软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

无重叠条件严格模式匹配的高效求解算法*

武优西[1,2,3], 刘 茜[1,2,3], 闫文杰[1,2,3], 郭 磊[2,4], 吴信东[5,6]

[1](河北工业大学 人工智能与数据科学学院, 天津 300401)

[2](河北工业大学 省部共建电工装备可靠性与智能化国家重点实验室, 天津 300401)

[3](河北省大数据计算重点实验室, 天津 300401)

[4](河北工业大学 电气学院, 天津 300401)

[5](明略科学院 明略科技集团, 北京 100084)

[6](教育部 大数据知识工程重点实验室（合肥理工大学）, 合肥 230009)

通讯作者: 武优西, E-mail: wuc567@163.com

摘 要: 无重叠条件序列模式挖掘是一种间隙约束序列模式挖掘方法,与同类挖掘方法相比,该方法更容易发现有价值的频繁模式,其核心问题是计算给定模式在序列中的支持度或出现数,进而判定该模式的频繁性.而计算模式支持度问题实质是无重叠条件模式匹配.当前研究采用迭代搜索无重叠出现,然后剪枝无用结点的方式计算模式的支持度,其计算时间复杂度为 $O(m*m*n*W)$,其中 m,n 和 W 分别为模式长度,序列长度及最大间隙.为了进一步提高无重叠条件模式匹配计算速度,从而有效地降低无重叠条件序列模式挖掘时间,本文提出了一种高效的算法,该算法将模式匹配问题转换为一棵网树,然后从网树的最小树根结点出发,采用回溯策略迭代搜索最左孩子方式计算无重叠最小出现,在网树上剪枝该出现后,无需进一步查找并剪枝无效结点即可实现问题的求解.本文理论证明了该算法的完备性并将该算法的时间复杂度降低为 $O(m*n*W)$.在此基础上,本文继续指明,该问题还存在另外三种相似的求解策略,分别是从最左叶子出发迭代查找最左双亲方式、从最右树根出发迭代查找最右孩子方式和从最右叶子出发迭代查找最右双亲方式.实验结果验证了本文算法的性能,特别是在序列模式挖掘中,应用本文方法的挖掘算法可以降低挖掘时间.

关键词: 模式匹配; 序列模式挖掘; 无重叠条件; 网树; 回溯策略

中图法分类号:

中文引用格式:

英文引用格式:

Efficient Solving Algorithm for Strict Pattern Matching under Nonoverlapping Condition

WU You-Xi[1,2,3], LIU Xi[1,2,3], YAN Wen-Jie[1,2,3], GUO Lei[2,4], WU Xin-Dong[5,6]

[1](School of of Artificial Intelligence, Hebei University of Technology, Tianjin 300401, China)

[2](Hebei Key Laboratory of Big Data Computing, Tianjin 300401, China)

[3](State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology, Tianjin 300401, China)

[4]((School of Electrical Engineering, Hebei University of Technology, Tianjin 300401, China)

[5](Mininglamp Academy of Sciences, Mininglamp Technology, Beijing 100084, China)

* 基金项目: 国家重点研发计划(2016YFB1000901); 国家自然科学基金(61976240,61702157,917446209); 河北省创新能力培养资助项目(CXZZSS2019023).

Foundation item: National Key Rearch and Development Program of China(2016YFB1000901); National Natural Science Foundation of China (61976240,61702157,917446209); Graduate Student Innovation Program of Hebei Province (CXZZSS2019023).

收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos 在线出版时间: 0000-00-00

CNKI 在线出版时间: 0000-00-00

[6](Key Laboratory of Knowledge Engineering with Big Data(Hefei University of Technology), Ministry of Education, Hefei 230009, China)

Abstract: Nonoverlapping conditional pattern mining is a method of gap constrained sequence pattern mining. Compared with similar mining methods, this method is easier to find valuable frequent patterns. The core of the problem is to calculate the support (or the number of occurrences) of a pattern in the sequence, and then determine whether the pattern is frequent. The essence of calculating the support is the pattern matching under nonoverlapping condition. The current studies employ the iterative search to find a nonoverlapping occurrence, and then prune the useless nodes to calculate the support of the pattern. The computational time complexities of these algorithms are $O(m*m*n*W)$, where m, n, and W are the pattern length, sequence length, and maximum gap, respectively. In order to improve the calculation speed of pattern matching under nonoverlapping condition, and effectively reduce sequence pattern mining time, this paper proposes an efficient and effective algorithm, which converts the pattern matching problem into a Nettree, and then starts from the minroot node of the Nettree, and adopts the backtracking strategy to iteratively search the leftmost child to calculate the nonoverlapping minimum occurrence. After pruning the occurrence on the Nettree, the problem can be solved without further searching and pruning invalid nodes. This paper proves the completeness of the algorithm and reduces the time complexity to $O(m*n*W)$. On this basis, the paper continues to indicate that there are other three similar solving strategies for this problem, iteratively finds the leftmost parent path from the leftmost leaf, the rightmost child path from the rightmost root, and the rightmost parent path from the rightmost leaf. Extensively experimental results verify the efficiency of the algorithm in this paper, especially, the mining algorithm adopting this method can reduce the mining time.

Key words: Pattern matching; Sequenc pattern mining; Nonoverlapping condition; Nettree; Backtracking strategy

模式匹配作为计算机科学的基本问题之一,包括大数据挖掘[1]在内的诸多研究都建立在模式匹配的基础上,如序列模式挖掘[2,3]、时间序列分析与预测[4]、生物序列数据分析[5]、网络入侵检测[6]、正负序列分析[7,8]以及大空间数据库中的字符串搜索[9,10].近年来,具有间隙约束的模式匹配逐渐引起研究者们的广泛关注,如 Manber 等人[11]最先研究了具有间隙约束的模式匹配问题,但其模式中只有一个可变间隙,具有一定的局限性.随着研究的深入更多学者将注意力转向具有多个可变间隙约束模式匹配问题(或称间隙约束模式匹配问题)[12].具有多个可变间隙的模式匹配在诸多领域具有广泛的应用,例如,在计算生物学中,Navarro 和 Raffinot[13]采用此种模式匹配提出了更为优化的搜索方法用来寻找特殊蛋白质位点;Drory 等人[14]实现了 RNA 序列的 motif 结构检测;在文本匹配方面,Cole 等人[15]在可变间隙近似模式匹配下有效的判断模式串是否在指定的文本或字典中. 特别需要指出的是,具有间隙约束模式匹配在序列模式挖掘领域具有重要应用[16,17].传统序列模式挖掘研究只关注模式在序列中是否存在,而不关注模式在序列中出现的次数.这导致序列串“AC”与序列串“ACACACAC”的作用相同,因此传统序列模式挖掘方式会在诸如 DNA 序列等长序列挖掘中丢失诸多重要信息.为了弥补传统序列模式挖掘的不足,可重复序列模式挖掘孕育而生[18],这种挖掘方法更加关注模式在序列中出现的次数[19,20].为了避免匹配的序列间隔过大,间隙约束序列模式挖掘孕育而生,其核心工作之一是计算模式在序列中的出现次数,即支持度,实质上就是间隙约束模式匹配问题[21].在间隙约束序列模式挖掘和模式匹配问题中,模式串可以写作 $P = p_1[min_1,max_1]p_2\cdots [min_{j-1},max_{j-1}]\ p_j\cdots [min_{m-1},max_{m-1}]p_m$ 的形式[22],其中 min_{j-1} 和 max_{j-1} 分别指 p_{j-1} 和 p_j 之间通配符的最小和最大个数[23].间隙约束作为一种新型的通配符,比传统通配符“?”和“*”更具灵活性与适用性,这是因为其允许在 p_{j-1} 和 p_j 之间通配的字符数量是一个范围值,而非一个确定值.

无重叠条件的模式匹配作为间隙约束模式匹配的一种方法,最早是在文献[24]中提出,其是指允许序列中的任意位置的字符重复使用,但不允许同一字符在相同位置多次使用.Wu 等人[25]严格形式化地给出了无重叠条件模式匹配的定义,并理论证明了该问题的复杂度为 P.之后, 文献[26]研究了无重叠条件序列模式挖掘,并有效地解决了间隙约束序列模式挖掘难以兼顾 Apriori 性质和挖掘完备性的问题[5,20].下面举例说明无重叠条件模式匹配问题.

例 1. 给定序列串 $S=s_1s_2s_3s_4s_5s_6s_7s_8$=aggcaaga,模式串 $P=p_1[min_1,max_1]p_2[min_2,max_2]p_3$ =a[0,1]g[0,1]a.

子模式 a[0,1]g 指在 a 与 g 之间可以没有间隙或者有一个通配符“?”,也就是说 a 与 g 之间可以通配 0 到 1 个字符.由图 1 可知,例 1 中满足间隙约束的出现共有 3 个,分别是<1,3,5>,<5,7,8>,<6,7,8>.出现<5,7,8>和<6,7,8>均在相同位置使用了 s_7 的“g”和 s_8 的“a”,因此这两个出现不满足无重叠条件.尽管出现<1,3,5>和<5,7,8>均使用

了字符 s_5="a",但这两条出现中的 s_5 分别与 p_3 和 p_1 进行匹配, 因此<1,3,5>和<5,7,8>构成了两个无重叠出现.

	1	2	3	4	5	6	7	8	
s =	a	g	g	c	a	a	g	a	
	a	·	g	·	a	·	·	·	1st occurrence
	·	·	·	·	a	·	g	a	2nd occurrence
	·	·	·	·	·	a	g	a	3rd occurrence

Fig.1 All occurrences of pattern P in sequence S

图 1 P 在 S 中的所有出现

尽管 Wu 等人[25,26]提出基于网树结构的模式匹配算法 NETLAP-Best 和 NETGap,这两种算法均采用在网树结构上剪枝无效结点的策略,虽然得到了问题的完备解,但时间复杂度均为 $O(m*m*n*W)$,这里 m,n 和 W 分别为模式长度,序列长度及最大间隙.研究具有更低时间复杂度的完备性无重叠模式匹配算法具有重要的意义.这是因为其是无重叠序列模式挖掘研究的重要基石.序列模式挖掘研究主要包含候选模式生成和支持度计算(即模式匹配问题)两个部分.因此影响模式挖掘速度的主要因素有两个:候选模式剪枝策略和模式支持度计算效率.虽然候选模式剪枝策略对挖掘效率具有重要影响,但是在候选模式剪枝策略不变的情况下,模式支持度计算效率在一定程度上也会影响挖掘效率,此外支持度计算的准确性能够保证挖掘算法的完备性.

