

第十二届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

第十二届“认证杯”数学中国

数学建模网络挑战赛 承 诺 书

我们仔细阅读了第十二届“认证杯”数学中国数学建模网络挑战赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

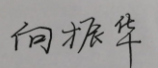
我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们接受相应处理结果。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

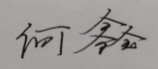
我们的参赛队号为：1928

参赛队员（签名）：

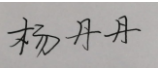
队员 1:



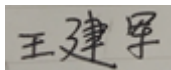
队员 2:



队员 3:



参赛队教练员（签名）：



参赛队伍组别（例如本科组）：本科组

第十二届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

第十二届“认证杯”数学中国

数学建模网络挑战赛 编号专用页

参赛队伍的参赛队号：（请各个参赛队提前填写好）：1928

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

第十二届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

2019 年第十二届“认证杯”数学中国 数学建模网络挑战赛第二阶段论文

题 目 基于改进混合算法的 *HMM* 多重序列比对
关 键 词 隐马尔可夫 遗传蚁群混合 等概率对数评分 相似度搜寻

摘 要：

已知发现了一种未知语言，目前只知道其文字是由 20 个字母构成。在已获得只有字母，但没有标点符号和空格的序列文本的条件下，本文建立了基于改进混合算法的 *HMM* 多重序列比对模型来研究这种文字。

针对问题一，基于第一阶段的隐马尔可夫模型上，对字符的匹配状态增加了“插入”和“删除”状态，建立状态耦合机制，并对马尔可夫模型的寻优路径求解进一步深化，利用遗传算法和蚁群算法的遍历性和全局搜索性，改进信息素更新机制，能较快的寻找出其优化路径，并针对遗传算法和蚁群算法的局部优化性，加入了变异因子和杂交因子进行改善，通过调整交叉率和变异率进行适当的仿真得出交叉率 $p_c = 0.8$ ，变异率 $p_m = 0.6$ 较为合适。对于错误录入的概率为等概率，在评分机制上进一步加入等概率对数评分来衡量其比对结果。引入 *SPS* 序列对准确对齐的比率和 *CS* 所有序列准确对齐的比率来检验模型的可行性。

针对算例的编纂，选取自生成的样本数据，利用问题一的改进混合算法进行比对分析，并利用 MATLAB 进行验证，可更直观反映算例的差异比对。

针对问题三，在基于问题一的基础上，对于没有样本情况下的讨论，为了找到相似比较高的序列部分，加入相似度搜寻机制，并对问题一的混合算法进一步改进，对其中信息素更新引入个体不同信息的交叉互换，能提高其相似度搜寻的寻找力度。

关键字：隐马尔可夫模型 遗传蚁群混合算法 等概率对数评分机制 *SPS* 序列对齐比率 相似度搜寻

参赛队号： 1928

所选题目： B 题

参赛密码 _____
(由组委会填写)

第十二届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

Abstract

It is known that an unknown language has been discovered, and currently only the text is composed of 20 letters. Under the condition that you have obtained sequence text with only letters but no punctuation and spaces, This paper establishes an *HMM* multiple sequence alignment model based on improved hybrid algorithm to study this text.

For the one problem, based on the first phase of the hidden Markov model, the "insertion" and "delete" states are added to the matching state of the characters, the state coupling mechanism is established, and the optimization path of the Markov model is further solved. Deepening, using the ergodicity and global searchability of genetic algorithm and ant colony algorithm, improving the pheromone update mechanism, it can quickly find its optimization path, and add the mutation factor to the local optimization of genetic algorithm and ant colony algorithm. And hybridization factors to improve, Appropriate simulations by adjusting the crossover rate $p_c=0.8$ and mutation rate $p_m=0.6$ is More suitable. For the probability of false entry, the probability is equal, and the equal probability logarithm score is further added to the scoring mechanism to measure the comparison result. The feasibility of the model was tested by introducing the ratio of the exact alignment of the *SPS* sequence to the exact alignment of all *CS* sequences.

For the compilation of the example, the self-generated sample data is selected, and the improved hybrid algorithm of Problem 1 is used for the comparison analysis, and the verification is performed by MATLAB, which can more directly reflect the difference comparison of the examples.

For the third problem, based on the problem one, for the discussion without the sample case, in order to find the similarly high sequence part, the similarity search mechanism is added, and the hybrid algorithm of the problem one is further improved, and the pheromone is updated. Introducing cross-interchange of different information of individuals can improve the search for similarity search.

Key words: Hidden Markov Model; Genetic Algorithm; Ant Colony Algorithm; Equal Probability Log Scoring Mechanism; *SPS* Sequence Alignment Ratio; Similarity Search

第十二届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会
电话：0471-4969085

邮编：010021

网址：www.tzmcm.cn
Email: service@tzmcm.cn

目录

一、问题重述.....	1
二、问题分析.....	1
2.1 问题一的分析.....	1
2.2 问题二的分析.....	1
2.3 问题三的分析.....	1
三、模型的假设.....	1
四、符号说明.....	2
五、模型的建立与求解.....	3
5.1 问题一的模型建立及求解.....	3
5.1.1 基于隐马尔可夫模型的改进序列比对.....	3
5.1.1.1 <i>HMM</i> 模型描述.....	3
5.1.1.2 状态耦合机制.....	3
5.1.2 求解优化状态序列.....	4
5.1.3 遗传与蚁群的混合改进.....	5
5.1.3.1 蚁群状态更新改进.....	5
5.1.3.1.1 局部状态.....	5
5.1.3.1.2 全局状态.....	5
5.1.3.2 蚁群变异因子改进.....	6
5.1.3.3 蚁群杂交因子改进.....	6
5.1.3.4 遗传算法的策略融合.....	7
5.1.3.4.1 交叉变异操作的改进.....	7
5.1.3.4.2 路径轨迹更新的改进.....	8
5.1.4 基于 <i>HMM</i> 模型改进遗传蚁群算法的多重序列优化比对.....	8
5.1.4.1 比对机制描述.....	8
5.1.4.2 比对算法的实现.....	8
5.2 算例编纂.....	9
5.3 问题三的模型建立及求解.....	10
5.3.1 相似度搜寻模型.....	10
5.3.1.1 评判准则.....	10
5.3.1.2 基于序列特征向量的相似度度量.....	10
5.3.1.3 基于上下语境的模型优化.....	11
5.3.2 混合算法的设置扩展.....	11
5.3.2.1 概率选择公式.....	11
5.3.2.2 改进算法的实现机制.....	11
5.3.2.2.1 适应度函数的改进.....	11
5.3.2.2.2 路径参数调整.....	12
六、模型的检验.....	12
6.1 多序列比对的模型检验.....	12
七、模型的优缺点及改进.....	13
7.1 模型优点.....	13
7.2 模型缺点.....	13
八、参考文献.....	13
九、附录.....	14

一、问题重述

我们发现了一种未知的语言，现只知道其文字是以20 个字母构成的。我们已经获取了许多段由该语言写成的文本，但每段文本只是由字母组成的序列，没有标点符号和空格，无法理解其规律及含义。我们希望对这种语言开展研究，有一种思路是设法在不同段文本中搜索共同出现的字母序列的片段。语言学家猜测：如果有的序列片段在每段文本中都会出现，这些片段就很可能具备某种固定的含义(类似词汇或词根)，可以以此入手进行进一步的研究。在文本的获取过程中，由于我们记录技术的限制，可能有一些位置出现了记录错误。可能的错误分为如下三种：

1. 删失错误：丢失了某个字母；
2. 插入错误：新增了原本不存在的字母；
3. 替换错误：某个字母被篡改成了其他的字母。

第二阶段问题： 现假设我们已经获取了30 段文本，每段文本的长度都在5000 - 8000 个字母之间。我们希望找到的片段的长度为15 个字母。由于技术的限制，当我们在记录每个字母时，都可能有五分之一的概率发生错误。错误类型可能为删失错误、插入错误或替换错误，每个错误只涉及一个字母，且每个错误的发生是独立的。请你设计合理的数学模型，快速而尽可能多地找到符合要求的片段，并自行编撰算例来验证算法的效果。如果我们事先不知道所寻找的片段的长度，算法又应当进行什么改进呢？

二、问题分析

2.1 问题一的分析

针对问题一，基于第一阶段的隐马尔可夫模型上，对字符的匹配状态增加了“插入”和“删除”状态，建立状态耦合机制，并对马尔可夫模型的寻优路径求解进一步深化，利用遗传算法和蚁群算法的遍历性和全局搜索性，改进信息素更新机制，能较快的寻找出其优化路径。对于错误录入的概率为等概率，在评分机制上进一步加入等概率对数评分来衡量其比对结果。

