周期性一般间隙约束的序列模式挖掘

武优西[1)] 周 坤[1)] 刘靖宇[1)] 江 贺[2)] 吴信东[3),4)]

[1)](河北工业大学计算机科学与软件学院 天津 300401)

[2)](大连理工大学软件学院 辽宁 大连 116621)

[3)](合肥工业大学计算机科学与信息工程学院 合肥 230009)

[4)](佛蒙特大学计算机系 佛蒙特州 伯灵顿 美国 05405)

摘 要 序列模式挖掘是从给定序列中发现出现频率高的模式,目前已在诸多领域被广泛应用.假定子模式 p_i 和 $p_j(i<j)$可以分别匹配事件 A 和事件 B,传统的序列模式挖掘方法能够对事件 B 在事件 A 之后的序列进行检测,而不能对事件 B 发生在事件 A 之前的序列进行识别.为了解决此问题,文中提出了周期性一般间隙约束的序列模式挖掘问题,该问题具有如下 5 个特点:间隙约束的最小值可为负值的一般间隙约束;每个间隙约束都相同的周期性模式;在支持数统计方面无特殊约束,即允许序列中事件多次使用;该挖掘问题满足 Apriori 性质;挖掘支持率大于给定的频繁度阈值的频繁模式.为了进行有效地挖掘,采用深度优先的方式建立模式树.文中采用模式匹配技术,在一遍扫描序列数据库的情况下,建立其所有超模式的不完整网树森林(不完整网树是网树的最后一层结点,可以存储在一个数组中,可以有效地表示一个模式在一个序列中的支持数),并对这些超模式的支持率进行有效地计算,进而挖掘出所有频繁模式,有效地提高了序列模式挖掘速度.实验结果验证了文中算法的可行性和有效性.

关键词 序列模式挖掘;一般间隙;频繁模式;模式匹配;Apriori 性质

中图法分类号 **DOI 号** 10.11897/SP.J.1016.2015.00000

Mining Sequential Patterns with Periodic General Gap Constraints

WU You-Xi[1)] ZHOU Kun[1)] LIU Jing-Yu[1)] JIANG He[2)] WU Xin-Dong[3),4)]

[1)](*School of Computer Science and Engineering, Hebei University of Technology, Tianjin* 300401)

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

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

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

Abstract Sequential pattern mining is to discover the frequent patterns in the sequences and plays an essential role in many critical data mining tasks with broad applications. Given sub-patterns p_i and $p_j(i<j)$ can match events A and B respectively, traditional pattern mining methods can detect the sequences in which event B is after event A, but fail to find the sequences with event B occurring before event A. To tackle this challenge, in this paper, we propose sequential pattern mining with periodic general gap constraints with five characteristics as follows. The minimal gap constraint, namely general gap constraint, can be a negative value. All gap constraints of the pattern are the same. Any event in the sequence can be used more than once in different supports. The problem satisfies the Apriori property under the new definition of offset sequences. If the support ratio of a pattern is greater than the given threshold, the pattern is a frequent pattern. To solve

收稿日期:2014- - ;最终修改稿收到日期:2015-06-23.本课题得到国家自然科学基金(61229301)、教育部创新团队项目(IRT13059)、河北省自然科学基金(F2013202138)、河北省教育厅重点项目(ZD2014009)和河北省教育厅青年基金(QN2014192)资助.**武优西**,男,1974 年生,博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为智能计算和数据挖掘. E-mail: wuc@scse.hebut.edu.cn. **周 坤**,女,1989 年生,硕士研究生,主要研究方向为序列模式挖掘. **刘靖宇**,男,1976 年生,博士,讲师,主要研究方向为网络存储. **江 贺**,男,1980 年生,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为软件工程、智能计算、数据挖掘. **吴信东**,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、万维网信息探索.

the problem effectively, a depth-first search strategy is used to create the pattern tree. An Incomplete Nettree structure, the last layer of a Nettree which can be stored in an array, can represent the support of a pattern in the sequence. Pattern matching method is used to create the Incomplete Nettrees for all super patterns of a frequent pattern by one way scan over the sequence database. Therefore, we can calculate the support ratios of these super patterns and find the frequent ones. Experimental results validate the feasibility and effectiveness of the proposed algorithms.

Keywords sequential pattern mining; general gap; frequent pattern; pattern matching; Apriori property

1 引 言

Agrawal 和 Srikant[1] 最早提出了序列模式挖掘概念,其核心工作是在序列中找到支持度大于指定阈值的高频项,即频繁模式. 目前序列模式挖掘已经在多种领域中具有重要应用,如 Shie 等人[2] 对移动商务环境的用户行为进行挖掘. 典型的序列模式挖掘算法有 AprioriAll 算法[1] 和 PrefixSpan 算法[3] 等.

由于挖掘领域和挖掘的目的不同,人们也设计了多种挖掘算法,如 Lo 等人[4] 在序列数据库上采用双向挖掘技术对非冗余的重复性规则进行挖掘. 由于在序列模式挖掘过程中各个事件之间并无间隔约束,因而存在某些事件的间隔较大的现象,不能满足用户需求. 近年来具有间隙约束的序列模式挖掘算法快速发展,尽管在增加间隙约束后,挖掘难度更大,但是由于可以灵活方便地满足用户的需求,所以这类研究在诸多领域内得到快速发展,如 Zhang 等人[5] 提出了一个有效算法 MPP-best 可以对 DNA 序列进行有效地挖掘; Ding 等人[6] 提出了 GSgrow 算法和 CloGSgrow 算法可以对序列数据库进行有效地挖掘; Li 等人[7] 建立了 GAP-BIDE 算法和 GAP-Connect 算法,其中 GAP-BIDE 可以挖掘封闭模式的全集,而且可以应用 GAP-BIDE 对分类或聚类的序列数据进行有效地特征抽取. 这一类研究的共同点就是挖掘的模式具有间隙约束,其模式 P 可以描述为 $p_0 g[M_0, N_0] p_1 g[M_1, N_1] p_2 \cdots g[M_{l-1}, N_{l-1}] p_{l-1}$ 的形式. 若对这类序列模式挖掘研究进行细分,有的需要预先给定间隙约束进行挖掘[5],有的则在间隙未知的情况下进行挖掘,如 He 等人[8] 开展的研究就是在间隙未知的情况下采用一次性条件限制实现序列模式挖掘. 此外,有些序列模式挖掘要求各个间隙都一致(这样的间隙约束被称为周期间隙约束[9]),而有些研究则对此不做要求. 在出现的计算方式方面,则有无特殊约束、无重叠约束和一次性约束共 3 种严格形式以及一种仅考虑首尾匹配(或仅考虑尾部匹配)的宽松形式.

目前无论何种形式的序列模式挖掘都是在有序序列上进行挖掘,因而挖掘的结果必然都是有序的,能够有效地找到具有一定因果关系的频繁模式. 由于不同消费者可能具有相同的购买模式,因此序列模式挖掘的一个典型例子就是"数月以前购买了单反相机的客户很可能在一个月内订购新的单反相机配件,如镜头或电池等". Yen 等人[10] 开展了具有间隙约束的序列模式挖掘,这样的挖掘方法可以使得经销商相对准确地判定出消费者再次购买某种商品的时间范围,所以具有间隙约束的序列模式挖掘具有重要的应用价值. 但是无论是序列模式挖掘还是建立在其基础上的具有间隙约束的序列模式挖掘(该类研究中,间隙约束的最小值大于或等于 0,即非负间隙)以及文献[11]的后续相关项挖掘,都只能针对后一事件发生在前一事件之后的理想情形进行挖掘,在客观上就严格地约定了消费者的购买顺序,对于那些因为某些因素产生的顺序颠倒现象不能识别. 例如在国内,通常家庭装修过后用户具有家具和家电等购买行为,如果采用序列模式挖掘能够识别的购买模式为先购买瓷砖,再购买地板,之后购买家具,最后购买家电,即购买模式〈瓷砖,地板,家具,家电〉. 但是在实际生活中,可能会受到促销等因素影响,使得实际购买行为序列可能是先购买地板,再购买瓷砖,之后购买家电,最后购买家具. 显然这样的购买序列不能为购买模式〈瓷砖,地板,家具,家电〉所识别. 此外,在网络入侵检测以及 Web 访问模式等经典序列模式挖掘应用领域均存在与购买模式相似的问题. 如果研究一般间隙约束下的序列模式挖掘,那么这些顺序颠倒的序列便可以被识别.

尽管关联规则挖掘不考虑事务间的顺序问题,

但其缺点在于也不考虑事务间内在间隔问题，因此在购买模式挖掘中，经销商难于判定出消费者何时再次购买何种商品，因而也会产生一定的局限性. 综上，更为有效的挖掘方式是，允许间隙为负的一般间隙序列模式挖掘，这样就既允许后一事件在前一事件之前发生，而且可以考虑事务间的间隔.

表 1 进一步对比说明了关联规则挖掘、序列模式挖掘、间隙约束的序列模式挖掘以及本文研究的一般间隙约束的序列模式挖掘的特点.