本文对具有长度约束的无重叠条件严格模式匹配问题进行研究,提出了一种高效求解算法 Nonoverlapping NettreeBacktracking(NetBack),该算法结合回溯策略,无需检测网树中无效结点,采用在网树上查找最左孩子路径(Leftmost child path)的方式获得一个无重叠出现,并迭代此过程,即可实现无重叠条件模式匹配问题的求解.NetBack 算法可将无重叠严格模式匹配算法的时间复杂度从 $O(m*m*n*W)$降为 $O(m*n*W)$.在此基础上,本文指出,除了最左孩子路径以外,还可以采用其他三种方式进行求解,分别为最左双亲路径(Leftmost parent path)、最右双亲路径(Rightmost parent path)和最右孩子路径(Rightmost child path),其中最左双亲路径和最左孩子路径所获得的无重叠出现完全一致;最右双亲路径和最右孩子路径所获得的无重叠出现完全一致.这四种方法均为完备性求解算法,能够找到的无重叠出现数目是一致的.本文在真实生物数据上进行了大量的实验,与多种算法进行了对比,并将 NetBack 算法应用于序列模式挖掘中进行实验,实验结果验证了本文算法的正确性与高效性.

本文第 1 节总结了相关工作进展;第 2 节给出问题定义及相关理论证明;第 3 节介绍了网树,并提出了本文求解算法 NetBack,分析了 NetBack 算法的空间复杂度及时间复杂度;第 4 节实验验证了算法的正确性及高效性;第 5 节得出结论.

1 相关工作

目前,具有间隙约束的模式匹配问题可从以下几个方面进行划分:严格模式匹配还是宽松模式匹配,其中严格模式匹配可以细分为一次性条件、无重叠条件和无特殊条件;是否具有长度约束;是否为精确模式匹配.

文献[27]首次将具有间隙约束的模式匹配问题划分为严格模式匹配和宽松模式匹配.严格模式匹配[28,29]指使用模式串中的对应字符在满足间隙约束的情况下在序列串中对应的位置索引来表示出现;而宽松模式匹配[13,30]只使用模式串中最后一个字符在序列串中的位置表示一个出现.严格模式匹配又存在三种不同条件的研究分别是:无特殊条件[31,32,33]、一次性条件[5,28]和无重叠条件[24,25,34].例 2 用来说明宽松模式匹配及三种条件下的严格模式匹配之间的关系.

例 2. 给定与例 1 相同的序列串 S 及模式串 P.表 1 给出了不同条件下的出现.

如表 1 所示,若按照宽松模式匹配,例 2 中有两个出现,分别为<5>和<8>.显然宽松模式匹配忽略了匹配过程,而严格模式匹配更注重匹配的细节.一次性条件指序列中任意位置的字符只能使用一次,所以满足该条件的严格模式匹配出现为<1,3,5>和<6,7,8>.根据无重叠的定义可知,满足无重叠条件的严格模式匹配的出现有两个,可以是{<1,3,5>,<5,7,8>}或{<1,3,5>,<6,7,8>}.无特殊条件指序列中任何位置的字符都可以多次使用,所以无特殊条件下的严格模式匹配出现有三个,即<1,3,5>,<5,7,8>和<6,7,8>.相对于一次性条件及无重叠条件,无特殊

条件可以得到问题的全部解,但这也会造成问题解空间呈指数级增长,并且可能得到过多的无用解.一次性条件虽然有效地限制了解空间,但其要求过于苛刻,也会遗漏许多有价值的信息.而无重叠条件处于两者之间,即限制了解的个数,又可得到更多的重要信息.

Table 1 The occurrences under the different methods

表 1 不同条件下的出现

条件	出现数	出现
宽松模式匹配	2	<5><8>
无特殊条件下严格模式匹配	3	<1,3,5><5,7,8><6,7,8>
一次性条件下严格模式匹配	2	<1,3,5><6,7,8>
无重叠条件下严格模式匹配	2	<1,3,5><5,7,8>或<1,3,5><6,7,8>

对出现的总长度进行约束,即出现中最大位置与最小位置的差需满足一定范围.目前许多模式匹配研究都使用了长度约束[29,35],并且在序列模式挖掘[36,37]中也引入了长度约束.假设例 1 中的最小长度约束和最大长度约束分别为 4 和 5,那么满足长度约束的出现只有<1,3,5>和<5,7,8>,因为这两个出现的长度分别为 5-1+1=5,8-5+1=4,在长度约束范围内.但出现<6,7,8>不合法,因为它的长度为 8-6+1=3,不在范围内.

模式匹配问题作为序列模式挖掘的核心,与序列模式挖掘的关系密不可分.因此表 2 中对比了几种具有间隙约束的模式匹配工作及序列模式挖掘工作.

Table 2 Comparison of related work

表 2 具有间隙约束的模式匹配及挖掘对比

相关工作	匹配/挖掘	间隙个数	严格/宽松	匹配类型	约束条件	长度约束
Manber 等人[11]	模式匹配	一个可变间隙	严格	精确	无	无
Bille 等人[38]	模式匹配	多个可变间隙	宽松①/严格	精确	-②/无	-
Fredriksson 等人[30]	模式匹配	多个可变间隙	宽松	近似	-	-
Zhang 等人[18]	序列模式挖掘	多个可变间隙	严格	精确	无	无
Chen 等人[29]	模式匹配	多个可变间隙	严格	精确	一次性	有
Liu 等人[39]	模式匹配	多个可变间隙	严格	精确	一次性	有
Ding 等人[24]	序列模式挖掘	多个可变间隙	严格	精确	无重叠	有
Wu 等人[26]	序列模式挖掘	多个可变间隙	严格	精确	无重叠	有
Wu 等人[34]	模式匹配	多个可变间隙	严格	近似	无重叠	有
Wu 等人[25]	模式匹配	多个可变间隙	严格	精确	无重叠	有
本文	模式匹配	多个可变间隙	严格	精确	无重叠	有

注:①该文献设计了两种算法,分别对严格模式匹配和宽松模式匹配进行研究

②宽松模式匹配通常不考虑一次性条件和长度约束问题.本文以“-”表示不予考虑

通过表 2 可以看出本文的研究与文献[25]是一致的,都是无重叠条件下具有多个可变间隙约束的严格精确模式匹配.而本文的研究与文献[25]的差异在于:

(1)降低了求解算法的时间复杂度.尽管 Wu 等人[25,26]利用网树结构得到了无重叠严格模式匹配问题的完备解,但每次获得一个无重叠出现后,需要在网树上查找并剪枝无效结点,以实现完备性求解.这造成上述文献的求解算法时间复杂度较高.本文拟研究一个更高效的完备性求解算法,无需查找并剪枝无效结点就可实现问题求解,并将完备性算法的时间复杂度从 $O(m*m*n*W)$降低为 $O(m*n*W)$.例 3 用来说明文献[25]提出的 NETLAP-Best 算法工作原理.

例 3. 假设序列串 S=actataagg,模式串 P=a[0,1]t[0,1]a[1,3]g.根据序列串及模式串,可得到如图 2 网树.

由图 2 可以看出,模式 P 在序列 S 中存在两条无重叠出现,分别为<1,3,4,8>和<4,5,7,9>.若采用文献[25]的 NETLAP-Best 算法进行求解,从最大叶子结点,即第四层的结点 9 开始向上迭代寻找最右双亲结点,找到第一条出现<4,5,7,9>后必须对网树进行无效结点的查找及剪枝,即必须剪枝第三层结点 6,这是因为 NETLAP-Best 算法不采用回溯策略,若不剪枝第三层结点 6,在找到子出现<6,8>后,会发现第二层结点 5 不可重用,从而放弃这条路径,进而丢失出现<1,3,4,8>.

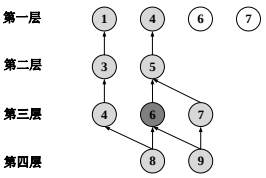

Fig.2 The nettree of pattern P in sequence S

图 2 模式 P 在序列 S 上的等价网树

(2)本文设计了具有回溯机制的最左孩子路径的求解算法,并理论证明了该算法的完备性.在此基础上,进一步证明了除了该方法以外,还可以存在其他三种方式,并对这四种求解方式的解的特点进行了分析.

2 问题定义及分析

2.1 问题定义

定义 1. 长度为 n 的序列串 S 可以表示为 $S=s_1s_2\dots s_i\dots s_n(1\leqslant i\leqslant n)$,序列串 S 是一个有序的字母表,$s_i\in\Sigma$,Σ 代表一种符号集.

在 DNA 序列中,Σ 为集合{A,C,G,T},而在音乐序列中 Σ 是由数字构成的集合.

定义 2. 具有间隙约束的模式 P 可以表示为 $P=p_1[min_1,max_1]p_2\dots[min_{j-1},max_{j-1}]p_j\dots[min_{m-1},max_{m-1}]p_m(1<j\leqslant m)$,其中 min_{j-1} 和 max_{j-1} 是给定的两个非负整数,分别指 p_{j-1} 和 p_j 之间可以通配的最小和最大间隙,这样的间隙称为非负间隙.本文只考虑非负间隙下的模式匹配问题.

定义 3. 序列 S 中一组位置索引 $L=<l_1,l_2,\dots,l_j,\dots,l_m>$,其中 $1\leqslant l_j\leqslant n,1<j\leqslant m$,并满足以下条件:

$$s_{l_j}=p_j$$

$$min_{j-1}\leq l_j-l_{j-1}-1\leq max_{j-1}$$

$$MinLen\leq l_m-l_1+1\leq MaxLen$$

则称 L 是模式 P 在序列 S 中满足间隙约束及长度约束的一个出现,其中 l_m-l_1+1 为出现的长度,而 $MinLen$ 和 $MaxLen$ 分别表示出现的最小长度和最大长度约束.长度约束可表示为 LEN=[$MinLen$,$MaxLen$].

