基于 Ant Colony 的基因序列分析方法

1 引言

意大利学者 M.Dorigo 等人在 20 世纪 90 年代时，根据研究蚂蚁群体觅食的特性，提出了模拟蚂蚁在觅食时通过关键物质“信息素”来寻找最短路径过程的 ACO（Ant Colony Optimazation）算法【1】【2】。该算法是一种启发式算法，其最显著的特点是并行计算与极强的鲁棒性，并被广泛用于解决如以旅行商问题（traveling salesman problem）为代表的组合优化问题【3】。与贪心算法或模拟退火算法等启发式算法一样，蚂蚁算法也存在容易陷入局部最优解的问题【4】。为了解决该问题，Dorigo 等其他学者在 ACO 的基础上提出了改进蚂蚁算法，其中应用最为广泛的有 MMAS（MAX-MIN Ant System）【5】，ACS（Ant Colony System）,P-ACO(Population-Based Ant Colony Optimization)【6】，ACE（Ant Colony Extended）【7】。这些算法通过限制信息素浓度，优化评分策略等方式提升算法性能，并都被证实在 TSP 问题上有更好的表现。

序列比对作为生物信息学的主要问题之一，取得了很多成果[8]。传统的序列比对算法均是通过动态规划算法进行求解，如 Needleman 与 Wunsch 提出了 Needleman-Wunsch 算法【9】，Smith 和 Waterman 通过引入空位罚分机制提出了 Smith-Waterman 算法【10】。动态规划算法要求遍历解空间，随着技术的发展，生物序列长度不断增加，动态规划的效率无法满足研究者的需求。为了解决该问题，科学家们开发了不同的启发式动态规划算法，其中主要两种方法是基于 hash 表的 seed 方法和基于后缀树的 LCS 方法。这其中很多算法都成为现在主流序列比对软件的核心算法：如通过将所有序列片段建立 Hash 表进行查询的 FASTA【11】，和通过设立种子（seeds）作为有代表性的序列集合充当部分关键路径的 BLAST 系列【12】【13】【14】。

因此本文提出将蚂蚁算法应用于序列比对算法的思路，通过应用蚂蚁算法并行计算的特性解决算法效率的问题，并根据算法过程中不断自适应调整寻径策略来提高准确性。本文将详细介绍蚂蚁算法原理及其在序列比对算法的应用，展示实验结果及结果分析，并最后进行总结。

2 DNA 序列比对问题描述

DNA 序列问题比对是生物信息学领域的重要研究领域。

序列 S 和 T 是有限长度的字符串，每个字符串由若干个 A，T，C，G 字符构成，并有特定的排列顺序。S[i]表示字符串 S 中第 i 个字符。用“—”来表示由删除插入而产生的空位。对于字符串中的元素 x ∈ S 和 y ∈ T,定义 $f(x, y)$为计

分函数。规定 S’和 T’是某一匹配结果，将 score(S’,T’)定义为 S 和 T 的某个匹配得分。对其中最大的 score(S’，T’)值，作为两个序列的最优比对。以下给出计分函数的一种简单形式:

$$f(x,y)=\begin{cases}2, x=y\in\{A,T,C,G\}\\-1, x\neq y\in\{A,T,C,G\}\\-2, x=' -' or\ y=' -'\end{cases} \quad (\ )$$

采用以上计分方式，对序列 S=“GCAATTC”与 T=“GGATTAC”的某个比对结果如图 2.1 所示：

表 2.1 序列 S=GCAATTC 与序列 T=G GATTAC 的一种比对结果

序列 S	G	G	A	A	T	T	-	C
序列 T	G	G	A	-	T	T	A	C
得分	2	-1	2	-2	2	2	-1	2

因此序列 S=“GCAATTC”与 T=“GGATTAC”的其中一个得分则为 score=6。

3 蚂蚁算法在序列比对问题的应用

3.1 蚂蚁算法的原理

算法的仿生学原理如下【15】:

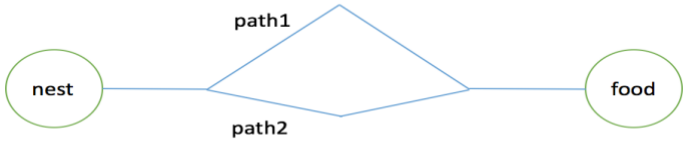

图 3.1 Ant Colony 算法的仿生学原理

蚂蚁在巢穴和食物间进行往返，并在路途中设置了不同长度的路径 1，2 其中路径 2 短于路径 1，该目的是使蚂蚁在觅食过程中必须进行选择。蚂蚁在从巢穴往食物或其他地方的过程中，在路上释放了称为“信息素”（pheromone）的物质。当遭遇了路口时，它们通过感知两条路径上信息素的浓度进行选择。在觅食最开始，由于两条路径上由于都未被放置过信息素（即两条路上的信息素浓度均为 0），因此蚂蚁将以相等概率随机选择两条路径中的一条。如图 1.1 所示，由于路径 2 的长度要短于路径 1，因此蚂蚁在路径 2 上来回更快。当它们回到巢穴，重新达到交叉口时，由于在路径 2 上耗时更少，信息素蒸发得少，更多的信息素被保留在路径 2 上。因此在感受不同信息素浓度的情况下，蚂蚁将会选择保留更多信息素的路径 2 而不选择信息素较少的路径 1。当路径 2 的信息素浓度远高于路径 1 时，所有蚂蚁都将选择路径 2，并最终成为所有蚂蚁觅食的路径。同时，伴随着两条路径的长度差逐步变大，蚂蚁也将更快地聚集在路径 2

