作业二报告

> 姓名：徐恒达
>
> 学号：1120151811

一、Hopfield网络

1. 算法原理

1982年，Hopfield提出了可用作联想存储器的互连网络，这个网络称为Hopfield网络模型，也称Hopfield模型。Hopfield神经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又反馈到其输入端，所以Hopfield网络在输入的激励下，会产生不断的状态变化。当有输入之后，可以求取出Hopfield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Hopfield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Hopfield网络就会输出一个稳定的恒值。对于一个Hopfield网络来说，关键是在于确定它在稳定条件下的权系数。

Hopfield 神经网络是最典型的反馈神经网络。从系统的观点看，它是一个非线性动力学系统；从计算角度看，它比前向神经网络具有更强的计算能力。该系统可分为连续型和离散型两种。下面从神经元输入输出关系，运动方程和网络能量函数三个方面的数学描述来描述这两种模型。

离散Hopfield网络

离散神经网络是离散时间系统。离散模型中每个神经元的输入u_i和输出v_i满足阶跃函数

$$v_i = f(u_i)$$

整个网络的状态矢量$v(t) \in 1, 0^n$，这样t时刻节点i即第i个神经元的状态$v_i(t)$的下一状态可由下式确定

$$V_i(t+1) = \begin{cases} 1, & H_i(t) > 0 \\ 0, & H_i(t) \leqslant 0 \end{cases}$$

其中

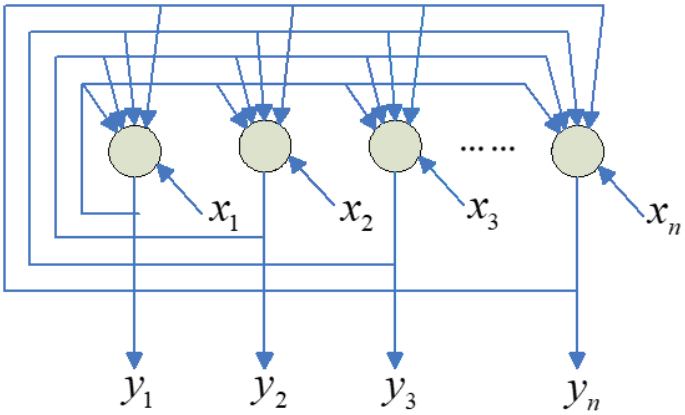
$$H_i(t) = \sum_{j=1}^{n} w_{ij} v_j(t) - \theta_i$$

为第i个神经元的输入强度，θ_i为第i个神经元的阈值。

所对应的神经网络的能量函数为

$$E=-\frac{1}{2}\sum_{i}\sum_{j}w_{ij}v_iv_j+\sum_{i=1}^{n}\theta_iv_i$$

如果网络工作在串行模式，w矩阵对称且对角元素为0，则能量函数总会收敛于一个稳定状态。



连续Hopfield网络

连续 Hopfield 神经网络中第i个神经元输出电压v_i与输入电压$u_i之间的非线性关系可以表示为

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

其中u_0为基准值，当$u_0\to\infty$时f_i成为硬限幅函数。

N个神经元相互作用的动力学性质可以用下面的微分方程描述

$$C\frac{du_i}{dt}=-\frac{u_i}{R}+\sum_{j=1}^{n}w_{ij}v_j+I_i$$

$$v_j=f(u_j)$$

其中R和C并联，用来模拟生物神经元输出的时间常数，w_{ij}模拟生物神经元之间互联的突触特性。

连续Hopfield神经网络的能量函数为

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

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，即所谓的旅行商问题(Traveling Salesman Problem) 。问题的提法是：在N个城市中各经历一次后 回到出发点 ,使所经过的路程最短。其不同选择方案有$N!/2N$种,在城市数较少的情况下可以用枚举等方法 , 但如果城市数量较大时， 使用枚举法求解就要考虑的情况是数量级， 计算量之大是不可想象的。将Hopfield网络应用于求解TSP问题，效果是显著的。

解决问题需要说明的是以下几个量:

- 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。
- N个城市间存在$N!/2N$条可能路径。
- d_{xy}为城市x与城市y之间的距离。
- V_{xi}为城市x的第i个神经元的状态。
- $w_{xi,yj}$ 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。