例 4. 假设序列串 S=aattatatt,模式串 P=a[0,1]t[0,1]a,$MinLen$=3,$MaxLen$=4.若不考虑长度约束那么 P 中一共有 4 个满足间隙约束的出现,分别为:<1,3,5>,<2,3,5>,<2,4,5>,<5,6,7>.出现<1,3,5>的长度是 5,即 5-1+1=5.在长度约束 LEN 为[3,4]的情况下,5 不在这个区间内,因此该出现不满足长度约束.

定义 4. 给定模式 P 在序列 S 中的两个出现 $L=<l_1,l_2,\dots,l_j,\dots,l_m>$和 $L'=<l'_1,l'_2,\dots,l'_j,\dots,l'_m>$,若对于所有的 $j(1\leqslant j\leqslant m)$都满足 $l_j\neq l'_j$,则称 L 和 L'是两个无重叠出现.

定义 5. 假设出现 $L=<l_1,l_2,\dots,l_j,\dots,l_m>$是模式 P 在序列 S 上的一个出现,对于模式 P 在序列 S 上的其他出现表示为 $L'=<l'_1,l'_1,\dots,l'_j,\dots,l'_m>$,若对于任意的 $j(1\leqslant j\leqslant m)$,$l_j\leqslant l'_j$ 恒成立,则称出现 L 为模式 P 在序列 S 上的最小出现.同理可得最大出现定义.

定义 6. 令集合 R 表示模式 P 在序列 S 中的所有出现,其出现数量用$|R|$表示.假设集合 C 是集合 R 的一个子集,若集合 C 中任意两个出现均满足无重叠条件,则称集合 C 为一个无重叠出现集,其出现数量用$|C|$表示.若不存在无重叠出现集 C',使得$|C'|>|C|$成立,则称集合 C 为最大无重叠出现集.无重叠严格模式匹配问题是指寻找模式 P 在序列 S 上的最大无重叠出现集 C,并计算$|C|$的值.

例 5. 给定与例 4 相同的序列串与模式串,且 LEN=[3,4].模式 P 在序列 S 中满足间隙约束及长度约束的出现有 3 个,即 R={<2,3,5>,<2,4,5>,<5,6,7>},$|R|$=3.根据定义 6 可知,最大的无重叠出现集 C={<2,3,5>,<5,6,7>}或 C={<2,4,5>,<5,6,7>},$|C|$=2,即模式 P 在序列 S 中的无重叠出现数为 2.

2.2 问题分析

引理 1. 给定子模式 $p_j[min_j,max_j]p_{j+1}$ 的两个子出现为$<a_j,a_{j+1}>$和$<b_j,b_{j+1}>$,如果 $a_j<b_j$ 且 $a_{j+1}>b_{j+1}$,则$<a_j,b_{j+1}>$

和<b_j,a_{j+1}>也是两个子出现.反之,若 $a_j<b_j$ 且 $a_{j+1}<b_{j+1}$,<a_j,b_{j+1}>和<b_j,a_{j+1}>不一定为子出现.

证明.我们之前的工作[25]根据最小数最大数原理已经证明,如果存在两个子出现<a_j,a_{j+1}>和<b_j,b_{j+1}>并满足 $a_j<b_j$ 且 $a_{j+1}>b_{j+1}$,那么<a_j,b_{j+1}>和<b_j,a_{j+1}>也一定是两个子出现.相反,若 $a_j<b_j$ 且 $a_{j+1}<b_{j+1}$,<a_j,b_{j+1}>和<b_j,a_{j+1}>不一定为子出现.证毕.

引理 2. 无重叠条件下的严格模式匹配问题是一个 P 问题.

证明.令集合 R 为出现全集,即 R 是由在不考虑无重叠条件时模式 P 在序列 S 中所有满足间隙约束与长度约束的出现构成的集合.现有无重叠出现集 C,包含 k 个无重叠出现,即 C={<$c_{1,1}$,$c_{1,2}$,…,$c_{1,m}$>,<$c_{2,1}$,$c_{2,2}$,…,$c_{2,m}$>,…,<$c_{k,1}$,$c_{k,2}$,…,$c_{k,m}$>},其中 $c_{i,1}<c_{i+1,1}$($1\leqslant i<k$).根据引理 1 和我们之前的工作[25],可得到一个同样包含 k 个无重叠出现的集合 D,且 D 中的出现是有序的,即 D={<$d_{1,1}$,$d_{1,2}$,…,$d_{1,m}$>,<$d_{2,1}$,$d_{2,2}$,…,$d_{2,m}$>,…, <$d_{k,1}$,$d_{k,2}$,…,$d_{k,m}$>},其中 $d_{h,j}<d_{h+1,j}$($1\leqslant h<k$,$1\leqslant j\leqslant m$).根据我们之前的工作[25,26],对于集合 D 存在以下两种情况:

(1)令<$f_{k,1}$,$f_{k,2}$,…,$f_{k,m}$>为 R 中最大出现.那么,无论出现<$d_{k,1}$,$d_{k,2}$,…,$d_{k,m}$>是否是最大出现,都可被最大出现<$f_{k,1}$,$f_{k,2}$,…,$f_{k,m}$>替换.从 R 中删除<$f_{k,1}$,$f_{k,2}$,…,$f_{k,m}$>以及与它有重叠的出现,此时<$f_{k-1,1}$,$f_{k-1,2}$,…,$f_{k-1,m}$>为最大出现,同理<$d_{k-1,1}$,$d_{k-1,2}$,…,$d_{k-1,m}$>可被<$f_{k-1,1}$,$f_{k-1,2}$,…,$f_{k-1,m}$>替换,重复上述过程直到 R 为空.可得到包含 k 个无重叠出现的无重叠最大出现集 F,F={<$f_{1,1}$,$f_{1,2}$,…,$f_{1,m}$>,<$f_{2,1}$,$f_{2,2}$,…,$f_{2,m}$>,…, <$f_{k,1}$,$f_{k,2}$,…,$f_{k,m}$>}.

(2)令<$g_{1,1}$,$g_{1,2}$,…,$g_{1,m}$>为 R 中最小出现.那么,无论出现<$d_{1,1}$,$d_{1,2}$,…,$d_{1,m}$>是否是最小出现,都可被最小出现<$g_{1,1}$,$g_{1,2}$,…,$g_{1,m}$>替换.从 R 中删除<$g_{1,1}$,$g_{1,2}$,…,$g_{1,m}$>以及与它有重叠的出现,此时<$g_{2,1}$,$g_{2,2}$,…,$g_{2,m}$>为最小出现,同理<$d_{2,1}$,$d_{2,2}$,…,$d_{2,m}$>可被<$g_{2,1}$,$g_{2,2}$,…,$g_{2,m}$>替换,重复上述过程直到 R 为空.可得到包含 k 个无重叠出现的无重叠最小出现集 G,G={<$g_{1,1}$,$g_{1,2}$,…,$g_{1,m}$>,<$g_{2,1}$,$g_{2,2}$,…,$g_{2,m}$>,…, <$g_{k,1}$,$g_{k,2}$,…,$g_{k,m}$>}.

所以,求解无重叠严格模式匹配问题可采取两种方式.一是选择最大出现得到无重叠最大出现集 F,我们称此方法为最大策略.二是选择最小出现得到无重叠最小出现集 G,我们称此方法为最小策略.综上,最大策略和最小策略都可得到无重叠严格模式匹配问题的完备解集,可证明此问题的计算复杂性为 P.证毕.

3 网树与 NetBack 算法

3.1 网树

我们首先介绍网树的基本定义.

定义 7. 网树结构 [25,26]类似于树结构,具有以下的特点:

(1)一棵网树可以有 n 个根结点,其中 $n\geqslant 1$.

(2)除根结点外的任意结点可以有不止一个双亲,并且一个结点的所有双亲结点必须在网树的同一层.

(3)网树允许同一结点出现在不同层,为了准确表示某一结点,使用 n^i_j 表示网树第 j 层的结点 i.

(4)从一个结点到其祖先结点(或子结点)的路径存在很多条.

定义 8. 若网树有 m 层,则称第 m 层的叶子为绝对叶子结点.

定义 9. 从树根结点到绝对叶子结点的一条路径称为完全路径.

定义 10. 在网树中,用<$n_1^{i_1}$, $n_2^{i_2}$,…, $n_m^{i_m}$>表示一条完全路径,相应的出现表示为<i_1, i_2,…, i_m>.一棵 m 层网树的完全路径长度是 m,因为绝对叶子结点在第 m 层.

引理 3. 网树上的一条完全路径对应模式 P 在序列 S 上的一个出现.

证明.在文献[40]中,我们已经证明模式 P 在序列 S 中的匹配问题可以转化为一棵网树进行求解,且每条完全路径对应一个出现,也就是说只要在网树中找到一条完全路径,就代表找到了一个出现.

定义 11. 对于结点 n^i_j 来说,在所有双亲结点中,最大标签的双亲称为最右双亲结点,最小标签的双亲称为最左双亲结点.在所有孩子结点中,最大标签的孩子称为最右孩子结点,最小标签的孩子称为最左孩子结点.

定义 12. 在网树中一个结点可以到达不止一个根结点,结点可到达的最小根结点称为 minroot,最大根结点称为 maxroot.根结点的 minroot 和 maxroot 都是它本身.同理,网树中的结点可以到达不止一个绝对叶子结点,结点可到达的最小绝对叶子结点称为 minleaf,最大绝对叶子结点称为 maxleaf.绝对叶子结点层的 minleaf 和

maxleaf 都是它本身.

定义 13. 选择最小树根结点,并迭代最左孩子形成的完全路径称为最左孩子路径.类似地,我们有如下三种定义:选择最小绝对叶子结点,并迭代最左双亲形成的完全路径称为最左双亲路径;选择最大绝对叶子结点,并迭代最右双亲形成的完全路径称为最右双亲路径;选择最大树根结点,并迭代最右孩子形成的完全路径称为最右孩子路径.

