免预设间隔约束的对比序列模式高效挖掘

王慧锋[1)] 段 磊[1),2),3)] 左 劼[1)] 王文韬[1)] 李钟麒[1)] 唐常杰[1)]

[1)](四川大学计算机学院 成都 610065)

[2)](四川大学华西公共卫生学院 成都 610041)

[3)](武汉大学软件工程国家重点实验室 武汉 430072)

摘 要 对比序列模式在识别不同类别序列样本集合的特征上有着重要的作用.已有对比序列模式挖掘算法需要用户预设间隔约束.在不具备充分先验知识情况下,用户不易准确地预设恰当的间隔约束,进而导致不能发现有用的模式.对此,文中设计了带紧凑间隔约束的最小对比序列模式挖掘算法,实现免预设间隔约束,并对候选模式自动计算最适合的间隔约束.此外,设计了 3 种剪枝策略来提高算法的执行效率.通过蛋白质序列、DNA 序列、行为序列数据集验证了提出的算法的有效性和高效率.

关键词 对比序列模式;间隔约束;序列数据挖掘

中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2016.01979

Efficient Mining of Distinguishing Sequential Patterns Without a Predefined Gap Constraint

WANG Hui-Feng[1)] DUAN Lei[1),2),3)] ZUO Jie[1)] WANG Wen-Tao[1)]

LI Zhong-Qi[1)] TANG Chang-Jie[1)]

[1)](*School of Computer Science, Sichuan University, Chengdu* 610065)

[2)](*West China School of Public Health, Sichuan University, Chengdu* 610041)

[3)](*State Key Laboratory of Software Engineering, Wuhan University, Wuhan* 430072)

Abstract Distinguishing sequential patterns with gap constraints are very useful for identifying important features for discriminating one class of sequences from sequences of other classes. A gap constraint should be predefined by users using the proposed mining methods for distinguishing sequential patterns. It is difficult for users to set suitable gap constraints without enough priori knowledge. As a result, useful patterns may be missed. To deal with this problem, this paper presents an algorithm for mining distinguishing sequential patterns with compact gap constraints. The proposed algorithm runs well without a predefined gap constraint. Instead, it computes the most suitable gap constraint for each candidate pattern. In addition, three pruning rules are designed to improve the efficiency of the algorithm. Experiments on protein sequences, DNA sequences, and activity sequences confirmed the effectiveness and efficiency of the proposed algorithm.

Keywords distinguishing sequential pattern; gap constraint; sequence data mining

收稿日期:2015-02-06;在线出版日期:2015-09-30.本课题得到国家自然科学基金(61103042)、教育部高等学校博士学科点专项科研基金(20100181120029)、软件工程国家重点实验室开放研究基金(SKLSE2012-09-32)和中国博士后科学基金(2014M552371)资助.**王慧锋**,男,1987 年生,硕士研究生,主要研究方向为数据挖掘.E-mail:huifengscu@163.com.**段 磊**(通信作者),男,1981 年生,博士,副教授,中国计算机学会(CCF)高级会员,主要研究方向为数据挖掘、医学信息分析.E-mail:leiduan@scu.edu.cn.**左 劼**,男,1977 年生,博士,副教授,主要研究方向为数据库与知识工程.**王文韬**,男,1991 年生,学士,主要研究方向为数据挖掘.**李钟麒**,男,1991 年生,博士研究生,主要研究方向为数据挖掘.**唐常杰**,男,1946 年生,教授,博士生导师,中国计算机学会(CCF)杰出会员,主要研究领域为数据库与知识工程.

1 引 言

序列模式挖掘作为数据挖掘的一项重要任务，有着广泛的应用.例如：广电服务商分析用户的收视纪录，发现用户的收视喜好，可为节目编排提供决策支持.再如：分析传染病传播的时空监测数据，可望发现传染病时空聚集性暴发规律，进而为防控工作提供参考.与此同时，序列模式挖掘也受到了众多研究者的关注，不同类型的序列模式被陆续提出，如频繁序列模式[1-2]、闭合序列模式[3]、对比序列模式[4]、周期模式[5]和偏序模式[6]等.

给定正类序列样本集合和负类序列样本集合，对比序列模式描述在正类序列样本集合中频繁(模式的支持度大于指定阈值 α)、且在负类序列样本集合中不频繁(模式的支持度小于指定阈值 β)的项集.对比序列模式能识别不同类别序列样本集合间的差异，并描述各类别样本集合的特征，因此适用于多个领域的序列数据分析.例如：在医学领域，分析阳性肿瘤和阴性肿瘤的 DNA 序列，识别对比序列模式，能够提高临床诊断的精度；在商业领域，对比不同年龄段顾客的购物行为，发现各年龄段顾客的购物模式，可以提高商品促销活动的针对性.