计算机完全模拟真实蚂蚁存在一些弊端【15】:

① 真实蚂蚁寻觅路径唯一的方式是通过感知路径上的信息素浓度(pheromone)在觅食最开始时，由于路径上没有信息素，蚂蚁在做选择时无法得到指引，因此它的选择都是随机的，这种随机性可能造成一定程度上的误导；而在之后的过程中，蚂蚁也只会依靠之前留下的信息素来选择路径。

② 真实蚂蚁并不存在“记忆”等辅助方向选择的手段，因此信息素的分布对路径搜索的效率有至关重要的作用。作为正反馈机制的信息素，虽然是引导蚂蚁前行的有效手段，但在一定程度也会造成“盲目性”，从而降低了觅食效率。

③ 真实蚂蚁和信息素都是连续变化的，这对计算机模拟蚂蚁进行判断起到了一定影响，也是算法需要改进的地方。

因此，根据真实蚂蚁存在的上述缺陷，在以不破坏真实蚁群行为基础的前提下进行改进，提出了人工蚂蚁：

① 在对方向的判定上不完全依赖于信息素，加入了启发信息等机制来限制信息素对蚂蚁的正反馈效应。

② 为蚂蚁添加“记忆”能力，能够记录访问的节点或选择过的路径。

③ 人工蚂蚁以离散时间为单位进行活动。

3.2 蚂蚁序列比对算法的设计

对于序列 S=“GAATTC”与 T=“GAATC”，按照动态规划算法的原理为其构建矩阵，结果如图 3.2 所示：

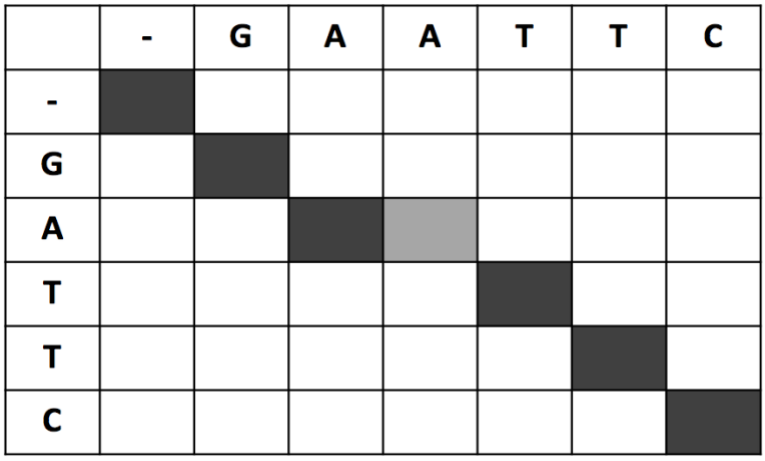

图 3.2 序列 S=GAATTC 与序列 T=GAATTC 的一种匹配情况

将矩阵左上角视为巢穴，右下角视为食物。蚂蚁从左上角出发，行进时共有四种情况：

（1）蚂蚁往水平方向向右移动至(i，j+1)，产生空位，得分为 a。

（2）蚂蚁往垂直方向向下移动至(i+1，j)，产生空位，得分为 a。

（3）蚂蚁向右下角移动至(i+1，j+1)，字符串相应位置字符不同，发生替换，

得分为 b。

（4）蚂蚁向右下角移动至(i+1，j+1)，字符串相应位置字符相同，得分为 c。

直到达到矩阵最右下角，完成一次比对，得到 S’与 T’。这其中分值最高的 S’和 T’即为最佳比对结果。

由于序列比对问题的方向选择只需要向右下方拓展，因此并不需要像 TSP 问题一样设置禁忌表(taboo)。

比对算法包含以下基本步骤：

（1）初始化:搜索轮次数NC=1;置所有蚂蚁的标志flag=0;

（2）对蚂蚁z=1, 2,...m: (A) 当蚂蚁未到达终点时,通过方向选择函数计算选择的方向，并将蚂蚁移动到新位置。 (B) 如果蚂蚁到达矩阵最右下角,置flag=1。

