一种求解 MPMGOOC 问题的启发式算法

武优西[1),5)] 吴信东[2),5)] 江 贺[3),5)] 闵 帆[4),5)]

[1)](河北工业大学计算机科学与软件学院 天津 300130)

[2)](合肥工业大学计算机科学与信息工程学院 合肥 230009)

[3)](大连理工大学软件学院 辽宁 大连 116621)

[4)](漳州师范学院粒计算重点实验室 福建 漳州 363000)

[5)](佛蒙特大学计算机系 佛蒙特州 伯灵顿 05405 美国)

摘 要 具有间隙约束和一次性条件的最大模式匹配(Maximum Pattern Matching with Gaps and One-Off Condition, MPMGOOC)是一种具有通配符长度约束的模式匹配问题,其任务是寻找彼此互不相关的最多出现.文中基于一种新的非线性数据结构——网树,提出了一种解决 MPMGOOC 问题的启发式算法.与树结构不同之处在于,除根结点外,网树中任何结点可以多于 1 个双亲结点.文中给出了网树的定义及其相关的概念和性质.基于这些概念和性质,提出了一种选择较优出现(Selecting Better Occurrence,SBO)的启发式算法.该算法在搜索一个出现的循环中,采用了贪婪搜索双亲策略(Strategy of Greedy-Search Parent,SGSP)和最右双亲策略(Strategy of RightMost Parent,SRMP)寻找相同叶子的两个出现并选择其中较好的出现作为 SBO 算法的结果.SGSP 策略的核心思想是每一步都寻找当前结点的一个近似最优双亲(Approximately Optimimal Parent,AOP);SRMP 策略的核心思想是每一步都寻找当前结点的最右双亲结点.实验结果表明,在多数情况下 SBO 算法可以获得更好的解且解的质量较其它算法有显著的提高.文中不但提供了一个解决 MPMGOOC 问题的启发式算法,更重要的是对于求解其它复杂问题具有一定的参考价值.

关键词 模式匹配;通配符;一次性条件;网树;启发式算法

中图法分类号 TP301 **DOI 号**: 10.3724/SP.J.1016.2011.01452

A Heuristic Algorithm for MPMGOOC

WU You-Xi[1),5)] WU Xin-Dong[2),5)] JIANG He[3),5)] MIN Fan[4),5)]

[1)](*School of Computer Science and Software, Hebei University of Technology, Tianjin* 300130)

[2)](*School of Computer Science and Information Engineering, Hefei University of Technology, Hefei* 230009)

[3)](*School of Software, Dalian University of Technology, Dalian, Liaoning* 116621)

[4)](*Laboratory of Granular Computing, Zhangzhou Normal University, Zhangzhou, Fujian* 363000)

[5)](*Department of Computer Science, University of Vermont, Burlington, VT* 05405, *USA*)

Abstract Maximum Pattern Matching with Gaps and the One-Off Condition (MPMGOOC) is an interesting and challenging pattern matching problem, which seeks to find the maximal number of occurrences of a pattern in a sequence. In this paper, a heuristic algorithm based on a new nonlinear data structure, Nettree, is proposed for this problem. A Nettree is different from a regular tree in that a node may have more than one parent. The algorithm is named Selecting Better Occurrence (SBO). SBO uses some special concepts and properties of the Nettree to solve the task. In the loop of finding an occurrence, SBO uses two strategies, Strategy of Greedy-Search Parent (SGSP) and Strategy of RightMost Parent (SRMP) to find two occurrences with the same

收稿日期:2010-11-13;最终修改稿收到日期:2011-03-24.本课题得到国家自然科学基金(60828005)资助.**武优西**,男,1974 年生,博士,教授,主要研究领域为智能计算与数据挖掘.**吴信东**(通信作者),男,1963 年生,博士,教授,博士生导师,主要研究领域为数据挖掘、基于知识的系统和万维网信息探索.E-mail: xwu@hfut.edu.cn.**江 贺**,男,1980 年生,博士,副教授,博士生导师,主要研究领域为智能计算与数据挖掘.**闵 帆**,男,1973 年生,博士,副教授,主要研究方向为粗糙集、粒计算和序列模式挖掘.

leaf, and then selects a better occurrence from the results of SGSP and SRMP. The main ideas of SGSP and SRMP are to find an Approximately Optimal Parent (AOP) and the rightmost parent of the current node at each step in the process of searching for an occurrence, respectively. Extensive experimental results on real-world biological data demonstrate that SBO achieves the best performance among all competitive algorithms in terms of solution quality. This paper not only provides a heuristic solution for the MPMGOOC problem, but also shows that the Nettree can be used to solve other complex problems.

Keywords　pattern matching; wildcards; one-off condition; Nettree; heuristic algorithm

1 引　言

模式匹配问题(也称串匹配问题)是计算机科学的基本问题之一,它是网络安全[1]、信息检索与过滤[2]、文字处理[3]、计算生物学[4]等重要领域的核心问题,同时也是模式挖掘技术的核心与基础[5-7]. 而带有通配符的模式匹配与模式挖掘使得问题变得更加复杂[8]. 在带有通配符的模式匹配研究中,主要有两类研究成果,一类偏向于通配符通配的字符长度是固定的;另一类偏向于通配符通配的长度是有约束的. Fischer 和 Paterson[9] 在 1974 年第一次从理论复杂性角度对带有通配符的模式匹配问题进行了研究. 近年来,通配符通配的长度是可变的问题正引起人们的广泛关注. Manber 等人[10] 提出的算法有效地解决了只有单个可变长度通配符的模式匹配问题;He 等人[8] 致力于通配长度小于指定值的频度模式挖掘问题;Huang 等人[11] 致力于具有多个可变长度通配符和一次性条件的频度模式挖掘问题;Min 等人[12] 提出了一个算法 PAIG-RST (Reduced Space and Time),该算法有效地解决了具有多个可变长度通配符的模式匹配问题;Chen 等人[13] 提出了用于在线求解具有间隙约束和一次性条件的最大模式匹配问题(Maximum Pattern Matching with Gap and One-Off Constraints, MPMGOOC) 的 SAIL 算法,然而该算法不是一个完备的离线算法[11]. 本文提出的基于网树的启发式算法,可很好地求解离线 MPMGOOC 问题.

网树结构(简称网树)[14] 是对树结构进行拓展的一种新的非线性数据结构. 网树不但具有很多树结构的概念,如树根结点、叶子结点、双亲、孩子、路径、层等概念,而且除根结点外网树结构中任何结点可以有多个双亲结点. 为了解决 MPMGOOC 问题,本文先将该问题转化为一棵网树,并在网树上构造了许多新的概念和性质,基于这些概念和性质,提出了一个选择较优出现 SBO(Selecting Better Occurrence)的启发式算法. SBO 算法在计算一个出现的循环中,利用结点的最小根和最大根的概念和性质对问题的全局约束进行了考虑;利用结点的树根路径数、叶子路径数和树根-叶子路径数等概念与性质实现对位置出现数的计算. 在此基础上形成了一个寻找出现的策略——贪婪搜索双亲策略 SGSP (Strategy of Greedy Search Parent),该策略的核心思想是每一步都寻找当前结点的近似最优双亲结点 AOP(Approximate Optimization Parent). 此外,本文利用结点的最右双亲结点、最小根和最大根等概念实现了另外一个寻找出现的策略——最右双亲策略 SRMP(Strategy of RightMost Parent). SBO 算法择优使用 SGSP 和 SRMP 策略的结果并以此求解 MPMGOOC 问题. SBO 算法的空间复杂度和时间复杂度分别是 $O(W\times m\times n)$ 和 $O(W\times n\times(n+m\times m))$,这里 m,n 和 W 分别是模式串 P 和序列串 S 的长度以及模式串 P 的最大间距. 本文在实际生物数据上进行了 72 组对比性实验,实验结果显示:(1) 在其中 69 个实例中,SBO 算法的结果好于或持平于 SAIL 算法的结果;(2) SBO 算法较大幅度地提高了 SAIL 算法的解的质量;(3) 在其中 56 个实例上 SBO 算法取得了 4 个算法中最好的结果,而另外 16 个实例均十分接近最好解. 因此通常情况下,SBO 算法不但可以获得更好的解,而且解的质量较 SAIL 算法有较为显著的提高.