2.2 问题二的分析

针对算例编纂，利用计算机随机生成 20 个样本序列，结合问题一的改进算法，作出评判与结果分析。

2.3 问题三的分析

针对问题三，在基于问题一的基础上，对于没有样本情况下的讨论，为了找到相似比较高的序列部分，加入相似度搜寻机制，并对问题一的混合算法进一步改进，对其中信息素更新引入个体不同信息的交叉互换，能提高其相似度搜寻的寻找力度。

三、模型的假设

H_0 ：假设任意序列对之间的比较是无相关性的。

H_1 ：假设不同序列样本之间的影响忽略不计。

H_2 ：假设样本序列的来源是正常程序测定而获取的。

H_3 : 忽略样本序列过错性缺失的影响。

H_4 : HMM 模型中与隐含状态相关联可直接观测而得到。

H_5 : HMM 模型的任一时刻 t 的某一状态只依赖于其前一时刻的状态, 与其它时刻的状态及观测无关, 也与时刻 t 无关。

H_6 : 观测独立性假设, 任一时刻的观测只依赖于该时刻的马尔科夫链的状态, 与其他观测及状态无关。

H_7 : 假设序列样本的抽取稳定。

H_8 : 假设随机因素不能样本序列的缺失、替换、重复造成干扰。

H_9 : 假定样本序列数据库足够充分。

H_{10} : 假设录入字符发生错误概率均等。

H_{11} : 假设蚁群转移时信息量浓度和蚁群间距离相互联系。

H_{12} : 假设状态路径在某一状态下输出序列的概率相等。

四、符号说明

符号	符号说明
S	不同状态集合
$A = (f_{ij})$	状态转移概率矩阵
$B = (e_k(x_k))$	字符释放概率矩阵
$e_k(x_k)$	状态 S 下释放字符 x_k 的概率
M_j	补偿空位概率
I_j	插入状态为
Π	产生 $X = \{x_1, x_2, \dots, x_n\}$ 的状态路径
L_{best}	蚂蚁最短回路路径
Q_L	路径参数
L_{worst}	最长回路路径
F_k	个体适应值
T_{max}	最大迭代次数
$\Delta\tau_{ij}(t)$	第 k 只蚂蚁在路径 ij 留下的信息素量
o_k	蚁群寻优的第 k 个优化解

五、模型的建立与求解

5.1 问题一的模型建立及求解

5.1.1 基于隐马尔可夫模型的改进序列比对

5.1.1.1 HMM 模型描述

$$M = (V, S, \theta) \quad (1.1)$$

其中定义 V 为 m 个字符集合：

$$V = \{V_1, V_2, \dots, V_m\} \quad (1.2)$$

定义 S 为不同状态集合：

$$S = \{M, I, D\} \quad (1.3)$$

定义 θ 为 (A, B) 的概率集合， $A = (f_{ij})$ 为状态转移概率矩阵， f_{ij} 表示在 t 时刻位于状态 S_i 在 $t+1$ 时刻位于状态 S_j 的条件概率，即：

$$f_{ij} = P(S_j | S_i) = \frac{q_j}{q_i}, i, j \in [1, 2], \quad (1.4)$$

$B = (e_k(x_k))$ 为字符释放概率矩阵， $e_k(x_k)$ 表示位于状态 S 下释放字符 x_k 的概率，即：

$$e_k(x_k) = P(Q = k | q = t, S \in M, I, D_k \in S) \quad (1.5)$$

5.1.1.2 状态耦合机制

对于长度为 N 的序列统计特征 P 为系列概率集合 $e_k(x_k)$ 。

在条件 P 下序列 $X = \{x_1, x_2, \dots, x_N\}$ 发生的概率有：

$$P(X|P) = \prod_{k=1}^N e_k(x_k) \quad (1.6)$$

基于第一阶段上对空格比对得分的忽略，在考虑录入时的错误操作，对全比对得分进行重新定义如下：

$$Score(X|P) = \sum_{k=1}^N \ln \frac{e_k(x_k)}{p(x_k)} + \sum_{k=1}^N \sum_{j \in N_G} \ln \frac{M_j}{e_k} \quad (1.7)$$

其中 (1.2) 第一项为非空位得分，第二项为空位补偿得分， $p(x_k)$ 为字符 x_k 的背景出现频率， N_G 为缺失字符集， M_j 为补偿空位概率，并有如下条件满足：

$$M_j = \begin{cases} 1, & e_j(x_j) > e_k(x_k) \\ \exp(-\frac{e_j(x_j)}{e_k(x_k)}), & else \end{cases} \quad (1.8)$$

在考虑到录入字符时，除却替换外，还有插入和删除状态，基于 Viterbi 算法基础，提出三种状态的耦合机制模型。

定义替换状态为 M_j ，插入状态为 I_j ，删除状态为 D_j ，耦合机制模型如图：

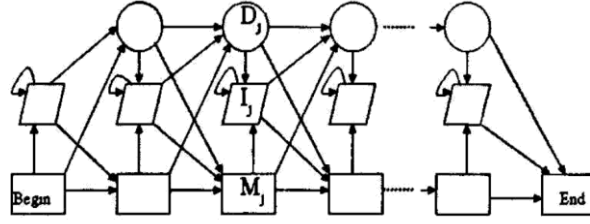


图 1 隐马尔可夫模型

对于 n 个序列比对，将第 j 个序列与 HMM 比对，插入空格得到多重比对机，在比对过程：

$$m_j^M(k) = Ln \frac{e_{M_j}(x_k)}{p(x_k)} + \max \begin{cases} m_{j-1}^M(k-1) + Lnf_{M_{j-1}M_j} \\ m_{j-1}^I(k-1) + Lnf_{M_{j-1}I_j} \\ m_{j-1}^D(k-1) + Lnf_{D_{j-1}M_j} \end{cases} \quad (1.9)$$

其中 $m_j^M(k)$ 表示子序列 $X^j = \{x_1, x_2, \dots, x_k\}$ 与 HMM 模型 P 的比对概率记分，该比对以替换 M_j 释放字符 x_k 为最终操作。

$$m_j^I(k) = Ln \frac{e_{I_j}(x_k)}{p(x_k)} + \max \begin{cases} m_{j-1}^M(k-1) + Lnf_{M_{j-1}I_j} \\ m_{j-1}^I(k-1) + Lnf_{I_{j-1}I_j} \\ m_{j-1}^D(k-1) + Lnf_{D_{j-1}I_j} \end{cases} \quad (1.10)$$

其中 $m_j^I(k)$ 表示子序列 $X^j = \{x_1, x_2, \dots, x_k\}$ 与 HMM 模型 P 的比对概率记分，该比对以插入 I_j 释放字符 x_k 为最终操作。

$$m_j^D(k) = Ln \frac{e_{D_j}(x_k)}{p(x_k)} + \max \begin{cases} m_{j-1}^M(k-1) + Lnf_{M_{j-1}D_j} \\ m_{j-1}^I(k-1) + Lnf_{I_{j-1}D_j} \\ m_{j-1}^D(k-1) + Lnf_{D_{j-1}D_j} \end{cases} \quad (1.11)$$

其中 $m_j^D(k)$ 表示子序列 $X^j = \{x_1, x_2, \dots, x_k\}$ 与 HMM 模型 P 的比对概率记分，该比对以删除 D_j 释放字符 x_k 为最终操作。

最终的优化得分为：

$$Score(X|P) = \max \begin{cases} m_N^M(L) + Lnf_{M_N end} \\ m_N^I(L) + Lnf_{I_N end} \\ m_N^D(L) + Lnf_{D_N end} \end{cases} + \sum_{k=1}^L \sum_{j \in N_G} Ln \frac{M_j}{e_k} \quad (1.12)$$

5.1.2 求解优化状态序列

对于一个路径 $\Pi = (\pi_1, \pi_2, \dots, \pi_N)$ 为已知的状态路径，该状态路径在状态 S 下输出序列的概率为：

$$P(X|S) = \sum_{\Pi} P(X, \Pi) \quad (1.13)$$

其中 Π 为产生 $X = \{x_1, x_2, \dots, x_n\}$ 的状态路径。

由于在给定的 $Markov$ 状态中产生的路径 Π 有诸多条，为保证每条路径的概率可能，

产生序列的概率是对在状态 S 中每条可能路径的叠加，为：

$$P(X|S) = \sum_{\Pi} f_{0,1} \prod_{k=1}^N e_k(x_k) f_{k,(k+1)} \quad (1.14)$$

为了简化算法的复杂度，在算法中引入前向变量和后向变量。

前向变量为：

$$\alpha_k(n) = P(x_1, x_2, \dots, x_n | \pi_n = k) \quad (1.15)$$

$\alpha_k(n)$ 表示在状态 S 序列 $\{x_1, x_2, \dots, x_n\}$ 终止于状态 S' 的最大概率的可能路径。

后项变量为：

$$\beta_k(n) = P(x_{n+1}, x_{n+2}, \dots, x_N | \pi_n = k) \quad (1.16)$$

$\beta_k(n)$ 表示在状态 S 中，序列 $\{x_{n+1}, x_{n+2}, \dots, x_N\}$ 起始于状态 S' 的最大概率的可能路径。 π_0 为起始状态， π_{N+1} 为终止状态，在具有普适意义下满足 $f_{N,(N+1)} = 1$ 。