网络的能量函数要满足下面四方面的要求

- 换位矩阵每行能有一个1
- 换位矩阵每列只能有一个1
- 换位矩阵中元素1之和应为N
- 所构造函数的极小值对应于最短路径

我们构造出与TSP相对应的计算能量函数为

$$
\begin{aligned}
E=&\frac{A}{2}\sum_x\sum_i\sum_j V_{xi}V_{xj}+\frac{B}{2}\sum_i\sum_x\sum_y V_{xi}V_{yi}+\\
&\frac{C}{2}\Big(\sum_x\sum_i V_{xi}-n\Big)^2+\frac{D}{2}\sum_x\sum_y\sum_i d_{xy}V_{xi}(V_{y,i+1}+V_{y,i-1})
\end{aligned}
$$

反推出网络的权重矩阵为

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

其中$\delta_{ij}=1$当且仅当$i=j$

网络的迭代公式为

$$\begin{aligned}\frac{u_{xi}}{dt} = &-\frac{u_{xi}}{\tau} - A\sum_j V_{xj} \\ &- B\sum_y V_{yi} \\ &- C\Big(\sum_x\sum_j V_{xj} - n\Big) \\ &- D\sum_j dxy(V_{y,i+1} + V_{y,i-1})\end{aligned}$$

其中A、B、C、D为惩罚因子，Hopfield 1985年的论文中给出的系数参考值为$A = B = 500$, $C = 200$, $D = 500$。

为了使系统开始收敛．在每个神经元初始值相同的基础上加入随机干扰

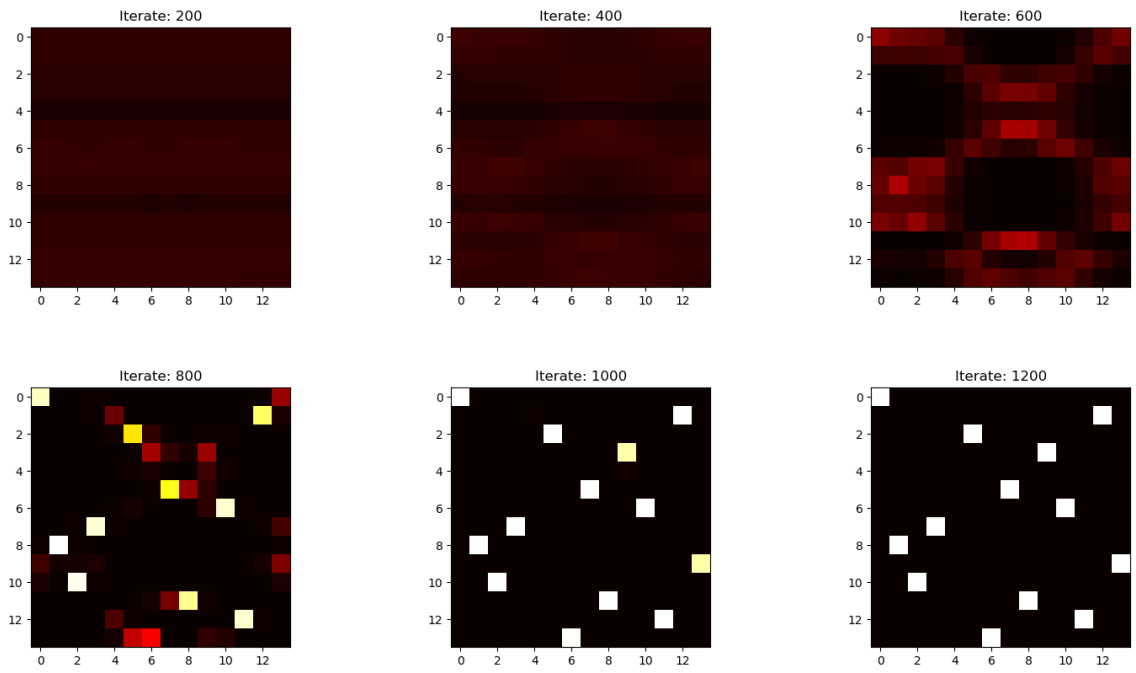
$$u_{x,i} = u_{00} + \delta u_{x,i}$$

其中$-0.1u_0 \leqslant \delta u_{xi} \leqslant 0.1u_0$。

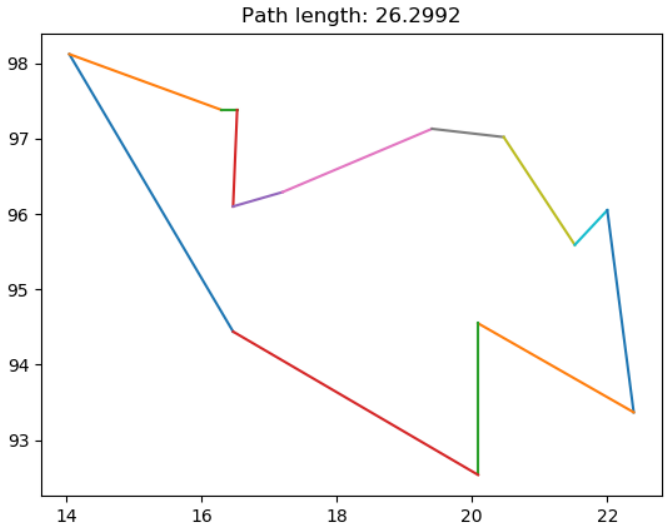
有了以上的条件，整个Hopfield网络的条件就完备了，由此就可以构建出一个网络进行迭代计算。

3. 实验效果