本文第 2 节论述问题的定义并通过实例来解释 MPMGOOC 问题及其相关概念;第 3 节给出网树的概念和性质;第 4 节给出 SBO 算法同时对 SBO 算法的时间复杂度和空间复杂度进行分析,用实例说明算法的工作原理;第 5 节给出实际生物数据的测试结果并对结果加以分析;第 6 节给出本文结论.

2 问题定义

尽管 Chen 等人[13]最早提出了该问题,但本节对该问题的定义进行重新描述,并增加一些新的概念和符号.通过一些实例对这些定义、概念和符号进行解释.

定义 1. 给定一个模式串 $P=p_0[min_0,max_0]p_1\cdots[min_{j-1},max_{j-1}]p_j\cdots[min_{m-2},max_{m-2}]p_{m-1}$ 和一个序列串 $S=s_0s_1\cdots s_i\cdots s_{n-1}$ 以及两个整数值最小长度 $Minlen$ 和最大长度 $Maxlen$,这里 m 和 n 分别是模式串 P 和序列串 S 的长度,$p_j\neq\phi(0<j\leq m-1)$,$s_i\neq\phi(0\leq i\leq n-1)$. ϕ 是通配符,可以匹配任何给定的字符. min_{j-1} 和 max_{j-1} 是给定整数值,代表模式字符 p_{j-1} 和 p_j 之间通配符可以匹配的最小和最大长度,这里 $0\leq min_j\leq max_j$. $W=\max(max_j-min_j+1)$ 称为模式串 P 的最大间距. $Minlen$ 和 $Maxlen$ 称为全局约束,min_j 和 max_j 称为局部约束[13].

定义 2. 如果一个位置索引序列 $A=\langle a_0,\cdots,a_j,\cdots,a_{m-1}\rangle$ 服从如下约束条件

$$\begin{cases} p_j=s_{a_j} \\ min_{j-1}\leq a_j-a_{j-1}-1\leq max_{j-1} \\ Minlen\leq a_{m-1}-a_0+1\leq Maxlen \\ a_{j-1}<a_j \end{cases} \tag{1}$$

这里 $0\leq j\leq m-1$ 且 $0\leq a_j\leq n-1$,则称 A 是 P 在 S 中的一个出现[13].

定义 3. 令集合 $T(S,P)$ 代表模式串 P 在序列串 S 中的所有出现,集合 $T(S,P)$ 的长度用 $|T(S,P)|$ 来表示.

定义 4. 给定两个出现 $B=\langle b_0,\cdots,b_j,\cdots,b_{m-1}\rangle$ 和 $C=\langle c_0,\cdots,c_k,\cdots,c_{m-1}\rangle$,如果存在 $b_j=c_k$,则称出现 B 与出现 C 相关并称出现 B 和 C 都包含位置 b_j,这里 $0\leq j<m$ 且 $0\leq k<m$;否则称出现 B 和 C 互不相关.

定义 5. $T(S,P)$ 中子集 $T_1(S,P)$ 满足的任何两个出现都是彼此不相关的,则称子集 $T_1(S,P)$ 是具有间隙约束和一次性条件的模式匹配 PMGOOC (Pattern Matching with Gaps and One-Off Condition). 满足 PMGOOC 的最大子集 $T_1(S,P)$ 称为具有间隙约束和一次性条件的最大模式匹配 MPMGOOC(Maximum Pattern Matching with Gaps and One-Off Condition).

定义 6. 给定一个出现 $B=\langle b_0,\cdots,b_j,\cdots,b_{m-1}\rangle$ 和一个集合 $D=\{d_0,\cdots,d_r,\cdots,d_{l-1}\}$,如果 $b_j=d_r$,则称出现 B 与集合 D 相关,这里 $0\leq j<m$ 且 $0\leq r<l$. 与集合 D 相关的所有出现的数目称为集合相关数,用 $RS(D)$ 来表示.

定义 7. 与出现 B 相关的所有出现的数目称为出现相关数,用 $RO(B)$ 表示;包含位置 $i(0\leq i\leq|S|-1)$ 的所有出现的数目称为位置相关数,用 $RP(i)$ 来表示.

定义 8. 假定 $e\notin D_1$ 且 $D_2=D_1\cup\{e\}$,增加 e 后,多增加的相关出现的数目称为新增数,用 $I(e,D_1)$ 来表示,其计算方法为 $I(e,D_1)=RS(D_2)-RS(D_1)$.

定义 9. 令 $B=\langle b_0,\cdots,b_j,\cdots,b_{m-1}\rangle$ 是给定模式串 P 和序列串 S 下的一个出现,在一次性条件(One-Off Condition)下,新序列串 $S^*=s_0^*s_1^*\cdots s_k^*\cdots s_{n-1}^*$ 是在出现 B 下的新序列,记为 $(S-B)$,其 s_k^* 计算方法如下:

$$s_k^*=\begin{cases} X, & k=b_j \\ s_k, & k\neq b_j \end{cases} \tag{2}$$

这里 X 表示为一个不可匹配的字符.

下面给出 3 个实例对问题的定义及其相关概念与符号进行解释.

例 1. 给定模式串 $P=$a[0,1]b[0,1]c,序列串 $S=$aabbcc 以及最小和最大长度分别为 $MinLen=3$,$MaxLen=5$.

依据给定条件可知,模式串 P 在序列串 S 中所有出现 $T(S,P)$ 是 $\{\langle0,2,4\rangle,\langle1,2,4\rangle,\langle1,3,4\rangle,\langle1,3,5\rangle\}$,故 $|T(S,P)|$ 为 4. 依据定义 5 可知,MPMGOOC 问题的解是 $\{\langle0,2,4\rangle,\langle1,3,5\rangle\}$,因为这两个出现是 $T(S,P)$ 中互不相关最大子集.

例 2. 与例 1 相同的 P,S 及全局约束条件,并令两个出现 $B=\langle0,2,4\rangle$ 和 $C=\langle1,3,4\rangle$,$D=\{2,3\}$ 和 $i=2$.

依据定义 4 可知,出现 B 和 C 是相关的,因为出现 B 和 C 中都包含位置 4;依据定义 6 可知,出现 B 和 C 都与集合 D 是相关的,因为出现 B 和 C 分别包含 2 和 3,而 2 和 3 是集合 D 的两个元素. 依据定义 6 和 7 可知,$RO(B)=3$,$RO(C)=4$,$RP(i)=2$ 以及 $RS(D)=4$. 依据定义 9 可知,在 one-off 条件下,在出现 B 下的新序列串为 aXbXcX.

例 3. 与例 1 相同的 P,S 及全局约束条件,并令 $e=3$ 且 $D_1=\{2,4\}$.

依据定义 6 和 8 可知,$RS(D_1)=3$ 且 $I(b,D_1)=RS(\{2,3,4\})-RS(\{2,4\})=1$.

3　网树的定义及性质

文献[14]最早给出了网树的定义，但是为了求解 MPMGOOC 问题，本节在网树定义的基础上，给出了网树的一些新概念和性质并对这些概念和性质进行了解释.