模式的支持度反映了序列集合中有多大比例的样本能够与之匹配.简而言之，匹配即模式在序列中出现.例如：模式 $P=aed$ 与序列 $S=aedcbfdc$ 匹配.为了提高模式的适用性，允许模式元素间出现通配符①，即允许模式元素不相邻地出现在序列中.例如：$P=a\$\$c\$f$ 与序列 $S=aedcbfdc$ 匹配.元素之间的连续的通配符个数称为间隔；为提高序列模式匹配的适用性并且方便表达，采用区间来表达序列模式元素两两间通配符个数的最小值和最大值，我们称这个区间为间隔约束.例如：给定模式 $P=ac$，设间隔约束为[1,2]，若 P 与序列 S 匹配，意味着在 S 中元素 a 和元素 c 间最少出现 1 个元素，最多出现 2 个元素.

在已有对比序列模式挖掘的研究[4,7-8]中，间隔约束作为挖掘参数与支持度阈值一同需要用户设定.实践表明，在没有足够先验知识时，用户很难设定恰当的间隔约束，失当的间隔约束，会使发现的模式偏离预期的结果，如例 1 所示.

例 1. 给定表 1 描述的正类序列样本集合 pos 和负类序列样本集合 neg，挖掘与所有正类序列样本匹配、但不与任何负类序列样本匹配的模式.结果表明：当间隔约束为[1,1]时，模式 ad 和 cf 满足条件；当间隔约束为[2,2]时，模式 df 满足条件；当间隔约束为[2,3]时，模式 acc 满足条件；当间隔约束为[0,3]时，模式 ab 满足条件.

表 1 序列数据集示例

序号	序列	类别
$S1$	$aedcbfdc$	pos
$S2$	$abdccfdc$	pos
$S3$	$eabdcefc$	pos
$S4$	$cebdcfac$	neg
$S5$	$adcdafec$	neg
$S6$	$debadcfa$	neg

例 1 表明，设置不恰当的间隔约束可能导致无法发现有用的对比序列模式.逐个预设所有可能的间隔约束显然效率低下，而且不可行.

能否自动计算最合适的间隔约束，使用户能更简便、高效地获得对比序列模式挖掘结果呢？

据我们所知，目前还没有对比序列模式挖掘研究工作考虑了在挖掘过程中自动计算合适的间隔约束.本文为响应这一需求，作了下列主要工作：(1)分析了间隔约束设置对对比序列模式挖掘的影响；(2)提出了免预设间隔约束的对比序列模式挖掘问题，避免了用户预设参数，有助于推动数据挖掘的实践；(3)设计了带紧凑间隔约束的最小对比序列模式挖掘算法，并通过翔实的实验验证了算法的有效性和执行效率.

本文第 2 节介绍相关工作；第 3 节定义本文的研究问题；第 4 节讲述本文设计的带紧凑间隔约束的最小对比序列模式挖掘算法，该算法不需要预设间隔约束；第 5 节在真实世界的 DNA 序列数据集、蛋白质序列数据集和用户行为序列数据集上验证本文提出算法的有效性和执行效率；第 6 节总结本文工作，并对下一步工作进行展望.

2 相关工作

分析序列数据并提取有用的模式，例如：频繁序列模式[1-2]、闭合序列模式[3]、对比序列模式[4]、周期模式[5]及偏序模式[6]等，能够提高人们对数据蕴含知识的发现能力，因而受到研究者的持续关注.由于篇幅有限，更多序列数据挖掘的研究及其应用请参阅文献[9-11].

在序列模式中考虑间隔约束，能够允许序列模

① 通配符可与任意符号匹配，本文用‘$’指代.

式与序列样本进行匹配时，序列模式元素间间隔一定数量的任意元素.这样，可以提高序列模式的适用性，因而间隔约束被普遍地应用于序列模式挖掘中.文献[12]在 PrefixSpan 算法[13]基础上设计了带间隔约束的序列模式挖掘算法.文献[14]研究了从给定 DNA 序列中挖掘满足支持度阈值和间隔约束的频繁序列模式的算法.文献[15]设计了 Gap-BIDE 算法挖掘带间隔约束的闭合序列模式.文献[5]针对单条序列样本，研究了满足给定间隔约束的频繁周期模式.文献[16]设计了利用带间隔约束的序列模式发现软件缺陷特征的方法.请注意，在这些研究工作中，间隔约束均作为算法运行参数之一，由用户预先指定.