在状态 S 中为产生序列 X 寻找一条最优路径 Π^* ，要求使得 $P(X|\Pi^*)$ 最大，记为：

$$\Pi^* = \arg \max_{\Pi} \{P(X|\Pi)\} \quad (1.17)$$

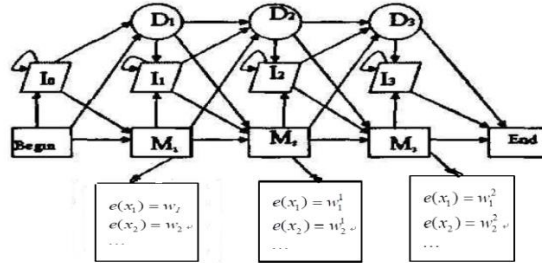


图2 耦合马尔可夫

对于最优路径 Π^* 的寻求，结合遗传与蚁群算法的优越性，可以有效的得到其解。

5.1.3 遗传与蚁群的混合改进

5.1.3.1 蚁群状态更新改进

蚁群的转移是由各条路径上留下的信息量浓度和蚁群间距离来变异的，对标准的蚁群群体的信息激素进行变异，可以增大蚁群对优解的寻找。

5.1.3.1.1 局部状态

蚂蚁完成一次循环后对全部信息的局部更新如下：

$$\tau_{ij}(t+n) = (1-\rho_1)\tau_{ij}(t) + \rho_1 A \quad (2.1)$$

其中 ρ_1 为信息素挥发参数，满足 $\rho_1 \in [0,1]$ ， A 为初始信息素。

$$\Delta\tau_{ij} = \frac{Q_L}{L_{best}} \quad (2.2)$$

其中有 L_{best} 为蚂蚁最短回路路径， Q_L 为路径参数。

5.1.3.1.2 全局状态

$$\tau_{ij}(t+n) = (1-\rho_1)\tau_{ij}(t) + \rho_2 \Delta\tau_{ij}(t) \quad (2.3)$$

$$\Delta\tau_{ij}(t) = \begin{cases} \frac{Q_L}{L_{best}}, & \text{if 边}(ij) \in L_{best} \\ \frac{Q_l}{L_{worst}}, & \text{if 边}(ij) \in L_{worst} \\ 0, & \text{else} \end{cases} \quad (2.4)$$

其中 L_{worst} 为最长回路路径, $\Delta\tau_{ij}(t)$ 为第 k 只蚂蚁在路径 ij 留下的信息素量。

完成信息量更新后, 将每条边上的信息量限制 $[\tau_{min}, \tau_{max}]$ 之间, 避免某些边上的信息量过大, 减小整个图上各条边的信息量差距, 能一定程度上增大蚁群的搜索空间。

5.1.3.2 蚁群变异因子改进

定义状态变量:

$$S = \begin{cases} \arg \max_{i,j \in set} \{ \tau_{ij}^\alpha(t) \eta_{ij}^\beta(t) \}, & q < q_0 \\ 0, & \text{else} \end{cases} \quad (2.5)$$

其中 $\eta_{ij}^\beta(t)$ 为蚂蚁所选择的路径由城市 i 转移到城市 j 的期望程度, τ_{ij}^α 表示蚂蚁在运动过程中城市 i 、 j 连线上所积累的信息素浓度, q 为 $[0,1]$ 之间的随机数。

由 (2.5) 可知, $q < q_0$ 时, 此时蚁群的状态采用确定性搜索, 蚂蚁 q_0 的概率选择 L_{best} , 反之, $q \geq q_0$ 时, 此时蚁群状态为随机搜索, 蚂蚁 $1 - q_0$ 的概率选择 L_{worst} 。

为了避免蚁群过早和过晚收敛, 引入变异因子, 使得蚂蚁在初始基因组合以外的空间进行搜索。

设选定的某只蚂蚁所搜索过的节点为 $\{a_{i1}, a_{i2}, \dots, a_{in}\}$, 从这一系列节点中随机决定变异点, 变异算子在算法的初期、中期、后期决定一定的概率, 重新选取一定的初始值, 并保存变异且更新信息素, 否则取消变异。

5.1.3.3 蚁群杂交因子改进

定义一个遗传种群:

$$a = \{\dots; a_{i1}, a_{i2}, \dots, a_{in}; \dots\} \quad (2.6)$$

蚂蚁所爬行回路的路径中各节点值的集合为:

$$L_s = \sum_{k=1}^{15} d_k \quad (2.7)$$

其中 d_k 表示蚂蚁所走回路中第 k 个节点值。

转盘式选择策略选择概率为:

$$p_k = \frac{1/L_k}{\sum 1/L_k} \quad (2.8)$$

对于 $1/L_k$ 较大的蚂蚁被选中的几率更大。选择蚂蚁杂交后, 允许按一定的比例吸收略差的蚂蚁或只吸收比原解对应的路径长度小一定百分比的蚂蚁。

用 (2.7) 计算所有个体符合优化性能指标的适应度, 并按适应度进行升序排列。

设得到符合优化性能指标的 m_k 个最优父体为:

参赛队号#1928

$$\begin{aligned} a_1 &= \{a_{11}, a_{12}, \dots, a_{1n}\} \\ a_2 &= \{a_{21}, a_{22}, \dots, a_{2n}\} \\ &\dots \\ a_{m_k} &= \{a_{m_k 1}, a_{m_k 2}, \dots, a_{m_k n}\} \end{aligned} \quad (2.9)$$

选却杂交段 $\{l_1, l_2, \dots, l_{m_k}\}$, 将父本 *one* 的杂交段 l_1 遗传给子代 m_k 相应部位, 将父本 *two* 的杂交段 l_2 遗传给子代 *three* 相应部位, 同理, 将父本 m_k 的杂交段 l_{m_k} 遗传给子代 *one*, 并且父本 *one* 删除与父本 *two* 相同的杂交内容, 父本 *two* 删除与父本 *three* 相同的杂交内容, 同理父本 m_k 删除父本 *one* 相同的杂交内容, 保持其余不变。

修改模型参数的仿真结果分析如下:

表 1-1

α	β	ρ	L_{best} 最短路径	T 进化次数
1	4	0.001	310.8758	162
1	4	0.5	322.5945	147
1	4	0.1	308.1685	150
2	2	0.001	271.1854	152
2	2	0.5	303.8851	143
4	1	0.1	287.8268	119
4	1	0.001	267.7795	168
4	1	0.1	290.4864	137

5.1.3.4 遗传算法的策略融合

5.1.3.4.1 交叉变异操作的改进

将蚁群输出与系统的期望进行误差调控:

$$F = \eta \left(\sum_{k=1}^N \text{abs}(y_k - o_k) \right) \quad (2.10)$$

其中 y_k 期望输出, o_k 蚁群寻优的第 k 个优化解, η 设定参数。

随机生成序列编码种群 $X = \{x_1, x_2, \dots, x_N\}$, 每个编码代表一个可遗传区域, 选定合适的适应函数计算误差调控。

结合 (2.8) 依旧基于转盘式选择策略, 对每个个体的选择概率为:

$$P_k = \frac{f_k}{\sum_{j=1}^N f_j} \quad (2.11)$$

$$f_k = \frac{\eta}{F_k} \quad (2.12)$$

其中 F_k 为个体适应值, 由 (3.1) 定义。

第 k 个染色体和第 1 个染色体 X_1 在 j 位的交叉操作如下定义:

$$x_{lj} \begin{cases} x_{kj} = x_{kj}(1-b) \nmid x_{lj}b \\ x_{lj} = x_{lj}(1-b) \nmid x_{kj}b \end{cases} \quad (2.13)$$