引理 4. 如果存在两条完全路径 A 和 B,若这两条路径没有经过相同的网树结点,则称 A 和 B 对应的出现为无重叠出现.

证明.完全路径 A 和 B 分别表示为$<n_1^{a_1}, n_2^{a_2},\ldots, n_m^{a_m}>$和$<n_1^{b_1}, n_2^{b_2},\ldots, n_m^{b_m}>$,对应的两个出现为 L_A=<a_1,a_2,…,a_m>和 L_B=<b_1,b_2,…,b_m>.因为 A 和 B 不包含相同结点,所以对于任意的 $i(1\leqslant i\leqslant m)$满足 $a_i\neq b_i$.因此,L_A 和 L_B 为两个无重叠出现.证毕.

根据无重叠出现的定义,序列中的字符并非是只可使用一次的,如何判断某一字符是否可以重复使用是一个难题.与其他无重叠模式匹配算法相比,使用网树结构可以高效地处理无重叠条件,只需保证每层每个结点最多使用一次即可.例 6 用于说明这一特点.

例 6. 根据给定序列串 S=atatgtagatgattga 和模式串 P=a[0,2]t[0,2]g[0,1]a,可得到如图 3 网树.

在图 3 中,可以找到所有满足间隙约束的出现,分别是<1,2,5,7>,<1,4,5,7><3,4,5,7>,<3,6,8,9>, <7,10,11,12>,<9,10,11,12>,<12,13,15,16>,<12,14,15,16>.易知,<1,2,5,7>与<3,4,5,7>是重叠的出现,因为这两个出现都使用了第三层的结点n_3^5和第四层的结点n_4^7.而出现<1,2,5,7>与出现<7,10,11,12>就是两个无重叠出现,因为在网树中结点n_1^7与结点n_4^7为不同结点.

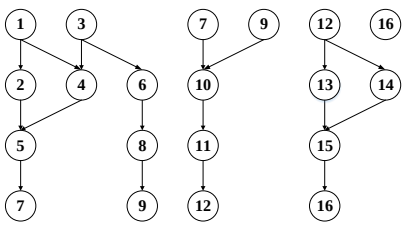

Fig.3 The nettree of pattern P in sequence S

图 3 模式 P 在序列 S 上的等价网树

3.2 NetBack算法

3.2.1 创建网树

我们通过例 7 来展示创建网树的原理并说明网树所具有的优势.

例 7. 给定与例 6 相同的模式串与序列串,若长度约束为 LEN=[5,7],那么利用网树结点的 minroot 和 maxroot 以及 minleaf 和 maxleaf 可以有效的处理长度约束,具有最小最大根及最小最大叶子标记的网树如图 4 所示.

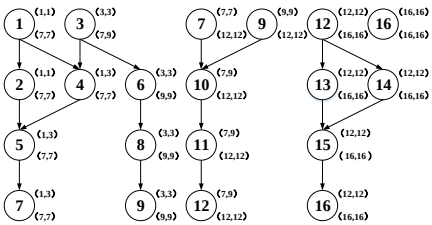

Fig.4 The Nettree with minroot, maxroot, minleaf, and maxleaf

图 4 具有最小最大根及最小最大叶子的网树

在图 4 中,结点右上的标记为此结点可到达的 minroot 和 maxroot,右下的标记为此结点可到达的 minleaf 和 maxleaf.这里以 s_4=t 为例,说明如何创建结点n_2^4并计算其 minroot 和 maxroot 值.由于 s_4=t=p_2 且 4-1-1=2 满足[0,2]间隙约束,此时创建结点n_2^4并与结点n_1^1建立父子关系,使其 minroot 和 maxroot 值与结点n_1^1的 minroot 和

maxroot 值一致,均为 1;又由于结点n_2^4与结点n_1^3也满足间隙约束,故结点n_2^4与结点n_1^3也建立父子关系,同时结点n_2^4的 minroot 从当前 1 和结点n_1^3的 minroot=3 中选择最小值,故结点n_2^4的 minroot=1;结点n_2^4的 maxroot 从当前 1 和结点n_1^3的 maxroot=3 中选择最大值,故结点n_2^4的 maxroot=3.因此结点n_2^4右上方标记为(1,3).当网树创建完毕后,从叶子层至树根层逐层更新每个结点的 minleaf 和 maxleaf.这里以结点n_1^3为例,易知其可抵达的最小叶子为 7,最大叶子为 9,于是结点n_1^3右下方的标记为(7,9).运用最小最大根和最小最大叶子,可以方便地确定该结点是否满足长度约束,若该结点的长度区间[minleaf-maxroot+1,maxleaf-minroot+1]与长度约束 *LEN* 存在交集,那么此结点就满足长度约束,否则就不满足长度约束.如结点n_1^9,其 minleaf 和 maxleaf 为(12,12),minroot 和 maxroot 为(9,9),那么此结点的长度区间为[4,4],与长度约束 *LEN*=[5,7]不存在交集,所以结点n_1^9不满足长度约束,即出现<9,10,11,12>不满足长度约束.因此本例中满足间隙约束及长度约束的出现全集 *R* 为{<1,2,5,7>,<1,4,5,7>, <3,4,5,7>, <3,6,8,9>, <7,10,11,12>,<12,13,15,16>,<12,14,15,16>}.

本文创建网树的基本原理如下[25,26]:逐一读取序列中每个字符s_i,若其与p_1相同,则直接创建网树树根n_1^i,并设置最小最大根均为 *i*;若其与p_j(*j*>1)相同且第 *j*-1 层存在与s_i满足间隙约束条件的结点,则创建结点n_j^i,并在第 *j*-1 层寻找所有与结点n_j^i满足间隙约束的双亲结点,建立父子关系并更新结点n_j^i的最小最大根.当全部字符读取完毕后,网树创建完毕.此时从网树的叶子层向树根层,逐层逐一更新各个结点的最小最大叶子.具体创建算法如算法 1 所示:

算法 1 CreNetTree

输入: 序列 *S* 及模式 *P*
输出: *Nettree*

```
1:    for i=1 to n step 1 do
2:      for j=1 to m step 1 do
3:        if p1 = si then
4:          Nettree[1].add(n1^i)并设置最小最大根均为 i;
5:        else if pj = si 且第 j-1 层存在与 i 满足间隙约束的结点 then
6:          Nettree[j].add(nj^i);
7:          与第 j-1 层满足间隙约束的所有结点建立父子关系,并更新nj^i的最小最大根;
8:        end if
9:      end for
10:   end for
11:   for j=m to 1 step -1 do
12:     for i=1 to 第 j 层结点数 step 1 do
13:       更新第 j 层第 i 个结点的最小最大叶子;
14:     end for
15:   end for
16:   return Nettree;
```

3.2.2 回溯算法

通过例 8 来说明采用回溯策略获得最左孩子路径的原理.

例 8. 给定与例 7 相同的模式串与序列串和长度约束,其网树如图 5 所示.

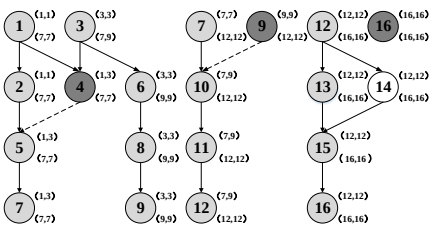

Fig.5 Leftmost child path

图 5 最左孩子路径

首先选择最小树根结点n_1^1,该结点满足长度约束,这是因为其[minleaf-maxroot+1, maxleaf-minroot+1] =[7-1+1,7-1+1]=[7,7]与长度约束 *LEN*=[5,7]存在交集.依次向下选择最左孩子结点,很容易得到一条完全路径<n_1^1,n_2^2,n_3^5,n_4^7>,对应的出现为<1,2,5,7>,然后删除这些结点.继续向后访问结点n_1^3,此结点满足条件,向下选择最

左孩子结点n_2^4,发现该结点没有可用孩子结点,那么此时需要回溯到结点n_1^3选择其他可用最左孩子结点,即n_2^6,继续向下遍历网树得到出现<3,6,8,9>.同理,依次向后遍历树根结点,可得到出现<7,10,11,12>和<12,13,15,16>.需要注意的是,在访问结点n_1^9时,发现该结点不满足长度约束,那么就不需继续向下寻找,只需直接向后访问结点n_1^{12},运用这一方法可以有效地加快搜索速度.

算法 2 给出了回溯算法.若当前处理层数 *level* 小于 1,则返回标志为-1,表示当前树根下不存在最左孩子路径;若 *level* 大于 *m*,则返回标志为 1,表示成功获得最左孩子路径 *G*;若 *level* 在 1 到 *m* 区间,*G*[*level*+1]获得下一个满足长度约束的最左孩子,若 *G*[*level*+1]不为空,则递归下一层结点寻找最左孩子路径,否则返回上一层结点寻找最左孩子路径.

算法 2 BackTracking

输入: 当前处理层数 *level*,网树 *Nettree*,网树深度 *m*,长度约束 *LEN* 以及最左孩子路径 *G*
输出: 结果标志

```
1:    if level<1 then return -1;
2:    if level>m then return 1;
3:    G[level+1]=G[level]的下一个满足长度约束的最左孩子;
4:    if G[level+1]<>NULL then
5:       return BackTracking(level+1, Nettree, m, LEN, G);
6:    else
7:       return BackTracking(level-1, Nettree, m, LEN, G);
8:    end if
```

3.2.3 求解算法 NetBack

定理 1. 无重叠出现 L_1 为最小出现,则无重叠出现 L_2 不会也不可能用到 L_1 左边的结点.