对比序列模式挖掘[4,7-8]旨在从两类序列样本中，发现在一类样本中频繁出现，但在另一类中不频繁出现的序列模式.文献[4]定义了对比序列模式问题并设计了挖掘算法 ConSGapMiner.文献[7]应用对比序列模式于蛋白质序列实现多肽折叠分类的预测.文献[8]在考虑支持度计算过程中引入了模式出现密度的概念.值得一提的是，文献[4,7-8]中对比序列模式间隔约束的范围需要用户预先指定，而本文设计了根据对比序列模式自动计算间隔约束的方法.

在生物序列分析中常用 motif[17]来描述在序列样本特定位置频繁出现的模式.但是，motif 挖掘与本文工作有以下本质区别.首先，motif 挖掘只考虑单一类别的序列样本集合，而对比序列模式挖掘分析对象为两类序列样本集合；其次，对比序列模式不要求模式中各元素必须在序列中某特定位置出现；最后，对比序列模式中各元素间的间隔是相对位置，并非 motif 中元素在序列中的特定位置.因此，motif 挖掘算法并不能直接适用于本文研究内容.

尽管间隔约束能够提高序列模式的适用性，但需要注意的是在不具备先验知识的前提下，用户很难恰当地设定间隔约束.而不恰当的间隔约束必然导致不能发现或者遗漏序列模式.对此，本文研究了免预设间隔约束的对比序列模式挖掘问题.文献[18]研究了生物序列数据中序列模式发现的问题，并且根据模式出现的位置自动计算间隔约束.虽然文献[18]自动计算间隔约束，但与本文工作相比仍然有以下本质区别：首先，文献[18]是针对一条长序列进行挖掘，其设计的基于 Apriori 产生候选模式的算法框架不适用于对比序列模式挖掘.其次，文献[18]的挖掘对象没有类别，而本文的挖掘对象则是两个属于不同类的序列样本集合.第三，文献[18]基于模式出现位置没有重叠的前提计算间隔约束.相比较而言，本文提出的计算方法（本文第 4 节）没有此限定，因而更加灵活、通用.据我们所知，目前还没有对比序列模式挖掘研究工作考虑了在挖掘过程中自动计算间隔约束.

3　问题定义

记 Σ 为序列元素的集合.例如：对于 DNA 序列，$\Sigma=\{G,C,A,T\}$；对于蛋白质序列，Σ 包含 20 种基本氨基酸的编码.

给定 Σ，任一序列 S 可形式化地描述为 $S=e_1e_2\cdots e_n$，其中 $e_k\in\Sigma$ 称为 S 的一个元素，k 是区间 $[1,n]$ 中的正整数，称为 S 元素的下标.为便于表述，用 $S_{[k]}$ 指代序列 S 中的第 k 个元素，用 $|S|$ 指代序列 S 的长度，即 S 包含的元素的个数.例如：$S=aedcbfdc$，则 $S_{[4]}=c$，$|S|=8$.

对于序列 S'，若存在正整数序列 $\langle k_1,\cdots,k_m\rangle$，满足 $1\leq k_1<\cdots<k_m\leq|S|$，且 $S'=S_{[k_1]}\cdots S_{[k_m]}$，则称序列 S' 是 S 的子序列（S 包含 S'），记为 $S'\subseteq S$.为便于描述，在给定 S 的前提下，我们用 S' 各个元素在 S 中的下标（$\langle k_1,\cdots,k_m\rangle$）指代 S'，并称 $\langle k_1,\cdots,k_m\rangle$ 是 S' 在 S 中出现的一个实例.例如：对于 $S=aedcbfdc$，用 $\langle 2,4,7\rangle$ 指代子序列 $S'=ecd$.

对序列 S 中任意的两个元素 $S_{[i]}$ 和 $S_{[j]}$（$1\leq i<j\leq|S|$），出现在 $S_{[i]}$ 和 $S_{[j]}$ 之间的元素个数称为 $S_{[i]}$ 和 $S_{[j]}$ 的间隔，记为 $\mathrm{Gap}(S,i,j)$.容易看出，$\mathrm{Gap}(S,i,j)=j-i-1$.例如：在 $S=aedcbfdc$ 中元素 $e(S_{[2]})$ 和元素 $b(S_{[5]})$ 之间的间隔 $\mathrm{Gap}(S,2,5)=2$.