定义 10.　网树[14]是一种每条边具有“双亲-孩子关系”或“孩子-双亲关系”边标签的有向无环图 DAG(Directed Acyclic Graph)且网树中每个结点都可以有 0 个或多个孩子结点以及 0 个或多个双亲结点. 网树还具有如下 5 个性质：

(1) 网树是树结构的拓展，它具备很多与树相似的概念，如根结点、叶子结点、层、双亲、孩子等概念；

(2) 一棵网树可以有多个根结点；

(3) 除根结点之外网树的其它结点可以有多个双亲结点；

(4) 从任意一个结点到达网树的一个根结点的路径数可以不唯一；

(5) 相同结点名称的结点可以在网树的不同层上多次出现.

定义 11.　如果网树中所有结点都仅出现一次，可以直接使用结点的名称来表示该结点；否则如果结点 i 在不同层上多次出现，用 n_j^i 来表示第 j 层的结点 i.

定义 12.　如果结点 b 在结点 c 与某一根结点的路径上，则称结点 b 是结点 c 的祖先. 当前结点看作是自身的一个祖先. 结点 c 的祖先集是由结点 c 的所有祖先构成的，用 $A(c)$ 表示.

定义 13.　给定一个结点集合 $D=\{d_0,d_1,\cdots,d_{l-1}\}$，集合 D 的所有元素的祖先集的交集称为集合 D 的共同祖先集(Common ascendant)，用 $C(D)$ 表示，其计算方法为

$$C(D)=A(d_0)\bigcap A(d_1)\bigcap\cdots\bigcap A(d_{l-1}) \tag{3}$$

定义 14.　从结点 n_j^i 到达根结点的路径数称为树根路径数 RPN(Root Path Number)，用 $N_r(n_j^i)$ 来表示. 根结点 n_1^i 的树根路径数为 1，即 $N_r(n_1^i)=1$.

性质 1.　结点 n_j^i 的树根路径数是其所有双亲结点的树根路径数之和，即

$$N_r(n_j^i)=\sum_{k=1}^{h}N_r(n_{j-1}^{i_k}) \tag{4}$$

这里 $n_{j-1}^{i_k}$ 是结点 n_j^i 的第 k 个双亲，h 是结点 n_j^i 的双亲数.

定义 15.　从结点 n_j^i 到达第 m 层叶子结点的路径数称为叶子路径数 LPN(Leaf Path Number)，用 $N_l(n_j^i)$ 来表示，这里 m 是网树的深度. 叶子 n_m^i 的叶子路径数为 1，即 $N_l(n_m^i)=1$.

性质 2.　结点 n_j^i 的叶子路径数是其所有孩子结点的叶子路径数之和，即

$$N_l(n_j^i)=\sum_{k=1}^{h}N_l(n_{j+1}^{i_k}) \tag{5}$$

这里 $n_{j+1}^{i_k}$ 是结点 n_j^i 的第 k 个孩子，h 是结点 n_j^i 的孩子数.

定义 16.　从所有根结点到第 m 层叶子结点的所有路径中包含结点 n_j^i 的路径数称为树根-叶子路径数 RLPN(Root-Leaf Path Number)，用 $N_p(n_j^i)$ 表示.

性质 3.　结点 n_j^i 的树根-叶子路径数是其树根路径数与其叶子路径数之积，即

$$N_p(n_j^i)=N_r(n_j^i)\times N_l(n_j^i) \tag{6}$$

性质 4.　位置 i 的位置相关数是网树中结点名称是 i 的结点的树根-叶子路径数之和，即

$$RP(i)=\sum_{j=1}^{m}N_p(n_j^i) \tag{7}$$

这里 m 是网树的深度.

定义 17.　位置 i 在集合 D 的共同祖先集中的所有树根路径数称为位置 i 在集合 D 的路径分支数，用 $pb(i,D)$ 来表示.

性质 5.　位置 i 在集合 D 的路径分支数是共同祖先集 $C(D)$ 中结点名称是 i 的结点的树根路径数之和，即

$$pb(i,D)=\sum_{j=1}^{l}N_r(n_j^i) \tag{8}$$

这里 l 是共同祖先集的深度.

为了解决 MPMGOOC 问题的全局约束，本文定义了结点的最小根和最大根的概念.

定义 18.　一个结点可以抵达的最小根结点名称称为该结点的最小根；其可以抵达的最大根结点名称称为该结点的最大根. 一个网树根结点的最小根和最大根都是其自身.

性质 6.　结点的最小根和最大根分别是其所有双亲结点的最小根集合中的最小值和最大根集合中的最大值.

定义 19.　结点的所有双亲结点中最后一个双亲结点称为该结点的最右双亲结点.

图 1 给出了一棵网树. 在这棵网树上，有一些结点名称多次出现. 例如结点 3 既在第 1 层上出现又

在第 2 层中出现. 本文采用 n_1^3 和 n_2^3 来分别描述第 1 层和第 2 层的结点 3. 结点 n_1^1、n_1^2 和 n_1^3 是图 1 中网树的 3 个根结点；n_2^2、n_3^5 和 n_3^6 是网树的 3 个叶子结点. 结点 n_2^3 有两个双亲结点，分别是 n_1^1 和 n_1^2. 网树中，每条边具有“双亲-孩子关系”或“孩子-双亲关系”的标签. 从其中任何一种边标签看网树，网树都是一种有向无环图，因此网树是一种具有边标签的有向无环图. 如果不考虑边标签，图 1 中网树将有一个环路 $\{n_1^2, n_2^3, n_3^6, n_2^4, n_1^2\}$. 网树的另外一个特征是从一个结点到达网树的一个根结点的路径数可以不唯一. 例如图 1 中，叶子结点 n_3^6 访问根结点 n_1^2 有两条不同的路径，分别为 $\{n_1^2, n_2^3, n_3^6\}$ 和 $\{n_1^2, n_2^4, n_3^6\}$.

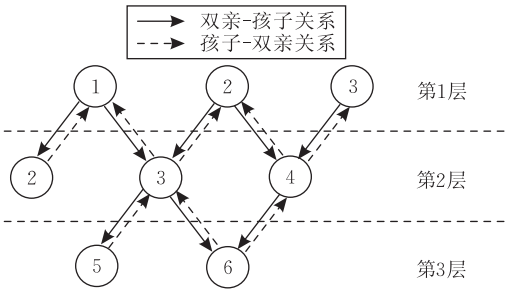


图 1 一棵网树

在图 1 中，结点 n_3^6 的祖先集（$A(n_3^6)$）为 $\{n_1^1, n_1^2, n_1^3, n_2^3, n_2^4, n_3^6\}$. 集合 $\{n_3^5, n_3^6\}$ 的共同祖先集是 $\{n_1^1, n_1^2, n_2^3\}$. 结点 n_2^3 的树根路径数 $N_r(n_2^3)$ 为 2，因为结点 n_2^3 有两个双亲结点 n_1^1 和 n_1^2 且 $N_r(n_1^1)=N_r(n_1^2)=1$. $N_l(n_2^3)=2$，因为结点 n_2^3 有两个孩子结点 n_3^5 和 n_3^6 且 $N_l(n_3^5)=N_l(n_3^6)=1$. $N_l(n_2^2)=0$，因为结点 n_2^2 不能到达第 3 层结点. $N_p(n_2^3)=N_r(n_2^3)\times N_l(n_2^3)=4$，因为图 1 的网树中，有 4 条不同的树根-叶子路径包含结点 n_2^3，即 $\{n_1^1, n_2^3, n_3^5\}$，$\{n_1^1, n_2^3, n_3^6\}$，$\{n_1^2, n_2^3, n_3^5\}$ 和 $\{n_1^2, n_2^3, n_3^6\}$. 同理，$N_p(n_2^2)=N_r(n_2^2)\times N_l(n_2^2)=0$，因为 $N_l(n_2^2)=0$. 依据性质 4 可以计算，包含 3 的位置相关数 $RP(3)=N_p(n_1^3)+N_p(n_2^3)=5$. $pb(3,\{n_3^6\})=N_r(n_1^3)+N_r(n_2^3)=3$，因为结点名称为 3 的结点在集合 $\{n_3^6\}$ 的共同祖先集中出现两次；$pb(2,\{n_3^6\})=N_r(n_1^2)=1$，因为结点名称为 2 的结点在集合 $\{n_3^6\}$ 的共同祖先集中仅出现一次；结点 n_3^6 的最小根为 1，因为结点 n_3^6 有两个双亲结点 n_2^3 和 n_2^4 且它们的最小根分别为 1 和 2；结点 n_3^6 的最大根是 3；结点 n_3^6 的最右双亲结点是结点 n_2^4.