在14个点的burma14.tsp数据集上进行测试，数据文件可在附带的文件中找到。每迭代200次对神经元矩阵进行可视化展示。可以看出，在迭代1200次之后网络收敛于一个稳定的状态，每行每列均只有一个神经元被激活，总共有n个神经元激活。



将迭代得到的路径画在一张图上，如下所示，每次迭代基本上可以得到最优解或近似最优解。



二、遗传算法GA

1. 算法原理

遗传算法(GA)是计算数学中用于解决最佳化的搜索算法，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。

在遗传算法里，优化问题的解被称为个体，它表示为一个变量序列，叫做染色体或者基因串。染色体一般被表达为简单的字符串或数字串，不过也有其他的依赖于特殊问题的表示方法适用，这一过程称为编码。首先，算法随机生成一定数量的个体，有时候操作者也可以干预这个随机产生过程，以提高初始种群的质量。在每一代中，都会评价每一个体，并通过计算适应度函数得到适应度数值。按照适应度排序种群个体，适应度高的在前面。这里的“高”是相对于初始的种群的低适应度而言。

下一步是产生下一代个体并组成种群。这个过程是通过选择和繁殖完成，其中繁殖包括基因重组和突变。选择则是根据新个体的适应度进行，但同时不意味着完全以适应度高低为导向，因为单纯选择适应度高的个体将可能导致算法快速收敛到局部最优解而非全局最优解。作为折中，遗传算法依据原则：适应度越高，被选择的机会越高，而适应度低的，被选择的机会就低。初始的数据可以通过这样的选择过程组成一个相对优化的群体。之后，被选择的个体进入重组过程。一般的遗传算法都有一个重组概率，这个交配概率反映两个被选中的个体进行交配的概率。每两个个体通过交配产生两个新个体，代替原来的“老”个体，而不交配的

个体则保持不变。交配父母的染色体相互交换，从而产生两个新的染色体，第一个个体前半段是父亲的染色体，后半段是母亲的，第二个个体则正好相反。不过这里的半段并不是真正的一半，这个位置叫做交配点，也是随机产生的，可以是染色体的任意位置。再下一步是突变，通过突变产生新的“子”个体。一般遗传算法都有一个固定的突变常数，这代表变异发生的概率。根据这个概率，新个体的染色体随机的突变。

经过这一系列的过程（选择、交叉和突变），产生的新一代个体不同于初始的一代，并一代一代向增加整体适应度的方向发展，因为总是更常选择最好的个体产生下一代，而适应度低的个体逐渐被淘汰掉。这样的过程不断的重复：评价每个个体，计算适应度，两两交配，然后突变，产生第三代。周而复始，直到终止条件满足为止。