选取第 i 个个体的第 j 个基因 x_{ij} 进行变异，其操作如下定义：

$$x_{ij} = \begin{cases} x_{ij} + (x_{ij} - x_{max})f(t), t \geq 0.5 \\ x_{ij} + (x_{min} - x_{ij})f(t), t < 0.5 \end{cases} \quad (2.14)$$

$$f(t) = \frac{r}{1 - \frac{t}{T_{max}}} \quad (2.15)$$

其中 r 为随机数， T_{max} 为最大迭代次数， x_{max} 为基因 x_{ij} 的上界， x_{min} 为基因 x_{ij} 的下界。

5.1.3.4.2 路径轨迹更新的改进

在蚁群群体中把各路径信息素初值为最大值 T_{max} 。通过遗传算法得到了一定的路径信息素，结合蚁群与遗传信息素综合水平，把信息素的初值设置为：

$$T_s = T_{max} + T \quad (2.16)$$

其中 T_G 是遗传算法结果转化的信息素值。

信息素更新模型只对圈中只有最短路径的蚂蚁才进行信息素修改加，则整体轨迹信息更新方程为：

$$\tau_{ij}(t+n) = \sum_{i \in N} (1-\rho) \tau_{ij}(t) + \rho \tau_{ij}(t) \quad (2.18)$$

5.1.4 基于 HMM 模型改进遗传蚁群算法的多重序列优化比对

5.1.4.1 比对机制描述

通过蚁群求出的每个路径集合，可找出对应的一个比对。路径中的结点便是蚂蚁所选择的相匹配的字符。对于不在路径上的字符，可以用其他方法简单的求出它们在比对中的位置。在所有蚂蚁完成路径的集合后，算法根据打分机制求出各个比对的得分，根据分值的高低对路径上的信息素进行更新，以增大分值高的路径上的信息素。这样当下一个蚂蚁选择结点时，就以新的信息素作为选择的依据，从而构成信息学习的正反馈机制。

5.1.4.2 比对算法的实现

基于遗传算法中，随机产生一个初始群体，群体中的每个个体代表一个可能的解，根据第 k 代群体计算第 $k+1$ 代群体的过程如下：

k 代群体中第 r 个个体的适应度值的计算公式是：

$$F_r = \sum_{i=1}^{N-1} \sum_{j=i+1}^N SE(S_i^p(d_{ri}), S_j^p(d_{rj})) + \sum_{i=1}^{N-1} \sum_{j=i+1}^N SE(S_i^s(d_{ri}), S_j^s(d_{rj})) \quad (3.1)$$

k 代群体中第 r 个个体被选择到下一代个体的概率为 (2.11) 所述。

对所选择的个体执行交叉操作和变异操作如上文所述机制。

利用 *Matlab* 对改进混合算法进行测试，改变交叉率 p_c 进行仿真如下：

表 1-2

p_c	最大时间	最小时间	平均时间	对数记分	适应值
0.65	7.51	7.19	7.20	-1592.0	8.0
0.70	7.82	7.32	7.44	-1570.3	27.7
0.75	8.90	8.45	8.66	-1562.1	29.9
0.80	9.51	9.05	9.41	-1549.2	50.8
0.85	10.60	9.54	10.02	-1548.5	51.5
0.90	11.25	10.68	10.91	-1548.2	51.8

在交叉率 $p_c = 0.8$ 时，改变变异率 p_m 的仿真如下：

表 1-3

p_m	平均时间	对数记分	适应值
0.3	9.19	-1566.3	33.7
0.4	9.17	-1558.9	41.1
0.5	9.22	-1552.0	48.0
0.6	9.33	-1551.5	48.5
0.7	9.58	-1552.2	47.8
0.8	9.64	-1552.4	47.6
0.9	9.75	-1549.9	47.9

由表可知，设定变异率 $p_m = 0.6$ ，交叉率 $p_c = 0.8$ 是比较合理的。

5.2 算例编纂

选取自生成的随机字符序列排列，见附录，结合模型一的改进混合算法得出结果分析如下：

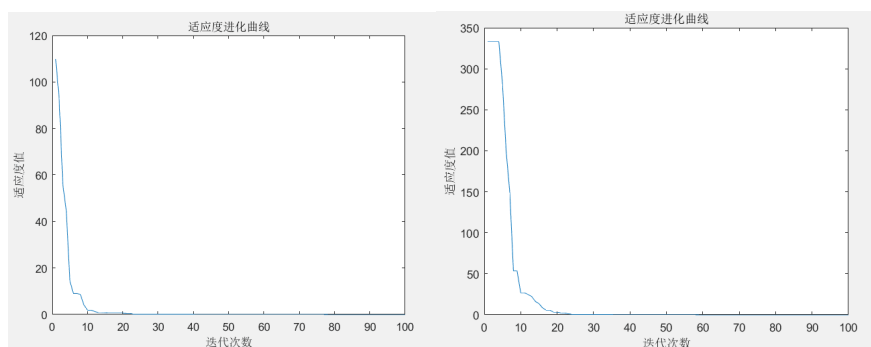


图 3

利用 *Python* 对隐马尔科夫路径序列优化的结果如下：

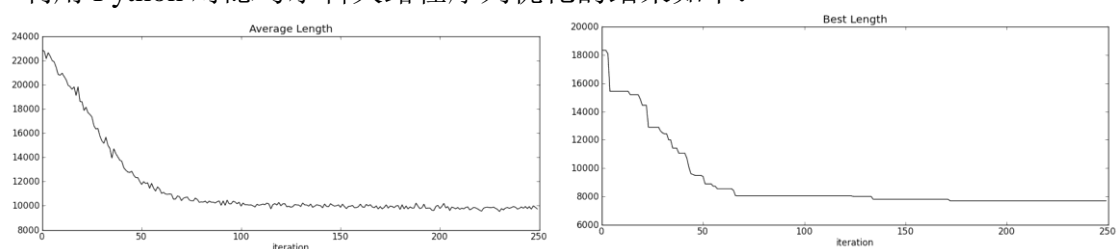


图 4

路径优化方式：

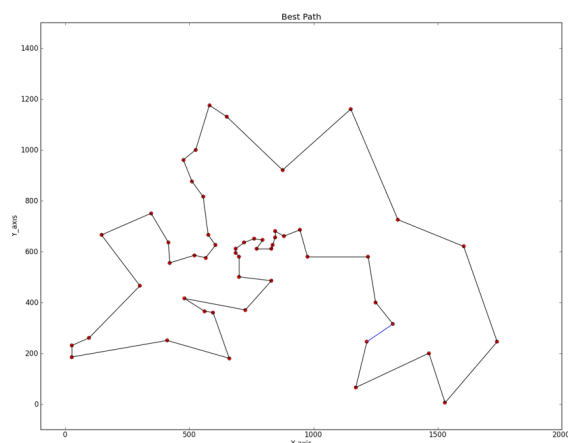


图 5

选取不同序列长度为参考序列的运行结果：

表 1-4

参考序列长度	等概率对数记分	适应值	搜索时长	相似度
5	-1990.6	9.4	3.06	0.78
10	-1929.7	70.3	3.96	0.65
15	-1884.1	115.9	4.84	0.68
20	-1771.3	191.3	5.75	0.73
25	-1688.8	228.7	6.62	0.56
30	-1685.6	311.2	7.54	0.47
35	-1677.7	314.4	8.43	0.56

可以看出随着参考序列长度增加，准确率在逐渐下降，搜索时长也逐渐增加，适应值的选取就变得比较困难。

5.3 问题三的模型建立及求解

5.3.1 相似度搜寻模型

5.3.1.1 评判准则

- (1) 序列自身相似度为 1；
 - (2) 不可替换序列，那么其相似度为 0；
 - (3) 相似度度量为单相关，即两个序列越相似，其相似度就越高。
- 为了方便处理序列相似度，采用归一化度量，其相似度值域为[0,1]。

5.3.1.2 基于序列特征向量的相似度度量

定义：

$$C = \{c_1, c_2, \dots, c_n\} \quad (4.1)$$

其中 C 为样本序列空间。

记序列 S_1 和 S_2 的特征向量分别为：

$$\begin{aligned} S_1 &= \{s_{11}, s_{12}, \dots, s_{1n}\} \\ S_2 &= \{s_{21}, s_{22}, \dots, s_{2n}\} \end{aligned} \quad (4.2)$$

其中 s_{ij} 表示第 i 个字符的特征向量中 c_j 的出现次数。

定义基于序列特征向量的相似度计算如下：

$$Sim(S_1, S_2) = \sqrt{\frac{2S_1 S_2^T}{S_1 S_1^T + S_2 S_2^T}} \quad (4.3)$$

其中 $Sim(S_1, S_2)$ 表示相似度。

定义差异度为：

$$Sim^2(S_1, S_2) - Sim^2(S_2, S_3) = \frac{2(S_2 S_2^T - S_1 S_3^T)(S_1 - S_3) S_2^T}{(S_1 S_1^T + S_2 S_2^T)(S_3 S_3^T + S_2 S_2^T)} \quad (4.4)$$