定义 1（间隔约束）.　间隔约束 γ 为一个整数域区间，记作 $\gamma=[\gamma.\min,\gamma.\max]$，其中 $\gamma.\min,\gamma.\max\in Z_0^+$，且 $\gamma.\min\leq\gamma.\max$.称 $\gamma.\min$ 为 γ 的下界，$\gamma.\max$ 为 γ 的上界，$\gamma.\max-\gamma.\min$ 为 γ 的宽度.

设 $\langle k_1,\cdots,k_m\rangle$ 为序列 S 的一个子序列，若 $\gamma.\min\leq k_{l+1}-k_l-1\leq\gamma.\max(1\leq l<m)$，则称子序列 $\langle k_1,\cdots,k_m\rangle$ 满足间隔约束 γ.

定义 2（γ-匹配）.　给定间隔约束 γ，序列 P 和序列 S.若 S 有子序列 $\langle k_1,\cdots,k_{|P|}\rangle$ 满足 γ，且 $P=S_{[k_1]}\cdots S_{[k_{|P|}]}$，则称 P,S 满足 γ-匹配.

例 2.　令序列 $P=ef$，序列 $S=aedcbfdc$.若 $\gamma=[2,3]$，那么由于 $P=S_{[2]}S_{[6]}$，且 $\gamma.\min\leq\mathrm{Gap}(S,2,6)\leq\gamma.\max$，$P,S$ 满足 γ-匹配；若 $\gamma=[1,2]$，那么 $\mathrm{Gap}(S,2,6)\notin[1,2]$，$P$ 不匹配 S.

对于任一序列 S 以及 S 的子序列 P' 和 P，且 $P'\subseteq P$，若 P,S 满足 γ-匹配，则 P',S 不一定满足 γ-匹配，如例 3 所示.

例 3. 给定序列 $S=aedcbfdc$，令序列 $P=adc$，序列 $P'=ac$. 容易看出 $P'\subseteq P$. 令间隔约束 $\gamma=[0,1]$，由于 $P=S_{[1]}S_{[3]}S_{[4]}$，且 $\gamma.\min\leq \mathrm{Gap}(S,1,3)\leq \gamma.\max$，$\gamma.\min\leq \mathrm{Gap}(S,3,4)\leq \gamma.\max$，$P,S$ 满足 γ-匹配；而 $P'=S_{[1]}S_{[4]}$，$\mathrm{Gap}(S,1,4)\notin[0,1]$，因此 P' 不匹配 S.

给定间隔约束 γ，序列集合 D，序列 P 在 D 中的支持度（记为 $sup_D(\langle P,\gamma\rangle)$）为 D 中与 P 满足 γ-匹配的序列的比例，即

$$sup_D(\langle P,\gamma\rangle)=\frac{|\{S\in D\mid P,S\text{ 满足 }\gamma\text{-匹配}\}|}{|D|} \quad (1)$$

序列模式表达了元素在序列中出现的先后顺序. 在序列模式匹配中考虑间隔约束，能避免因为序列中噪声元素影响而不能匹配的情况，从而提高模式的适用性. 但是，间隔约束的宽度过大会降低序列模式本身表达含义的精度. 因此，本文在满足支持度阈值的相同前提下，选择宽度小的间隔约束作为挖掘结果. 对宽度相同的间隔约束，则优先选择下界小的作为结果.

为提高序列模式的实用性以及计算效率，序列模式挖掘通常以发现满足特定条件的“最小”序列模式为目标[4,8,19]，即包含“最小”序列模式的序列不作为挖掘结果输出. 例如，给定序列 P 和 P'，且 $P'\subseteq P$，若 P 和 P' 均满足支持度阈值要求，则只将 P' 作为挖掘结果输出. 不失一般性，我们沿用此约束，挖掘最小对比序列模式.

定义 3 给出了本文提出的带紧凑间隔约束的最小对比序列模式的定义.

定义 3（带紧凑间隔约束的最小对比序列模式）. 给定两个序列集合 pos（正类样本集合）和 neg（负类样本集合），设 α 和 β 为支持度阈值，如果存在间隔约束 γ，使得序列 P 满足如下条件(i)～(iv)，那么称 $\langle P,\gamma\rangle$ 为带紧凑间隔约束的最小对比序列模式.