证明.采用反证法证明.假设无重叠出现 $L_1=<a_1,a_2,\ldots a_j,a_{j+1},\ldots,a_m>$,为当前最小出现,且无重叠出现 L_2 可用 L_1 左边的结点,即同时存在出现 $L_2=<b_1,b_2,\ldots b_j,b_{j+1},\ldots,b_m>$,其中 $1\leqslant k\leqslant j$ 时 $a_k<b_k$,且 $j+1\leqslant k\leqslant m$ 时 $a_k>b_k$.根据引理 1 可知,若同时存在出现 L_1 和 L_2,那么一定存在出 $L_1'=<a_1,a_2,\ldots a_j,b_{j+1},\ldots,b_m>$和出现 $L_2'=<b_1,b_2,\ldots b_j,a_{j+1},\ldots,a_m>$可替换出现 L_1 和 L_2.又因为 $j+1\leqslant k\leqslant m$ 时 $a_k>b_k$,那么 L_1 与 L_1'相比,L_1'为当前最小出现,这与假设 L_1 为最小出现矛盾.因此,若无重叠出现 L_1 为最小出现,那么无重叠出现 L_2 一定不会用到 L_1 左边的结点.证毕.

同理,若无重叠出现 L_3 为当前最大出现,则无重叠出现 L_4 不会也不可能用到 L_3 右边的结点.

算法 3 给出本文 NetBack 算法.该算法首先调用算法 1 创建网树;然后对最小网树树根结点调用算法 2 回溯获得最左孩子路径 *G*;如果结果为 1,则无重叠出现集 *C* 获得路径 *G* 所对应的最小出现,并依据定理 1 删除路径 *G* 及其以左结点.迭代这一过程直至网树中所有树根结点处理完毕.

算法 3 NetBack

输入: 序列 *S*,模式 *P* 以及长度约束 *LEN*
输出: 无重叠出现集 *C*

```
1:    使用算法 1 创建网树;
2:    for l=1 to 根结点个数 step 1 do
3:       G[1]=第 l 个根结点;
4:       ret=BackTracking(1,Nettree,|P|,G);
5:       if ret=1 then
6:          C=C∪G.nodelabel;
7:          依据定理 1,删除路径 G 及其以左结点;
8:       end if
9:    end for
10:   return C;
```

3.3 算法分析

定理 2. NetBack 算法是完备算法.

证明.首先证明 NetBack 算法采用算法 2 获得最左孩子路径的方式是一种最小策略.显然算法 2 中 g_1 是最小树根结点,因此 g_1 是所有出现中的最小值.现在假设 g_k 是所有出现在第 *k* 层的最小值且 g_{k+1} 是 g_k 的最左孩子结点($1\leqslant k<m$).我们将证明 g_{k+1} 是所有出现在 *k*+1 层的最小值.采用反证法证明.假设 l_{k+1} 是所有出现在第 *k*+1 层的最小值,即 $g_{k+1}>l_{k+1}$.设 l_{k+1} 的最左双亲是 l_k.则 l_k 与 g_k 存在以下三种情况:

(1) $l_k<g_k$.这与假设 g_k 是所有出现在第 *k* 层的最小值矛盾.

(2) $l_k>g_k$.因此<g_k,g_{k+1}>和<l_k,l_{k+1}>均为子出现.因为 $g_{k+1}>l_{k+1}$,根据引理 1,<g_k,l_{k+1}>也是一个子出现.又由于 g_k 是所有出现在第 k 层的最小值,因此 $g_k<l_k$.这与假设 l_k 是 l_{k+1} 的最左双亲矛盾.

(3) $l_k=g_k$.这样<g_k,g_{k+1}>与<g_k,l_{k+1}>也是两个子出现.但是 $g_{k+1}>l_{k+1}$,这与假设 g_{k+1} 为 g_k 的最左孩子矛盾.

综上,所有情况均不成立,所以 g_{k+1} 是所有出现在 k+1 层的最小值.故算法 2 采用最左孩子路径得到的出现为模式在序列中的最小出现.通过引理 2 可知,最小策略可得到完备解集,因此 NetBack 算法是完备算法.证毕.

定理 3. NetBack 算法的空间复杂度为 $O(m*n*W)$,其中,m,n 和 W 分别为模式长度,序列长度及最大间隙.

证明.由于网树最多有 m 层,每层结点最多有 n 个,每个结点最多有 W 个双亲结点,因此 NetBack 算法在最坏情况下的空间复杂度为 $O(m*n*W)$.证毕.

定理 4. NetBack 算法的时间复杂度为 $O(m*n*W)$.

证明. NetBack 算法时间开销由两部分组成:创建网树和计算出现.在创建网树方面,由于算法 1 的第 7 行结点n_j^i最多存在 W 个双亲结点,易知网树创建及更新最小最大根的时间复杂度为 $O(m*n*W)$.同理更新最小最大叶子的时间复杂度也为 $O(m*n*W)$.在计算出现方面,根据定理 1 可知,每个结点最多被访问一次.根据定理 3 可知,网树上最多有个 $m*n$ 结点,因此计算出现的时间复杂度为 $O(m*n)$.因此 NetBack 算法的时间复杂度为 $O(m*n*W+m*n*W+m*n)=O(m*n*W)$.证毕.

例如,在例 8 中采用最左孩子路径方式,从最小树根结点开始结合回溯策略得到无重叠最小出现集,在这个过程中每个结点最多被访问一次.找到第二个无重叠出现<3,6,8,9>后,它左边的结点均不会被后面的出现使用,也就是说结点n_2^4不可能与结点n_1^7及它之后的根结点存在父子关系,于是无需再次访问结点n_2^4,而结点n_2^{14}从未被访问.在图 5 中,白色结点均未被访问过,其他颜色结点均被访问过一次.

3.4 其他三种寻径算法及分析

除最左孩子路径(Leftmost child path)外,还存在三种类似的寻径方式,分别是最左双亲路径(Leftmost parent path)、最右双亲路径(Rightmost parent path)、最右孩子路径(Rightmost child path).这四种不同的寻径方式分别对网树进行不同方向的遍历,均可与回溯策略相结合求得无重叠严格模式匹配问题的完备解集.

定理 5. 最左双亲路径得到的出现与最左孩子路径得到的出现相同,也是最小出现.

证明.采用反证法证明.假设当前网树上最小出现 L_A=<a_1,a_2,…,a_m>与最左双亲路径策略所获的第一个出现 L_B=<b_1,b_2,…,b_m>是不同的.由最左双亲定义可知,b_m 为第 m 层可到达树根结点层的最小绝对叶子结点.这样存在如下两种情况:

(1)a_j 与 b_j 完全不同,即所有的 $a_j\neq b_j(1\leqslant j\leqslant m)$.由于 L_A 是最小出现,因此有 $a_j<b_j$（$1\leqslant j\leqslant$m）,这样 $a_m<b_m$,这就是说能够从绝对叶子层到达树根层的最小结点是 a_m,这与假定 b_m 是可到达树根结点层的最小绝对叶子结点矛盾.

(2)a_j 与 b_j 局部有不同.由于 L_A 是最小出现,所以有 $a_j<b_j(1\leqslant j\leqslant m)$.我们从如下两方面进行分析:

①假定 $j=m$,即 $a_m<b_m$,这与情形(1)相同,这种情况与假定 b_m 为第 m 层可到达树根结点层的最小绝对叶子结点矛盾.

②假定 $1\leqslant j<m$,且 $a_{j+1}=b_{j+1}$.由于 $a_j<b_j$,且 b_{j+1} 也是 a_j 的孩子结点,即 a_j 也是 b_{j+1} 的双亲结点.因此对与 b_{j+1} 结点来说,最左双亲结点是 a_j,这与在出现 L_B 中 b_{j+1} 选择 b_j 作为最左双亲结点矛盾.

综上,所有假设均存在矛盾,因此 L_B 是与 L_A 完全一致的出现,即最左双亲路径得到的出现与最左孩子路径得到的出现都是最小出现.证毕.

例 9. 给定与例 7 相同的模式串与序列串和长度约束,采用最小策略与回溯策略结合寻找无重叠出现.

根据例 8 已知采用最左孩子路径找到的第一个出现为<1,2,5,7>,而<1,2,5,7>是全集 R 中的最小出现.从 R 中删除<1,2,5,7>及与它重叠的出现后,出现<3,6,8,9>为当前最小出现,最左孩子路径找到的第二个出现就是<3,6,8,9>.以此类推,最左孩子路径可得到无重叠最小出现集{<1,2,5,7>,<3,6,8,9>,<7,10,11,12>,<12,13,15,16>}.若采用最左双亲路径,如图 6 所示,首先访问最小绝对叶子结点n_4^7,向上选择最左双亲结点n_3^5,迭代此过程找到第一个出现<1,2,5,7>.以此类推,得到无重叠出现集{<1,2,5,7>,<3,6,8,9>,<7,10,11,12>,<12,13,15,16>}.这与最左孩

子路径得到的结果是一致的.

综上所述,只要采用最小策略,无论是最左孩子路径方式还是最左双亲路径方式,均能得到无重叠最小出现集.

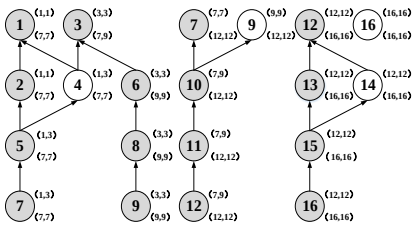

Fig.6 Leftmost parent path

图 6 最左双亲路径

例 10. 给定与例 7 相同的模式串与序列串和长度约束,最右双亲路径方式寻径过程如图 7 所示.

若采用最右双亲路径,如图 7 所示首先选择最大绝对叶子结点n_4^{16},向上选择最右双亲结点n_3^{15},迭代此过程可得到第一个出现<12,14,15,16>.向前访问绝对叶子结点n_4^{12},迭代选择其最右双亲结点,当选择结点n_2^{10}的最右双亲时,发现结点n_1^9不满足长度约束,根据回溯策略,此时选择结点n_1^7,得到第二个出现<7,10,11,12>.同理,依次向前遍历绝对叶子结点,可得到出现<3,6,8,9>和<1,4,5,7>.特别地,在寻找出现<1,4,5,7>时,向上选择结点n_2^4的最右双亲结点时发现其最右双亲结点n_1^3已被使用,所以根据回溯策略此时应选择结点n_1^1作为双亲结点.