$|S_{31} - S_{32}| < |S_{11} - S_{21}|$ 时，有 $Sim(S_3, S_2) > Sim(S_1, S_2)$ ；

$|S_{31} - S_{32}| > |S_{11} - S_{21}|$ 时，有 $Sim(S_3, S_2) < Sim(S_1, S_2)$ 。

当序列 S_3 在样本空间的投影（某一字符的频率）相比序列 S_1 更接近 S_2 时，其相似度也就越高。

5.3.1.3 基于上下语境的模型优化

针对外星语的语言特点，对上、下文语境空间分别赋以不同权重，对（4.3）相似度模型进行优化如下：

$$Sim(S_1, S_2) = \alpha \sqrt{\frac{2S_{U,1} S_{U,2}^T}{S_{U,1} S_{U,1}^T + S_{U,2} S_{U,2}^T}} + \beta \sqrt{\frac{2S_{D,1} S_{D,2}^T}{S_{D,1} S_{D,1}^T + S_{D,2} S_{D,2}^T}} \quad (4.5)$$

其中 $S_{U,i}$ 、 $S_{D,i}$ 分别表示序列 S_i 的上、下特征向量， (α, β) 为权重向量。

5.3.2 混合算法的设置扩展

5.3.2.1 概率选择公式

定义序列概率为如下：

$$P(k, l, n, m) = \frac{phe(k, l, n, m) \times \alpha + mat(k, l, n, m) \times b + dev(k, l, n, m) \times c}{\sum_{r=loc(k, l, n)} phe(k, l, n, m) \times \alpha + mat(k, l, n, m) \times b + dev(k, l, n, m) \times c} \quad (4.7)$$

其中， $phe(k, l, n, m)$ 表示第 k 个序列的第 l 个字符选择第 n 个序列中的第 m 个字符的信息素指标， $mat(k, l, n, m)$ 表示第 k 个序列的第 l 个字符和第 n 个序列的第 m 个字符的匹配得分， $dev(k, l, n, m)$ 表示第 n 个序列的第 m 位置与第 k 个序列的第 $l-1$ 个字符的第 m 个字符的匹配得分 a 为信息素， b 为匹配程度， c 为位置偏差， $loc(k, l, n)$ 表示第 k 个序列的第 l 个字符出发选择第 n 个序列时起始字符的字符位置。

5.3.2.2 改进算法的实现机制

5.3.2.2.1 适应度函数的改进

以各个个体对应的解的得分值作为其适应度函数值，对于所有序列中的第 j 个字符的 SP 得分记为：

$$SP - Score(j) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N-1} p(c_{ij}, c_{kj}) \quad (5.1)$$

其中 $p(c_{ij}, c_{kj})$ 为字符 c_{ij} 及 c_{kj} 的记分，它的定义如下：

$$p(c_{ij}, c_{kj}) = \begin{cases} +2, & \text{if } (c_{ij} = c_{kj} \text{ and } c_{ij}, c_{kj} \in \Sigma) \\ -1, & \text{if } (c_{ij} \neq c_{kj} \text{ and } c_{ij}, c_{kj} \in \Sigma) \\ -2, & \text{if } ((c_{ij} \in \Sigma, c_{kj} \in \Sigma' - ') \text{ or } (c_{kj} \in \Sigma, c_{ij} \in \Sigma' - ')) \\ 0, & \text{if } (c_{ij} \in \Sigma' - ', c_{kj} \in \Sigma' - ') \end{cases} \quad (5.2)$$

其中 Σ 代表字符集，其匹配字符记+2 分，不匹配字符记-1 分，字符与空位记-2 分，而空位与空位的匹配记 0 分。

混合模型的适应度函数值：

$$alignsum = \sum_{j=1}^L SP - Score(j) \quad (5.3)$$

5.3.2.2.2 路径参数调整

$$a = \begin{cases} 1, & t \leq T_1 \\ 2, & T_1 < t \leq T_2 \end{cases} \quad (5.5)$$

$$b = \begin{cases} 1.8, & t \leq T_1 \\ 1.2, & T_1 < t \leq T_2 \end{cases} \quad (5.6)$$

$$c = \begin{cases} 1.8, & t \leq T_1 \\ 1.2, & T_1 < t \leq T_2 \end{cases} \quad (5.7)$$

其中， T_1, T_2 的值为常数， a, b, c 的值对应阶梯函数的不同取值。

个体在搜索的初始创优所得到的这些路径的整体匹配程度并不一定好，导致了搜索所得的结果是局部最优而不是全局最优。可以选择阶梯参数，根据迭代次数来调整参数 a, b, c 的值。

六、模型的检验

6.1 多序列比对的模型检验

在进化计算中，目标函数不仅是评价一个结果的适应度的标准，而且目标函数的值还反映这个比对的生物信息并预示序列的结构或比对序列之间存在的进化关系。根据 SP 目标函数，在比对结果的每一列中，将每对对比序列给定一个分值 P_{SCORE} ，然后将 P_{SCORE} 分值累加起来，得到每列的分值 C_{SCORE} ，最后将每列的分值累加，即可得到 SP_{SCORE} 。

假设比对结果为 $S_0 = (s_{ij})$ ，则 SP_{SCORE} 的计算如下：

$$SP_{SCORE}(S_0) = \sum_{k=1}^L C_{SCORE} \{s_{1j}, s_{2j}, \dots, s_{Nj}\} \quad (6.1)$$

$$C_{SCORE} \{s_{1j}, s_{2j}, \dots, s_{Nj}\} = \sum_{1 \leq P \leq Q \leq N} P_{SCORE}(S_0 P_k, S_0 Q_k) \quad (6.2)$$

如果输入数据是标准比对库中的序列，即有一个标准的比对结果，可以计算相对的 SP_{SCORE} ，定义为 SPS 。假定对于标准库的输入序列，标准库中比对结果为 S_0 ，某方法的比对结果为 S_1 ，则 SPS 定义如下：

$$SPS = \frac{SP_{SCORE}(S_0)}{SP_{SCORE}(S_1)} \quad (6.3)$$

如果没有对比库， SPS 定义如下：

$$SPS = \frac{SP_{SCORE}(S_0)}{\frac{LN(N-1)}{2}} \quad (6.4)$$

SPS 分值反映的是序列对准确对齐的比率，使用 CS 值来反映所有序列准确对齐的比率。 CS 值计算方法为：如果一列上的所有序列对均相等，则 $c_i = 1$ ，反之 $c_i = 0$ 。同样，对于比对结果 S_0' ， CS 值计算公式为：

$$CS = \sum_{i=1}^L \frac{c_i}{L} \quad (6.5)$$

对于 SPS 值和 CS 值越高，说明比对结果越准确，越能反映序列的特性。

七、模型的优缺点及改进

7.1 模型优点

1. 改进信息量更新后，将每条边上的信息量限制 $[\tau_{min}, \tau_{max}]$ 之间，避免某些边上的信息量过大，减小整个图上各条边的信息量差距，能一定程度上增大蚁群的搜索空间。
2. 本文结合遗传算法、蚂蚁算法的优点，提出了一种基于融合遗传算法与蚂蚁算法的改进算法，加强了遗传算法的全局搜索能力，对遗传算法中使用随机生成种群，减少了人为主观因素判定，同时加快了蚂蚁算法的收敛速度。
3. 蚁群算法具有正反馈、较强的鲁棒性，易于与其他方法相结合。
4. 改进遗传算法使用概率机制进行迭代，具有随机性，具有可扩展性，容易与其他算法结合。
5. 隐马尔科夫序列能够计算出具有维修能力和多重降级状态的系统的概率，该方法对过程的状态预测效果良好。

7.2 模型缺点

1. 随着算法迭代次数的不断增加，信息激素会逐渐积累在某条特定的路径上，使该条路径上的信息激素浓度远远大于其他路径上的信息激素浓度，导致特定个体蚂蚁都集中到这条局部最优的路径上。对于群体而言会出现多个优化路径
2. 混合算法如果参数设置不当，导致求解速度很慢，且所得解的质量特别差。
3. 蚁群算法遍历性计算量大，求解所需时间较长
4. 遗传算法的编程实现比较复杂。
5. 没有能够及时利用网络的反馈信息，故算法的搜索速度比较慢，要得要较精确的解需要较多的训练时间。
6. 隐马尔科夫模型只依赖于每一个状态和它对应的观察对象，目标函数和预测目标函数不匹配容易出现不匹配现象。