(i) 在序列集合 pos 中频繁：$sup_{pos}(\langle P,\gamma\rangle)\geq\alpha$；

(ii) 在序列集合 neg 中不频繁：$sup_{neg}(\langle P,\gamma\rangle)\leq\beta$；

(iii) 模式最小化：不存在序列 P' 和间隔约束 γ'，满足 $P'\subseteq P$，$P'\neq P$，使得对 P' 和 γ'，条件(i)、(ii)成立；

(iv) 间隔约束最紧凑：对于 P，不存在符合如下任一条件的间隔约束 γ'，使得条件(i)、(ii)成立；

(a) 间隔约束宽度最小：$(\gamma'.\max-\gamma'.\min)<(\gamma.\max-\gamma.\min)$；

(b) 宽度相等时下界最小：$(\gamma'.\max-\gamma'.\min)=(\gamma.\max-\gamma.\min)$ 且 $\gamma'.\min<\gamma.\min$.

请注意，定义 3 中模式最小化条件只是限定模式之间的包含关系，并不限定模式的长度. 本文第 5 节的实验结果表明，带紧凑间隔约束的最小对比序列模式的长度可以为任意值.

对比文献[4]中最小对比序列模式的定义，本文提出的带紧凑间隔约束的最小对比序列模式是由序列 P 和间隔约束 γ 组成的二元组 $\langle P,\gamma\rangle$. 这样，既保留了对比序列模式能有效识别不同类别序列样本集合间差异的性质（条件(i)和(ii)）和最小化（条件(iii)），并且 γ 作为计算结果不需要在挖掘前预设. 因此，本文提出的模式表达更加灵活，同时在实践中算法运行参数设置更简单，更容易为用户使用，具有更强的实用性.

简而言之，本文研究的免预设间隔约束的对比序列模式挖掘问题为：在仅指定支持度阈值 α 和 β 的情况下，从序列集合 pos 和 neg 中挖掘出所有带紧凑间隔约束的最小对比序列模式，即 $\{\langle P,\gamma\rangle\mid P,\gamma$ 满足定义 3 中的条件(i)～(iv)$\}$.

例 4. 根据表 1 所示正类序列样本集合 pos 和负类序列样本集合 neg，令 $\alpha=1.0$，$\beta=0.0$，则带紧凑间隔约束的最小对比序列模式如表 2 所示.

表 2 带紧凑间隔约束的最小对比序列模式实例

$\langle ab,[0,3]\rangle$	$\langle df,[2,2]\rangle$
$\langle ad,[1,1]\rangle$	$\langle cf,[1,1]\rangle$
$\langle acc,[2,3]\rangle$	

观察表 2 可看出，不同的带紧凑间隔约束的最小对比序列模式其序列长度、间隔约束并不相同.

4 算法设计

为解决因为用户设置不恰当间隔约束导致不能或部分发现对比序列模式的问题. 我们提出了免用户预设间隔约束的算法. 算法要点：(1) 紧凑间隔约束；(2) 最小对比序列模式，称为 Algorithm for Minimal Distinguishing Sequential Patterns with Compact Gap Constraints，简称 MDSP-CGC.

MDSP-CGC 的主要步骤包括：(1) 产生候选序列模式；(2) 对每一个候选序列模式挖掘能满足支

持度阈值条件的紧凑间隔约束.

4.1　算法框架

MDSP-CGC 算法通过枚举所有可能的候选序列模式来保证挖掘的无遗漏，即挖出所有支持度阈值条件的带紧凑间隔约束的最小对比序列模式. MDSP-CGC 采用在序列数据挖掘中广泛应用的集合枚举树方法[20]产生候选序列模式. 对给定集合，如序列元素集合 Σ，集合枚举树依照集合元素的全序，枚举集合元素的所有可能组合. 图 1 示例了对 $\Sigma=\{a,b,c\}$，用集合枚举树枚举所有候选序列模式.

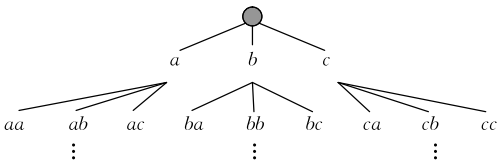


图 1　集合枚举树示例

MDSP-CGC 算法按深度优先的方式遍历集合枚举树. 令 P 为当前访问的结点表达的序列模式，若存在间隔约束 γ，使得$\langle P,\gamma\rangle$满足定义 3 中的条件，那么$\langle P,\gamma\rangle$是一个带紧凑间隔约束的最小对比序列模式. 同时，根据模式最小化性质，所有包含 P 的序列模式，都不可能成为最终解. 因此，我们得到候选序列模式枚举剪枝条件 1.