（3）n只蚂蚁都到达终点后,通过计分公式计算每只蚂蚁记录的序列得分，并以此计算该路径的信息素增量；记录本轮寻优找到的最优解。

（4）将所有蚂蚁初始化。

（5）如果轮次超过最大轮次N_{max}，或出现收敛，输出结果；否则,回（2）。

比对算法的具体设计如下：

共 m 只蚂蚁从左上角出发，其行进的三个方向 k=0，1，2 分别代表垂直向下，水平向右和向右下角。用(i，j)表示蚂蚁所处的位置，将蚂蚁在 t 时刻所处位置对三个方向的选择定义一个分数【16】:

$$Score_{ijk} = \tau_{ijk}(t)^{\alpha} M_k^{\beta} {d_p}^{\gamma} \tag{3.1}$$

其中，$\tau_{ijk}(t)$表示在 t 时刻(i，j)位置上沿 k 方向的信息素浓度。

M_k（k=0，1，2，3）是启发信息函数，它与蚂蚁下一次移动将产生的变化有关，共有以下三种情况：

① k=0 或 1 时，蚂蚁向下或向右移动，将引入空位，将 M 设置为空位罚分值的绝对值的倒数 0.5。

② k=2，蚂蚁向右下角移动，将发生替换，将 M 设置为替换罚分值的绝对值的倒数 1。

③ k=2，蚂蚁向右下移动，两字符相同，将 M 设置为匹配得分值 2。

另一启发信息函数 d_p 是用来控制蚂蚁偏离主对角线的方向，其意义即控制比对中空位的数量。对给定长度的序列 S 和 T，我们设置一个可接受的偏差值 D，同时我们对蚂蚁的实时位置(i，j)进行监测：

① 当蚂蚁的位置(i，j)中 i 和 j 的差值的绝对值小于 D，则认为蚂蚁的位置仍处于可接受范围内，对三个方向赋予相同的 d 值 d_0。

② 当蚂蚁偏离了对角线且位置偏下，即|i-j|>H 且 i>j 时，则为向右和向右下方向分配较大的 d 值 d_1。

③ 当蚂蚁偏离了对角线且位置偏右，即|i-j|>H 且 i<j 时，则为向下和向右

下方向分配较大的 d 值 d_l。

通过该函数可以限制蚂蚁大致在一个“沿对角线”的方向行进，同时可以限制引入空位的数量。α，β，γ则是分配给三个函数在方向选择时的权值，体现了其对决策的影响：

① 当 α 大于另两个权值，则表明在选择方向时蚂蚁主要依靠信息素的浓度进行判断，在搜索早期影响不大，但在中期容易陷入较优解而错过最优解。

② 当 β 大于另两个权值，则表明在选择方向时蚂蚁主要靠下一位置所引起的变化进行判断，类似于“贪心”策略，较容易陷入局部最优。

③ 当 γ 大于另外两个权值，则表明在选择时蚂蚁比较注重空位限制，对于扩展解空间的搜索有一定帮助，但可能会减缓收敛速度。

因此需要考虑空位，启发信息和空位三者对结果的影响。在选择方向时则选择公式(4.1)中 Score 值最大的方向。

对 m 只蚂蚁得到的序列进行排序，并对排序结果分层，对前 m/4 只蚂蚁赋予信息素更新值 Q1，对位于 m/4-m/2 的蚂蚁赋予信息素更新值 Q2，以此类推。定义 ρ∈(0，1)为衰减系数，表明信息素的挥发程度，其更新公式如下：

$$\tau_{ijk}(t+n) = \rho \times \tau_{ij}(t) + \Delta\tau_{ijk}^{z} \tag{3.2}$$

其中$\Delta\tau_{ijk}^{z}$是第 t 轮搜索过后的信息素增量，其值为：

$$\Delta\tau_{ijk}^{z} = \begin{cases} Q_i, if \text{ 蚂蚁} z \text{在} (i,j) \text{选择了方向 k} \\ 0\text{，蚂蚁未在} (i,j) \text{选择方向} k \end{cases} \tag{3.3}$$

其中 Q_i 值即是前文提到的信息素更新值 Q。该更新公式即是 ant-cycle 模型。将路径长短的信息替换为了序列比对的优劣程度，用以反应全局信息，能加快找到最优解的速度。

蚂蚁搜索到的序列长短不一，其最终到达右下角的时间也不尽相同，因此无法设置统一的时序信号。因此为每只蚂蚁加上标记 flag=0，表示搜索过程未完成，当达到右下角时，使 flag=1 表示搜索完成。当所有蚂蚁的 flag 都为 1，表示该轮搜索完成，对每只蚂蚁的序列统计分数并进行信息素更新。

由于启发式算法并没有遍历解空间，因此需要判定其收敛条件或控制其终止时间。在该算法中，我们限定搜索次数超出了次数上限或得到的分数大于预期分数，即判断算法终止。预期分数的公式如下：

$$Score_{exp} = \min(|S|, |T|) \times 2 \times G \tag{3.4}$$

这样的判定较为不精确，因为无法确定在之后的搜索中是否还能找到更好的解，容易陷入较优解（甚至结果较差的解）就停止。

算法伪代码如图4.2所示：

```
1  procedure Ant Colony Sequence Alignment（ants ）
2     t=ants.id，create_PheromoneTable();
3        for each i∈|S|*|T|,j∈{0，1，2},do τij=1
4           while(（loop<Nmax）||(bestScore<scoreexp))
5              for each ant[t]∈ants
6                 while（ant[t].flag!=0）do Score_ijk = τ_ijk(t)^α M_k^β d_p^γ;
7                    ant[t].[i,j]=JudgePosition(scoreijk);
8                    if (ant[t].position==[i,j]) do ant[t].flag==1
9                 end while
10             τ_ij(t+n) = ρ×τ_ij(t) + Δτ_ij;
11          if（bestScoreLoop>10） do break;
12        end while
13    free_all_ants();
14 end procedure
```