4 算法设计

为了求解 MPMGOOC 问题，一个合理的启发式策略是每次选择出现相关数较小的出现 B 作为结果. 而为了找到这个出现 B，第 2 个合理的启发式策略是每次选择新增数较小的点以便形成出现 B. 而每次查找并计算新增数较小的点的时间复杂度较高，第 3 个启发式策略是用位置相关数较小的点来替代新增数较小的点.

本文利用每个结点的树根路径数 RPN、叶子路径数 LPN 及树根-叶子路径数 RLPN 等网树的特殊概念及其性质实现对位置相关数的计算. 本文还利用了网树的最大根和最小根的概念与性质，对 MPMGOOC 问题的全局约束条件进行考虑. 这样将 MPMGOOC 问题等价地转换为一棵网树，并在此基础上设计出 SGSP 策略，实现了计算一个出现 $B1$. 因此在 SGSP 策略中应用了第 2 和第 3 个启发式策略. 此外，本文还利用最右双亲结点、最小根和最大根等概念设计了 SRMP 策略，实现了求解一个出现 $B2$. SBO 算法是择优使用 $B1$ 或 $B2$，进而进一步实现每次选择出现相关数较小的出现这个启发式策略.

4.1 提出算法

4.1.1 SBO 算法

SBO 算法的第 1 步是将模式匹配问题转化为一棵网树，即依据 P 和 S 创建一棵网树[14]. 当接收一个字符 $s_i(0\leq i<n)$，检查 s_i 是否满足如下 3 条规则，如果满足相应规则，则按照对应规则创建一个结点或一条边，具体如下.

规则 1. 如果 $s_i=p_0$，则在第 1 层创建结点 n_1^i；

规则 2. 如果 $s_i=p_j$ 且 i 与第 j 层某个结点 n_{j-1}^e 的距离满足局部约束条件（$min_{j-1}\leq i-e-1\leq max_{j-1}$），则在第 j 层创建结点 n_j^i 并在结点 n_{j-1}^e 与新建结点 n_j^i 之间建立“双亲-孩子关系”和“孩子-双亲关系”；

规则 3. 如果结点 n_j^i 和 n_{j-1}^q 之间的距离满足局部约束条件（$min_{j-1}\leq i-q-1\leq max_{j-1}$），则可以在这两个结点之间建立“双亲-孩子关系”和“孩子-双亲关系”.

SBO 算法的第 2 步是依据定义 18 和性质 6 计算每个结点的最小根和最大根.

之后，SBO 算法在该网树下解决 MPMGOOC 问题，具体方案如下：如果出现 B 的相关出现数越小，则出现 B 越有可能是一个最优出现. 为了找到相关出现数较小的出现，SBO 算法从网树最后一个叶子结点开始依次向前查找包含该叶子结点的出现. 为了找到局部最优解，SBO 算法采用了 SGSP

和 SRMP 两种策略寻找具有相同叶子结点的两个出现，并在两个出现中选择相关出现数较小的出现作为 SBO 算法的一个最优出现. 之后依据定义 9 重新计算新序列串，并在新序列串中寻找下一个最优出现. 迭代此过程，直至所有叶子结点都被检测一遍为止. 因此 SBO 算法给出如下.

算法 1.　SBO.

输入：$P=p_0[min_0,max_0]p_1\cdots[min_{m-2},max_{m-2}]p_{m-1}$，$S=s_0s_1\cdots s_{L-1}$，*Minlen* and *Maxlen*

输出：解 C

1. 依据 P 和 S 建立一棵网树；
2. 计算每个结点的最小和最大根；
3. for k=第 m 层叶子结点数 downto 1 step −1
4. 　$B1$=SGSP(第 k 个叶子结点)；
5. 　$y1=RO(B1)$；
6. 　$B2$=SRMP(第 k 个叶子结点)；
7. 　$y2=RO(B2)$；
8. 　if ($y1<y2$)　$B=B1$　else　$B=B2$；
9. 　$C=C\cup B$；
10. 　$S=S-B$；
11. 　依据新的 S 重新计算各个结点的 RPN；
12. next k
13. return C.

4.1.2　SGSP 策略

为了找到包含第 m 层叶子结点 f 的出现 B，SGSP 的核心思想是采用贪婪策略寻找局部最优解，即 $m-1$ 次寻找当前结点的近似优化双亲结点 AOP. 所谓当前结点的 AOP 是指在满足全局约束的双亲结点中查找位置相关数最小的双亲结点作为当前结点的 AOP；若两个双亲结点的位置相关数同样小，则在已获得路径 B 的共同祖先集中选择路径分支数最大的双亲结点作为当前结点的 AOP. 因此 SGSP 策略可描述为：首先计算每个结点的 RPN、LPN 和 RLPN，然后计算每个位置的位置相关数，之后 SGSP 迭代 $m-1$ 次寻找当前结点的 AOP，具体给出如下.

算法 2.　SGSP.

输入：叶子结点 f

输出：出现 B

1. 依据性质 1 计算每个结点的树根路径数；
2. 依据性质 2 计算每个结点的叶子路径数；
3. 依据性质 3 计算每个结点的树根-叶子路径数；
4. 依据性质 4 计算每个位置的位置相关数；
5. $B[m-1]=f$；
6. for $j=m-2$ downto 0 step −1 do
7. 　依据性质 5 计算每个位置 x 的在已有路径 B 下的路径分支数 $pb(x,B)$；
8. 　$r=B[j+1].number\_of\_parents$；
9. 　$B[j]=B[j+1].parent[r-1]$；
10. 　for $k=r-2$ downto 0 step −1 do
11. 　　if $B[j+1].parent[k]$ 满足全局约束 then
12. 　　if ($RP(B[j])>RP(B[j+1].parent[k])$) then $B[j]=B[j+1].parent[k]$；
13. 　　if (($RP(B[j])=RP(B[j+1].parent[k])$ and ($pb(B[j+1].parent[k],B)>=pb(B[j],B)$)) then $B[j]=B[j+1].parent[k]$；
14. 　　end if
15. 　end for
16. end for
17. return B

4.1.3　SRMP 策略

为了找到包含第 m 层叶子结点 f 的出现 B，SRMP 算法核心思想是每次迭代过程中都选择满足全局约束的最右双亲结点，具体给出如下.

算法 3.　SRMP.

输入：叶子结点 f

输出：一个出现 B

1. $B[m-1]=f$；
2. for $j=m-2$ downto 0 step −1 do
3. 　迭代查找一个满足全局约束的最右双亲结点作为当前结点的双亲结点；
4. end for
5. return B

4.2　复杂性分析

SBO 算法的空间复杂度是 $O(W\times m\times n)$，因为网树的深度为 m，网树上每层最多有 n 个结点，每个结点最多有 m 个双亲结点，这里 m、n 和 W 分别是模式串 P 和序列串 S 的长度以及模式串 P 的最大间距.