剪枝条件 1(最小模式剪枝).　给定序列模式 P 和间隔约束 γ，若$\langle P,\gamma\rangle$是一个带紧凑间隔约束的最小对比序列模式，那么剪去 P 在集合枚举树中的所有子孙结点.

例 5.　在例 1 中，当得到$\langle ab,[0,3]\rangle$满足条件，那么，在集合枚举树上 ab 的所有子孙结点都被剪去. 即，以 ab 开头的所有序列都不再作为候选模式.

性质 1.　给定序列集合 D，设 P'是序列模式 P 的子序列，Γ 为候选间隔约束集合，那么 $\mathrm{MAX}\{sup_D(\langle P',\gamma\rangle)\mid\gamma\in\Gamma\}\geq\mathrm{MAX}\{sup_D(\langle P,\gamma\rangle)\mid\gamma\in\Gamma\}$ 成立.

证明.　我们用反证法证明.

假设存在一个$\gamma'\in\Gamma$，使得 $\mathrm{MAX}\{sup_D(\langle P',\gamma\rangle)\mid\gamma\in\Gamma\}<sup_D(\langle P,\gamma'\rangle)$成立. 那么，$sup_D(\langle P',\gamma'\rangle)\leq\mathrm{MAX}\{sup_D(\langle P',\gamma\rangle)\mid\gamma\in\Gamma\}<sup_D(\langle P,\gamma'\rangle)$

根据式(1)可得

$$sup_D(\langle P',\gamma'\rangle)=\frac{|\{S\in D\mid P',S\text{ 满足 }\gamma'\text{-匹配}\}|}{|D|}<sup_D(\langle P,\gamma'\rangle)=\frac{|\{S\in D\mid P,S\text{ 满足 }\gamma'\text{-匹配}\}|}{|D|}\quad(2)$$

$\because P'\subseteq P$

$\therefore$ 若序列 S 与 P 满足 γ'-匹配，则一定与 P'满足 γ'-匹配，反之不一定成立.

$\therefore sup_D(\langle P',\gamma'\rangle)\geq sup_D(\langle P,\gamma'\rangle)$，与式(2)矛盾.

$\therefore$ 假设不成立.

$\therefore \mathrm{MAX}\{sup_D(\langle P',\gamma\rangle)\mid\gamma\in\Gamma\}\geq\mathrm{MAX}\{sup_D(\langle P,\gamma\rangle)\mid\gamma\in\Gamma\}$成立.　证毕.

性质 1 蕴含了下列候选序列模式剪枝条件.

剪枝条件 2(α 剪枝).　给定正类序列样本集合 pos，序列模式 P 和候选间隔约束集合 Γ，若 $\mathrm{MAX}\{sup_{pos}(\langle P,\gamma\rangle)\mid\gamma\in\Gamma\}<\alpha$，那么剪去 P 在集合枚举树中的所有子孙结点.

算法 1 描述了 MDSP-CGC 算法框架.

算法 1.　MDSP-CGC 框架.

输入：正类序列样本集合 pos，负类序列样本集合 neg，正类支持度阈值 α，负类支持度阈值 β

输出：带紧凑间隔约束的最小对比序列模式集合 MC

```
1.  FOR 深度优先遍历集合枚举树，产生候选模式 P
        DO
2.      计算 P 的候选间隔约束集合 Γ;
3.      IF ∃γ∈Γ 使得⟨P,γ⟩满足定义 3 条件(i),(ii),
            (iv) THEN
4.          移除 MC 中包含 P 的模式; //定义 3 条件(iii)
5.          MC←MC∪{⟨P,γ⟩};
6.          剪去 P 的所有子孙结点; //剪枝条件 1
7.      END IF
8.      IF 剪枝条件 2 成立 THEN
9.          剪去 P 的所有子孙结点; //剪枝条件 2
10.     END IF
11. END FOR
12. RETURN MC;
```

算法 1 步 3、步 4 保证挖掘结果符合定义 3 中的各项条件. 接下来，我们将介绍步 2、步 3 的详细实现过程.

4.2　间隔约束挖掘朴素算法

给定序列模式 P，MDSP-CGC 首先产生 P 的所有候选间隔约束(算法 1 步 2). 易知，若序列样本的最大长度为 l，则候选间隔约束下界的最小值为 0，上界的最大值为 $l-2$.

例 6.　在例 1 中，序列样本的最大长度为 8. 那么任意候选间隔约束 γ 满足：$0\leq\gamma.\min\leq\gamma.\max\leq6$.