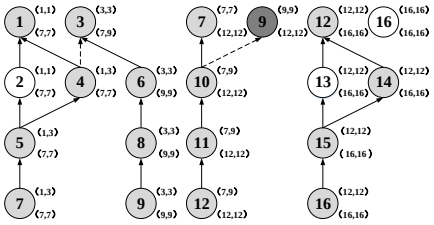

Fig.7 Rightmost parent path

图 7 最右双亲路径

定理 6. 最右双亲路径得到的出现是模式在序列中的最大出现.

证明.同定理 2 的证明,可以证明最右双亲路径得到的出现是模式在序列中的最大出现.

定理 7. 最右孩子路径得到的出现与最右双亲路径得到的出现相同,也是最大出现.

证明.同定理 5 的证明,可以证明最右孩子路径得到的出现与最右双亲路径得到的出现是一致的,是模式在序列中的最大出现.

例 11. 给定与例 7 相同的模式串与序列串和长度约束,采用最大策略与回溯策略结合寻找无重叠出现.

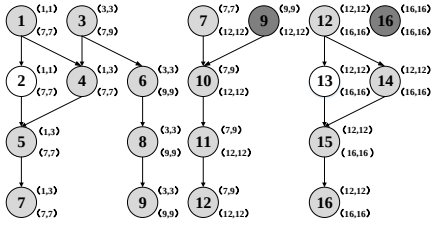

Fig.8 Rightmostchild path

图 8 最右孩子路径

根据例 10 可知采用最右双亲路径找到的第一个出现为<12,14,15,16>,在全集 R 中最大出现为<12,14,15,16>.从 R 中删除<12,14,15,16>及与它重叠的出现后,出现<7,10,11,12>为当前最大出现.最右孩子路径找到的第二个出现就是<7,10,11,12>.以此类推,最右双亲路径可得到无重叠最大出现集{<1,4,5,7>,<3,6,8,9>,

<7,10,11,12>,<12,14,15,16>}.若采用最右孩子路径,如图 8 所示,首先选择最大根结点n_1^{16},但该结点不满足约束条件,向前访问结点n_1^{12},选择其最右孩子结点n_2^{14},迭代此过程找到第一个出现<12,14,15,16>.以此类推,得到无重叠出现集{<1,4,5,7>,<3,6,8,9>,<7,10,11,12>,<12,14,15,16>},这与最右双亲路径得到的结果是一致的.

综上,只要采用最大策略,无论是最右孩子路径方式还是最右双亲路径方式,均能得到无重叠最大出现集.

4 实验结果与分析

4.1 实验环境与数据

本文采用真实生物数据为测试序列可以从美国国家生物计算信息中心下载(http://www.ncbi.nlm.nih.gov/-genomes/FLU/SwineFlu.html),具体数据如表 3 所示,*S*1-*S*8 为测试序列,表 4 给出了实验所需的 9 种模式串.

实验运行的环境为:Intel(R)Core(TM)i5-7200U 处理器,主频 2.50GHZ,内存 8GB,Windows 7,64 位操作系统的计算机.程序开发环境为 VC++6.0.

Table 3 Real biologycial sequence

表 3 真实生物数据片段

序号	来源	长度
*S*1	Homo Sapiens CY058563	2286
*S*2	Homo Sapiens CY058562	2299
*S*3	Homo Sapiens CY058561	2169
*S*4	Homo Sapiens CY058556	1720
*S*5	Homo Sapiens CY058559	1516
*S*6	Homo Sapiens CY058558	1418
*S*7	Homo Sapiens AX829178	5393
*S*8	Homo Sapiens AX829174	10011

Table 4 Patterns

表 4 模式串

序号	模式串	长度约束
*P*1	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	[5,49]
*P*2	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	[7,65]
*P*3	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	[10,101]
*P*4	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	[8,96]
*P*5	a[0,10]a[0,10]t[0,10]c[0,10]g[0,10]g	[6,56]
*P*6	a[0,5]t[0,7]c[0,9]g[0,11]g	[5,37]
*P*7	a[0,5]t[0,7]c[0,6]g[0,8]t[0,7]c[0,9]g	[7,49]
*P*8	a[5,6]c[4,7]g[3,8]t[2,8]a[1,7]c[0,9]g	[22,52]
*P*9	c[0,5]t[0,5]g[0,5]a[0,5]a	[5,25]

4.2 对比算法简介

为了测试本文 NetBack 算法的求解质量与效率,与四种已有算法进行对比:INSGrow[22]、NETLAP-Nonpruning[25]、NETLAP-Best[25]及 NETGap[26].这四种算法概要说明如下:

1. INSGrow[24]算法:该算法是最早提出的无重叠严格模式匹配算法,其不采用回溯策略,而是基于最左原则寻找出现,其时间复杂度为 $O(m*n)$,但会造成丢失可行解的现象.与 INSGrow 算法相比,本文算法能够得到问题的完备解集.

2. NETLAP-Nonpruning[25]算法:该算法的时间复杂度与本文算法一致,但因其不采用回溯策略,也会造成可行解的丢失.与 NETLAP-Nonpruning[25]算法的解数量进行对比,验证本文算法中回溯策略的必要性.

3. NETLAP-Best[25]算法和 NETGap[26]算法:这两种算法均采用网树结构进行求解,其求解原理基本相似,都是在找到一个无重叠出现后必须对网树进行剪枝操作,以便实现完备性求解,因此这两种算法的时间复杂均为 $O(m*m*n*W)$.这两种算法的差异在于,NETLAP-Best 算法采用最右双亲路径方式,而 NETGap 算法采用最左孩

子路径方式.为了验证本文算法解的完备性,与 NETLAP-Best 算法和 NETGap 算法的解数量进行对比.同时,为了验证本文算法的高效性,与 NETLAP-Best 算法和 NETGap 算法的运行时间进行对比.

除本文 NetBack 算法外,还存在另外三种不同寻径方式的算法:NetBack-rrtl 算法、NetBack-rltr 算法和 NetBack-lltr 算法.这三种算法均采用网树结构与回溯策略相结合的方式，但分别采用最右孩子路径(从最大根结点开始向下迭代搜索最右孩子结点)方式、最右双亲路径(从最大绝对叶子结点开始向上迭代搜索最右双亲结点)方式和最左双亲路径(从最小绝对叶子结点开始向上迭代搜索最左孩子结点)方式实现求解.

4.3 实验结果与分析

五种算法在 9 个模式 8 种序列 *S*1-*S*8 上关于无重叠严格模式匹配问题的解如图 9 所示,五种算法在 9 个模式 8 种序列上的运行时间如表 5 所示.我们从如下几方面进行分析:

(1) NetBack 算法和 NETLAP-Best 算法、NETGap 算法都是完备算法,而 INSGrow 算法和 NETLAP-Nonpruning 算法会丢失可行解.在图 9 中 72 个实例上,NetBack 算法和 NETLAP-Best 算法、NETGap 算法结果个数是相同的,且总是大于或者等于 INSGrow 算法和 NETLAP-Nonpruning 算法的结果,这同时也验证了 NetBack 的正确性.例如,模式 *P*3 在序列 *S*1 上 NetBack 算法和 NETLAP-Best 算法、NETGap 算法的出现数均为 203,而 INSGrow 算法和 NETLAP-Nonpruning 算法得到的出现数分别为 80,151.因此,真实生物数据的实验结果验证了 NetBack 算法的完备性,同时也验证了回溯策略的必要性.

(2) NetBack 算法的求解速度快于 NETLAP-Best、NETLAP-Nonpruning 及 NETGap 算法.INSGrow 算法运行时间最短,这是因为 INSGrow 算法是最简单的,而其他算法均比 INSGrow 算法复杂,但是 INSGrow 算法会丢失大量可行解,解的质量明显差于其他算法.通过表 5 可以看出,在 72 个实例上,除 INSGrow 算法外,NetBack 算法在大多数情况下求解速度是最快的,个别情况下 NETLAP-Nonpruning 算法求解速度最快,均比 NETLAP-Best 算法和 NETGap 算法要快.例如,在表 5 中,9 个模式在序列 *S*7 上 NetBack 算法的运行时间有 7 个是最快的,NETLAP-Nonpruning 算法的运行时间有 2 个是最快的,在其他实例中也有类似现象.在全部 72 个实例中,NetBack 算法有 50 个是求解速度最快的,这说明 NetBack 算法具有较优的求解效率.

(3) 尽管 NetBack 算法运行效率整体最优,但与 NETLAP-Best 算法和 NETGap 算法的差异不大,原因如下:从图 5 中可以看出,NETLAP-Best 算法和 NETGap 算法需要剪枝的结点数量在整个网树中的占比很小,其剪枝消耗的时间在整个算法运行时间中占比也很小.因此,尽管本文算法采用回溯策略降低了算法的求解复杂度,但是运行时间差异不大.

(4) NetBack 算法与 NETLAP-Nonpruning 算法运行时间相差较小.通过表 5 可以看出,NetBack 算法与 NETLAP-Nonpruning 算法运行时间相近,这是因为这两种算法的时间复杂度均为 $O(m*n*W)$.NetBack 算法与 NETLAP-Nonpruning 算法都不对网树进行无效结点的查找及剪枝,网树中的结点最多被访问一次,而 NETLAP-Best 算法和 NETGap 算法需要多次访问网树结点,时间复杂度较高.虽然运行时间相差较小,但 NETLAP-Nonpruning 算法不能得到问题的完备解.

本文 NetBack 算法采用最左孩子路径方式寻找出现,为了说明其他三种寻径方式的可行性与正确性,将 NetBack 算法与 NetBack-rrtl 算法(最右孩子路径方式)、NetBack-rltr 算法(最右双亲路径方式)、NetBack-lltr 算法(最左双亲路径方式)进行对比,图 10 给出了四种算法在 *S*1-*S*5 上的匹配结果总和,表 6 给出了四种算法在 *S*1-*S*5 上的运行时间.分析结果如下:

(1) 四种不同寻径方式的算法均为完备算法.通过图可以看到,其他三种算法的结果与 NetBack 算法一致,这验证了本文其他三种不同遍历方向的算法均为完备算法.例如模式 *P*1 在序列 *S*1-*S*5 上四种算法匹配到出现总数均为 127 个,其他模式均有相同现象.