八、参考文献

- [1] 闫磊、马健、董辉、高梦，基于遗传算法与模拟退火算法在多重 DNA 序列比对中的应用研究，海南热带海洋学院学报，第 24 卷第 2 期，64-69，2017
- [2] 张树波，一种用于多重生物序列比对的协同进化算法，计算机应用与软件，第 27 卷第 8 期，37-40，2010
- [3] 陈娟、陈峻，多重序列比对的蚁群算法，计算机应用，第 26 卷，124-128，2006
- [4] 张维梅，基于遗传算法和蚁群算法的多重序列比对，潍坊学院学报，第 8 卷第 4 期，88-90，2008
- [5] 高强、吕文芝、杜小山等，遗传算法性能评价准则研究，西安交通大学学报，第 40 卷第 7 期，803-806，2006
- [6] 孙瑞祥、屈梁生，遗传算法优化效率的定量评价，自动化学报，第 26 卷第 4 期，552-556，2000
- [7] 王晓瑜、原思聪、李曼，基于自适应蚁群遗传混合算法的 PID 参数优化，计算机应用研究，第 32 卷第 5 期，1376-1378，2015
- [8] 蒋卫丽、陈振华、邵党国等，基于领域词典的动态规划分词算法，南京理工大学学报，第 43 卷第 1 期，63-71，2019
- [9] 彭沛夫、张桂芳，遗传融合蚁群算法的改进与仿真，计算机工程与应用，第 46 卷第 4 期，43-45，2010
- [10] 王娟、巩建平、冯蕾洁，一种改进的遗传蚁群混合算法，制造业自动化，第 36 卷第 2 期，78-80，2014
- [11] 王颖、谢剑英，一种自适应蚁群算法及其仿真研究，系统仿真学报，第 14 卷第 1 期，31-33，2002
- [12] 刘哲、李风军，遗传算法与蚂蚁算法的融合在神经网络中的研究，科技广场，第 10 期，6-9，2012
- [13] 李建珍，基于遗传算法的人工神经网络学习算法，西北师范大学学报(自然科学版)，第 38 卷第 2 期，33-38，2002
- [14] 李敏强、徐博艺、寇纪淞，遗传算法与神经网络的结合，系统工程理论与实践，第 19 卷第 2 期，1-7，1999
- [15] 冯通、杜文才，一种基于理想隐形马尔科夫模型的地图匹配，海南大学学报(自然科学版)，第 32 卷第 3 期，212-216，2014/9
- [16] 梁栋、霍红卫，自适应蚁群算法在序列比对中的应用，计算机仿真，第 22 卷第 1 期，100-103，2005/1
- [17] 陈娟、陈峻，多重序列比对的蚁群算法，计算机应用，第 26 卷，124-128，2006/6
- [18] 沈世镒，关于多重序列比对距离矩阵的一点注记*，工程数学学报，第 20 卷第 3 期，1-7，2003/8
- [19] 刘立芳、霍红卫、王宝树，PHGA-COFFEE：多序列比对问题的并行混合遗传算法求解，第 29 卷第 5 期，728-733，2006/5

九、附录

样本数据：

cfhgfgjgfhgjbryjglibyuiyvvtuvtyuyuguytubjbitrjbvirthjbihtgikbjrtjbjrybkltbylkrkjbljrkl hgfuy gyuy dyd
xsjbxsjnxhs gljfhubvwjnrgcnjdvmv qwesnhedgeskcdcdpsdic znxjbchduqhbdstdefds ncjdgshdiedhdj
sbshgshawqpoiuewyqasqndwb snjhde xsjgfgjw dsjfh swjehrcbdwjqwqeb qwrpoefrukjxnshdiejndfjh
znahsgdqundegu xhbsbajdhueguw cbdbqqhsqejwhddcdgs dsdjehfdjvdjnakdiofeo nzjadhuefhwiqpr
zanjgegdwjnd xnhsgdesdjadbshfd sjhafiudfyehwqbcfue ihsauidhawiudhueafhu sncjhdgsuhajdhus

参赛队号#1928

anjbashshqwyqwd qwqwdhwbxdvbfjdnrh hueytuvbhvdsbv ncdhgfuewfn vfdghttpuigrhmiryujfeu
xsbcvudsicn vpfioboioetjew xbasfzadmkefhe xiewugemhytphothorg cjhguqjqdnbsjdnxsabchewfj
zajdewhfhjsknutgioefj vjfhgrugjfbntr ifrjgnfjkbjngttht djfdgktjjbnfjgng proetpgoirehojrefdjngbjdfb
znjabduywiqndejgg qiwqhqudbnhscyuds dbhsbcdshfhwd hdsbcjcnbdsjigv xahgdsh qkwehdwjbcsdb
dhbdxhsabchsdbjc sqdewpogwiefew fmjerhobvhf djshfuwbdcbs fbhegyuwe sndjhdfugwudhdhvq
zjqgdywchavdgy dhwfxbshfawhjbch whagdhwbbhjsbghe sbwjddjsfgfore fjhgeruhvehdsncj whfgwdb
amjhdwytjj shuadywycbsbcasribf grghuhvs cxx chdssfguerc cdusignvibd disuhuefbhew iehfewihud
zmamjifgewuncud qidhwuqvbdsbv idhscubbnwvuh sfewioiwufo dqwfcqeihrqnfcn mdoefwepg
vgaxhagdyabxcvshb xnadycavchascdsjk jhsbagfyuc sdghvkdjhdjbvdj sdvhbdsj sdhghjcvb sajffeshv
sjdfshdbchjaegue wqtqwihupoe pogjweviosgnveur hgvhwduiwegskdfjenj nfjegjenfjiegewjfnejnsgk
vgdtrqwdwbhvbvbus dfjreiyvufuyguief fruguewboowydiw qwqjfielhncvduhg whegfeuwbvurgherui
gzabhyxwjxjgyg bxhsgsqsqdpsfierojdjv hregwjcxcnchrufewb sjhgiwefiufbdwuhfu chwgfeufewgd
hgdyxbhsfteqei qedwodncewufe xhdsedwncewudyw wqetybfreyfwuiqfnreu iewghfewifhuef duheu
qwgwydgwhcbsuhcvshdbwhfhe hjxsnaduigwdbwxwub nhwdbwuxbuwbc xqwuduwxusxbnwqubxx
zaqdhwbqbxqhsbhs hsacghsubcds auisfygevbfeub djsfnciqwudfhewn djgfwncvegur smgjirfnzfgeyf
xnjadhjcwddmkcdsask xjsbcajsnnw sjsdhnsajacgbasds csjdcjsjndsbfcd xhisjakwhfiudsb bsbsajnsnjx
snjgyefjsbcdodd djejkjewurgb wpewrieufnv dewbfffrehgjejnre rehfgufre fnjdgfyrbjsd whegvhevfe
wjejuwfviegehwr griopnjfrevjdffhrkvjffjjfj vnfjvbjdnjvrjevknkfjvhr cbnvhdhfdjfbdrj ksdjiefjeinfrjjf
sfrfecjthgeirf chfveruejiwhf ffdhwouretiewhfiew dsbfhkhwdnejfkhkefn jsdbjehfewkjdwknrjihreoig
dfneghtpwqirt htopregjeohgefncjdger fbhregtioewfhuhe cbjc fvwijfghwhfvh bhwrehwfbhewufhhd
bxhdgfwjncdnhgweg wqiewfhewocrehewjkfn dbwjgfsnchjdvsfjb wbdewhqrhioqwhdbewu dbhegw
zjbxhjsdghasvdyuew vmodshgiusegeiqhf hdgfiuegwncfu djhwguwbdw bdjwgfiwjhhufwegriqowruhi
areefryhjvcjt mfrjgureopiyehbndskjgitg njsdgoqwjfewhguweifio dbwuegfwqio dhwqighihroiwtuhuyj
cnjdjfhscjbfhreg zbzhsgfjendviuhicrpeutyiowfiewghi cbvbjsdbslahdgvuwehfjibjdegfu efghebfiwigfiue
shdvfebfjisabfuieicn cjhbdfjwncdufh jshruereuifuiwefh jmkauwrwiojdiw podoejwjinshgqwudhab