表 1　不同类型挖掘的特点

挖掘类型	特点
关联规则挖掘	解决事务之间的内在联系，但是该挖掘不考虑事务间的顺序.
序列模式挖掘	注重事务间的顺序，然而该挖掘未对事务之间的间隔进行考虑，即不存在间隙约束.
间隙约束的序列模式挖掘	注重事务间顺序的同时，在事务间存在非负间隙约束，正是非负间隙的限定，导致严格地规定了事务间的顺序，若有违反，不能识别.
一般间隙约束的序列模式挖掘	对事务发生序列进行分析，通过采用允许间隙值为负的一般间隙约束对事务间的间隔进行考虑，这样可以对一定范围内事务顺序颠倒的序列进行识别.

为了解决文献[5]等周期性间隙约束序列模式挖掘中，间隙为非负的问题，本文研究了周期性一般间隙约束的序列模式挖掘问题.

本文第 2 节总结相关研究工作；第 3 节对一般间隙约束的序列模式挖掘问题进行形式化定义，并证明其满足 Apriori 性质；第 4 节对一个相关求解算法的工作原理做简要地介绍；第 5 节提出求解算法，通过实例介绍本文算法的工作原理，并对该算法的时间复杂度和空间复杂度进行分析；第 6 节通过对比性实验，验证本文方法的有效性；第 7 节得出本文结论.

2　相关工作

目前对于序列模式挖掘问题可从以下几个方面进行划分：挖掘的序列类型、是否具有间隙约束及是否指定间隙、间隙是否可以为负的一般间隙、出现的统计方法、是否支持 Apriori 性质和输出频繁模式类型.

针对挖掘的序列类型可以分为只能在单序列上进行挖掘的单序列模式挖掘和在序列数据库上进行挖掘的多序列模式挖掘. 能够在多序列上进行挖掘的算法也可以在单序列上进行有效地挖掘，反之则不一定.

对是否具有间隙约束以及是否指定间隙可分为不具有间隙约束、具有不指定间隙约束和具有指定间隙约束 3 种挖掘类型. 间隙就是指两个事件的间隔(或两个字符串中间可以通配的字符数量)，可定义为 $g[M,N]$的形式. 根据 M 和 N 的取值可以分为非负间隙和一般间隙，非负间隙下 M 和 N 均为大于等于 0 的整数，而一般间隙下，两者可以为负整数. 易知如果两个事件 A 和 B 的最小间隙 M 为正，则 A 一定在前，B 一定在后；若最小间隙 M 为负，则 B 可以在 A 的前面. 有些挖掘方法是在序列模式挖掘之前给定间隙约束，有些则是挖掘之后形成间隙约束. 周期间隙序列模式挖掘是一种特殊的具有指定间隙约束的序列模式挖掘，它要求模式每两个字符之间的间隙是相同的，如 a[1,2]b[1,2]c 是周期间隙序列模式，而 a[0,2]b[1,2]c 则不是.

在支持数统计方面，具有严格形式与宽松形式的区分，在严格形式中又有 3 种情况，下面举例说明一下这 4 种方式的不同.

例 1.　给定序列 $S=s_0\,s_1\,s_2\,s_3\,s_4=$atata 和模式 $P=p_0\,p_1\,p_2=$ata.

方式 1：无特殊约束. 这种方式的最大特点是允许任何一个字符被多次重复使用. 由于序列中 s_0，s_1 和 s_2 分别为字符 a，t 和 a，这与 p_0，p_1 和 p_2 分别相同，因此〈0,1,2〉是一个合法的出现. 除此之外，还有另外 3 个合法的出现分别是〈0,1,4〉，〈0,3,4〉和〈2,3,4〉，因而模式 P 在序列 S 的支持数为 4. 文献[13]的序列模式挖掘研究属于此类研究.

方式 2：无重叠约束. 这种方式的最大特点在于不能任意重复使用任何字符，只能在出现的不同位置重复使用字符. 例如〈0,1,2〉和〈0,3,4〉就是重叠出现，因为 s_0 两次与 p_0 进行匹配. 但是出现〈0,1,2〉和〈2,3,4〉属于无重叠出现，尽管 s_2 被两次使用，但是第 1 次使用 s_2 时是与 p_2 进行匹配，而第 2 次使用时是与 p_0 进行匹配. 例 1 中无重叠出现最多的一组出现是〈0,1,2〉和〈2,3,4〉，因此 P 在 S 中的支持数为 2. 文献[6]的序列模式挖掘研究属于此类研究.

方式 3：一次性约束. 这种方式的最大特点在于任何情况下都不能重复使用任何字符，因此在一次性约束下〈0,1,2〉和〈2,3,4〉是不可以同时存在的，因为 s_2 被两次使用，在此约束下 P 在 S 中支持数为 1. 文献[14]的序列模式挖掘研究属于此类研究.

方式 4：首尾匹配. 这种方式仅仅考虑首尾形成

出现的位置,对中间匹配的位置则不做考虑,更为简洁的方式是仅仅考虑最后字符匹配的位置.因此例1中,首尾匹配下,有〈0,2〉,〈0,4〉和〈2,4〉共3个出现(若仅考虑尾部匹配位置,就只有2和4两个出现).文献[15]的序列模式挖掘研究属于此类研究.

由于前3种方式都对模式串中任意字符匹配的位置进行衡量,因此称为严格形式.而最后一种则对形成匹配的中间过程不进行衡量,因此称为宽松形式.尽管方式1的支持数为指数形式,但是易于在多项式时间内构造完备性求解算法.而方式2和方式3的解都属于方式1的子集,都要面临一个问题,即尽可能多的求出一个模式在序列中的支持数,而这属于优化问题,需探索如何有效地建立完备性求解算法[6,16].因此基于方式2和方式3的序列模式挖掘结果都是不完备的.

由于在长度为$|\Sigma|$的字符集合上挖掘长度为l的模式,会有$|\Sigma|^l$种可能性,依次探寻每种模式是否为频繁模式是不可能的,为此需要建立有效方法以便有效地去除不可能成为频繁模式的模式.采用最多的方法就是Apriori性质,即子模式不频繁,其所有超模式均不频繁.但在某些情况下的序列模式挖掘并不满足Apriori性质,例如当允许任意位置字符被多次使用的情况下便不满足,为此Zhang等人[5]建立了Apriori-like性质以实现减少搜索范围的目的.

在输出频繁模式方面:如果输出全部的频繁模式,存在模式的数量过多的问题.为了有效地解决这个问题,可以采用闭合模式的方式进行无损输出,即如果子模式的支持度与其超模式的支持度一致,则仅仅保留超模式忽略子模式.如文献[6]就是采用闭合模式方式减少频繁模式的数量.此外,还可以采用Top k的方法,即输出最频繁的k种模式或各个长度模式下的k个模式[13].表2对相关研究进行了比较.

表2 相关研究比较

文献	序列类型	间隙个数及范围	间隙约束	支持度统计方法	满足Apriori性质与否	输出频繁模式
Agrawal and Stikant[1]	序列数据库	无	—	子序列个数	是	全部/闭合
El-Ramly等人[15]	序列数据库	多个	非负间隙	宽松	否	全部
Zhang等人[5]	单序列	多个且指定范围	非负间隙	严格并无特殊约束	否	全部
Min等人[12]	单序列	多个且指定范围	非负间隙	严格并无特殊约束	是	全部
Li等人[7]	序列数据库	多个且指定范围	非负间隙	严格并无特殊约束	是	全部/闭合
Ding等人[6]	序列数据库	多个且不指定范围	非负间隙	严格并无重叠约束	是	全部/闭合
吴信东等人[14]	单序列	多个且指定范围	非负间隙	严格并一次性约束	是	全部
本文	序列数据库	多个且指定范围	一般间隙	严格并无特殊约束	是	全部

通过表2不难发现,本文特点在于可以对一般间隙的模式进行挖掘。这样的挖掘在实际中具有重要意义.例如文献[10]的研究就约定了事件发生的顺序,但是从背景分析可知,如果不限定事件的发生顺序,在实际应用中将具有更大的意义.

3 问题定义

定义1. 序列S是有序的事件列表,标记为$S=\langle e_0,e_1,\cdots,e_{L-1}\rangle$,其中$e_i$是一个事件并且$e_i\in\varepsilon$,$\varepsilon$为不同事件的集合,$0\leq i\leq L-1$,为了方便,序列$S$也可写作$S=e_0e_1\cdots e_{L-1}$,一个序列中事件的个数叫做序列的长度,长度为$L$的序列也称为$L$-序列.

定义2. 输入序列数据库SDB是二元组(sid,S)的集合,sid是序列标号,S是一个输入序列.SDB中二元组的个数叫做SDB的规模,记为$|SDB|$,为了方便起见,SDB也可以标记为$\{S_0,S_1,\cdots,S_{|SDB|-1}\}$.表3给出了一个序列数据库的实例.为了下文中引用方便,我们令$|SDB|=n$.