2. TSP问题

基因表示

在TSP问题中，问题的一个解是所有城市的一个全排列，就用这个全排列来作为基因序列。设有共n个城市，编号为0到$n-1$，则每一个基因就是0到$n-1$的一个全排列。其后的评价、选择和基因操作全全部以全排列为操作对象，设n个数据的全排列构成的集合为$Perm(n)$，则基因$gene$的取值为

$$gene \in Perm(n)$$

适应度评价

每一个全排列代表一种n个城市的周游策略，周游的总路径越长则这个解的效果越差，总路径越短则效果越好，评估效果与路径长度成负相关。因此选择总路径的倒数作为基因的竞争力度量。设路径长度为$length$，则适应度$fitness$为

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

选择策略

为了对选择的力度有所把控，引入参数α，定义适应度的α次方作为选择基因时每个基因的权重，选择时为了概率之和为1，要对权重进行归一化处理。设$sum(Genes)$表示基因集合$Genes$中左右基因权重之和，则基因$gene_i$被选中的概率为

$$P(gene_i) = \left(\frac{1}{length}\right)^{\alpha} \cdot \frac{1}{sum(Genes)}$$

重组

每次从所有基因中依概率选则两个基因进行重组，重组策略为随机选择两个分段点将每个基因分为3段，交换中间的一段。这样重组的效果是有概率把一个基因中“好”的那一部分排列移植到另一段基因当中。当然，按这样的交叉策略处理后的两个基因很有可能是不合法的，于是引入如下图所示的调整方法，次方法可以保证在不超过n次调整之后使一段基因重新合法化。

本实例中交叉采用部分匹配交叉策略，其基本实现的步骤是：

步骤1： 随机选取两个交叉点

步骤2： 将两交叉点中间的基因段互换

步骤3：将互换的基因段以外的部分中与互换后基因段中元素冲突的用另一父代的相应位置代替，直到没有冲突。

本例中路径实例如图交叉点为2，7，交换匹配段后A中冲突的有7 、6、5，在B的匹配段中找出与A匹配段中对应位置的值7-3，6-0，5-4，继续检测冲突直到没有冲突。对B 做同样操作，得到最后结果。