图 4.2 Ant Colony 算法的伪代码

通过改变参数，可以改变蚂蚁方向选择的策略【17】。根据公式(3.1)，在蚂蚁所处位置失去启发信息 M 值的指引时，蚂蚁的选择就显得比较随机，同时由于 d 限制了空位的上限数量，所以当引入空位数量过多时会使蚂蚁被强行扭转移动方向，从而没法得到最优解。简而言之，控制蚂蚁在局部进行选择的参数 M_k 和控制空位个数的参数 d 之间存在一定的矛盾性，因此需要对其进行调和。同时，算法由于只存在“信息素”这一正反馈机制而不存在校正机制，因此当一定时间后某路径上的信息素浓度达到了一个优势值，蚂蚁无法再选择别的路径，容易陷入局部最优解。实验结果将在第 5 章进行展示。

4 改进自适应蚂蚁比对算法

4.1 改进比对算法设计：

由前文所述，蚂蚁算法容易停滞于局部最优解而降低求解效率【18】。本文借鉴了模拟退火算法 (Simulated Annealing，SA)的思想提出自适应蚂蚁比对算法，动态调整信息素分配及优化方向选择【19】【20】。

模拟退火算法由 Kirkpatrick 在 1982 年【21】，根据固体退火的原理而进行模仿，并引入组合优化领域。算法的基本思想是：固体升温时，其内部的粒子随着温度的不断升高而逐步朝无序的状态变化；在逐步降温后，内部的粒子又重新趋于稳定，并最终在某个温度（如常温）达到平衡态。根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为：

$$p=e^{\left(-\frac{\Delta E}{kT}\right)} \tag{4.1}$$

其中 E 为温度 T 时的内能，ΔE 为其改变量，k 为 Boltzmann 常数。

其在组合优化问题上的应用，是将目标函数值视为物体的内能，温度 T 则

是调控参数 t。通过不断进行“产生新解—计算与目标函数的差值—选择或舍弃”，逐步降温衰减 t 值，直到达到某个温度后得到的解则为近似最优解。其最主要的思想就是以一定概率接受一个新解，通过该新解可以拓宽解空间，从而减少陷入局部最优的概率【22】【23】。

前述算法中，蚂蚁在进行方向选择时完全依靠 Score 值进行选择，因此很容易出现过快收敛于某几个解的情况。为此我们提出如下改进方法：

① 对每轮蚂蚁搜索到的解进行监控，统计其出现频率最高的解的路径，分数以及在该轮的占比，并一直记录从搜索开始找到的最优解。

② 对给定的序列 S 和序列 T，设置期望分数 Sp 是根据两序列中较短的序列长度来确定的，其公式一般为：

$$Score_{exp} = \min(|S|, |T|) \times 2 \times G \tag{4.2}$$

式中 $Score_{exp}$ 代表预期分数，G 则代表研究者预设的比例系数，由于使用的是公式（2.1），所以最高的预期分数即长度较短序列长度的两倍。

③ 引入退火值 r 作为控制机制：在经过一定轮次后，当某轮蚂蚁 70%（或其他阈值）的解都集中在某个特定解上时，对该解和预期函数值进行比较，如果两者差值较小，表明该解的得分较为满意，但仍可以上升，因此则对当前最优路径上另两个方向上加入 r 值，以加大该两个方向的分数；当两个分数的差值过大，表明该收敛解并不好，对其所在的路径进行“升温”操作，即破坏和该解有关的信息素表的结构，这将在之后阐述。具体操作即是在该路径上对除已选择方向的令两个方向乘上因子 r，扩大其被选择的概率。

④ 对方向选择函数添加一个界限值 q∈(0，1)，并在蚂蚁每次选择路径时生成一个随机值 p_0∈(0，1)。当 p_0<q 时，根据直接选择 $Score_{ijk}$ 中最大值所代表的方向；当 p_0>q，则根据三个方向的得分值比例来选择方向：比如当方向 k=0，1，2 的比例为 1：1：1 时，则各以 1/3 的概率选择方向 k=0，1，2；当方向 k=0，1，2 的比例为 1：1：2，则以 1/4,1/4 的概率选择方向 k=0，k=1，以 1/2 的概率选择方向 k=2。

因此，通过改进后的方向函数为：

$$Score_{ijk} = \tau_{ijk}(t)^{\alpha} M_k^{\beta} {d_p}^{\gamma} r^{\theta} \tag{4.3}$$