表3 序列数据库实例

Sid	Input sequence
0	〈A,T,G,G,A,G,A,G〉
1	〈A,G,G,A,A,G,G,C〉

定义3. 通配符(用一个小圆点"·"表示)是一个特殊的符号,它可以匹配事件集中任意一个事件.间隙是一个通配符序列,间隙的尺寸表示其中通配符的个数,例如,"……"的尺寸是6.用$g(N)$表示尺寸为N的间隙,用$g[M,N]$表示尺寸在$[M,N]$之间的间隙,区间$[M,N]$称为间隙约束.本文用W表示最大允许间隔,其计算方法为$W=N-M+1$.

定义4. 长度为l且周期间隙约束为$[M,N]$的模式P是l个事件的有序序列,标记为$P=p_0g[M,N]p_1g[M,N]\cdots p_{l-2}g[M,N]p_{l-1}$,或简写为$P[M,N]=p_0p_1\cdots p_{l-2}p_{l-1}$,对$0\leq j\leq l-1$,$p_j\in\varepsilon$.在一般间隙下有$M<0$且$N\geq 0$.

定义 5. 如果间隙约束模式 P 在序列 S 中的一个位置索引序列 $I=\langle i_0,\cdots,i_j,\cdots,i_{l-1}\rangle$ 满足如下约束条件：

$$S_{i_j}=P_j \tag{1}$$

$$i_{j-1}\neq i_j \tag{2}$$

$$\begin{cases} M\leq i_j-i_{j-1}-1\leq N, & i_{j-1}<i_j \\ M\leq i_j-i_{j-1}\leq N, & i_{j-1}>i_j \end{cases} \tag{3}$$

其中，$0\leq j\leq l-1$ 且 $0\leq i_j\leq L-1$，称 I 是 P 在 S 中的一个出现，也称 P 匹配 S. P 在 S 中的出现个数定义为 $sup(P,S)$. P 在 SDB 中的支持数为 $sup(P,SDB)=\sum_{i=0}^{n-1}sup(P,S_i)$，也称为 P 的绝对支持度.

假设模式 P 在序列 S 中的出现为 $\langle i_0,i_1,\cdots,i_{l-1}\rangle$，在非负间隙下，对任意 $0\leq j\leq l-2$，都有 $i_j<i_{j+1}$，但是在一般间隙下，这个条件是不成立的. 例如 S=ATCCG 在正间隙下，模式 CT 的支持数为 0，但是在一般间隙 $g[M,N]=[-1,1]$ 下，CT 的支持数为 1(对应的出现为〈2,1〉，而〈3,1〉不是 CT 的出现，因为位置为 3 的 C 和 A 的间隔为 -2).

定义 6. 如果 $P_a=a_0a_1\cdots a_{l-1}$ 和 $P_b=b_0b_1\cdots b_{l-1}b_l$ 是两个长度分别为 l 和 $l+1$ 的模式，又知对于任意的 $0\leq i<l$ 有 $a_i=b_i$，则称 P_a 是 P_b 的子模式，P_b 是 P_a 的超模式.

定义 7. 如果序列 S 的长度为 L，模式 P 的长度为 l，如果一个索引序列 $D=\langle d_0,d_1,\cdots d_{l-1}\rangle$ 满足

$$0\leq d_0\leq L-1 \tag{4}$$

$$d_{j-1}\neq d_j \tag{5}$$

$$\begin{cases} M\leq d_j-d_{j-1}-1\leq N, & M\geq 0 \\ M\leq d_j-d_{j-1}\leq N, & M<0 \end{cases} \tag{6}$$

D 就是间隙约束模式 P 在 S 中关于周期间隙 $[M,N]$ 的一个偏移序列，P 在 S 中偏移序列的个数定义为 $ofs(P,S)$，其表示的含义是 P 在 S 中理论上能够产生出现的最大上界. P 在 SDB 中偏移序列的个数定义为 $ofs(P,SDB)=\sum_{i=0}^{n-1}ofs(P,S_i)$.

Zhang 等人在文献[5]中提出了一种求解偏移序列个数的方法，但是其算法过于复杂，而且不满足 Apriori 性质. 为此 Min 等人[12]对文献[5]的偏移序列的定义进行了修改. 在 Zhang 等人的论文中，模式 P 的偏移序列为 $\langle c_0,c_1,\cdots,c_{l-1}\rangle$，对所有 $j(0\leq j\leq l-1)$ 都满足 $0\leq c_j\leq L-1$ 且 $M\leq c_{j+1}-c_j-1\leq N$. 在文献[12]中把 $0\leq c_j\leq L-1$ 修改成了 $0\leq c_0\leq L-1$，而其他的 c_j 没有限制. 该定义就相当于在序列的尾部添加了虚假的字符，该方法使得偏移序列的求解变的十分简单，而且满足 Apriori 性质. 本文就是借鉴的这种定义方式.

定义 8. 间隙约束模式 P 的相对支持度(也称为支持率)用 $r(P,SDB)$ 来表示，其计算方法为绝对支持度和偏移序列的比值，即 $sup(P,SDB)/ofs(P,SDB)$. 如果 $r(P,SDB)$ 不小于一个给定的阈值 ρ，那么 P 就是一个频繁模式，否则就是非频繁模式.

引理 1. 令 P' 是仅比 P 少一个字符的子模式，那么 P 的偏移序列数量为 P' 的偏移序列数量的 W 倍，这里 W 是最大允许间隔.

证明. 由于在定义 7 偏移序列的定义中，仅仅约束了 d_0 的大小，而对其后的 d_i 值(这里 $0<i<l$)均不做约束，因此 d_i 可以为超过 $L-1$ 的任意值，只要其满足间隙约束即可. 因此如果 P' 的一个偏移序列为 $\langle d_0,d_1,\cdots,d_{l-2}\rangle$，那么 $\langle d_0,d_1,\cdots,d_{l-2},d_{l-1}\rangle$ 也是 P 的一个偏移序列，这里 d_{l-1} 为任意一个在 d_{l-2} 基础上满足间隙约束 $[M,N]$ 的值，这就决定了 d_{l-1} 有 W 中可能值. 因此对于任意一个 P' 的偏移序列，都将对应 W 个可能的 P 的偏移序列. 证毕.

通过引理 1 易知，长度为 l 的模式 P 在长度为 L 的序列 S 上，偏移序列个数为 $ofs(P,S)=LW^{l-1}$.

定理 1. 本文问题在单序列下满足 Apriori 性质.

证明. 由于 P' 的每个匹配最多对应 W 个 P 的匹配，因此 $sup(P,S)\leq sup(P',S)W$. 因而由引理 1 可知，$r(P,S)=\dfrac{sup(P,S)}{ofs(P,S)}\leq\dfrac{sup(P',S)}{ofs(P',S)}=r(P',S)$. 证毕.

定理 2. 文本问题在序列数据库上仍然满足 Apriori 性质.

证明. 给定规模为 n 的序列数据库 SDB，S_i 的长度为 $L_i(0\leq i\leq n-1)$，若模式 $P=p_0p_1\cdots p_{l-1}$，则子模式 $P'=p_0p_1\cdots p_{l-2}$，其偏移序列存在如下关系：

$$\begin{aligned} ofs(P,SDB)&=\sum_{i=0}^{n-1}ofs(P,S_i)\\ &=ofs(P,S_0)+ofs(P,S_1)+\cdots+ofs(P,S_{n-1})\\ &=L_0W^{l-1}+L_1W^{l-1}+\cdots+L_{n-1}W^{l-1}\\ &=(L_0+L_1+\cdots+L_{n-1})W^{l-1}. \end{aligned}$$

令 L 表示 SDB 中所有字符串长度的和，则有 $ofs(P,SDB)=L\times W^{l-1}$，同理可得 $ofs(P',SDB)=L\times W^{l-2}$，因此有 $ofs(P,SDB)=ofs(P',SDB)\times W$. 另外 P' 在每个序列中的每个匹配最多对应 P 的 W 个匹配，因此 $sup(P,SDB)\leq sup(P',S_0)W+$

$sup(P',S_1)W+\cdots+sup(P',S_{n-1})W=sup(P',SDB)W$，所以可得到 $r(P,SDB)=\frac{sup(P,SDB)}{ofs(P,SDB)}\leq\frac{sup(P',S)}{ofs(P',S)}=r(P',SDB)$. 证毕.

定义 9. 所有满足 Apriori 性质的频繁模式，都可以用一棵模式树来表示，其中模式树的树根为空，从树根到每个结点所形成的路径即可构成一棵模式树.

例 2. 假定一个 DNA 序列的频繁模式为{A，T，C，G，AA，AC，AG，TA，TC，CA，CC，GA，GT，GC，GG，ACA，ACC}，则其可以用图 1 所示的模式树表示.