（图 4）

突变

突变发生在重组之后的每一条基因上，突变定义为随机选择两个位点，交换这两个位点上的值。突变操作后的基因仍然是一个合法的基因。

迭代策略

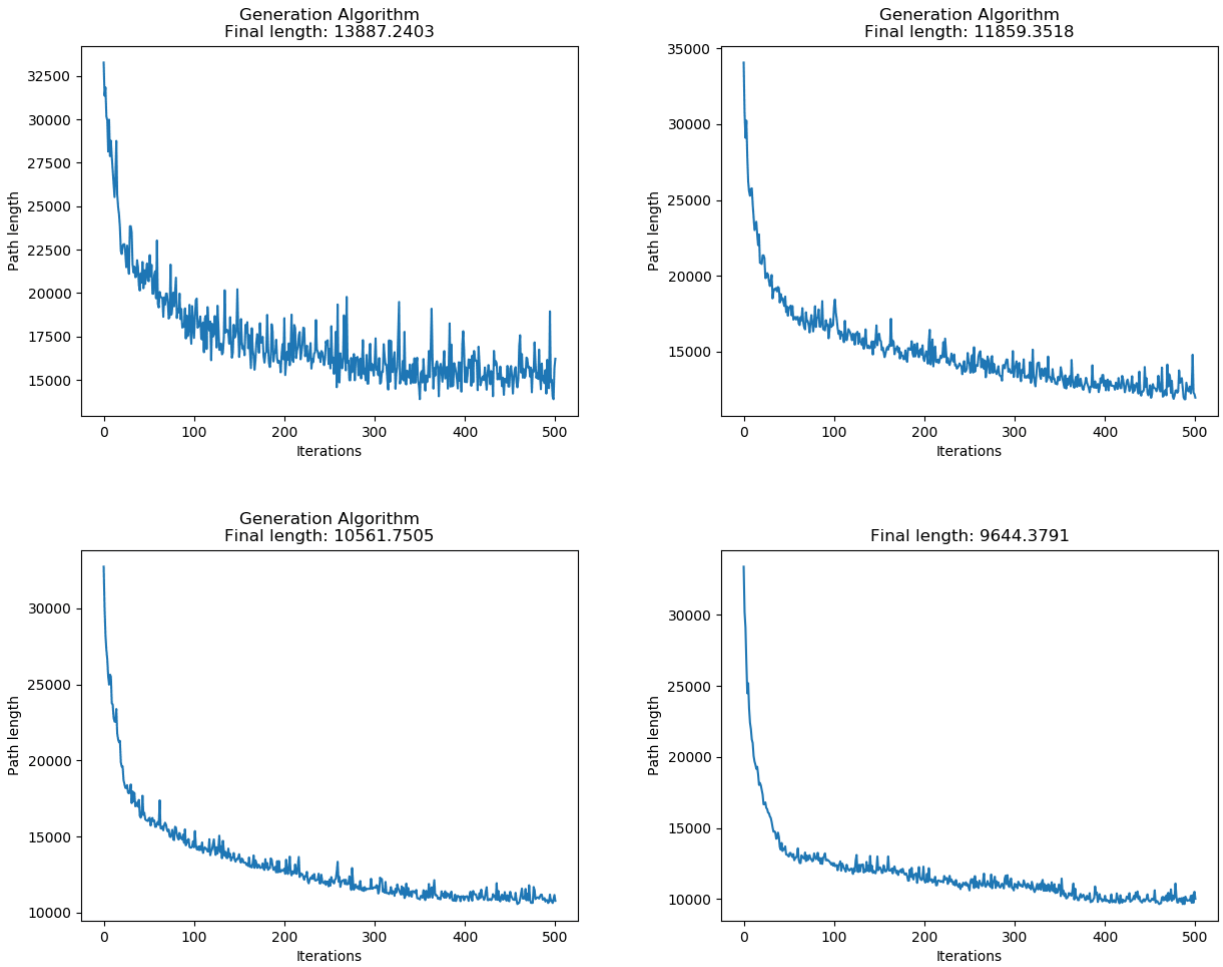
设种群规模为n，每一轮迭代新产生的后代数量为m，则进行$m/2$次选择，每次选出2个基因依概率进行重组和突变，然后把现有的$n+m$个个体混合，统一进行选择，依概率选出n个优胜个体进入下一轮竞争。

3. 实验效果

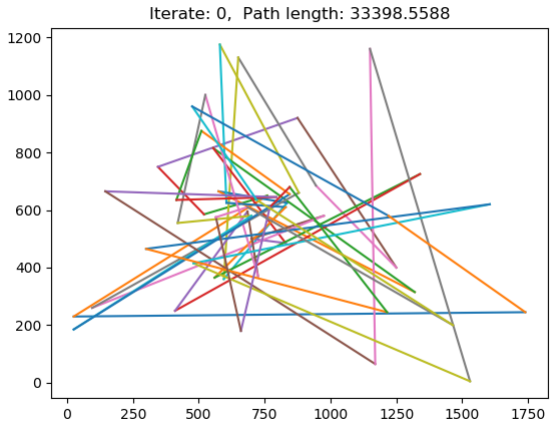
在52个点的berlin52.tsp数据集上进行测试，数据文件可在附带的文件中找到。

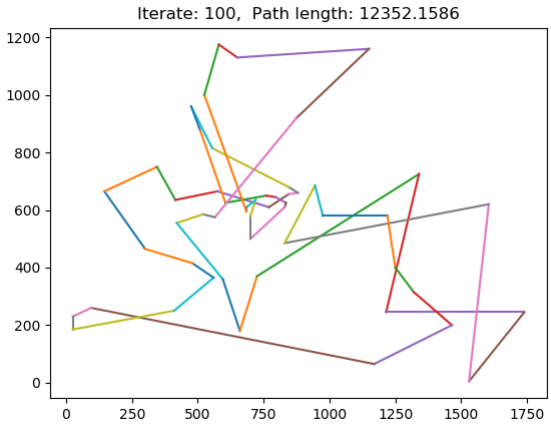
核心的参数有三个，分别是与选择力度有关的α，重组概率P_c和突变概率P_m。α越大，则优秀个体与劣质个被选中的概率差距越大，选择越倾向于优秀个体，算法的exploitation能力越强，P_c和P_m越大，则基因发生变化的可能性越高，算法的exploration能力约强。