在讨论 SBO 算法的时间复杂度之前首先讨论 SGSP 和 SRMP 策略的时间复杂度. 易知策略 SRMP 的时间复杂度为 $O(W\times m)$. 而 SGSP 的时间复杂度分析如下：SGSP 的第 1 行和第 2 行(为每个结点计算 RPN 和 LPN)的时间复杂度都是 $O(W\times m\times n)$，由于每个双亲-孩子关系都需要考虑，而按照算法的空间复杂度分析可知网树最多有 $W\times m\times n$ 个双亲-孩子关系；SGSP 的第 3 行和第 4 行的时间复杂度都是 $O(m\times n)$，因为每个结点都需要被计算，而网树上最多有 $m\times n$ 个结点；第 7 行的时间复杂度为 $O(m^2\times W)$，因为计算 $pb(x,B)$ 的时间复杂度是 B 的共同祖先集下结点的数量(其为 $O(m^2\times W)$)，这样第 6～16 行的时间复杂度为 $O(m^3\times W)$. 故 SGSP 策略的时间复杂度为 $O(W\times$

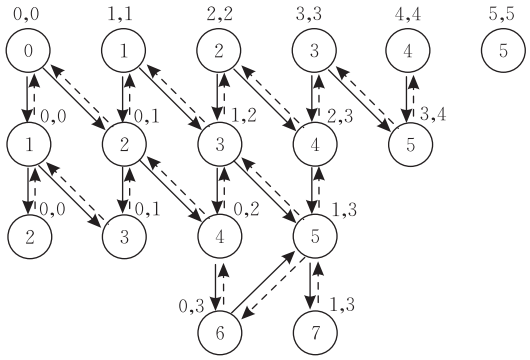
$m\times(n+m^2))$.

SBO 算法的时间复杂度分析如下：由算法的空间复杂度分析易知，SBO 算法的第 1 行和第 2 行的时间复杂度都是 $O(W\times m\times n)$；算法的第 4 行的时间复杂度是 $O(W\times m\times(n+m^2))$；而给定一个出现 B，计算 $S^*=S-B$ 的时间复杂度是 $O(m)$；$RO(B)$ 的计算方法采用 $|T(S,P)|-|T(S^*,P)|$ 的方法，因为计算 $|T(S,P)|$ 的时间复杂度是 $O(W\times m\times n)$，所以计算 $RO(B)$ 的时间复杂度也是 $O(W\times m\times n)$. SBO 算法第 6 行的时间复杂度是 $O(W\times m)$；第 8～10 行的时间复杂度均是 $O(m)$，因为模式串 P 的长度为 m；第 11 行的时间复杂度是 $O(W\times m\times n)$. 这样算法从第 3 行到第 12 行的时间复杂度为 $O(W\times m\times(n+m^2)\times n/m)=O(W\times n\times(n+m^2))$，因为该问题的出现数最多为 n/m 个. 故算法 SBO 的时间复杂度为 $O(W\times n\times(n+m^2))$.

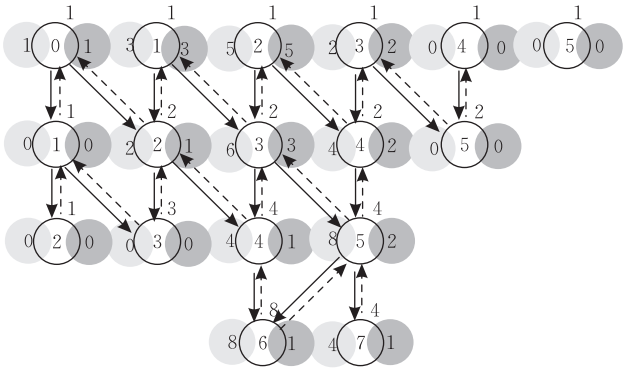
4.3 运行实例

例 4. 给定模式串 $P=$a[0,1]a[0,1]a[0,1]b，序列串 $S=$aaaaaabb 以及最小和最大长度 $Minlen=4$，$Maxlen=7$.

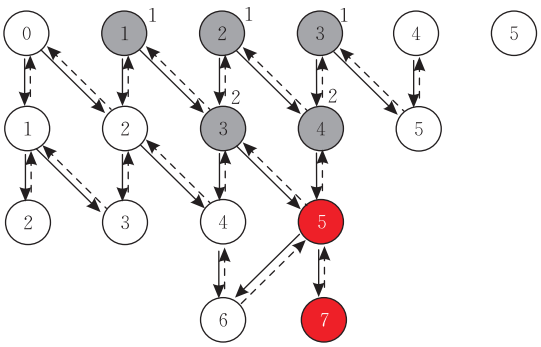
依据网树的创建规则，创建结果如图 2(a)所示. 从图中可以看出，某些位置索引被创建为多个网树结点，例如：位置索引 2 被创建为 3 个网树结点，分别位于网树的第 1、2 和 3 层，因为 $s_2=a$ 可以与 $p_0=a$，$p_1=a$ 和 $p_2=a$ 分别匹配. 依据性质 6 对每个结点的最小根和最大根进行了计算，其结果给出在每个结点的上方. 使用结点的最小根和最大根可以判断当前结点及其祖先集结点是否满足全局约束. 例如因为结点 7 的最小根和最大根分别为 1 和 3 且 $4=Minlen\leq 7-3+1\leq 7-1+1\leq Maxlen=7$，所以结点 7 及其祖先集结点都满足全局约束. 易知本实例中所有匹配都满足全局约束.



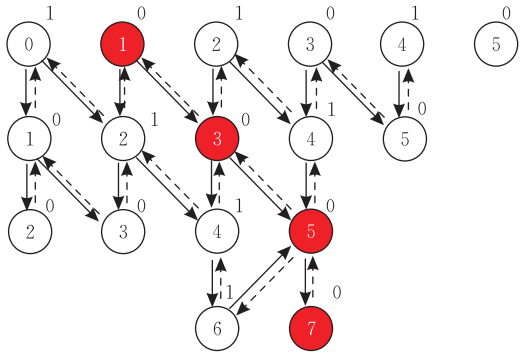
(a) 网树及其最小和最大根



(b) 网树及其RPN、LPN和RLPN



(c) 集合$\{n_3^5,n_4^7\}$的共同祖先集



(d) 新网树

图 2 实例求解示意图

网树第 4 层的最后叶子结点是 n_4^7，因此 SBO 算法调用 SGSP 和 SRMP 两种策略来分别计算两个包含结点 n_4^7 的出现. SGSP 策略工作过程如下：

第 1 行按照性质 1 计算每个结点的 RPN 值. 例如 $N_r(n_2^3)$ 是 2，因为结点 n_2^3 有两个双亲结点 n_1^1 和 n_1^2 且 $N_r(n_1^1)=N_r(n_1^2)=1$，每个结点的 RPN 值在图 2(b)中每个结点的上方给出. 第 2 行按照性质 2 计算每个结点的 LPN 值. 例如 $N_l(n_3^2)=N_l(n_3^3)=0$，因为结点 n_3^2 和 n_3^3 都不能抵达网树的第 4 层叶子结点；$N_l(n_2^3)=3$，因为结点 n_2^3 有两个孩子结点 n_3^4 和 n_3^5 且 $N_l(n_3^4)=1$，$N_l(n_3^5)=2$，每个结点的 LPN 值在图 2(b)中每个结点的右边给出. 第 3 行按照性质 3 计算每个结点的 RLPN 值. 例如 $N_p(n_2^3)=N_r(n_2^3)\times N_l(n_2^3)=6$，每个结点的 RLPN 值在图 2(b)中每个