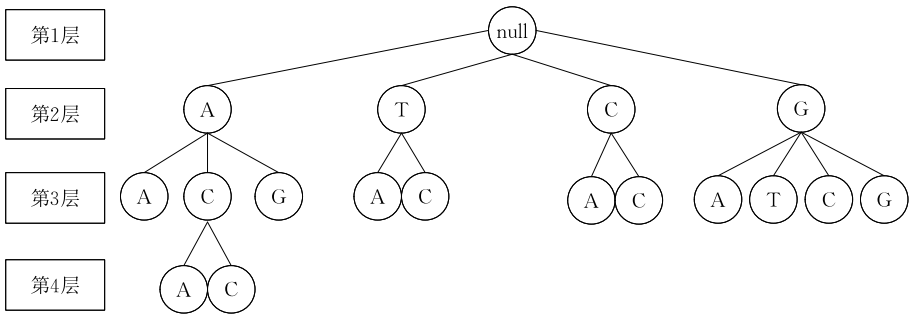


图 1 一棵模式树

从图 1 中可知，TG 不是一个频繁模式，依据 Apriori 性质，任意以 TG 为前缀的模式都不是频繁模式. 这样就可以在图 1 的模式树中，删除第 2 层 T 的子树中以 G 为树根的子树. 本文就是采用以深度优先方式建立模式树，发现所有频繁模式的.

4 相关算法介绍

文献[5]最早提出了在支持数计算方面无约束的序列模式挖掘，但本文选择同样具有此类性质的文献[18]的算法进行介绍，其原因在于一方面文献[18]的挖掘效率大大地优于文献[5]的挖掘效率，另外一个方面文献[18]提出的 MCPaS 算法也是采用模式匹配技术计算模式的支持数，这与本文方法有些相似，因而本文介绍一下 MCPaS 算法，并在第 6 节的实验中与增强型的 MCPaS 算法，即 MCPaS-PRO 算法进行对比.

MCPaS 算法可以对多序列进行有效挖掘，该文献是通过将多序列转换为单序列，并在单序列上采用名为 MGCS 的模式匹配方法计算各个模式的支持数，进而计算其支持率，以便判别其是否为频繁模式. 下面举例说明 MCPaS 算法的工作原理.

例 3. 求解模式 A[0,2]G 及其超模式 A[0,2]G[0,2]A 在序列 ATGGAGAG 中的支持数.

为了求解模式 A[0,2]G 的支持数，文献[18]首先采用模式匹配方法建立如图 2 所示的二维表. 通过二维表中最后一行数字和 1+1+1+2=5 来获得 A[0,2]G 模式的支持数.

	A	T	G	G	A	G	A	G
A	1	0	0	0	1	0	1	0
G	0	0	1	1	0	1	0	2

图 2 MCPaS 计算子模式的工作原理图

在依据子模式 A[0,2]G 计算超模式 A[0,2]G[0,2]A 的过程中，MCPaS 算法仅仅存储了子模式 A[0,2]G 的最后一行，并与 G[0,2]A 的映射头做点积，并将结果存储在可能生成 A 的位置上. 图 3 给出了 MCPaS 求解超模式的工作原理.

	A	T	G	G	A	G	A	G
AG	0	0	1	1	0	1	0	2
GA	0	0	1	2	0	1	0	0
AGA	0	0	0	0	2	0	2	0

图 3 MCPaS 计算超模式的工作原理图

从这个原理图中，可以看出 MCPaS 算法的两点不足：(1) MCPaS 算法无论是计算子模式还是超模式都是在稀疏二维表上进行操作，这样既增加了空间消耗，存储无意义的 0 数据，在计算的过程中，又需要对 0 数据进行扫描，增加了算法的计算时间；(2) 由于每扫描一次数据库只对一个超模式计算其支持数，当字符集大小为 $|\Sigma|$ 时，具有 $|\Sigma|$ 种超模式，需要扫描 $|\Sigma|$ 次数据库，这显然不如只扫描一次数据库就对 $|\Sigma|$ 种超模式进行计算的算法效率高.

5 求解算法

5.1 网 树

网树结构是为了解决各种间隙约束模式匹配问

题提出的与树结构类似的结构，不同之处在于，一棵网树可以有 n 个根结点，其中 $n\geq 1$；并且网树的非树根结点可以有多个双亲结点；以及任意一个结点到达其祖先结点的路径不唯一[19].

由于模式 P 在序列 S 中的模式匹配问题可以表示为一棵网树，并且其全部无特殊约束的支持数可以用网树的第 m 层结点的树根路径数之和来表示[19]. 又由于创建网树的下一层结点仅仅依靠上一层结点信息情况，因而文献[13]在计算长度为 $m+1$ 的模式的支持数时，仅仅使用第 m 层网树结点信息，去除了前 $m-1$ 层网树结点，并以此形成了不完整网树. 在具体实现上，可以将不完整网树存储在一个一维结构体数组中. 下面举例来说明如何构建一棵网树以及如何用不完整网树求解模式 P 在序列 S 中的支持数.

例 4.　给定模式 $P=\mathrm{A}[-1,2]\mathrm{G}$ 以及 $S_0=s_0s_1s_2s_3s_4s_5s_6s_7=\mathrm{ATGGAGAG}$，求解模式 P 在序列 S_0 中的支持数.

首先按照文献[19]的模式匹配方法计算模式 P 在 S_0 中的支持数，并将该计算过程转换化为一棵网树，如图 4 所示，图中白色圆圈和灰色圆圈内数字分别为结点名称及其树根路径数. 图 4 中第 2 层结点的树根路径数之和为 $1+2+2+2=7$，表明 P 在 S_0 中有 7 个出现，即出现〈0,2〉、〈0,3〉、〈4,3〉、〈4,5〉、〈4,7〉、〈6,5〉和〈6,7〉. 图 4 中黑框部分即为一棵不完整的网树.

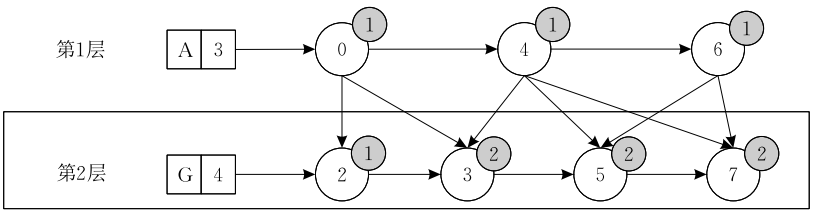


图 4　由模式 P 在 S_0 中出现所转化的一棵网树

5.2　MAPD-PRO 算法

本文的挖掘算法名为 MAPD-PRO，是因为该算法借鉴了文献[13]所提出的 MAPD 算法的思想，是对 MAPD 算法的改进增强算法，其与 MAPD 算法的差异在于：(1) MAPD 算法只能对非负间隙进行挖掘，而本文方法可以对一般间隙进行挖掘；(2) MAPD 算法满足 Apriori-like 性质，而 MAPD-PRO 满足 Apriori 性质；(3) MAPD 算法只能对单序列进行挖掘，而 MAPD-PRO 算法可以在序列数据库上进行挖掘.

下面介绍一下 MAPD-PRO 算法的工作原理：MAPD-PRO 算法是采用堆栈结构实现的以深度优先方式对模式树进行遍历的算法. 由于本问题满足 Apriori 性质，因此如果当前模式 P 是频繁的，则需要判定其所有可能的超模式是否为频繁模式. 由于模式 P 在单个序列中的匹配可以用一棵网树来表示，因而在序列数据库 SDB 上则需要用多棵网树所构成的森林来表示，其中第 i 棵网树表示模式 P 在子序列 S_i 上的匹配. MAPD-PRO 算法工作过程为：每次从栈中取出模式 P 及其不完整网树所构成的森林，再一次扫描序列数据库，对 P 的所有超模式建立其相应的不完整网树森林，并计算对应超模式的支持数，进而判定超模式是否频繁，如果是频繁，则模式串及其不完整网树所构成的森林入栈. 迭代这一过程，直到栈空为止. 下面我们用实例来说明不完整网树所构成的森林.

例 5.　使用例 3 的模式 $P=\mathrm{A}[-1,2]\mathrm{G}$ 在表 3 序列数据库上构建不完整网树所构成的森林.

在例 4 中，已经介绍了模式 P 在表 3 的序列 S_0 上构建不完整网树的过程，对于表 3 中 $S_1=s_0s_1s_2s_3s_4s_5s_6s_7=\mathrm{AGGAAGGC}$ 也可以构造如图 5 所示的不完整网树. 这样为了求解模式 P 在表 3 的序列数据库中的出现数，使用了由图 4 和图 5 两棵不完整网树所构成的森林.