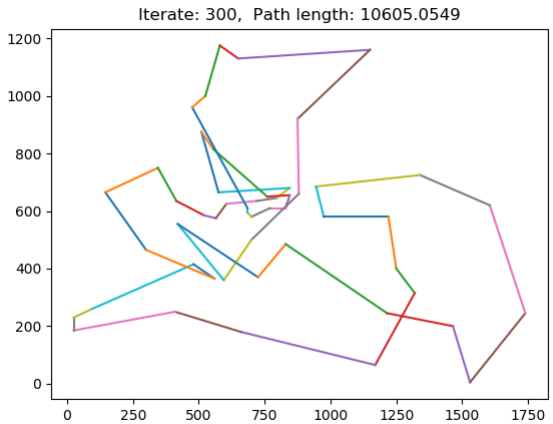
若将每一次迭代中最优秀个体，也就是总路径最短的个体的路径长度记下来绘制成曲线，则算法的exploration能力越强表现为曲线却粗糙，“毛刺”越多，exploitation能力越强则表现为曲线越平滑。一下四幅图为不同的参数设置下曲线由粗糙到平滑的变化。

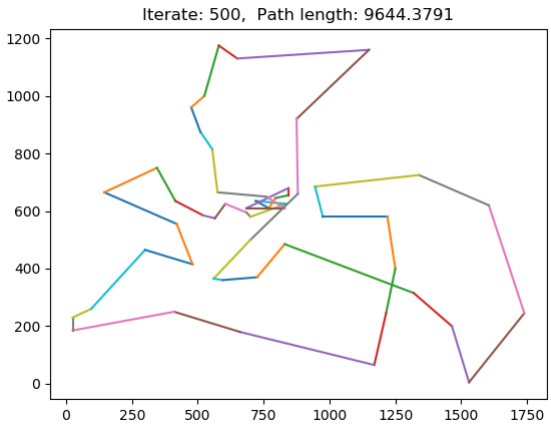


最终选择种群数量$n=100$，每轮迭代产生的后代数量$m=100$，选择力度$\alpha=50$，交叉概率$P_c=0.9$，突变概率$P_m=1.0$进行测试，可以看到在迭代500次时算法可以找到一个比较好的解。









三、蚁群算法ACO

1. 算法描述

蚁群算法(ACO)是一种用来在图中寻找优化路径的机率型算法。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。

蚂蚁在路径上前进时会根据前边走过的蚂蚁所留下的分泌物选择其要走的路径。其选择一条路径的概率与该路径上分泌物的强度成正比。因此，由大量蚂蚁组成的群体的集体行为实际上构成一种学习信息的正反馈现象：某一条路径走过的蚂蚁越多，后面的蚂蚁选择该路径的可能性就越大。蚂蚁的个体间通过这种信息的交流寻求通向食物的最短路径。

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。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，最终找到最佳路线。

这种优化过程的本质在于：

- 选择机制：信息素越多的路径，被选择的概率越大。

- 更新机制：路径上面的信息素会随蚂蚁的经过而增长，而且同时也随时间的推移逐渐挥发消失。
- 协调机制：蚂蚁间实际上是通过分泌物来互相通信、协同工作的。通过个体之间的信息交流与相互协作最终找到最优解，使它具有很强的发现较优解的能力。
- 出错机制：显然如果蚂蚁都往信息素多的地方移动，会导致局部最优解的问题。可是，总有些具有叛逆精神的蚂蚁，会不往信息素较多的地方移动，从而可以跳出局部最优解，找到全局的最优解。

2. TSP问题

初始化每条路径上的信息素为0，随着算法的迭代，蚂蚁不断地在路径上留下信息素，同时已有的信息素也会随着时间的流逝而蒸发。

选择策略

每次迭代中，一直位于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，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