其中 r 值为退火值，用来“扰乱”蚂蚁的选择。Θ 是控制该扰乱因素作用大小的权值。

p<q 时，选择最大 $Score_{ijk}$ 值代表的方向；p>q 时，通过如下公式进行选择：

$$p_{ijk} = Score_{ijk} / \sum_{k=0}^{2} Score_{ijk} \tag{4.4}$$

该改进方式的意义是：增大了选择之前被舍弃解的概率，增大了解空间。通过调整 M 值的权重也可以达到此效果，但容易影响初期对较优解的搜索，从而

减慢了收敛速度。

在对蚂蚁的方向选择策略进行改进的同时，也可以对信息素表进行改进。具体做法是通过对当前找到的最优解和预期解进行比较判断，并根据差值对信息素表进行一定程度的破坏：当找到的最优解和预期分数达到一定差值，而大部分蚂蚁又已收敛在该路径上，则对该路径上各方向的信息素值按照一定比例进行重置，如该路径上的方向是 k=2，则表明该位置 k=2 的信息素值已远大于 k=0 和 k=1，因此可以按照 k=0，k=1，k=2 比值为 1，1，2 的比例重新分配信息素，这样就加大了其搜索的空间。由于在之前的操作已保存了当前和全部搜索过程中的最优比对值，因此并不需要担心在扰动后导致解收敛在了更差的结果上。

由于算法中的方向函数都有多个幂函数，因此过大的浓度差会导致结果无法逆转。其意义是使某个方向的浓度无法永远成为优势方向，其余方向在通过前述基于模拟退火思想的方向选择和优势方向的信息素衰减后，可以逐步缩小与优势方向的差距，从而扩大搜索的范围。因此不妨将信息素规定在[1，2]区间内，当信息素更新后大于最大值或小于最小值时使它强制恢复为最大值与最小值。同时还需要改变信息素更新值 Q，将其投射于[0，1]区间，使其增加趋势不会过快，从而保证了解的多样性。本次实验的具体做法是将其与蚂蚁数量关联，如公式(4.5)所示：

$$\Delta\tau_{ijk}^{z}=\begin{cases}\mathrm{Q_i/m, if}\ \text{蚂蚁 z 在(i, j)选择了方向 k}\\ 0\ ,\ \text{蚂蚁未在(i, j)选择方向 k}\end{cases} \qquad (4.5)$$

同时引入时变因子 R(t)来控制信息素更新的增量【16】。在不同轮次中，动态调整 R(t)值：如在搜索早期赋予较小的 R(t)值，以此减缓搜索早期收敛过快的缺陷；在搜索后期逐步加大 R(t)值，以此加快搜索后期对最优匹配方向的收敛速度。

不妨将 T_i 作为搜索的时段，可将 R（t）值定义为如下：

$$R(t)=\begin{cases}Q1(0<t<T_1)\\ Q2(T_1<t<T_2)\\ \cdots.\\ Qn(t>T_{n-1})\end{cases} \qquad (4.6)$$

则调整后的信息素更新方式为：

$$\Delta\tau_{ijk}^{z}=\begin{cases}\mathrm{Q_i\times R(t), if}\ \text{蚂蚁 z 在(i, j)选择了方向 k}\\ 0\ ,\ \text{蚂蚁未在(i, j)选择方向 k}\end{cases} \qquad (4.7)$$

这样做的好处是，结合前述改进后的方向选择函数及“干扰信息素表”的方式，在前期可以极大地拓宽可选择的解，同时在后期可以通过“接受较差解”的思想再逐步逼近最优解。

4.2 收敛性分析：

蚂蚁算法一直没有公理化的收敛性证明。在文献【24】中提出了一种通过马尔科夫模型的证明方法，同时提到之前已提出的两种证明方法，分别是gutjahr【25】用于证明GBAS的方法和dorigo【26】证明MMAS的方法。本节将借鉴文献【24】中证明的思路，建立序列比对蚂蚁算法的马尔科夫序列模型，进而证明蚂蚁算法的收敛性。以下给出马尔科夫模型的定义：

定义1：设{x_t :t=0,1,2,……}是有限空间S={$a_1,a_2,a_3,….a_n$}的随机变量序列，若对任意k>=0及a_i，a_j，$a_{i_0},a_{i_1},a_{i_2}\ldots\ldots a_{i_{k-1}} \in S$，有

$$P(x_{k+1}=a_j|x_0=a_{i_0},x_1=a_{i_1},\ldots\ldots x_{k-1}=a_{i_{k-1}},x_k=a_i)=P(x_{k+1}=a_j|x_k=a_i) \quad (4.8)$$

则称{ x_t :t = 0,1,2,……}为一个马尔科夫链。条件概率P（$x_{t+1}=a_j|x_t=a_i$）称为该马尔科夫链在时刻t处于状态a_i的条件下，在时刻t+1转移到状态a_j的转移概率，记为p_{ij}（t）。

定义2：若转移概率和时间t无关，即对任意a_i，$a_j \in S$，和任两个时刻t_1，t_2，都有$p_{ij}(t_1)= p_{ij}(t_2)= p_{ij}$，则称该马尔科夫链是齐次的。