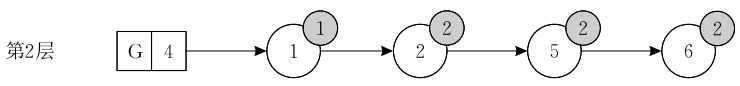


图 5　由模式 P 在 S_1 中出现所转化的一棵不完整网树

在 MAPD-PRO 算法具体实现中 *Cells* 是一个结构体数组，*Cells*[k]表示当前模式 P 的第 k 种超

模式 $P\&\Sigma_k$ 的模式串及其对应的多个不完整网树所构成的森林，其中 Σ_k 表示字符集中第 k 种字符. 在 $Cells[k]$ 结构体中有两个成员分量，其中 *pattern* 是用来存储模式串的；而 *Forest* 表示由不完整网树所构成的森林，$Forest[i]$ 表示第 i 棵不完整网树，其用来表示模式串 *pattern* 在子序列 S_i 中的对应不完整网树；$Forest[i].size$ 表示第 i 棵不完整网树中结点的数量；$Forest[i].node[j]$ 表示第 i 棵不完整网树中第 j 个结点；$Forest[i].node[j].name$ 表示第 i 棵不完整网树中第 j 个结点的名称；$Forest[i].node[j].NRP$ 表示第 i 棵不完整网树中第 j 个结点的树根路径数；$Forest[i].sum$ 表示第 i 棵不完整网树所有结点的树根路径数之和，即模式串 *pattern* 在子序列 S_i 中的支持数. C 表示所有出现频率不小于 ρ 的频繁模式集. 算法可以描述为

算法 1. MAPD-PRO.

输入：$SDB=S_0S_1\cdots S_{n-1}$，M，N，频繁度阈值 ρ，这里 $N>=0$，M 为小于 N 的整数，且可以小于 0

输出：所有支持率不小于 ρ 的频繁模式集合 C

1. L=所有序列长度总和；
2. 采用算法 2 为所有模式串长度为 1 的模式建立 *Cells*；
3. FOR ($k=0$; $k<|\Sigma|$; k++)
4. *sum*=*Cells*[k]的所有不完整网树的树根路径数总和；
5. IF *sum*/$L>=\rho$ THEN 将*Cells*[k]放入堆栈*meta*中；
6. END FOR
7. WHILE (!*meta.empty*())
8. *oldcell*=*meta.pop*()；
9. *length*=*oldcell.pattern* 的长度；
10. $C_{length}=C_{length}\cup oldcell.pattern$；
11. 采用算法 3 建立 *oldcell* 的所有超模式的 *Cells*；
12. FOR ($k=0$; $k<|\Sigma|$; k++)
13. *sum*=*Cells*[k]的所有不完整网树的树根路径数总和；
14. IF *sum*/($L\times W$^*length*)$>=\rho$ THEN 将 *Cells*[k]放入堆栈 *meta* 中；
15. END FOR
16. END WHILE
17. RETURN $C=\cup C_i$.

算法 2 的功能为初始化算法，即为字符集中每个字符建立一个具有模式名和多棵网树森林的树根结点. 其原理是对 S_i 中的每个字符 j，确定其是字符集中的第 t 个元素，然后为 $Cells[t]$ 的第 i 个网树建立一个名称为 j 的树根结点，并使其树根路径数为 1. 算法可以描述为

算法 2. Initialization.

输入：$SDB=S_0S_1\cdots S_{n-1}$，M，N

输出：*Cells*

1. FOR ($k=0$; $k<|\Sigma|$; k++)
2. *Cells*[k].*pattern*=Σ_k；
3. END FOR
4. FOR ($i=0$; $i<n$; i++)
5. FOR ($j=0$; $j<len(S_i)$; j++)
6. t 为 S_{i_j} 在字符集合中 Σ 的第 t 个字符；
7. *Cells*[t].*Forest*[i].*node*[*size*].*name*=j；
8. *Cells*[t].*Forest*[i].*node*[*size*].*NRP*=1；
9. *Cells*[t].*Forest*[i].*sum*++；
10. *Cells*[t].*Forest*[i].*size*++；
11. END FOR
12. END FOR
13. RETURN *Cells*.

算法 3 的功能是扫描一遍序列数据库，计算出当前模式的所有超模式的支持数. 其工作原理是对序列 S_i 范围内每个字符 x 依次进行处理，先确定 x 是字符集中的第 t 个字符，之后在 $Cells[t]$ 的第 i 棵不完整网树上建立结点 x，并在第 i 棵不完整网树 *oldcell*.*Forest*[i]的 *start* 开始位置寻找满足间隙约束的双亲结点，并依据双亲结点的树根路径数来更新当前结点的树根路径数. 其算法可以描述为

算法 3. SUPPORTS.

输入：$SDB=S_0S_1\cdots S_{n-1}$，M，N，*oldcell*

输出：*Cells*

1. P=*oldcell.pattern*；
2. $dt=1$；
3. IF ($M<0$) THEN $dt=0$；
4. FOR ($k=0$; $k<|\Sigma|$; k++)
5. *Cells*[k].*pattern*=$P\&\Sigma_k$；
6. END FOR
7. FOR ($i=0$; $i<n$; i++)
8. *start*=0；
9. FOR ($j=0$; $j<|S_i|$; j++)
10. t 为 S_{i_j} 在字符集合中 Σ 的第 t 个字符；
11. *first*=1；
12. FOR (x=*start*; $x<oldcell.Forest[i].size$; x++)
13. *nodename*=*oldcell*.*Forest*[i].*node*[x].*name*；
14. IF $j-nodename-1>N$ THEN
15. *start*++；
16. CONTINUE；
17. END IF
18. IF $j-nodename-dt<M$ THEN BREAK；
19. IF $j==nodename$ THEN CONTINUE；
20. IF *first*=1 THEN
21. *first*=0；

```
22.        Cells[t].Forest[i].node[size].name=x;
23.        Cells[t].Forest[i].node[size].NRP=
                   oldcell[i].node[j].NRP;
24.        Cells[t].Forest[i].size++;
25.      ELSE
26.        Cells[t].Forest[i].node[position].NRP+=
                   oldcell[i].node[j].NRP;
27.      END IF
28.      Cells[t].Forest[i].sum+=
                   oldcell [i].node[j].NRP;
29.    END FOR
30.  END FOR
31. END FOR
32. RETURN Cells.
```

5.3　运行实例

例 6.　在表 3 的序列数据库上挖掘支持率大于 0.05 的频繁模式.

这里在例 5 的基础上，继续说明 MAPD-PRO 算法是如何建立模式 P 的所有超模式的不完整网树森林，并计算各自的支持度和支持率，进而判定是否为频繁模式的.

P 在 S_0 和 S_1 中的支持度均为 7，而 S_0 和 S_1 两个序列的长度相同均为 8，因此偏移序列值均为 $8\times(2-(-1)+1)=32$，所以 $r(P,SDB)=(7+7)/(32+32)\approx 0.22>0.05$，因此模式 P 为频繁模式. 在字符集{A，C，G，T}下，以模式 P 为子模式的超模式有 4 种，分别为 AGA、AGC、AGG 和 AGT. 我们以模式 P 在 S_1 中的不完整网树为例说明如何对这 4 种超模式的支持数进行计算. 由于 S_1 中的第一个字符为 s_0='A'，因此为超模式 AGA 建立第一个结点 0，由于在[−1，2]的间隙作用下，该结点在 AG 模式的不完整网树中只有一个双亲结点，因此超模式 AGA 的第一个结点 0 的树根路径数为 1. S_1 中的第 2 个字符为 s_1='G'，因此为超模式 AGG 建立第一个结点 1，在间隙作用下，其树根路径数也为 1. 以此类推，为 S_1 中的第 3 个字符、第 4 个字符直到最后一个字符在对应的超模式上依次构建结点，并计算这些结点的树根路径数. 对序列进行一次扫描后，同时为 AGA 模式、AGC 模式、AGG 模式和 AGT 模式分别建立了不完整网树，结果如图 6 所示，并分别求得其各自的支持数分别为 1+3+5=9、4、2+1+4+2=9 和 0. 同理可以求得 AGA 模式、AGC 模式、AGG 模式和 AGT 模式这 4 种模式在 S 上各自的支持数分别为 11、0、8 和 1. 按照定义 8 可以计算 AGA 模式、AGC 模式、AGG 模式和 AGT 模式的支持率分别为 0.078、0.016、0.066 和 0.004，进而判定 AGA 模式和 AGG 模式为频繁模式，将这里两个超模式分别压入堆栈，再分别判定 AGA 模式的所有超模式以及 AGG 模式的所有超模式是否为频繁模式. 由于计算方法相同，不再详述.

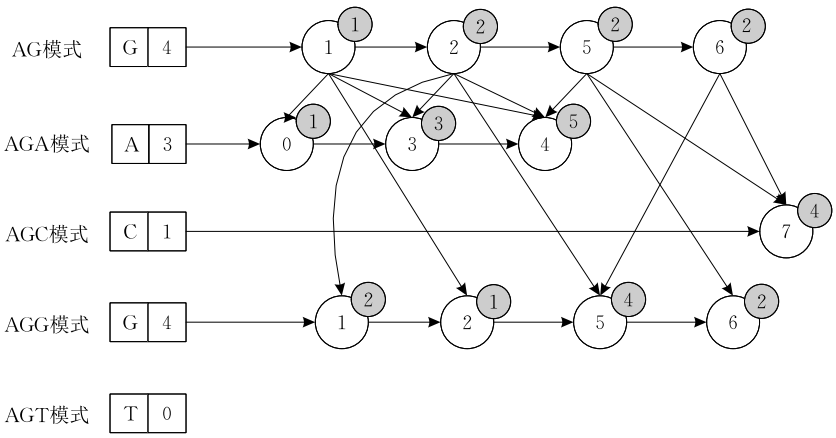


图 6　求解一个模式所有超模式的原理图

5.4　复杂度分析

这里首先分析算法的空间复杂度. 由于不完整网树中每个结点存储结点标签及其树根路径数，因此结点的空间复杂度为 $O(1)$. 由于序列数据库的总大小为 L，在每个字符使用概率近似相同的情况下，每种字符在序列中平均出现 $O(L/|\Sigma|)$ 次，假定每次出现都能形成一个结点，那么一棵不完整网树所构成的森林的空间复杂度为 $O(L/|\Sigma|)$. 假定堆栈的最大深度为 d，因此算法的存储堆栈的空间需要 $O(d\times L/|\Sigma|)$. 而计算频繁模式的所有超模式的支持数时，序列数据库的每个字符都可以转换为一个结点，因而其所需空间为 $O(L)$. 因而算法整体空间

复杂度为 $O(d\times L/|\Sigma|+L)$.