这样，根据序列样本最大长度 l，可以得到 $\Gamma=\{\gamma\mid0\leq\gamma.\min\leq\gamma.\max\leq(l-2)\}$，作为候选间隔约束集合. 算法 2 给出了实现该过程的伪代码.

算法 2 的复杂度为 $O(l^2)$. 对算法 2 中产生的所有候选间隔约束,根据宽度和下界按升序排列,然后依次取出 Γ 中的间隔约束 $\gamma\in\Gamma$,测试$\langle P,\gamma\rangle$是否满足支持度阈值(算法 1 步 2).

可以看出,算法 2 对所有候选序列模式产生相同的候选间隔约束集合,这导致算法 2 的执行结果包含冗余,即不可能成为最终解的间隔约束. 为避免这个情况,本文设计基于序列模式实例的间隔约束产生算法.

算法 2. 间隔约束挖掘朴素算法.

输入:序列样本最大长度 l

输出:候选间隔约束集合 Γ

1. $\Gamma\leftarrow\varnothing$;
2. FOR $i\leftarrow 0$ TO $(l-2)$ DO
3. FOR $j\leftarrow i$ TO $(l-2)$ DO
4. $\Gamma\leftarrow\Gamma\cup\{[i,j]\}$;
5. END FOR
6. END FOR
7. RETURN Γ;

4.3 基于序列模式实例的间隔约束挖掘

首先,我们定义间隔约束的合并运算如下.

定义 4(间隔约束的合并). 给定间隔约束 γ_1 和 γ_2,构造间隔约束 γ 使得 γ 的下界为 γ_1 和 γ_2 下界的最小值,γ 的上界为 γ_1 和 γ_2 上界的最大值,称 γ 为 γ_1 和 γ_2 的合并,记为 $\gamma=\gamma_1\hat{\cup}\gamma_2$. 即 $\gamma=\gamma_1\hat{\cup}\gamma_2=[\mathrm{MIN}\{\gamma_1.\mathrm{min},\gamma_2.\mathrm{min}\},\mathrm{MAX}\{\gamma_1.\mathrm{max},\gamma_2.\mathrm{max}\}]$.

例如,$[2,3]\hat{\cup}[4,6]=[\mathrm{MIN}\{2,4\},\mathrm{MAX}\{3,6\}]=[2,6]$.

给定间隔约束 γ_1 和 γ_2,若满足 $\gamma_1.\mathrm{min}\leq\gamma_2.\mathrm{min}$,$\gamma_1.\mathrm{max}\geq\gamma_2.\mathrm{max}$,则称间隔约束 γ_1 覆盖 γ_2.

例 7. 给定间隔约束 $\gamma_1=[0,4]$,$\gamma_2=[1,2]$,$0<1,4>2$ 可知 γ_1 覆盖 γ_2.

回忆一下第 3 节定义的序列包含关系. 给定序列 P,若序列 S 中存在一组正整数序列$\langle k_1,\cdots,k_{|P|}\rangle$,满足 $S_{[k_1]}=P_{[1]}$,$S_{[k_2]}=P_{[2]}$,$\cdots$,$S_{[k_{|P|}]}=P_{[|P|]}$. $\langle k_1,\cdots,k_{|P|}\rangle$称为 P 在 S 中的一个实例. 给定实例$\langle k_1,\cdots,k_{|P|}\rangle$,我们按规则 1 产生区间 γ,称 γ 为 P 在 S 上的一个间隔约束实例.

规则 1.

$$\gamma.\mathrm{min}=\mathrm{MIN}\{\mathrm{Gap}(S,k_i,k_{i+1})\mid 1\leq i<|P|\}$$

$$\gamma.\mathrm{max}=\mathrm{MAX}\{\mathrm{Gap}(S,k_i,k_{i+1})\mid 1\leq i<|P|\}$$

例 8. 设 $S=aedcbfdc$,$P=acc$. 那么,$\langle 1,4,8\rangle$是 P 在 S 中的一个实例,且$[2,3]$为一个间隔约束实例.

序列 P 在序列 S 中有多个实例,将得到多个间隔约束实例. 例如,$S=aedcbfdc$,$P=ac$. 那么,可以得到间隔约束实例$[2,2]$和$[6,6]$.

给定序列模式 P,序列样本集合 D,记 $\mathcal{G}(P)$为 P 在 D 中所有序列上产生的间隔约束实例,即 $\mathcal{G}(P)=\{\gamma\mid\exists S\in D,\gamma$ 为 P 在 S 上的一个间隔约束实例$\}$.