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

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

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

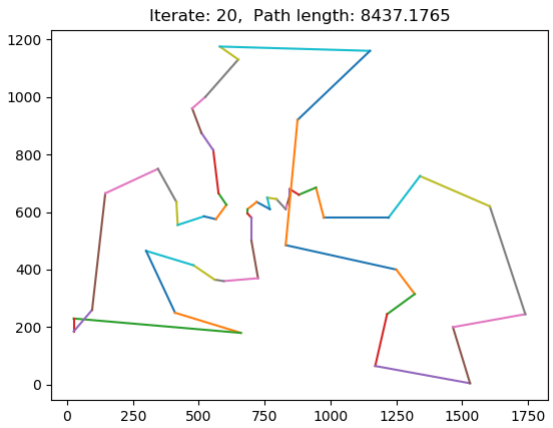
其中ρ表示信息素挥发系数；$\Delta\tau_{ij}(t)$表示本次循环中路径(i,j)上的信息素增量。

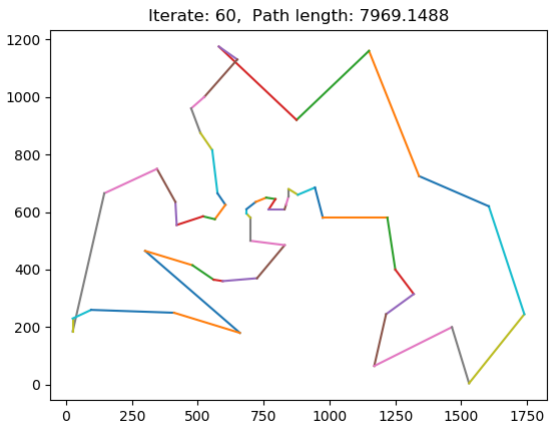
Q表示信息素强度，它在一定程度上影响算法的收敛速度；$Path(k)$表示第k只蚂蚁所走过的路径的集合；L^k表示第k只蚂蚁在本次迭代中所走路径的长度。

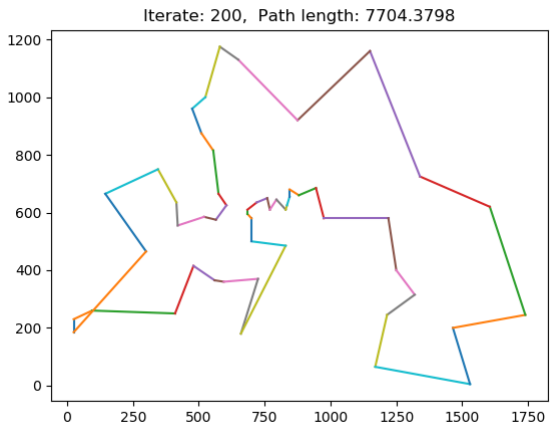
3. 实验效果

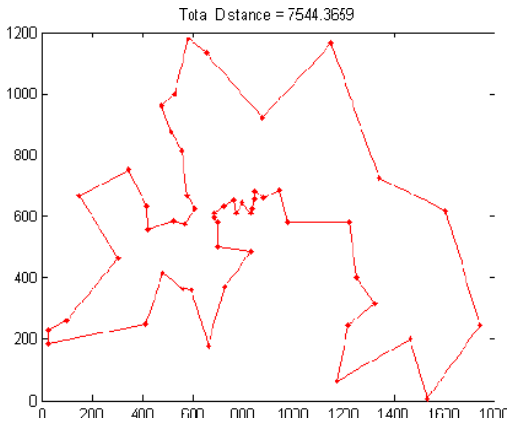
与GA算法一样在52个点的berlin52.tsp数据集上进行测试。蚂蚁数量设为$n=50$只，$\alpha=1$，$\beta=2$，$\rho=0.5$，$Q=100$，迭代200轮，记录中间的路径长度和每一轮迭代中的最短路径。

下面前3辐图分别取自迭代20次、迭代60次和迭代200次时的最短路径，可以看出，蚁群算法优化的效果非常好，迭代不多次后产生的解的效果已经非常好，非常接近最优解。其中第4幅图是berlin52.tsp数据集的最优解。