在时间复杂度方面,我们先分析算法 2 的时间复杂度.算法 2 的 1～3 行时间复杂度为 $O(|\Sigma|)$.尽管算法中 4～5 行是一个双重循环,但是其表示序列数据库的总体长度,因而其循环次数为 L 次.而 6～10 行语句均可在 $O(1)$时间内完成,又由于 $|\Sigma|\ll L$,因此算法 2 的时间复杂度为 $O(L)$.显然算法 3 的 7 行和 9 行的双重循环次数也是 L 次,而算法 12 行表示结点的双亲个数,在 $W=N-M+1$ 间隙的作用下,每个结点最多有 W 个双亲结点,但是平均为 $O(W/|\Sigma|)$,因此算法 3 的时间复杂度为 $O(L\times W/|\Sigma|)$.而算法 1 的第 7 行循环次数为频繁模式个数 $|C|$.因此 MAPD-PRO 时间复杂度为 $O(|C|\times L\times W/|\Sigma|)$.

6 实验结果及分析

6.1 实验环境

由于本文是建立在具有间隙约束的序列模式挖掘基础上的新挖掘方法且为了方便地获得大量可靠的序列,本文采用文献[5]和[20]中所使用的 DNA 和蛋白质数据集,DNA 数据集为人类基因序列 Homo Sapiens AX829174,在该数据集中我们选择不同长度的 DNA 片段做实验并把它们分成了多序列,蛋白质序列为 ASTRAL95_1_161 和 ASTRAL95_1_171,并截取不同的长度形成序列数不等的序列数据库. AX829174 可以从国家生物信息中心网站上下载,蛋白质序列可以从 http://gi.cebitec.uni-bielefeld.de/comet/force/indexOld.html 上下载.为了对比 MAPD-PRO 的挖掘性能,我们对文献[18]的 MCPaS 挖掘算法进行改造,在保留原始的支持数和超模式支持数计算方法不变的前提下,将其计算偏移序列的方法改为本文方法,并使其能够在多序列上进行周期性一般间隙的序列模式挖掘.改造后的算法名为 MCPaS-PRO,并使其按照 MAPD-PRO 的方式计算偏移序列,因此 MCPaS-PRO 也满足 Apriori 性质.此外,本文还对文献[12]的挖掘方法进行改造,使其可以支持一般间隙序列模式挖掘,改造后的算法称为 AMIN-PRO.本文涉及的算法 MCPaS-PRO、AMIN-PRO 和 MAPD-PRO 均使用 JAVA 语言实现,它们的源代码可以在 http://wuc.scse.hebut.edu.cn/mapdpro/mapdpro.htm 处下载.本文实验所用计算机的硬件环境为 Intel(R) Core(TM) i5 处理器、2.4 GHz 主频、2.0 GB 内存;软件环境为 32 位 Windows 7 操作系统,Eclipse Helios 3.6.2 编程环境.实验中蛋白质序列的支持率阈值为 0.0001,DNA 序列的支持率阈值均为 0.005.

为了测试算法在不同长度的字符集上的求解速度问题,我们对蛋白质数据集设定了 6 种情况,其中 SDB2 和 SDB3 分别是 SDB1 的前 338 和 169 个序列,SDB5 和 SDB6 分别是 SDB4 的前 400 和 200 个序列.这样设置序列目的在于使他们尽量保持大小为 3∶2∶1 的关系,以便对挖掘时间与序列长度之间的关系进行实验.同样,为 DNA 数据集设定了 7 种情况,具体数据集特征见表 4.从表 4 中可以看出 SDB10～SDB13 的总长度都近似相同,仅在拆分规则和拆分的序列个数上有所差异,目的在于测试挖掘时间是否受序列个数或子序列长度等因素的影响.

表 4 实验数据集特征

序列数据库名	数据集名称	序列类型	序列个数	总长度	备注
SDB1	ASTRAL95_1_161	蛋白质序列	507	91875	各个序列不等长
SDB2	ASTRAL95_1_161	蛋白质序列	338	62985	各个序列不等长
SDB3	ASTRAL95_1_161	蛋白质序列	169	32503	各个序列不等长
SDB4	ASTRAL95_1_171	蛋白质序列	590	94205	各个序列不等长
SDB5	ASTRAL95_1_171	蛋白质序列	400	73425	各个序列不等长
SDB6	ASTRAL95_1_171	蛋白质序列	200	37327	各个序列不等长
SDB7	AX829174	DNA 序列	10	1000	各个序列等长
SDB8	AX829174	DNA 序列	10	3000	各个序列等长
SDB9	AX829174	DNA 序列	10	6000	各个序列等长
SDB10	AX829174	DNA 序列	10	10010	各个序列等长
SDB11	AX829174	DNA 序列	10	10010	各个序列不等长(按照奇数序列长度相等,偶数序列长度相等,并且奇数序列长度是偶数序列长度的 2 倍)
SDB12	AX829174	DNA 序列	10	10011	各个序列不等长(按等差序列的方式递增)
SDB13	AX829174	DNA 序列	50	10011	各个序列等长

6.2　实验结果及分析

6.2.1　一般间隙与非负间隙挖掘结果比较

为了对比一般间隙$[-N,N]$与非负间隙$[0,N]$下挖掘结果的差异，我们分别在$N=1,2,3$时做了实验，限于篇幅，本文仅给出$N=2$时各个序列数据库挖掘的结果(见表5).

表5　挖掘结果对比

序列数据库名	[0,2]间隙挖掘结果(最长模式长度,{各个长度下频繁模式的数量}总频繁模式数量)	[−2,2]间隙挖掘结果(最长模式长度,{各个长度下频繁模式的数量}总频繁模式数量)
SDB1	4,{20,400,3888,7},4315	6,{20,400,3440,330,12,1},4203
SDB2	4,{20,400,3921,5},4346	6,{20,400,3480,324,12,1},4237
SDB3	4,{20,400,3899,7},4326	6,{20,399,3477,317,10,1 },4224
SDB4	4,{20,400,3917,3},4340	6,{20,400,3508,280,9,1},4218
SDB5	4,{20,400,3884,4},4308	6,{20,400,3515,289,9,1},4234
SDB6	4,{20,400,3919,6},4345	6,{20,400,3512,287,10,1},4230
SDB7	6,{4,16,64,42,2,1},129	11,{4,16,64,49,10,4,2,1,1,1,1},153
SDB8	4,{4,16,64,37},121	6,{4,16,64,49,5,2},140
SDB9	4,{4,16,64,41},125	6,{4,16,64,45,3,2},134
SDB10	5,{4,16,64,49,2},135	9,{4,16,64,45,5,2,2,1,1},140
SDB11	5,{4,16,64,48,2},134	9,{4,16,64,45,5,2,2,1,1},140
SDB12	5,{4,16,64,49,2},135	9,{4,16,64,45,5,2,2,1,1},140
SDB13	5,{4,16,64,45,2},131	9,{4,16,64,45,5,2,2,1,1},140

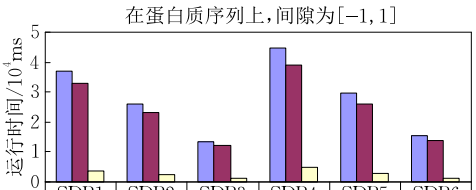
通过表5可以看出，一般间隙挖掘能够挖掘到更多频繁的长模式.这体现在两个方面：一方面是挖掘到的最长频繁模式的长度增加，例如在SDB1序列上[0,2]间隙下，最长频繁模式长度为4，而在[−2,2]间隙下最长频繁模式长度为6，而表5中其余12组序列均呈现了相同的特点，说明了挖掘到的最长频繁模式的长度增加；另外一方面是挖掘到的长频繁模式的数量增加，例如SDB1序列上[0,2]间隙下长度为4的频繁模式有7种，而在[−2,2]间隙下，长度为4的频繁模式有330种，通过对比发现，这7种模式均存在于这330种模式之中，此外，在一般间隙下，长度为5和6的频繁模式分别有12个和1个，而因此新增加的长模式数据量达到了336个，这种现象在DNA序列上依然存在，又如在SDB8序列库上，在非负间隙下长度为4的最长频繁模式为37个，而在一般间隙下，不但有长度为4的频繁模式，而且还有长度为5和6的频繁模式，长度是4及以上的频繁模式数量达到了49+5+2=57个.由于任何频繁模式的子模式都是频繁模式，因而频繁模式越长，其意义和应用价值越大.通过表5不难看出，一般间隙挖掘不但能够挖掘到非负间隙下最长频繁模式，而且能够发现出更多长模式，故此一般间隙挖掘较非负间隙挖掘具有更大意义和更多的应用价值.