结点的左边给出. 第 4 行按照性质 4 计算每个位置的位置相关数. 例如 $RP(2)=N_P(n_1^2)+N_P(n_2^2)+N_P(n_3^2)=7$；同理 $RP(0)=1$，$RP(1)=3$，$RP(3)=8$，$RP(4)=8$，$RP(5)=8$，$RP(6)=8$ 且 $RP(7)=4$. 第 6～16 行是迭代寻找一个包含叶子结点 n_4^7 的出现 B. SGSP 首先找结点 n_4^7 的近似优化双亲结点 AOP，结点 n_3^5 是结点 n_4^7 的 AOP，因为结点 n_4^7 只有一个双亲结点 n_3^5. 接下来寻找结点 n_3^5 的 AOP，从结点 n_3^5 的双亲结点中选择位置相关数最小的结点作为其 AOP，结点 n_3^5 有两个双亲结点，分别是结点 n_2^3 和 n_2^4 且 $RP(3)=RP(4)=8$. 在这种情况下，需要比较位置 3 和 4 在结点集 $\{n_3^5,n_4^7\}$ 的共同祖先集中的路径分支数，即 $pb(3,\{n_3^5,n_4^7\})$ 和 $pb(4,\{n_3^5,n_4^7\})$. 图 2(c)给出了结点集 $\{n_3^5,n_4^7\}$ 的共同祖先集，由于 $pb(3,\{n_3^5,n_4^7\})=N_r(n_2^3)+N_r(n_1^3)=3$ 且 $pb(4,\{n_3^5,n_4^7\})=N_r(n_2^4)=2$，这样结点 n_2^3 被看作是结点 n_3^5 的 AOP，因为 SGSP 策略选择路径分支数较大的双亲结点作为当前结点的 AOP 且 $I(3,\{n_3^5,n_4^7\})<I(4,\{n_3^5,n_4^7\})$. 最后 SGSP 计算结点 n_2^3 的 AOP，结点 n_2^3 有两个双亲结点 n_1^1 和 n_1^2. 结点 n_2^3 的 AOP 是结点 n_1^1 因为 $RP(1)=3<RP(2)=7$. 因此 SGSP 策略计算结果为〈1,3,5,7〉.

依据 SRMP 策略，当叶子结点 n_4^7 给定后，容易找到结点 n_3^5 是结点 n_4^7 的最右双亲结点；而结点 n_2^4 是结点 n_3^5 的最右双亲结点；而结点 n_1^3 是结点 n_2^4 的最右双亲结点. 因此 SRMP 策略计算结果为〈3,4,5,7〉.

SBO 算法选择 SGSP 和 SRMP 策略的结果中相关出现数较小的出现作为算法的一个出现. 易知 $RO(\langle 1,3,5,7\rangle)$ 和 $RO(\langle 3,4,5,7\rangle)$ 分别是 7 和 8，所以 SBO 算法选择〈1,3,5,7〉作为算法的一个出现.

在出现〈1,3,5,7〉不再被使用的情况下，新序列串为 aXaXaXbX. SBO 算法的第 11 行重新计算了在新序列串下各个结点的 RPN. 图 2(d)中给出了在新序列串下的网树，此时 SBO 算法的结果为〈0,2,4,6〉. 故 SBO 算法针对本问题找到的解为〈0,2,4,6〉和〈1,3,5,7〉.

5　实验结果与分析

5.1　实验结果

本节采用真实生物数据用来对比 SAIL 算法[13]和 SBO 算法的性能. 此外，本文还将 SGSP 和 SRMP 两种策略分别形成了两个可以单独计算 MPMGOOC 问题的算法，并分别命名为贪婪搜索双亲算法 AGSP(Algorithm of Greedy Search Parent)和最右双亲算法 ARMP(Algorithm of RightMost Parent). 所有 4 种算法的源代码已全部公开①. 实验运行的软硬件环境为：酷睿 2 双核 T7100、主频 1.80GHZ、内存 1.0 GB、Windows XP SP2 操作系统的笔记本.

猪流感 H1N1 病毒在 2009 年大流行，其病毒的 DNA 序列可在美国国家生物计算信息中心的网上下载②. 该病毒有很多候选序列，本文选择于 2010 年 3 月 30 日公布的一个结果(A/Managua/2093.01/2009(H1N1))中的全部 8 个片段③作为测试序列(见表 1).

表 1　真实生物数据片段

序号	片段名称	位点	片段长度
S1	Segment 1	CY058563	2286
S2	Segment 2	CY058562	2299
S3	Segment 3	CY058561	2169
S4	Segment 4	CY058556	1720
S5	Segment 5	CY058559	1516
S6	Segment 6	CY058558	1418
S7	Segment 7	CY058557	982
S8	Segment 8	CY058560	844

Min 等人[12]在其研究工作中给出了一些模式串，由于其中 $P5$ 模式串不能在 DNA 序列中应用，所以本文选择其余的 4 个模式串($P1$～$P4$)作为本文的部分测试模式串. 此外本文又另外构造了 5 个新的模式串，表 2 给出了本文的全部 9 种模式串.

表 2　模式串

序号	模式串	最小长度	最大长度
$P1$	a[0,3]t[0,3]a[0,3]t[0,3]a[0,3]t[0,3]a[0,3]t[0,3]a[0,3]t[0,3]a	11	41
$P2$	g[1,5]t[0,6]a[2,7]g[3,9]t[2,5]a[4,9]g[1,8]t[2,9]a	24	57
$P3$	g[1,9]t[1,9]a[1,9]g[1,9]t[1,9]a[1,9]g[1,9]t[1,9]a[1,9]g[1,9]t	21	101
$P4$	g[1,5]t[0,6]a[2,7]g[3,9]t[2,5]a[4,9]g[1,8]t[2,9]a[1,9]g[1,9]t	27	73
$P5$	a[0,10]a[0,10]t[0,10]c[0,10]g[0,10]g	6	56
$P6$	a[0,5]t[0,7]c[0,9]g[0,11]g	5	37
$P7$	a[0,5]t[0,7]c[0,6]g[0,8]t[0,7]c[0,9]g	7	49
$P8$	a[5,6]c[4,7]g[3,8]t[2,8]a[1,7]c[0,9]g	22	52
$P9$	c[0,5]t[0,5]g[0,5]a[0,5]a	5	25

① 所有 4 种算法的源代码可在 http://wuc.scse.hebut.edu.cn/下载.

② 美国国家生物计算信息中心的网址 http://www.ncbi.nlm.nih.gov/.

③ A/Managua/2093.01/2009(H1N1)中的全部 8 个片段可在 http://www.ncbi.nlm.nih.gov/genomes/FLU/SwineFlu.html 下载.

表 3 给出了这 4 种算法的运行时间复杂度. 4 种算法在 $S2$ 这个最长序列上的全部 9 种模式下的运行时间对比表以及在 $P1$ 模式下的全部 8 种序列上的运行时间对比表分别见表 4 和表 5.

表 3 算法时间复杂度对比

算法名称	时间复杂度
SAIL[13]	$O(W^2\times n\times m^2)$①
ARMP	$O(W\times n^2)$②
AGSP	$O(W\times n\times(n+m^2))$
SBO	$O(W\times n\times(n+m^2))$

注:①文献[13]给出 SAIL 算法时间复杂度为 $O(n+klmW)$,k 和 l 分别为 p_{m-1} 在 S 中出现的频度和出现的最大跨度. 由于 k 的数量级为 $O(n)$,l 的数量级为 $O(W\times m)$,因此 SAIL 算法时间复杂度可以描述为 $O(W^2\times n\times m^2)$.