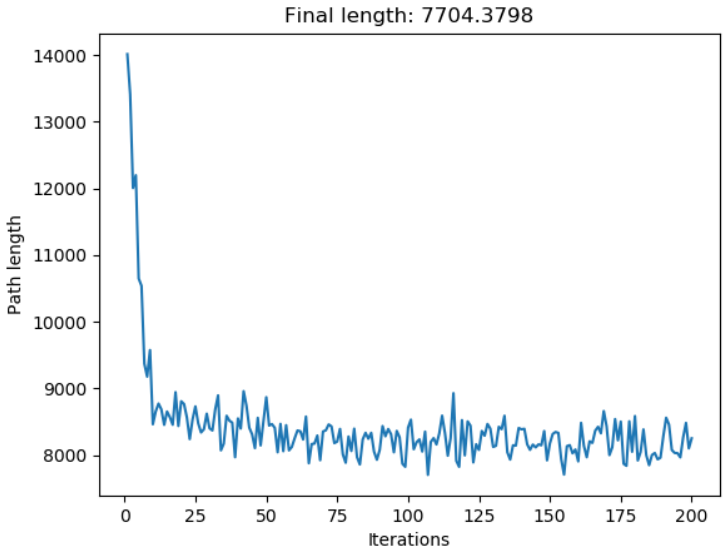








将每一轮迭代中产生的最短路径绘制成折线图如下图所示，可以看出蚁群算法优化TSP问题的效果非常好。



四、搜索算法

1. 概述

搜索算法是一类可以有效解决优化问题的通用算法。它的基本原理是在临近解中迭代，使目标函数逐步优化，直至不能再优化为止。

搜索算法可以这样描述：假设问题的解空间表示为S，局部搜索算法从一个初始解i§in 开始，然后根据具体问题定义的具体邻域结构，在当前解i的邻域S_i内按一定规则找到一个新解，再用这个新解取代$_i成为当前解，判断是否满足算法结束条件，如果不满足再对当前解继续使用算法，如果满足则算法结束，当前解就作为算法的最终解。

局部搜索算法具有以下特点：

- 算法结构通用易实现。只要定义好具体问题相关的邻域，就能有效的求解该问题。
- 算法性能和邻域的定义以及初始状态有关。邻域定义的不同或初始状态选取的不同会对算法的性能产生决定性的影响。
- 算法的局部优化特性。算法容易陷入局部最优解，达到全局最优比较困难。

局部搜索算法主要包含五大要素：

- 目标函数：用来判断解的优劣。
- 邻域的定义：根据不同问题，有着不同的邻域定义。
- 初始解的产生规则
- 新解的产生和接受规则
- 算法终止准则

2. 统一框架

局部搜索算法是对一类算法的统称，符合其框架的算法很多，但在优化流程上呈现出很大的共性。我们可以将局部搜索算法的统一框架描述为：算法从一个或若干个初始解出发，在算法参数控制下由当前状态的邻域中产生若干个候选解，并以某种策略在候选解中确定新的当前解。伴随控制参数的调节，重复执行上述搜索过程，直至满足算法终止准则，结束搜索过程并输出优化结果。

在具体设计算法时，需要考虑以下几个方面：

搜索机制的选择

搜索机制是构造算法框架和实现优化的关键，是决定算法搜索行为的根本点。比如爬山法是基于局部贪婪的搜索机制，而模拟退火算法时基于概率分布的搜索机制，禁忌搜索采用避免迂回的策略进行搜索。

邻域函数的设计

邻域函数决定了邻域结构和邻域解的产生方式。算法对问题解的不同描述方式，会直接影响邻域函数的设计，进而影响算法的搜索行为。同时，即使在编码机制确定的情况下，邻域结构也可以采用不同的形式，以考虑新状态产生的可行性、合法性和对搜索效率的影响。在确定邻域结构后，当前状态邻域中候选解的产生方式即可以是确定的，也可以是随机性地。

状态更新方式的设计

更新方式是指以何种策略在新旧状态中确定新的当前状态，是决定算法整体优化特性的关键步骤之一。基于确定性地状态更新方式的搜索，容易陷入局部最优；而随机性地状态更新方式，尤其是概率性劣向转移，往往取得较好的全局优化解，但是计算时间也比较长。

参数的控制方式

控制参数是决定算法搜索进程和行为的又一关键因素。合适的参数控制有助于增强算法在邻域中的优化能力和效率，同时也必须以一定的准则和方式进行修改以适应算法性能的动态变化。

算法终止准则的设计

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。