此外，我们注意到不是在所有模式长度下，一般间隙挖掘结果都多于非负间隙挖掘结果.例如在SDB1下，当频繁模式长度为3时，一般间隙挖掘结果仅为3440个，而非负间隙挖掘结果为3888个，造成这一现象的原因是我们的序列模式挖掘方法在出现数计算中无任何约束，为了满足Apriori性质我们采用了新的偏移序列的定义，并采用模式的支持率来评价其频繁性.在一般间隙下，模式的支持数一定会大于非负间隙下的支持数，而偏移序列$ofs(P,S)$的计算公式为LW^{l-1}，这里$W=N-M+1$.当间隙为[0,2]时，其$W=2-0+1=3$，而当间隙变为[−2,2]时，其$W=2-(-2)+1=5$，当偏移序列的增长速度大于支持数的增长速度时，会使得挖掘到的模式数量有所减少.如果采用无重叠约束[6]或一次性约束[14]进行挖掘，那么将采用支持数作为模式是否频繁的判定依据，就可以避免这一现象.但是在这两种条件下计算模式的支持数均属于NP-Hard问题，因而也会引发新的问题和挑战，所以目前看无论是采用无约束还是采用无重叠约束或一次性约束进行序列模式挖掘均存在各自的优缺点.在不同阈值和N下，均能得到相同的结论，限于篇幅，略去了其他挖掘结果的对比.

6.2.2　挖掘速度

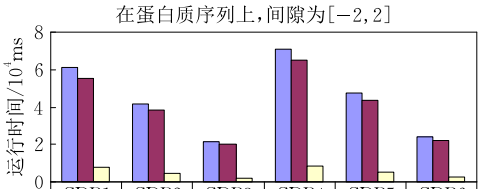
为了对比MCPaS-PRO、AMIN-PRO和MAPD-PRO的挖掘速度，我们在不同数据集上以及不同间隙下做了实验.图7、图8和图9分别显示了在蛋白质数据集上间隙分别为[−1,1]、[−2,2]和[−3,3]时三者的挖掘速度对比图；图10、图11和图12分别显示了在DNA数据集上间隙分别为[−1,1]、[−2,2]和[−3,3]时三者的挖掘速度对比图.

从图 7 到图 12 可以看出，字符集越大，MAPD-PRO 算法的优势越显著. 从图 7 到图 9 可以明显的看出，在蛋白质序列上 MAPD-PRO 算法的挖掘速



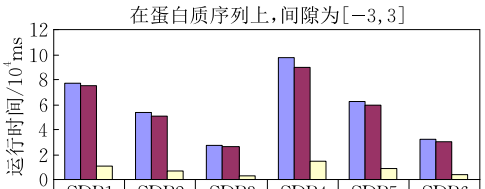
	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
MCPaS-Pro	37 133	25 870	13 600	44 780	29 709	15 393
AMIN-PRO	32 766	23 031	12 301	38 909	26 114	13 638
MAPD-PRO	3760	2378	1156	4720	2863	1366

图 7 [−1,1]间隙下挖掘速度对比图(蛋白质,0.0001)



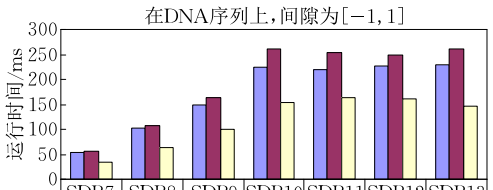
	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
MCPaS-Pro	60 929	41 649	21 193	70 935	47 595	24 342
AMIN-PRO	55 414	38 551	19 927	65 354	43 723	22 311
MAPD-PRO	7568	4452	2057	8700	5465	2459

图 8 [−2,2]间隙下挖掘速度对比图(蛋白质,0.0001)



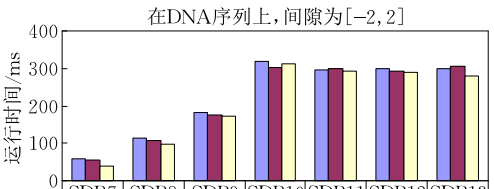
	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
MCPaS-Pro	774 86	53 322	27 722	97 742	62 668	32 020
AMIN-PRO	754 72	51 174	26 821	89 704	59 585	30 647
MAPD-PRO	104 76	7096	3201	14 177	8909	3782

图 9 [−3,3]间隙下挖掘速度对比图(蛋白质,0.0001)



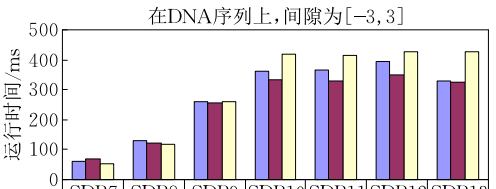
	SDB7	SDB8	SDB9	SDB10	SDB11	SDB12	SDB13
MCPaS-PRO	53	102	150	225	220	226	230
AMIN-PRO	56	108	164	260	254	250	262
MAPD-PRO	34	64	101	154	163	160	147

图 10 [−1,1]间隙下挖掘速度对比图(DNA,0.005)



	SDB7	SDB8	SDB9	SDB10	SDB11	SDB12	SDB13
MCPaS-PRO	60	113	181	320	296	300	298
AMIN-PRO	54	107	177	302	298	293	305
MAPD-PRO	40	97	172	311	293	289	280

图 11 [−2,2]间隙下挖掘速度对比图(DNA,0.005)



	SDB7	SDB8	SDB9	SDB10	SDB11	SDB12	SDB13
MCPaS-PRO	62	132	260	363	365	395	328
AMIN-PRO	68	121	256	333	328	351	324
MAPD-PRO	53	118	260	420	413	427	426

图 12 [−3,3]间隙下挖掘速度对比(DNA,0.005)

度较 MCPaS-PRO 算法以及 AMIN-PRO 算法提升十分显著. 而从图 10 到图 12 也可以看到，在 DNA 序列的大多数情况下，MAPD-PRO 算法较 MCPaS-PRO 和 AMIN-PRO 算法略好. 这说明了字符集越大，MAPD-PRO 算法的优势就越明显.

此外，间隙约束越小，MAPD-PRO 算法的优势越显著. 从图 7 可以看出 MAPD-PRO 算法较 MCPaS-PRO 以及 AMIN-PRO 算法平均提高了 9～10 倍左右；而图 8 就大约只有 8～9 倍；到图 9 时，MAPD-PRO 算法仅仅提高了 7 倍多. 而图 7 到图 9 间隙约束分别为[−1,1]、[−2,2]和[−3,3]，间隙是逐渐增大的. 图 10 到图 12 也呈现类似特点. 在[−1,1]下图 10 反映出 MAPD-PRO 算法较 MCPaS-PRO 算法和 AMIN-PRO 算法优势明显；在[−2,2]下图 11 反映出 MAPD-PRO 算法全部优于 MCPaS-PRO 算法，仅在 SDB10 数据集上略逊于 AMIN-PRO；但是在[−3,3]下图 12 反映出 MAPD-PRO 算法仅在 SDB7～SDB9 等序列数据库长度略小的数据集上优于 MCPaS-PRO 和 AMIN-PRO 算法，说明了间隙越小 MAPD-PRO 算法优势越突出.

造成 MAPD-PRO 在 DNA 上间隙为[−2,2]和[−3,3]时优势没那么显著的原因是，MAPD-PRO 的时间复杂度为 $O(L\times(N-M+1)/|\Sigma|)$形式，因此当$|\Sigma|$较大时，MAPD-PRO 算法的性能提升较快，同时 MAPD-PRO 算法较 MCPaS-PRO 算法及 AMIN-PRO 算法逻辑相对复杂，这样当间隔较大且字符集相对较小时，算法的时间复杂度会有所上升. 而 MCPaS-PRO 及 AMIN-PRO 算法逻辑简单，尽管易于实现，但是在较大字符集上，性能较差. 因而 MAPD-PRO 是较好的算法，特别是当字符集较大且间隔较小时，MAPD-PRO 算法的优势非常显著.