(2) NetBack 算法与 NetBack-rrtl 算法、NetBack-rltr 算法、NetBack-lltr 算法运行时间大致相同,差距较小.说明这四种算法在效率上是一致的,这是因为这四种算法都使用了回溯策略和网树结构,只是寻径方式有所不同.

Fig.9 Comparison of the results on *S*1-*S*8

图 9 9 种模式串在序列 *S*1-*S*8 上匹配的结果比较

4.4 在序列模式挖掘中的应用

为了进一步说明本文算法的高效性,我们在 DNA 序列上进行了序列模式挖掘实验,具体数据如表 7 所示. 将使用 NetBack 算法计算支持度的挖掘算法 NOSEP-back 与文献[26]提出的不采用回溯策略的挖掘算法 NOSEP 算法进行对比,预设最小支持度为 500,长度约束为[1,30],间隙约束为[0,5].图 11 为 NOSEP-back 算法和

NOSEP 算法在序列 DNA1-DNA5 上频繁模式的个数,表 8 为这两种挖掘算法在序列 DNA1-DNA5 上的运行时间.

Table 5 Comparison of running time on *S*1-*S*8 (ms)

表 5 五种算法在 *S*1-*S*8 上运行时间比较 (ms)

序列	算法名称	不同模式下算法的运行时间								
		P1	P2	P3	P4	P5	P6	P7	P8	P9
S1	INSGrow	2.8	2.48	3.44	3.12	3.12	3.42	2.5	2.5	1.86
	NETLAP-Nonpruning	**13.11**	22.31	**41.65**	26.67	**26.37**	17.32	17.79	12.48	12.64
	NETLAP-Best	14.04	23.87	46.05	29.98	30.26	17.52	19.71	13.64	12.97
	NETGap	14.97	24.18	46.65	30.89	30.73	17.47	19.65	13.41	13.89
	NetBack	13.58	**21.84**	43.68	**26.52**	27.45	**15.76**	**17.54**	**12.17**	**11.86**
S2	INSGrow	2.8	2.5	2.82	2.8	3.44	2.48	3.12	3.12	2.5
	NETLAP-Nonpruning	**12.16**	24.02	**42.27**	29.8	**27.08**	16.38	**19.04**	11.85	14.35
	NETLAP-Best	14.05	24.88	44.62	31.2	29.84	18.01	20.66	12.16	15.76
	NETGap	14.35	25.43	44.78	32.07	31.83	18.09	21.21	13.41	15.29
	NetBack	12.78	**23.24**	43.37	**29.64**	28.46	**16.22**	19.34	**11.71**	**13.57**
S3	INSGrow	2.8	2.5	3.44	2.82	3.1	3.12	2.82	2.5	2.5
	NETLAP-Nonpruning	**11.39**	21.69	**39.94**	**26.68**	**25.28**	16.38	18.74	10.14	13.42
	NETLAP-Best	13.22	22.77	44.3	29.86	28.64	17.63	20.23	10.61	14.32
	NETGap	13.42	24.18	46.49	30.73	29.95	18.41	20.91	11.38	14.66
	NetBack	12.01	**21.37**	41.34	26.83	25.46	**15.91**	**18.61**	**10.14**	**12.79**
S4	INSGrow	2.8	2.18	2.82	2.8	2.8	1.88	2.5	2.48	1.86
	NETLAP-Nonpruning	12.5	21.9	43.7	26.5	29.7	14	**15.6**	9.3	10.9
	NETLAP-Best	15.6	20.53	45.2	28.1	26.2	15.6	16.84	9.36	11.54
	NETGap	14.68	19.66	38.7	25.28	25.28	15.3	16.86	9.68	10.92
	NetBack	**11.86**	**17.48**	**33.08**	**21.22**	**22.14**	**12.8**	**15.6**	**8.74**	**9.98**
S5	INSGrow	1.86	2.18	2.5	2.18	3.12	2.5	2.2	2.2	2.18
	NETLAP-Nonpruning	**7.8**	14.1	29.6	**17.2**	**17.2**	12.5	**12.4**	7.8	9.4
	NETLAP-Best	8.74	15.3	32.44	19.34	21.8	13.12	14.04	8.12	9.66
	NETGap	9.36	15.6	30.58	19.66	19.34	12.48	13.74	8.1	9.36
	NetBack	8.1	**14.04**	**28.4**	17.48	18.1	**11.26**	12.8	**7.5**	**8.74**
S6	INSGrow	2.18	2.2	2.5	2.2	2.5	2.5	2.5	2.5	2.18
	NETLAP-Nonpruning	9.4	14.1	**26.5**	17.1	17.2	10.9	15	7.8	9.4
	NETLAP-Best	9.06	14.34	29.34	17.78	17.46	12.16	13.4	7.48	9.98
	NETGap	9.66	14.98	29.64	18.1	18.42	11.22	13.1	7.18	9.98
	NetBack	**8.74**	**13.4**	27.14	**16.54**	**16.86**	**10.28**	**12.18**	**6.88**	**8.72**
S7	INSGrow	6.54	4.68	5.94	4.68	7.18	4.36	5	3.74	3.12
	NETLAP-Nonpruning	43.83	53.82	97.34	63.98	**59.12**	38.53	44.93	**22.46**	32.76
	NETLAP-Best	46.64	65.5	113.9	74.9	74.9	51.54	57.7	26.5	45.2
	NETGap	49.92	68.6	120.1	78	76.5	51.5	59.2	29.7	45.3
	NetBack	**43.6**	**53.1**	**96.7**	**62.4**	62.4	**37.4**	**44.62**	22.78	**30.88**
S8	INSGrow	18.42	11.54	14.66	12.18	23.72	13.1	15.3	9.98	9.06
	NETLAP-Nonpruning	80.34	97.82	**182.2**	113.9	121.7	82.68	**94.22**	**57.25**	71.76
	NETLAP-Best	85.33	101.4	192.1	118.4	130.7	86.31	98.43	59.91	72.86
	NETGap	89.39	104.9	200	124.8	135.8	89.08	104.7	62.55	75.51
	NetBack	**80.18**	**94.36**	182.8	**111.2**	**121.3**	**81.59**	94.38	58.18	**71.6**

通过图 11 可以看出,NOSEP-back 算法与 NOSEP 算法得到的频繁模式数量是一致的,这验证了 NetBack 算法支持度计算的正确性.通过表 8 可以看出,NOSEP-back 算法的运行速度在整体上要略快于 NOSEP 算法.造成这种现象的原因如下:一方面,我们优化了支持度计算方法,使 NetBack 算法较 NETGap 算法具有更低的时间复

杂度,但是由于网树中可剪枝结点的占比较小,所以虽然 NetBack 算法的实际运行速度快于 NETGap 算法,但差异不大.另一方面,候选空间剪枝对挖掘效率有着重要影响,本文只对支持度计算方法进行了优化,未对候选空间剪枝进行优化.因此 NOSEP-back 算法运行速度只是略快于 NOSEP 算法,但没有显著差异.

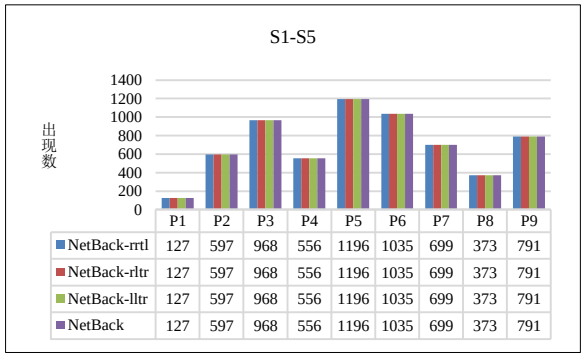

Fig.10 Comparison of the results on *S*1-*S*5

图 10 种模式串在序列 *S*1-*S*5 上匹配的结果总和

Table 6 Comparison of running time on *S*1-*S*5 (ms)

表 6 四种算法在 *S*1-*S*5 上运行时间比较 (ms)

运行时间(ms)	P1	P2	P3	P4	P5	P6	P7	P8	P9
NetBack-rrtl	58.73	99.96	185.51	121.33	125.62	72.76	84.99	51.81	57.79
NetBack-rltr	59.02	99.71	186.27	120.78	125.74	73.46	86.11	52.26	58.89
NetBack-lltr	57.78	99.68	186.13	121.83	123.94	72.52	84.47	52.22	58.67
NetBack	58.34	100.28	184.73	121.32	123.85	72.79	84.26	51.8	57.38

Table 7 Real biologycial sequence

表 7 真实生物数据片段

序号	来源	长度
DNA1	Homo Sapiens AL158070	6000
DNA2	Homo Sapiens AL158070	8000
DNA3	Homo Sapiens AL158070	10000
DNA4	Homo Sapiens AL158070	12000
DNA5	Homo Sapiens AL158070	14000

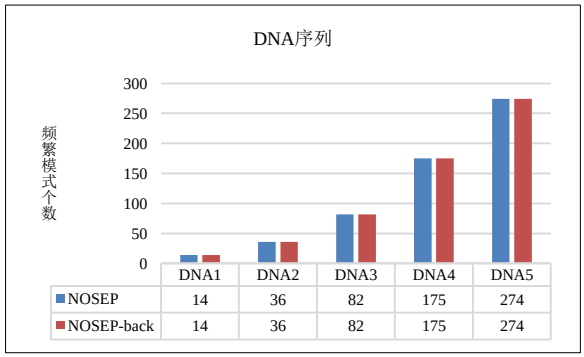

Fig.11 Comparison of the results on DNA

图 11 两种挖掘算法在 DNA 序列上候选模式个数对比

Table 8 Comparison of running time on DNA1-DNA5(ms)

表 8 两种挖掘算法在序列 DNA1-DNA5 上运行时间比较(ms)

运行时间(ms)	DNA1	DNA2	DNA3	DNA4	DNA5
NOSEP	920	3962	13525	39140	78749
NOSEP-back	**889**	**3759**	**12932**	**38221**	**77329**

5 结论

本文研究了具有多个可变间隙约束的无重叠严格模式匹配问题,它是一种允许序列中的任意位置的字符重复使用,但不允许同一字符在相同位置多次使用、模式串中包含多个可变间隙约束且具有长度约束的严格精确模式匹配.本文针对该问题当前求解算法存在的不足,提出了基于网树结构及回溯策略的求解算法 NetBack 算法,有效的降低了时间复杂度,提高了算法的效率,为提高无重叠序列模式挖掘的效率提供了基础.本文理论证明了 NetBack 算法的完备性与正确性,并理论证明了该算法的空间复杂度和时间复杂度都为 $O(m*n*W)$,其中,m,n 和 W 分别为模式长度,序列长度及最大间隙.通过真实生物数据实验验证了 NetBack 算法的正确性与高效性.此外,本文指出除 NetBack 算法所采用的最左孩子路径外,还存在其他三种求解策略,分别为最左双亲路径、最右双亲路径和最右孩子路径,并通过理论证明和实验验证了这三种求解策略的可行性与正确性.

