期望启发式因子，表示能见度的相对重要性，反映蚂蚁在运动过程中启发信息在蚂蚁选择路径中的受重视程度，其值越大，则该状态状态转移概率越接近于贪心规

则，即算法的exploitation能力更强。

更新策略

$t+1$时刻路径(i,j)上的信息素浓度的更新为

$$\tau_{ij}(t+1)=\rho\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t)$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t)$$

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}Q/L^{k} & (i,j)\in Path(k)\\ 0 & else\end{cases}$$

其中ρ表示信息素挥发系数；$\Delta\tau_{ij}(t)$表示本次循环中路径(i,j)上的信息素增量。

Q表示信息素强度，它在一定程度上影响算法的收敛速度；$Path(k)$表示第k只蚂蚁所走过的路径的集合；L^{k}表示第k只蚂蚁在本次迭代中所走路径的长度。

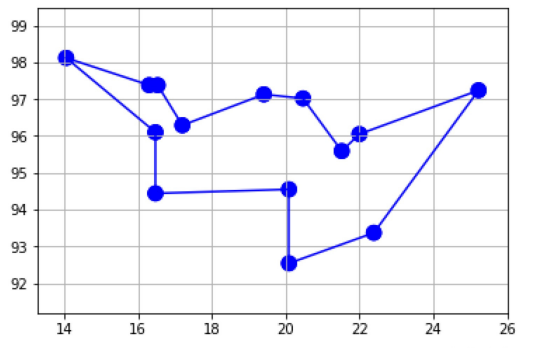
3.实验结果

蚂蚁数量设置为50只
迭代次数设置为200
$\alpha=1$
$\beta=2$
$\rho=0.5$
$Q=100$
在14、29、38、52个城市的数据集进行测试：

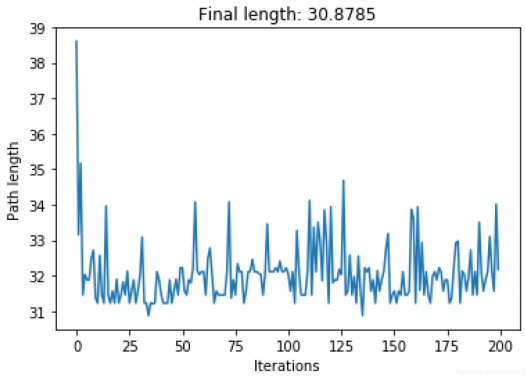
使用14个城市的数据集测试

最终得到的最优路径为：

```
通过200代的蚁群寻找，最短路径出现在第35代，路径为：
8,10,7,12,6,11,5,4,3,2,13,1,0,9
最短路径长度为：30.878503892588
```



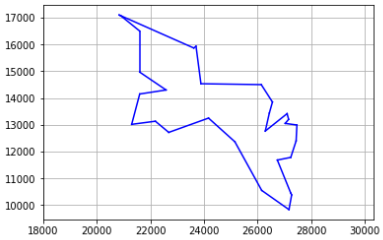
最优路径长度随迭代次数变化：



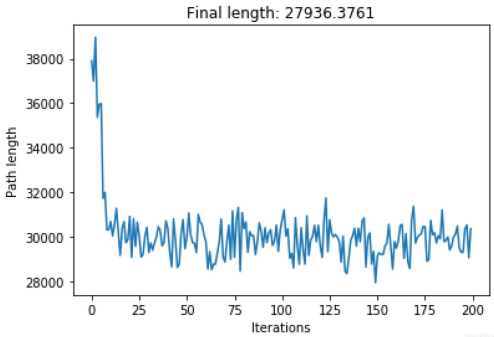
使用29个城市的数据集测试

最终得到的最优路径为：

```
通过200代的蚁群寻找，最短路径出现在第150代，路径为：
21,22,20,28,27,25,19,24,26,23,15,13,12,8,6,2,3,7,4,5,1,0,9,10,11,14,18,17,16
最短路径长度为：27936.376114029175
```



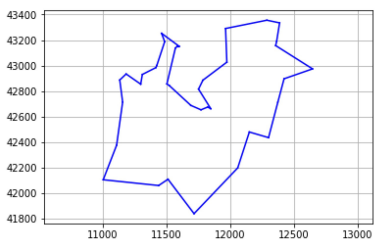
最优路径长度随迭代次数变化：



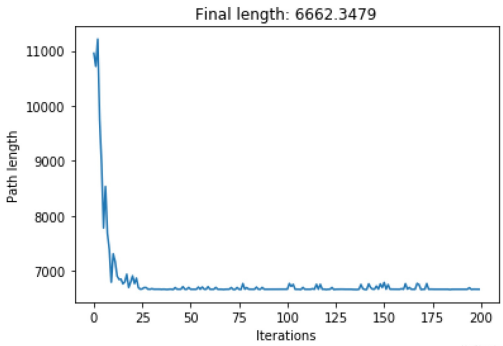
使用38个城市的数据集测试

最终得到的最优路径为：

```
通过200代的蚁群寻找，最短路径出现在第125代，路径为：
33,32,37,36,34,31,29,28,20,13,9,0,1,3,2,4,5,6,7,8,11,10,18,17,16,15,12,14,19,
22,24,25,21,23,27,26,30,35
最短路径长度为：6662.3478701540225
```