此外，通过图 7 到图 12 说明了 MAPD-PRO、MCPaS-PRO 以及 AMIN-PRO 算法的挖掘速度与

总序列数据库的大小线性相关.我们以图 8 为例进行说明,在 SDB1、SDB2 和 SDB3 三种序列数据库上 MAPD-PRO、MCPaS-PRO 以及 AMIN-PRO 算法的运算速度均大致呈现了 3∶2∶1 的关系,而我们通过表 4 也可以看到 SDB1、SDB2 和 SDB3 三种序列数据库的大小也近似呈现 3∶2∶1 的关系,这种现象不但在蛋白质序列上很好地呈现出来,而且在 DNA 序列数据上依然存在.通过表 4 可以看到 SDB8、SDB9 和 SDB10 三个序列数据库的大小为1∶2∶3.3 的关系,而在图 10 到图 12 这 3 个图中,MAPD-PRO、MCPaS-PRO 以及 AMIN-PRO 算法的挖掘时间也近似呈现此关系.此外,我们可以看到 SDB10 至 SDB13 这 4 种序列数据库的大小都接近一致,仅仅是在子序列的个数上以及各个子序列的大小关系方面存在差异,而在图 10 到图 12 上可以看到,MAPD-PRO、MCPaS-PRO 以及 AMIN-PRO 算法在 4 种序列数据库上挖掘时间都近似相同,因而说明算法的挖掘时间与序列的个数以及各个子序列的长短无关.这与 MAPD-PRO 算法时间复杂度分析一致.

序列挖掘时间与间隙大小相关,间隙越大,挖掘时间越长.从图 7 到图 9 可以看出,在蛋白质序列上,当间隙 N 由 1 增大到 3 时,算法的挖掘时间也大致增加到 3 倍.我们以 SDB1 序列为例进行说明,MAPD-PRO 在[−1,1]、[−2,2]和[−3,3]的挖掘时间分别为 3867ms、6855ms 和 10154ms,因此运行时间大致为 1∶2∶3 的关系.而其他 5 种蛋白质序列数据库也都有类似的特点.在 DNA 序列数据库上,依然呈现类似特征.

综上,实验结果表明,MAPD-PRO 是较好的算法,而且验证了其时间复杂度与序列数据库总长度 L 及间隙 W 呈线性关系,与字符集大小 $|\Sigma|$ 呈反比关系,这与 MAPD-PRO 算法时间复杂度分析一致.

7　结　论

本文针对序列模式挖掘在有序序列上挖掘有序结果的不足进行研究,提出了间隙可负的一般间隙的序列模式挖掘问题.在给出问题定义的基础上,采用堆栈结构实现了深度优先遍历模式树的方式,对频繁模式进行挖掘.由于计算模式在序列数据库中的支持数时可以采用不完整网树所构成的森林进行求解,因此本文采用子模式的不完整网树森林通过一次扫描序列数据库对其所有超模式的支持数进行计算,并进而判定这些超模式的频繁性.实验结果验证了本文算法的有效性.

在未来研究方面,将有如下 3 个方面可以开展研究:

(1)本文目前仅针对给定的周期性一般间隙约束进行序列模式挖掘,如何针对未知的、非周期的一般间隙约束进行序列模式挖掘值得深入探索.

(2)本文仅在 DNA 序列以及蛋白质序列上进行了实验研究,如何针对真实消费序列进行挖掘同样值得探索.

(3)目前在支持数统计方面存在多种方式.本文仅在无约束下探讨了一般间隙序列模式挖掘问题,而一次性约束或无重叠约束下一般间隙序列模式挖掘问题同样值得研究;此外不同约束下挖掘结果之间有何区别与联系,也同样值得我们去思考与研究.

致　谢　在此,我们向对本文提出宝贵修改意见的论文审稿专家表示衷心地感谢!

参　考　文　献

[1] Agrawal R, Srikant R. Mining sequential patterns//Proceedings of the 11th IEEE International Conference on Data Engineering. Taipei, China, 1995: 3-14

[2] Shie B E, Yu P S, Tseng V S. Mining interesting user behavior patterns in mobile commerce environments. Applied Intelligence, 2013, 38(3): 418-435

[3] Pei J, Han J, Mortazavi-Asl B, et al. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth//Proceedings of the 17th IEEE International Conference on Data Engineering. Heidelberg, Germany, 2001: 215-224

[4] Lo D, Ding B, Lucia, Han J. Bidirectional mining of non-redundant recurrent rules from a sequence database//Proceedings of the 27th IEEE International Conference on Data Engineering. Hannover, Germany, 2011: 1043-1054

[5] Zhang M, Kao B, Cheung D W, Yip K Y. Mining periodic patterns with gap requirement from sequences. ACM Transactions on Knowledge Discovery from Data, 2007, 1(2): Article No. 7

[6] Ding B, Lo D, Han J, Khoo S C. Efficient mining of closed repetitive gapped subsequences from a sequence database//Proceedings of the 25th IEEE International Conference on Data Engineering. Shanghai, China, 2009: 1024-1035

[7] Li C, Yang Q, Wang J, Li M. Efficient mining of gap-constrained subsequences and its various applications. ACM Transactions on Knowledge Discovery from Data, 2012, 6(1): Article No. 2

[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. Las Vegas, USA, 2007: 329-334

[9] Li Z, Han J, Ding B, Kays R. Mining periodic behaviors of object movements for animal and biological sustainability studies. Data Mining and Knowledge Discovery, 2012, 24(2): 355-386

[10] Yen S J, Lee Y S. Mining non-redundant time-gap sequential patterns. Applied Intelligence, 2013, 39(4): 727-738

[11] Zhang S, Zhang J, Zhu X, Huang Z. Identifying follow-correlation itemset-pairs//Proceedings of the 6th IEEE International Conference on Data Mining. Hong Kong, China, 2006: 765-774

[12] Min F, Wu Y, Wu X. The Apriori property of sequence pattern mining with wildcard gaps. International Journal Functional Informatics and Personalised Medicine, 2012, 4(1): 15-31

[13] Wu Y, Wang L, Ren J, et al. Mining sequential patterns with periodic wildcard gaps. Applied Intelligence, 2014, 41(1): 99-116

[14] Wu Xin-Dong, Xie Fei, Huang Yong-Ming, et al. Mining sequential patterns with wildcards and the One-Off condition. Journal of Software, 2013, 24(8): 1804-1815(in Chinese)
(吴信东, 谢飞, 黄咏明等. 带通配符和 One-Off 条件的序列模式挖. 软件学报, 2013, 24(8): 1804-1815)

[15] El-Ramly M, Stroulia E, Sorenson P. From run-time behavior to usage scenarios: An interaction-pattern mining approach//Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining. Edmonton, Canada, 2002: 315-324

[16] Wang Hai-Ping, Hu Xue-Gang, Xie Fei, et al. Impact of pattern feature on pattern matching problem with wildcards and length constraints. Pattern Recognition and Artifical Intelligence, 2012, 25(6): 1013-1021(in Chinese)
(王海平, 胡学钢, 谢飞等. 模式特征对带有通配符和长度约束的模式匹配问题的影响. 模式识别与人工智能, 2012, 25(6): 1013-1021)

[17] Wu You-Xi, Liu Ya-Wei, Guo Lei, Wu Xin-Dong. Sub-nettrees for strict pattern matching with general gaps and length constraints. Journal of Software. 2013, 24(5): 915-932(in Chinese)
(武优西, 刘亚伟, 郭磊, 吴信东. 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)

[18] Zhu X, Wu X. Mining complex patterns across sequences with gap requirements//Proceedings of the 20th International Joint Conference on Artificial Intelligence. Hyderabad, India, 2007: 2934-2940

[19] 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. Las Vegas, USA, 2010: 109-114

[20] Chandonia J M, Hon G, Walker N S, et al. The ASTRAL compendium in 2004. Nucleic Acids Research, 2004, 32(supply 1): 189-192

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

ZHOU Kun, born in 1989, M. S. candidate. Her research interest is data mining.

LIU Jing-Yu, born in 1976, Ph. D., assistant professor. His research interests include network storage and storage security.

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

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

Background

Sequential pattern mining, which is a very important data mining problem, discovers frequent patterns in a sequence database. Recently, Sequential pattern mining with gap constraints, which is one of the hottest tasks in sequential pattern mining field, is to find patterns with gap constraints since this kind of pattern can flexibly reflect sequential behaviours and is often exhibited in many real-world fields. For example, miners can discover the common sequential

purchasing behaviours with time gaps for most of the customers according to the transactional database.

Traditional sequential pattern mining algorithms are used to discover the patterns with non-negative gap constraints. However, a more general and useful issue is to find the patterns with general gap constraints. To deal with this challenge, we address sequential pattern mining with periodic general gap constraints which satisfies the Apriori property. In this paper, depth first search strategy is adopted to construct the pattern tree and pattern matching approach is employed to calculate the support of the pattern effectively. The mining algorithm employs an Incomplete Nettree structure, the last layer of a Nettree, to calculate the supports of all super patterns of a frequent pattern by one way scan over the sequence database and then determines whether the super patterns are frequent or not. The experimental results show that the mining algorithm has a good performance.

This research is supported by the National Natural Foundation of China under Grant No. 61229301, the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China under Grant No. IRT13059, the Natural Science Foundation of Hebei Province of China under Grant No. F2013202138, the Key Project of the Educational Commission of Hebei Province under Grant No. ZD2014009, and the Youth Foundation of Education Commission of Hebei Province under Grant No. QN2014192.