②尽管 SRMP 策略时间复杂度为 $O(W\times m)$,但是在形成 ARMP 算法后,在每次求解一个出现前,需要执行依据新的 S 重新计算各个结点的 RPN 操作,所以 ARMP 算法时间复杂度为 $O(W\times n^2)$.

表 4 S2 序列上的全部 9 种模式下的运行时间

算法名称	运行时间/ms								
	$P1$	$P2$	$P3$	$P4$	$P5$	$P6$	$P7$	$P8$	$P9$
SAIL	47	47	63	47	31	16	32	31	16
ARMP	47	297	609	437	735	656	344	234	562
AGSP	47	438	875	563	844	735	421	265	656
SBO	61	468	922	594	969	797	469	282	687

表 5 P1 模式下的全部 8 种序列上的运行时间

算法名称	运行时间/ms							
	$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	$S7$	$S8$
SAIL	31	47	31	31	31	≤16	≤16	≤16
ARMP	63	47	63	79	63	≤16	≤16	≤16
AGSP	78	47	63	94	47	≤16	≤16	≤16
SBO	78	61	63	94	63	≤16	≤16	≤16①

注:当运行时间过短情况下,计算机不能准确报告运行时间,所以本文采用"≤16"表示运行时间小于或等于 16 ms.

这 4 种算法在全部的 72 个实例上的测试结果见表 6. 为了直观地显示每个问题的最好解,表 6 中最好解都采用加粗方式显示.

表 6 生物数据测试结果

模式	算法名称	解的数目/个							
		$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	$S7$	$S8$
$P1$	SAIL	**13**	**9**	**10**	**15**	**11**	**5**	**3**	**3**
	ARMP	**13**	**9**	**10**	**15**	**11**	**5**	**3**	**3**
	AGSP	**13**	**9**	**10**	**15**	10	**5**	**3**	**3**
	SBO	**13**	**9**	**10**	**15**	**11**	**5**	**3**	**3**
$P2$	SAIL	66	69	59	54	42	39	31	27
	ARMP	67	71	62	54	42	41	**33**	28
	AGSP	65	71	61	53	44	**42**	31	28
	SBO	**69**	**74**	**64**	**55**	**45**	**42**	32	**29**
$P3$	SAIL	66	69	66	54	**45**	42	**33**	**28**
	ARMP	64	70	**68**	52	43	43	**33**	26
	AGSP	**68**	71	67	50	42	43	29	**28**
	SBO	67	**75**	**68**	**55**	**45**	**45**	32	27
$P4$	SAIL	49	50	49	40	32	31	24	20
	ARMP	**51**	**58**	52	46	37	30	26	21
	AGSP	48	56	51	46	**38**	33	26	22
	SBO	**51**	56	**53**	**47**	**38**	**34**	**27**	**22**

（续 表）

模式	算法名称	解的数目/个							
		$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	$S7$	$S8$
$P5$	SAIL	207	204	204	147	143	132	100	75
	ARMP	215	**208**	208	**151**	147	132	101	76
	AGSP	213	204	206	146	146	133	101	76
	SBO	**218**	206	**209**	150	**148**	**134**	**102**	**77**
$P6$	SAIL	186	192	198	144	129	124	94	68
	ARMP	**197**	**198**	**203**	**147**	**141**	**127**	**97**	**70**
	AGSP	196	197	201	142	140	125	96	70
	SBO	196	197	202	143	**141**	126	**97**	**70**
$P7$	SAIL	86	94	97	76	66	64	50	39
	ARMP	88	**100**	100	83	69	67	50	**40**
	AGSP	**90**	**100**	100	81	69	67	**51**	**40**
	SBO	**90**	**100**	**103**	**84**	**74**	**68**	**51**	**40**
$P8$	SAIL	68	66	54	42	46	42	32	26
	ARMP	**72**	**78**	59	**49**	**49**	**45**	**34**	**27**
	AGSP	**72**	77	57	**49**	47	44	**34**	**27**
	SBO	**72**	77	**60**	**49**	48	44	**34**	**27**
$P9$	SAIL	150	174	168	126	112	108	73	61
	ARMP	**153**	**175**	**170**	**128**	**113**	**109**	**74**	**63**
	AGSP	**153**	173	168	127	112	**109**	**74**	**63**
	SBO	**153**	**175**	169	**128**	**113**	**109**	**74**	**63**

5.2 实验结果分析

(1) 在解较少的情况下,SAIL 算法可以取得较好的性能且 ARMP、AGSP 和 SBO 算法的运行时间较短. 由表 6 可以看出,$P1$ 模式在 8 个测试序列上解较少. 在此情况下,SAIL 算法都能够取得最好解;然而当解较多的情况下,SAIL 算法很难取得最好解. 尽管 ARMP、AGSP 和 SBO 等 3 种算法时间复杂度较高,但是在解较少的情况下,这 3 种算法实际运行时间较短;然而当解较多的情况下,这 3 种算法实际运行时间较长并与算法的时间复杂度相吻合. 这是由于当解较少的情况下,问题相对较为简单,对应的网树结构也比较简单,这样 SAIL 算法就能够取得最好解并且基于网树的 3 种算法也能很快找到解. 因此 $P1$ 模式不具有普遍意义,在后面的讨论中,将忽略 $P1$ 模式下的 8 个实例的结果.

(2) SAIL 算法不适合求解离线 MPMGOOC 问题. 表 3 和表 4 可以看出,SAIL 算法的时间复杂度最低且 SAIL 算法的实际求解速度最快,这充分说明 SAIL 算法适用于求解在线问题. 但是在求解离线问题时,SAIL 算法解的质量差. 在 64 个实例中,SAIL 算法仅有 3 次取得了最好解,说明 SAIL 算法在求解复杂的离线问题时较难获得最好解. 此外,SAIL 算法解的质量相对较差,例如在"$P8$-$S2$"实例上,SAIL 算法的解为 66,但是求解到的最好解为 78. 在 64 个实例中,SAIL 算法的解与最好解之差大于或等于 4 的实例共有 26 个. 这些都充分地说明了 SAIL 算法解的质量较差,所以 SAIL 算法不适

合求解离线 MPMGOOC 问题.

(3)在运行时间增加不大的情况下,SBO 算法的质量最好,说明了 SBO 算法可以较好地求解离线 MPMGOOC 问题.尽管 SBO 算法中包含 SGSP 和 SRMP 两种策略并择优使用其结果,但是从表 4 可以看出这样的时间开销并不大.而在解的质量方面,SBO 算法较其它 3 种算法都有显著提高.具体分析如下:

与 SAIL 算法相比,在 64 个实例中,有 60 个实例 SBO 算法的解结果好于 SAIL 算法且其中 23 个实例两种算法解之差大于或等于 4;在"$P3$-$S5$"实例上,SBO 算法与 SAIL 算法同时取得了最好解;在"$P3$-$S7$"、"$P3$-$S8$"和"$P6$-$S4$"实例上,SBO 算法的解较 SAIL 算法的解都只差 1.此外,在很多实例上 SBO 算法大幅度地提高了 SAIL 解的质量.例如在实例"$P8$-$S2$"上,采用 SAIL 算法的结果仅为 66,而采用 SBO 算法的结果为 77,提高幅度非常显著.这充分地说明了 SBO 算法显著地改善了 SAIL 算法的解的质量.