程序代码:

```
function [P1,P2,p1] = ForwardBackwardAlgo(A,B,Pi,0)
alpha_1 = zeros();
sum_alpha_j = zeros();
sum_beta_i = zeros();
% sum_p = zeros();
P1 = zeros();
A_size = size(A);
O_size = size(0);
N = A_size(1,1);
M = A_size(1,2);
K = O_size(1,1);
P2 = zeros(N,K);
% -----前向算法-----
%求解初值
for i = 1:N
    alpha_1(i,1) = Pi(i,1) * B(i,0(1,1));
end
P1= alpha_1;
%递推求解前向概率, 存放在矩阵 P1 中
for k = 1:K-1
    for j = 1:M
        for i = 1:N
            sum_alpha_j(i,1) = P1(i,k) .* A(i,j);
        end
        P1(j,k+1) = sum(sum_alpha_j) .* B(j,0(k+1,1));
    end
end
%计算观测概率 P(0|lambda)
```

```

p1 = sum(P1(:,K));
% -----后向算法-----
% 对最终时刻所有状态有 beta_T(i) = 1
P2(:,K) = 1;
% 递推求解后向概率, 存放在矩阵 P2 中
for k = K-1:-1:1
    for i = 1:M
        for j = 1:N
            sum_beta_i(j,1) = A(i,j) .* B(j,0(k+1,1)) .* P2(j,k+1);
        end
        P2(i,k) = sum(sum_beta_i);
    end
end
蚁群路径寻找:
import numpy as np
import matplotlib.pyplot as plt
%pylab
coordinates
np.array([[565.0, 575.0], [25.0, 185.0], [345.0, 750.0], [945.0, 685.0], [845.0, 655.0],
[880.0, 660.0], [25.0, 230.0], [525.0, 1000.0], [580.0, 1175.0], [650.0, 1130.0],
[1605.0, 620.0], [1220.0, 580.0], [1465.0, 200.0], [1530.0,
5.0], [845.0, 680.0],
[725.0, 370.0], [145.0, 665.0], [415.0, 635.0], [510.0, 875.0], [560.0, 365.0],
[300.0, 465.0], [520.0, 585.0], [480.0, 415.0], [835.0, 625.0], [975.0, 580.0],
[1215.0, 245.0], [1320.0, 315.0], [1250.0, 400.0], [660.0, 180.0], [410.0, 250.0],
[420.0, 555.0], [575.0, 665.0], [1150.0, 1160.0], [700.0, 580.0], [685.0, 595.0],
[685.0, 610.0], [770.0, 610.0], [795.0, 645.0], [720.0, 635.0], [760.0, 650.0],
[475.0, 960.0], [95.0, 260.0], [875.0, 920.0], [700.0, 500.0], [555.0, 815.0],
[830.0, 485.0], [1170.0,
65.0], [830.0, 610.0], [605.0, 625.0], [595.0, 360.0],
[1340.0, 725.0], [1740.0, 245.0]])
def getdistmat(coordinates):
    num = coordinates.shape[0]
    distmat = np.zeros((52,52))
    for i in range(num):
        for j in range(i,num):
            distmat[i][j] = distmat[j][i]=np.linalg.norm(coordinates[i]-coordinates[j])
    return distmat
distmat = getdistmat(coordinates)
numant = 40 #蚂蚁个数
numcity = coordinates.shape[0] #城市个数
alpha = 1 #信息素重要程度因子
beta = 5 #启发函数重要程度因子

```

```

rho = 0.1    #信息素的挥发速度
Q = 1
iter = 0
itermax = 250

etatable = 1.0/(distmat+np.diag([1e10]*numcity)) #启发函数矩阵，表示蚂蚁从城市 i 转移到
矩阵 j 的期望程度
pheromonetable = np.ones((numcity,numcity)) # 信息素矩阵
pathtable = np.zeros((numant,numcity)).astype(int) #路径记录表

distmat = getdistmat(coordinates) #城市的距离矩阵

lengthaver = np.zeros(itermax) #各代路径的平均长度
lengthbest = np.zeros(itermax) #各代及其之前遇到的最佳路径长度
pathbest = np.zeros((itermax,numcity)) # 各代及其之前遇到的最佳路径长度
while iter < itermax:
    # 随机产生各个蚂蚁的起点城市
    if numant <= numcity:#城市数比蚂蚁数多
        pathtable[:,0] = np.random.permutation(range(0,numcity))[:numant]
    else: #蚂蚁数比城市数多，需要补足
        pathtable[:numcity,0] = np.random.permutation(range(0,numcity))[:]=
        pathtable[numcity:,0]
np.random.permutation(range(0,numcity))[:numant-numcity]
length = np.zeros(numant) #计算各个蚂蚁的路径距离
    for i in range(numant):#
        m,n = bestpath[i],bestpath[i+1]
        print m,n
        plt.plot([coordinates[m][0],coordinates[n][0]], [coordinates[m][1],coordinates[n][1]
]], 'k')
        plt.plot([coordinates[bestpath[0]][0],coordinates[n][0]], [coordinates[bestpath[0]]
[1],coordinates[n][1]], 'b')
    ax=plt.gca()
    ax.set_title("Best Path")
    ax.set_xlabel('X axis')
    ax.set_ylabel('Y axis')
    plt.savefig('Best Path.png',dpi=500,bbox_inches='tight')
    plt.close()
    序列比对搜索:
    //动态规划 递归 序列比对
    //得分矩阵: 匹配+1, 失配 0, 空位-1
    # include <stdio.h>
    # include <iostream.h>
    # define SEQULEN 23
    int scores[SEQULEN][SEQULEN]={0};
    char sequ1[SEQULEN+1]="ergtrgryhythh6uhjyy5",
        sequ2[SEQULEN+1]="54h56hgtrh56h65j64";
    int p(int i,int j);
    int score(int i,int j);
    int max(int a,int b,int c);

```

```

void main()
{
    cout<<"sequ1:"<<sequ1<<endl;
    cout<<"sequ2:"<<sequ2<<endl;

    cout<<" |"<<sequ1<<endl;
    cout<<"-+"<<"-----"<<endl;
    for(int i=0;i<SEQULEN;i++)
    {
        cout<<sequ2[i]<<"|";
        for(int j=0;j<SEQULEN;j++)
        {
            if(sequ2[i]==sequ1[j]) cout<<"*";
            else cout <<" ";
        }
        cout<<endl;
    }

    for(i=0;i<SEQULEN+1;i++)
    {
        for(int j=0;j<SEQULEN+1;j++)
            cout<<score(i, j)<<" ";
        cout<<endl;
    }
    char cntn;
    cout<<"输入'f'+' 回车键' 继续"<<endl;
    cin>>cntn;
}

int score(int i, int j)
{
    if(i==0)
混合遗传蚁群:
clear
clc
Ant = 300;%蚂蚁数量
Times = 80;%移动次数
Rou = 0.9;%荷尔蒙发挥系数
P0 = 0.2;%转移概率
Lower_1 = -1;%搜索范围
Upper_1 = 1;
Lower_2 = -1;
Upper_2 = 1;

for i=1:Ant
    X(i, 1)=(Lower_1+(Upper_1-Lower_1)*rand);
    X(i, 2)=(Lower_1+(Upper_2-Lower_2)*rand);
    Tau(i)=F(X(i, 1), X(i, 2));
end

step=0.05;

```

参赛队号#1928

```

f='-(x.^2+3*y.^4-0.2*cos(3*pi*x)-0.4*cos(4*pi*y)+0.6)';

[x,y]=meshgrid(Lower_1:step:Upper_1,Lower_2:step:Upper_2);
z=eval(f);
figure(1);
subplot(1,2,1);
mesh(x,y,z);
hold on;
plot3(X(:,1),X(:,2),Tau,'k*')
hold on;
text(0.1,0.8,-0.1,'蚂蚁的初始位置分布');
xlabel('x');ylabel('y');zlabel('f(x,y)');

for T=1:Times
    lamda=1/T;
    [Tau_Best(T),BestIndex]=max(Tau);
    for i=1:Ant
        P(T,i)=(Tau(BestIndex)-Tau(i))/Tau(BestIndex);%计算转移状态概率
    end
    for i=1:Ant
        if P(T,i)<P0%局部搜索
            temp1=X(i,1)+(2*rand-1)*lamda;
            temp2=X(i,2)+(2*rand-1)*lamda;
        else%全局搜索
            temp1=X(i,1)+(Upper_1-Lower_1)*(rand-0.5);
            temp2=X(i,2)+(Upper_2-Lower_2)*(rand-0.5);
        end
        if temp1<Lower_1%越界处理
            temp1=Lower_1;
        end
        if temp1>Upper_1
            temp1=Upper_1;
        end
        if temp2<Lower_2
            temp2=Lower_2;
        end
        if temp2>Upper_2
            temp2=Upper_2;
        end

        if F(temp1,temp2)>F(X(i,1),X(i,2))%更新位置
            X(i,1)=temp1;
            X(i,2)=temp2;
        end
    end
    for i=1:Ant
        Tau(i)=(1-Rou)*Tau(i)+F(X(i,1),X(i,2));%更新荷尔蒙
    end
end