记X(n)={t(n),p(n)},其中p(n)表示n次觅食后该次得到的分数最高的匹配，τ(n)表示n次迭代后的信息素分布。由信息素更新的方式及寻找匹配的步骤，可以看出p(n)只与τ(n-1)有关，而τ(n)的取值与p(n)与τ(n-1)有关，因此X(n)的分布只与X(n-1)有关，满足马尔科夫性，因此认为X(n)={p(n),t(n),n=0,1,2,…..}是离散的马尔科夫链。且根据定义2，{X(n),n=1,2,…}是齐次的马尔科夫链。

设在n次迭代中，算法至少一次得到最佳匹配的概率为B(n)。根据前文，信息素的浓度已被设置上下限，即$\tau_{ij} \in [\tau_{min},\tau_{max}]$。根据公式(4.3)与公式(4.4)，设匹配的最佳结果为w，对（i,j）∈w，算法从(i,j)选择下一个节点(m,n)的概率为p_{min}，其中m的取值为i或i+1，j的取值为j或j+1。不妨考虑最差情况，对(m,n)∈w，有$\tau_{ij\to mn}=\tau_{min}$，而对(m,n)∉w，有$\tau_{ij\to mn}=\tau_{max}$（即蚂蚁通过最小的信息素值选取了最优路径上的节点）。根据公式（4.4），可知$p_{min}>0$。则k次觅食后，蚂蚁得到最优解的概率为P（G）=（p_{min}）$^G>0$，其中G代表蚂蚁从原点到最佳匹配节点时选择方向的次数。因此在n次觅食中，算法得不到最优解概率为$[1-P(G)]^n$。因此$B(n)=1-[1-P(G)]^n$，当n足够大时，有任意小$\varepsilon=[1-P(G)]^n>0$,使$B(n)=1-\varepsilon=1$。即当$n\to\infty$时算法必可取到最优解。

4.3 时间复杂度和空间复杂度分析

假设给定的两条序列 S,T 的长度在同一数量级，以|S|和|T|来表示序列 S 和序列 T 的长度，且蚂蚁数量 m 由前文所述，远小于序列长度|S|或|T|，因此在计算复杂度时主要考虑序列长度。

算法首先设定二维数组形式的信息素表，其大小为|S|×|T|×3，该步骤的复杂度为O（n^2）。信息素表也是算法主要的空间消耗，因此算法的空间复杂度为O（n^2）。共m只蚂蚁进行搜索，每只蚂蚁可得到的最大路径长度为|S|+|T|，每次判断下一个方向需要计算该位置三个方向的分数，共需完成3次计算。因此对于m只蚂蚁，最多花费（|S|+|T|）×m×3的操作进行搜索，其时间复杂度为O（n×m）。其后计算m只蚂蚁得到序列的总分和信息素增量并进行信息素更新，耗费的时间为m×n^2，其时间复杂度为O（n^2）。第四步重置所有蚂蚁，其共需要m个操作，因此其时间复杂度为O（m）。最终，算法的轮次上限为N_{max}，所以其整个算法的时间最多为N_{max}。综上，可以认为整个算法的时间复杂度为O（N_{max}×n^2）。但考虑到算法存在收敛条件，因此并不会达到该数量级。

5 实验结果展示

5.1 仿真实验结果

实验 1 以 GCboc 序列进行测试，序列分别是 S=“GGCCGCC”,T=“GGCGCC”。其最优比对分数分别为 8，最佳匹配如图 5.1 所示。未改进的算法与改进后的算法同样极快地在 2 轮内找到最优解，并在 10 轮内收敛，算法效率无太大差别。

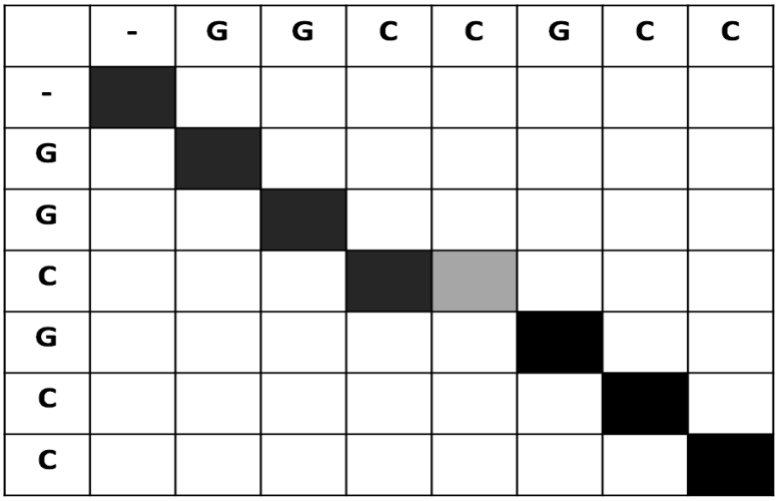

图 5.1 序列 S=GGCGCC 与序列 T=GGCCGCC 的一种最佳匹配

与未改进的蚂蚁算法出现明显差别的是实验 2，该实验的序列如下，其最高分为 81：

S: tgcgagcgtcctatgaaaaagtctagataactgagtcctgccgacgattatg
T: tggagcgtcctctgaaaaatgtcttagatactgagtctgccgtcgataatg