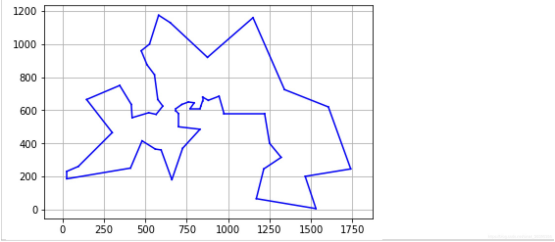
最优路径长度随迭代次数变化：



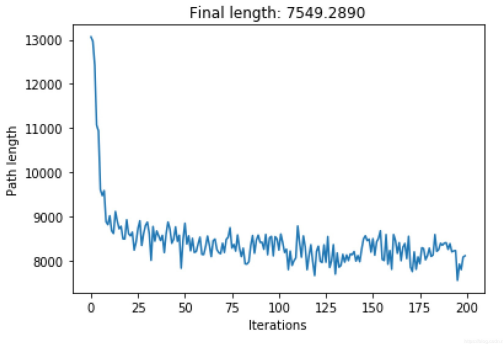
使用52个城市的数据集测试

最终得到的最优路径为：

```
通过200代的蚁群寻找，最短路径出现在第196代，路径为：
28,15,45,43,33,34,35,38,39,37,36,47,23,4,14,5,3,24,11,27,26,25,46,13,12,51,10
,50,32,42,9,8,7,40,18,44,31,48,0,21,30,17,2,16,20,41,6,1,29,22,19,49
最短路径长度为：7549.289019371725
```



最优路径长度随迭代次数变化：



四、搜索算法

1.搜索问题和搜索算法的统一表述

搜索问题的目标就是在可接受的时间或资源花费下，在问题的解空间中找到最优解或者可行解。
搜索算法就是从一个解或一组解，以某种方式变化到另一个解或另一组解，直至找到最优解或者终止条件满足为止。
其中主要包含四大要素：

初始解的产生

首先需要确定针对该问题解的编码方式，初始解可以是一个解或者一组解。

终止条件的设置

终止准则是判断算法是否收敛的标准，决定了算法的最终优化性能。算法收敛理论为终止理论提供了明确的设计方案，但是基于理论分析所得的收敛准则往往很苛刻的，甚至难以应用。实际设计时，应兼顾算法

的优化质量和搜索效率等多方面性能，或根据问题需要着重强调算法的某方面性能。

解的适应度计算

解的适应度计算与具体问题相关。

选择下一组解的启发式规则

不同搜索方式的不同主要在于启发式规则的不同。比如爬山法是基于局部贪婪的搜索机制，而模拟退火算法则是基于概率分布，遗传算法使用了重组和突变，蚁群算法采用基于信息素的策略。

2.搜索算法的 Exploration 与 Exploitation 能力

Exploitation：根据当前信息做出的最佳的决策，重在深度。
Exploration：探索未知的领域，重在广度。

当搜索空间太大，在给定的时间内无法遍历空间。如果在一小块区域仔细考察下去，得到这块区域的最佳解，就是Exloitation；如果在整个空间中挑最有可能包含最优解的区域，就是exploration。
如果模型的exploitation能力太强，那么模型比较容易陷入局部最优，如果exploration能力太强，模型收敛速度太慢。平衡exploitation和exploration的目的就是获得一种长期收益最高的策略，这个过程可能对短期收益有损失。
在模型拥有的时间和计算资源一定的情况下，就需要平衡分配到exploitation和exploration的资源，以求达到更好的效果。具体可以通过调整参数的方式。如蚁群算法，若要增强exploitation能力，可以提高β值；若要增强exploration能力，可以提高α值。在遗传算法中，若要增强exploitation能力，可以提高突变概率降低重组概率，或者提高子代选择更高适应度的概率；若要增强exploration能力，则可以提高重组概率降低突变概率，或者以一定概率选择较差的子代。
达到exploitation和exploration的平衡没有具体的准则，还是以模型在具体数据上的表现为准。