本文基于网树结构提出了 NetBack 算法,该算法较以往算法具有较低的时间复杂度,具有较高的求解效率.虽然采用网树结构可以直观易懂地表示所有出现,并解决无重叠条件严格模式匹配问题,但网树结构的创建过程较为复杂,这会增加算法的时间开销.特别是从定理 4 的证明可以看出,若不建立网树,则可以将算法的时间复杂度降低为 $O(m*n)$.因此,如何在不建立网树结构的情况下,有效地减少无重叠模式匹配及挖掘问题的求解时间值得进一步探索.同时,无重叠模式匹配在其他领域的应用也值得进一步研究.

References:

[1] Wu X, Zhu X, Wu G, et al. Data mining with big data. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(1): 97-107.

[2] Li C, Yang Q, Wang J, et al. Efficient mining of gap-constrained subsequences and its various applications. ACM Transactions on Knowledge Discovery from Data, 2012, 6(1): 2.

[3] Song W, Rong K. Mining high utility sequential patterns using maximal remaining utility. International Conference on Data Mining and Big Data. Springer, Cham, 2018. 466-477.

[4] Tan C D, Min F, Wang M, et al. Discovering patterns with weak-wildcard gaps. IEEE Access, 2016, 4(1): 4922-4932.

[5] Wu X, Zhu X, He Y, et al. PMBC: Pattern mining from biological sequences with wildcard constraints. Computers in Biology and Medicine, 2013, 43(5): 481-492.

[6] Peng H, Li J, Li B, et al. Fast multi-pattern matching algorithm on compressed network traffic. China Communications, 2016, 13(5): 141-150.

[7] Jiang X, Xu T, Dong X. Campus data analysis based on positive and negative sequential patterns. International Journal of Pattern Recognition and Artificial Intelligence, 2019, 33(5): 1959016.

[8] Dong X, Qiu P, Lv J, et al. Mining top-k useful negative sequential patterns via learning. IEEE Transactions on Neural Networks and Learning Systems, 2019, DOI: 10.1109/TNNLS.2018.2886199.

[9] Li F, Yao B, Tang M, et al. Spatial approximate string search. IEEE Transactions on Knowledge & Data Engineering, 2013, 25(6): 1394-1409.

[10] Wu M, Wu X. On big wisdom. Knowledge and Information Systems, 2019, 58(1): 1-8.

[11] 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.

[12] Lewenstein M. Indexing with gaps. International Symposium on String Processing and Information Retrieval. Springer, Berlin, Heidelberg, 2011. 135-143.

[13] Navarro G, Raffinot M. Fast and simple character classes and bounded gaps pattern matching, with applications to protein searching. Journal of Computational Biology, 2003, 10(6): 903-923.

[14] Drory Retwitzer M, Polishchuk M, Churkin E, et al. RNAPattMatch: a web server for RNA sequence/structure motif detection based on pattern matching with flexible gaps. Nucleic Acids Research, 2015, 43(W1): W507-W512.

[15] Cole R, Gottlieb L A, Lewenstein M. Dictionary matching and indexing with errors and don't cares. Proceedings of 36th annual ACM symposium on Theory of computing. ACM, 2004. 91-100.

[16] Song W, Liu Y, Li J. Mining high utility itemsets by dynamically pruning the tree structure. Applied Intelligence, 2014, 40(1): 29–43.

[17] Zhou Z, Pi D. Minimal rare pattern mining method for satellite telemetry data streams. Chinese Journal of Computers, 2019, 42(6): 1351-1366. (in Chinese). http://cjc.ict.ac.cn/qwjs/view.asp?id=5193

[18] Zhang M, Kao B, Cheung D, et al. Mining periodic patterns with gap requirement from sequences. ACM Transactions on Knowledge Discovery from Data, 2007, 1(2): 7.

[19] Wu Y, Wang Y, Liu J, et al. Mining distinguishing subsequence patterns with nonoverlapping condition. Cluster Computing, 2019, 22(3): 5905-5917.

[20] Wang L, Wang S, Liu S, et al. An algorithm of mining sequential pattern with wildcards based on index-tree. Chinese Journal of Computers, 2019, 42(3): 554-565. (in Chinese). http://cjc.ict.ac.cn/qwjs/view.asp?id=5144

[21] Wu Y, Zhu C, Li Y, et al. NetNCSP: Nonoverlapping closed sequential pattern mining. Knowledge-Based Systems, 2020, 196: 105812.

[22] Min F, Zhang Z, Zhai W, et al. Frequent pattern discovery with tri-partition alphabets. Information Sciences, 2020, 507: 715-732.

[23] Shi Q, Shan J, Yan W, et al. NetNPG: Nonoverlapping pattern matching with general gap constraints. Applied Intelligence. DOI: 10.1007/s10489-019-01616-z.

[24] Ding B, Lo D, Han J, et al. Efficient mining of closed repetitive gapped subsequences from a sequence database. IEEE 25th International Conference on Data Engineering. IEEE, Shanghai, 2009. 1024-1035.

[25] Wu Y, Shen C, Jiang H, et al. Strict pattern matching under non-overlapping condition. Science China Information Sciences, 2017, 60 (1): 012101.

[26] Wu Y, Tong Y, Zhu X, et al. NOSEP: Nonoverlapping sequence pattern mining with gap constraints. IEEE Transactions on Cybernetics, 2018, 48(10): 2809-2822.

[27] Wu Y, Liu Y, Guo L, et al. Subnettrees for strict pattern matching with general gaps and length constraints. Ruan Jian Xue Bao/Journal of Software, 2013, 24(5): 915-932 (in Chinese). http://www.jos.org.cn/1000-9825/4381.htm

[28] Chai X, Jia X, Wu Y, et al. Strict pattern matching with general gaps and one-off condition. Ruan Jian Xue Bao/Journal of Software, 2015, 26(5): 1096-1112 (in Chinese). http://www.jos.org.cn/1000-9825/4707.htm

[29] Chen G, Wu X, Zhu X, et al. Efficient string matching with wildcards and length constraints. Knowledge and Information Systems, 2006, 10(4): 399-419.

[30] Fredriksson K, Grabowski S. Efficient algorithms for pattern matching with general gaps, character classes, and transposition invariance. Information Retrieval, 2008, 11(4): 335-357.

[31] Han C, Duan L, Lin Z, et al. Discovering relationship patterns among associated temporal event sequences. International Conference on Database Systems for Advanced Applications. Springer, Cham, 2019. 107-123.

[32] Wu Y, Fu S, Jiang H, et al. Strict approximate pattern matching with general gaps. Applied Intelligence, 2015, 42(3): 566-580.

[33] Yang H, Duan L, Hu B, et al. Mining top-k distinguishing sequential patterns with gap constraint. Ruan Jian Xue Bao/Journal of Software, 2015,26(11): 2994-3009 (in Chinese). http://www.jos.org.cn/1000-9825/4906.htm

[34] Wu Y, Li S, Liu J, et al. NETASPNO: Approximate strict pattern matching under nonoverlapping condition. IEEE Access, 2018, 6(1): 24350-24361.

[35] Xie F, Wu X, Zhu X. Efficient sequential pattern mining with wildcards for keyphrase extraction. Knowledge-Based Systems, 2017 115: 27-39.

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

[37] Ferreira P G, Azevedo P J. Protein sequence pattern mining with constraints. European Conference on Principles of Data Mining and Knowledge Discovery. Springer, Berlin, Heidelberg, 2005. 96-107.

[38] Bille P, Gørtz IL, Vildhøj HW. String matching with variable length gaps. Theoretical Computer Science, 2012, 443: 25-34.

[39] Liu H, Liu Z, Huang H, et al. Sequential pattern matching with general gap and one-off condition. Ruan Jian Xue Bao/Journal of Software, 2018, 29(2): 363-382 (in Chinese). http://www.jos.org.cn/1000-9825/5255.htm

[40] Wu Y, Tang Z, Jiang H, et al. Approximate pattern matching with gap constraints. Journal of Information Science, 2016, 42(5): 639-658.

附中文参考文献:

[17] 周忠玉,皮德常.面向卫星遥测数据流的最小稀有模式挖掘方法. 计算机学报. 2019, 42(6): 1351-1366.

[20] 王乐,王水,刘胜蓝,等.基于索引树的带通配符序列模式挖掘算法. 计算机学报. 2019, 42(3): 554-565.

[27] 武优西,刘亚伟,郭磊,等.子网树求解一般间隙和长度约束严格模式匹配. 软件学报, 2013, 24(5): 915-932.

[28] 柴欣,贾晓菲,武优西,等.一般间隙及一次性条件的严格模式匹配. 软件学报, 2015, 26(5): 1096-1112.

[33] 杨皓,段磊,胡斌,等.带间隔约束的 Top-k 对比序列模式挖掘. 软件学报, 2015, 26(11): 2994-3009.

[39] 刘慧婷,刘志中,黄厚柱,等.一般间隙与 One-Off 条件的序列模式匹配. 软件学报, 2018, 29(2): 363-382.