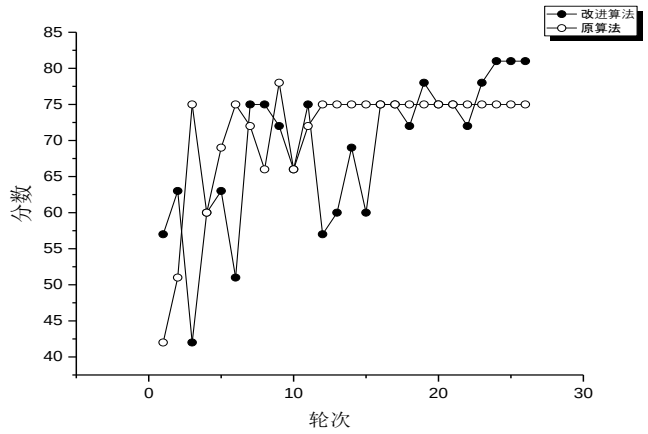

图 5.3 实验结果的折线图

图 5.1 展示了蚂蚁搜索过程中分数变化的过程。未改进的蚂蚁算法在 12 轮后开始收敛在较优解 75 并最终收敛于该值，而改进后的蚂蚁算法虽然收敛速度变慢，但于 22 轮找到最优值 81 并最终收敛于 81。从总体来看改进算法在最优解的搜索效率上比未改进的算法有很大提升，因此我们增大序列长度继续测试。

实验 3 是在实验 2 的序列上进行倍增和修改并且添加了空位。两个序列的长度均为 106.其最优比对得分为 180。由于序列较长，因此将 $Score_{exp}$ 的 G 值调整为 0.75 上下。实验结果显示，未改进的算法大多收敛在 150-160 分数的路径上，这与最优解仍有较大的差距。但改进后的算法在前期收敛于 150-160 分时由于未达到预期分数儿信息素表进行破坏后，蚂蚁重新进行搜索。可能在重复 1-2 次后蚂蚁重新跳出该分段的较优解，逐步往 170 分以上的路径靠拢，此时已满足 $Score_{exp}$ 的要求。若算法搜索到了分数更高的解则进行输出，否则就输出该收敛解。

改进算法的仿真试验结果如表 5.1 所示。可见该算法有较好的准确性。

表 5.1 改进算法的所有实验结果

实验编号	序列长度	最高分	最低分	平均分	最佳比对
1	7	8	8	8	8
2	50	81	78	80.1	81
3	100	180	174	176.25	180

5.2 数据库实验结果

本次实验选择的是 Ebola 病毒亚种的 DNA 序列片段数据及从人类 1 号染色体的 DNA 序列片段中进行截取并进行修改的片段，每个序列的测试均进行 20 次并进行统计。

Ebola 病毒亚种的实验结果如表 5.2 所示。数据特征是两序列有相同长度，因此没有插入或删除对匹配进行影响。实验结果显示算法有很高的准确性，大部

分都极快收敛于最优解。最低分的出现是因为蚂蚁一定概率自发选择插入或删除的方向，从而影响了分数。

表 5.2 Ebola 病毒亚种序列实验结果

实验编号	序列长度	最高分	最低分	平均分	最佳比对
4	242	481	464	479.3	481
5	242	477	443	471.9	477
6	253	501	498	500.9	556

为了进一步验证准确性，从人类染色体 1 号上截取片段并进行修改进行实验。这三个实验在序列中加入了较长的空位，以验证空位对算法的影响。三个实验的实验结果如表 5.3 所示

表 5.3 人类染色体 1 号序列的实验结果

实验编号	序列长度	最高分	最低分	平均分	最佳比对
7	200	372	351	366.3	372
8	319	603	580	592.3	608
9	305	556	541	546	556

从实验结果可以看出，连续空位对算法影响比较大。因为在算法中蚂蚁无法自发调整方向选择函数各个参数的权重，因此当参数选择不佳时，可能会因为空位太多而选错方向；但若调低蚂蚁向右下方的权重，则可能会导致收敛速度变慢。

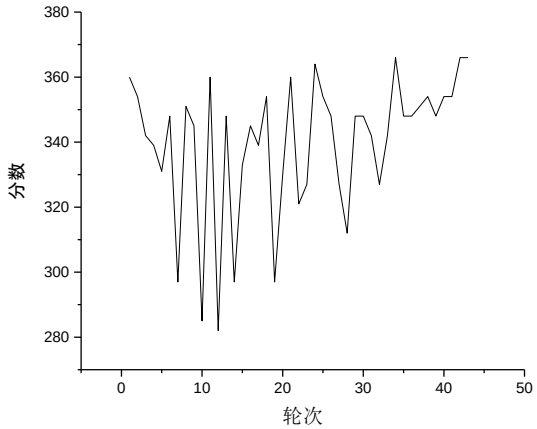

图 5.2 实验编号为 7 的实验结果折线图

图 5.2 展示了实验编号为 7 的序列比对结果，可以看出在经历了震荡后分数逐渐上升并最终还是收敛在了较高的分数上。同样的，实验 8 和实验 9 在序列中也加入了较多的连续空位，因此匹配结果并不如之前的实验结果理想。但对于实验 1，2，3，序列中的空位数量都在较少且较短，因此匹配的结果也较为理想。从总体看，改进后,算法比未改进算法在收敛效率和最优解的搜索能力上都提高了很多。