与 ARMP 算法相比,有 32 个实例 SBO 算法的解好于 ARMP 算法且其中有 9 个实例两种算法解之差大于或等于 3;在 18 个实例上 SBO 算法的解与 ARMP 算法同为最好解;在余下的 14 个实例中,仅在"$P6$-$S4$"实例上差距显著(SBO 算法的解为 143,而 ARMP 算法的解为 147),在"$P4$-$S2$"实例上 SBO 算法较 ARMP 算法差 2,而其余的 12 个实例 SBO 算法较 ARMP 算法都仅差 1.这充分地说明了 SBO 算法解的质量好于 ARMP 算法.

与 AGSP 算法相比,有 41 个实例 SBO 算法的解好于 AGSP 算法且其中有 15 个实例两种算法解之差大于或等于 3;在 21 个实例上 SBO 算法的解与 AGSP 算法相同;在"$P3$-$S1$"和"$P3$-$S8$"两个实例 SBO 算法的解比 AGSP 算法差 1.这充分地说明了 SBO 算法解的质量好于 AGSP 算法.

(4)SBO 算法之所以能够取得良好的解是因为 SBO 算法多次运用启发式策略,实现了每次选择出现相关数较小的出现这个启发式策略.但是本文注意到,应用这个启发式策略在求解 MPMGOOC 问题时,不一定都能得到最好解.实验结果显示,SBO 算法在 16 个实例上未能取得最好解.因此即使存在第 3 种策略求解出现并择优使用这 3 种策略结果的算法,也不能保证其结果一定优于 SBO 算法.

综上所述,与其它 3 种算法相比,SBO 算法的解的质量最好且与基于网树的 SRMP 和 SGSP 两种算法相比,时间开销增加不大,充分说明了 SBO 算法适用于求解离线 MPMGOOC 问题.

6　结　论

由于网树是一个新的数据结构,其区别于树结构的概念在于:一个非根网树结点可以有多个双亲结点.因此网树有许多特有概念和性质,如最小根、最大根、树根路径数、叶子路径数、树根-叶子路径数、共同祖先集、最右双亲结点等概念.为了求解 MPMGOOC 问题,一个合理的启发式策略是每次选择一个出现相关数较小的出现 B 作为结果,而为了找到这个出现 B,一个合理的启发式策略是选择新增数较小的双亲结点作为当前结点的双亲结点,但是由于计算新增数的时间复杂度较高,所以采用位置相关数最小的双亲结点作为当前结点的 AOP.本文利用了网树的每个结点的 RPN、LPN 和 RLPN 等概念和性质对位置相关数进行了计算.在此基础上,本文又利用共同祖先集、最小根、最大根、最右双亲结点等概念设计出 SGSP 策略和 SRMP 策略用来计算出现.本文提出的 SBO 算法择优地使用这两种策略的结果.通过实际的生物医学数据测试表明,SBO 算法的解的质量明显优于 AGSP、ARMP 和 SAIL 等算法,这是由于在 SBO 算法中多次运用了启发式策略.SBO 算法的空间复杂度和时间复杂度分别是 $O(W\times m\times n)$ 和 $O(W\times n\times(n+m^2))$,这里 m、n 和 W 分别是模式串 P 和序列串 S 的长度以及模式串 P 的最大间距.我们相信网树将在更多的领域得到应用,并引起更广泛的理论研究.

参　考　文　献

[1] Lunteren J V. High-performance pattern-matching for intrusion detection//Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006). Barcelona, Spain, 2006: 1-13

[2] Califf M E, Mooney R J. Bottom-up relational learning of pattern matching rules for information extraction. Journal of Machine Learning Research, 2003, 4(6): 177-210

[3] Cole R, Gottlieb L A, Lewenstein M. Dictionary matching and indexing with errors and don't cares//Proceedings of the 36th ACM Symposium on the Theory of Computing. New York, USA, 2004: 91-100

[4] Cole J R, Chai B, Farris R J, Wang Q, Kulam S A, McGarrell D M, Garrity G M, Tiedje J M. The ribosomal database

project (RDP-II): Sequences and tools for high-throughput rRNA analysis. Nucleic Acids Research, 2005, 33(Sup. 1): 294-296

[5] Zhang M, Kao B, Cheung D, Yip K. Mining periodic patterns with gap requirement from sequences//Proceedings of the ACM SIGMOD International Conference on Management of Data. Maryland, USA, 2005: 623-633

[6] Han J, Cheng H, Xin D, Yan X. Frequent pattern mining: Current status and future directions. Data Mining and Knowledge Discovery, 2007, 15(1): 55-86

[7] Ji X, Bailey J, Dong G. Mining minimal distinguishing subsequence patterns with gap constraints. Knowledge and Information Systems, 2007, 11(3): 259-286

[8] He Y, Wu X, Zhu X, Arslan A N. Mining frequent patterns with wildcards from biological sequences//Proceedings of the 2007 IEEE International Conference on Information Reuse and Integration (IRI-07). Las Vegas, USA, 2007: 329-334

[9] Fischer M J, Paterson M S. String matching and other products//Proceedings of the 7th SIAM AMS Complexity of Computation. Cambridge, USA, 1974: 113-125

[10] Manber U, Baeza-Yates R. An algorithm for string matching with a sequence of don't cares. Information Processing Letters, 1991, 37(3): 133-136

[11] Huang Y, Wu X, Hu X, Xie F, Gao J, Wu G. Mining frequent patterns with gaps and one-off condition//Proceedings of the 12th IEEE International Conferences on Computational Science and Engineering. Vancouver, Canada, 2009: 180-186

[12] Min F, Wu X, Lu Z. Pattern matching with independent wildcard gaps//Proceedings of the 8th International Conference on Pervasive Intelligence and Computing. Chengdu, China, 2009: 194-199

[13] Chen G, Wu X, Zhu X, Arslan A N, He Y. Efficient string matching with wildcards and length constraints. Knowledge and Information Systems, 2006, 10(4): 399-419

[14] Wu Y, Wu X, Min F, Li Y. A nettree for pattern matching with flexible wildcard constraints//Proceedings of the 2010 IEEE International Conference on Information Reuse and Integration (IRI2010). Las Vegas, USA, 2010: 109-114

WU You-Xi, born in 1974, Ph. D., professor. His research interests include intelligent computation, data mining.

WU Xin-Dong, born in 1963, Ph. D., professor, Ph. D. supervisor. His research interests include data mining, knowledge-based systems, and Web information exploration.

JIANG He, born in 1980, Ph. D., associate professor, Ph. D. supervisor. His research interests include intelligent computation, data mining.

MIN Fan, born in 1973, Ph. D., associate professor. His research interests include rough sets, granular computing, and sequence data mining.

Background

This research focuses on solving the off-line MPM-GOOC problem, which deals with pattern matching with gaps and the one-off condition. Chen et al put forward the problem and proposed an online algorithm, SAIL, to solve this task. However, SAIL is not a complete off-line algorithm. In this paper, we proposed a heuristic algorithm SBO which used a new nonlinear structure Nettree to solve the problem. In order to solve the problem, we brought forward some special concepts and properties of the Nettree, such as the root path number, leaf path number, root-leaf path number, common descendants, the min root, the max root, the rightmost parent, and so on. Experimental results showed that SBO has the best performance among all competitive algorithms. We believe that the Nettree will attract more theoretical studies.

Dr. Wu Xindong is a Professor and also a former Chair (2001～2010) of the Department of Computer Science at the University of Vermont (UVM). Dr. Wu Youxi, Dr. Jiang He and Dr. Min Fan were research scholars of UVM, and during their visits to UVM, they completed most of this research under the supervision of Prof. Wu Xindong.

This research is supported by the National Natural Foundation of China (NSFC) under grant No. 60828005, with the project title of Pattern Matching and Mining with Wildcards and Length Constraints. This research deals with one of the key problems of this NSFC grant.