```



```

subplot(1,2,2);
mesh(x,y,z);
hold on;
x=X(:,1);
y=X(:,2);
plot3(x,y,eval(f),'k*');
hold on;
text(0.1,0.8,-0.1,'蚂蚁的最终位置分布');
xlabel('x');ylabel('y');zlabel('f(x,y)');

[max_value,max_index]=max(Tau);
maxX=X(max_index,1);
maxY=X(max_index,2);
maxValue=F(X(max_index,1),X(max_index,2));

function f = F(x,y)
f = -(x.^2+3*y.^4-0.2*cos(3*pi*x)-0.4*cos(4*pi*y)+0.6);
function main()
G=[0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 1 1 1 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0;
    0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0;
    0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1 1 1 0;
    0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1 1 1 0;
    1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0;
    1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0;
    0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0;
    0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0;];
MM=size(G,1); % G 地形图为 01 矩阵，如果为 1 表示障碍物
Tau=ones(MM*MM,MM*MM); % Tau 初始信息素矩阵
Tau=8.*Tau;
K=100; %迭代次数（指蚂蚁出动多少波）
M=50; %蚂蚁个数
S=1; %最短路径的起始点
E=MM*MM; %最短路径的目的点
Alpha=1; % Alpha 表征信息素重要程度的参数
Beta=7; % Beta 表征启发式因子重要程度的参数
Rho=0.3; % Rho 信息素蒸发系数
Q=1; % Q 信息素增加强度系数
minkl=inf;

```

```

mink=0;
minl=0;
D=G2D(G);
N=size(D,1);           %N 表示问题的规模（像素个数）
a=1;                    %小方格像素的边长
Ex=a*(mod(E,MM)-0.5);   %终止点横坐标
if Ex==0.5
Ex=MM-0.5;
end
Ey=a*(MM+0.5-ceil(E/MM)); %终止点纵坐标
Eta=zeros(N);           %启发式信息，取为至目标点的直线距离的倒数
%以下启发式信息矩阵
for i=1:N
ix=a*(mod(i,MM)-0.5);
if ix==0.5
ix=MM-0.5;
end
iy=a*(MM+0.5-ceil(i/MM));
if i~=E
Eta(i)=1/((ix-Ex)^2+(iy-Ey)^2)^0.5;
else
Eta(i)=100;
end
end
end
ROUTES=cell(K,M);       %用细胞结构存储每一代的每一只蚂蚁的爬行路线
PL=zeros(K,M);          %用矩阵存储每一代的每一只蚂蚁的爬行路线长度
%启动 K 轮蚂蚁觅食活动，每轮派出 M 只蚂蚁

for k=1:K
for m=1:M
%状态初始化
W=S;                    %当前节点初始化为起始点
Path=S;                  %爬行路线初始化
PLkm=0;                  %爬行路线长度初始化
TABUkm=ones(N);          %禁忌表初始化
TABUkm(S)=0;             %已经在初始点了，因此要排除
DD=D;                    %邻接矩阵初始化
%下一步可以前往的节点
DW=DD(W,:);
DW1=find(DW);
for j=1:length(DW1)
if TABUkm(DW1(j))==0
DW(DW1(j))=0;
end
end
end
LJD=find(DW);
Len_LJD=length(LJD); %可选节点的个数
%蚂蚁未遇到食物或者陷入死胡同或者觅食停止
while W~=E&&Len_LJD>=1
%转轮赌法选择下一步怎么走
PP=zeros(Len_LJD);

```

```

for i=1:Len_LJD
    PP(i)=(Tau(W, LJD(i)) ^Alpha)*((Eta(LJD(i))) ^Beta);
end
sumpp=sum(PP);
PP=PP/sumpp;%建立概率分布
Pcum(1)=PP(1);
    for i=2:Len_LJD
        Pcum(i)=Pcum(i-1)+PP(i);
    end
Select=find(Pcum>=rand);
to_visit=LJD(Select(1));
%状态更新和记录
Path=[Path,to_visit]; %路径增加
PLkm=PLkm+DD(W,to_visit); %路径长度增加
W=to_visit; %蚂蚁移到下一个节点
    for kk=1:N
        if TABUkm(kk)==0
            DD(W,kk)=0;
            DD(kk,W)=0;
        end
    end
    end
TABUkm(W)=0; %已访问过的节点从禁忌表中删除
    DW=DD(W,:);
    DW1=find(DW);
    for j=1:length(DW1)
        if TABUkm(DW1(j))==0
            DW(j)=0;
        end
    end
    end
LJD=find(DW);
Len_LJD=length(LJD);%可选节点的个数
end
%记下每一代每一只蚂蚁的觅食路线和路线长度
ROUTES{k,m}=Path;
    if Path(end)==E
        PL(k,m)=PLkm;
        if PLkm<minkl
            mink=k;minl=m;minkl=PLkm;
        end
    end
    else
        PL(k,m)=0;
    end
end
end
%更新信息素
Delta_Tau=zeros(N,N);%更新量初始化
    for m=1:M
        if PL(k,m)
            ROUT=ROUTES{k,m};
            TS=length(ROUT)-1;%跳数
            PL_km=PL(k,m);

```

```

        for s=1:TS
            x=ROUT(s);
            y=ROUT(s+1);
            Delta_Tau(x,y)=Delta_Tau(x,y)+Q/PL_km;
            Delta_Tau(y,x)=Delta_Tau(y,x)+Q/PL_km;
        end
    end
end
Tau=(1-Rho).*Tau+Delta_Tau;%信息素挥发一部分，新增加一部分
end
%绘图
plotif=1;%是否绘图的控制参数
if plotif==1 %绘收敛曲线
    minPL=zeros(K);
    for i=1:K
        PLK=PL(i,:);
        Nonzero=find(PLK);
        PLKPLK=PLK(Nonzero);
        minPL(i)=min(PLKPLK);
    end
figure(1)
plot(minPL);
hold on
grid on
title('收敛曲线变化趋势');
xlabel('迭代次数');
ylabel('最小路径长度'); %绘爬行图
figure(2)
axis([0,MM,0,MM])
for i=1:MM
    for j=1:MM
        if G(i,j)==1
            x1=j-1;y1=MM-i;
            x2=j;y2=MM-i;
            x3=j;y3=MM-i+1;
            x4=j-1;y4=MM-i+1;
            fill([x1,x2,x3,x4],[y1,y2,y3,y4],[0.2,0.2,0.2]);
            hold on
        else
            x1=j-1;y1=MM-i;
            x2=j;y2=MM-i;
            x3=j;y3=MM-i+1;
            x4=j-1;y4=MM-i+1;
            fill([x1,x2,x3,x4],[y1,y2,y3,y4],[1,1,1]);
            hold on
        end
    end
end
hold on
title('运动轨迹');

```

```

xlabel('坐标 x');
ylabel('坐标 y');
ROUT=ROUTES{mink,minl};
LENROUT=length(ROUT);
Rx=ROUT;
Ry=ROUT;
for ii=1:LENROUT
Rx(ii)=a*(mod(ROUT(ii),MM)-0.5);
if Rx(ii)==-0.5
Rx(ii)=MM-0.5;
end
Ry(ii)=a*(MM+0.5-ceil(ROUT(ii)/MM));
end
plot(Rx,Ry)
end
plotif2=0;%绘各代蚂蚁爬行图
if plotif2==1
figure(3)
axis([0,MM,0,MM])
for i=1:MM
for j=1:MM
if G(i,j)==1
x1=j-1;y1=MM-i;
x2=j;y2=MM-i;
x3=j;y3=MM-i+1;
x4=j-1;y4=MM-i+1;
fill([x1,x2,x3,x4],[y1,y2,y3,y4],[0.2,0.2,0.2]);
hold on
else
x1=j-1;y1=MM-i;
x2=j;y2=MM-i;
x3=j;y3=MM-i+1;
x4=j-1;y4=MM-i+1;
fill([x1,x2,x3,x4],[y1,y2,y3,y4],[1,1,1]);
hold on
end
end
end
for k=1:K
PLK=PL(k,:);
minPLK=min(PLK);
pos=find(PLK==minPLK);
m=pos(1);
ROUT=ROUTES{k,m};
LENROUT=length(ROUT);
Rx=ROUT;
Ry=ROUT;
for ii=1:LENROUT
Rx(ii)=a*(mod(ROUT(ii),MM)-0.5);
if Rx(ii)==-0.5

```

```

Rx(ii)=MM-0.5;
end
Ry(ii)=a*(MM+0.5-ceil(ROUT(ii)/MM));
end
plot(Rx,Ry)
hold on
end
end
function D=G2D(G)
l=size(G,1);
D=zeros(l*l,l*l);
for i=1:l
    for j=1:l
        if G(i,j)==0
            for m=1:l
                for n=1:l
                    if G(m,n)==0
                        im=abs(i-m);jn=abs(j-n);
                        if im+jn==1 || (im==1&&jn==1)
                            D((i-1)*l+j,(m-1)*l+n)=(im+jn)^0.5;
                        end
                    end
                end
            end
        end
    end
end
end
end
end

```