6 结论

Ant Colony 算法中包含的参数众多，如何协调各参数使算法发挥最佳功能是

很大的难题；同时提出的多种改进措施能否达到协同效果也需要再进行进一步研究。根据实验结果，基本算法在相似度较高的序列上效果尚可，但对于最优解的搜索效率仍有缺陷，可能无法从早期收敛到的较差结果中挣脱，同时对于亲缘性较低（如需要引入多个空位）的序列，效果并不理想。改进算法针对这两个缺陷进行改进，在搜索效率和精度上有显著的提升，但对于最优解的搜索精度上仍未达到理想效果。因此如何通过动态改变参数来提升准确性是一个新的难题。

7 参考文献

[1]D. Corne, M. Dorigo and F. Glover. The Ant Colony Optimization Meta-Heuristic[J]. New Ideas in Optimization, 1999:1-7.

[2] Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1):53-66.

[3]T. Stu¨tzle and M. Dorigo. ACO algorithms for the quadratic assignment problem. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, (McGraw-Hill, London, UK, 1999) 33–50.

[4]秦玲. 蚁群算法的改进与应用[D].扬州大学,2004.

[5] Tzle T, Hoos H H. MAX-MIN, Ant system[J]. Future Generation Computer Systems, 2000, 16(9):889-914.

[6] Angus D. Niching for Population-Based Ant Colony Optimization[C]// IEEE International Conference on E-Science and Grid Computing, 2006. E-Science. 2007:115-115.

[7]Escario J B, Jimenez J F, Giron-Sierra J M. Ant Colony Extended: Experiments on the Travelling Salesman Problem[J]. Expert Systems with Applications, 2015, 42(1):390-410.

［8］艾冬梅, 赵清玉, 张德坤. 生物序列比对算法综述[J].中国科技纵横,2013(18):78.

[9] Needleman S B, Wunsch C D. A General Method Applicable to Search for Similarities in Amino Acid Sequence of 2 Proteins[J]. Journal of Molecular Biology, 1970, 48(3):443-53.

[10] Smith T F, Waterman M S. Identification of common molecular subsequences[J]. Journal of Molecular Biology, 1981, 147(1):195–197.

[11]Pearson W R. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms.[J]. Genomics, 1991, 11(3):635-650.

[12]Altschul S F, Gish W, Miller W, et al. Basic Local Alignment Search Tool[J]. Journal of Molecular Biology, 1990, 215(3):403-10.

[13]Altschul S F, Madden T L, Schäffer A A, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.[J]. Nucleic Acids Research, 1997, 25(17):3389-402.

[14]Schwartz S, Kent W J, Smit A, et al. Human-mouse alignments with BLASTZ.[J]. Genome Research, 2003, 13(1):103-7.

[15]Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1):53-66.
[16]梁栋. 蚁群算法在生物序列比对中的应用[D]. 西安电子科技大学, 2004.
[17]叶志伟, 郑肇葆. 蚁群算法中参数α、β、ρ设置的研究——以TSP问题为例[J]. 武汉大学学报(信息科学版), 2004, 29(7):597-601.
[18]周明秀，程 科，汪正霞. 动态路径规划中的改进Ant Colony算法[J].计算机科学，2013,40(1):314-316.
[19]梁爽, 莫忠良.多序列联配问题的模拟退火求解[J]. 武汉大学学报（理学版）2002, 48(1):23-27.
[20]明华.基于模拟退火的多序列比对算法的研究[D]，西安电子科技大学，西安电子科技大学，2006.
[21]Kirkpatrick S. Optimization by Simulated Annealing[J]. Science, 1987, 220(4598):606-615.
[22]刘 阳,王小磊,李江域,毛逸清,赵东升 . 局部序列比对算法及其并行加速研究进展 [J].军事医学，2012,36(7):556-557.
[23]张淑萍.自适应Ant Colony算法在dna序列比对中应用研究[J].计算机仿真，2012,29(6):211-212.
[24] 朱勇. 一类改进蚂蚁算法收敛性分析及数值实验[D]. 上海交通大学, 2005.
[25] Gutjahr W J. A GENERALIZED CONVERGENCE RESULT FOR THE GRAPH-BASED ANT SYSTEM METAHEURISTIC[J]. Probability in the Engineering & Informational Sciences, 2003, 17(4):545-569.
[26] Stutzle T, Dorigo M. A short convergence proof for a class of ant colony optimization algorithms[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(4):358-365.