例 9. 考虑表 1 中的正类序列样本集合(pos),序列模式 $P=acc$. 那么,可得到 P 在 $S1$ 的实例为$\langle 1,4,8\rangle$,在 $S2$ 的实例为$\langle 1,4,5\rangle$,$\langle 1,4,8\rangle$和$\langle 1,5,8\rangle$,在 $S3$ 的实例为$\langle 2,5,8\rangle$. 根据规则 1,$\mathcal{G}(P)=\{[0,2],[2,2],[2,3]\}$.

观察 1. 给定序列模式 P 和间隔约束 γ,假设$\langle P,\gamma\rangle$是一个带紧凑间隔约束的最小对比序列模式,那么

(1) γ 至少覆盖一个间隔约束实例($\langle P,\gamma\rangle$才可能满足支持度阈值).

(2) γ 或者等于一个间隔约束实例,或者等于多个间隔约束实例的合并(γ 满足紧凑条件).

根据观察 1,得到 P 的候选间隔约束集合 Γ.

$$\Gamma=\left\{\hat{\bigcup_{\gamma\in\mathcal{G}'}}\gamma\mid\mathcal{G}'\subseteq\mathcal{G}(P)\right\}\tag{3}$$

性质 2. 给定序列集合 D,序列模式 P,间隔约束 γ 和 γ',若 $\gamma\sqsubset\gamma'$,则 $sup_D(\langle P,\gamma'\rangle)\geq sup_D(\langle P,\gamma\rangle)$.

证明. 根据定义 2,$\{S\in D\mid P,S$ 满足 γ-匹配$\}\subseteq\{S\in D\mid P,S$ 满足 γ'-匹配$\}$,由式(1)可得 $sup_D(\langle P,\gamma\rangle)\leq sup_D(\langle P,\gamma'\rangle)$. 证毕.

由性质 2,可得到候选间隔约束剪枝条件 3.

剪枝条件 3(β 剪枝). 给定负类序列样本集合 neg,序列模式 P,和候选间隔约束集合 Γ. 令 $\gamma\in\Gamma$,若 $sup_{neg}(\langle P,\gamma\rangle)>\beta$,那么将 Γ 中所有覆盖 γ 的间隔约束移除.

给定间隔约束 $\gamma\in\Gamma$,若$\langle P,\gamma\rangle$满足支持度阈值且 γ 最紧凑,那么算法 1 步 3 为真. 若 γ 满足剪枝条件 3,则移除 Γ 中冗余的元素.

接下来介绍,如何从 Γ 中选择当前最紧凑的候选间隔约束. 我们借鉴了归并排序[21]的思想.

首先,求出 $\mathcal{G}(P)$ 中所有间隔约束实例的下界值,记为 $\mathcal{G}(P).\mathrm{min}$. 即 $\mathcal{G}(P).\mathrm{min}=\{\gamma.\mathrm{min}\mid\gamma\in\mathcal{G}(P)\}$.

然后,对 $\forall v\in\mathcal{G}(P).\mathrm{min}$,按式(4)找出 $\mathcal{G}(P)$中的间隔约束实例,通过合并运算产生所有以 v 为下界的候选间隔约束的上界,记为 $U(v)$.

$$U(v)=\{\gamma'.\max|\exists\gamma,\gamma'\in\mathcal{G}(P),$$
$$\text{s.t. } v=\gamma.\min, v\leq\gamma'.\min, \gamma.\max\leq\gamma'.\max\} \quad (4)$$

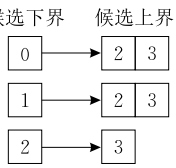
显然，

$$\hat{\bigcup_{v\in\mathcal{G}(P).\min}}\{[v,w]|w\in U(v)\}=\Gamma \quad (5)$$

按升序排序 $U(v)$ 中的元素. 那么，当前最紧凑间隔约束一定在$\{[v,w]|v\in\mathcal{G}(P).\min, w$ 是 $U(v)$ 的第一个元素$\}$.

设$[v,w]$为当前选择的候选间隔约束，若$\langle P,[v,w]\rangle$满足要求，返回$[v,w]$. 否则，从 $U(v)$中移除第一个元素(w). 重复在$\{[v,w]|v\in\mathcal{G}(P).\min, w$ 是 $U(v)$的第一个元素$\}$中选择最紧凑的候选间隔约束，直至找到满足条件的间隔约束，或测试完全部候选间隔约束.

例 10.　以 $\mathcal{G}(P)=\{[0,2],[1,2],[2,3]\}$ 为例. $\mathcal{G}(P).\min=\{0,1,2\}$，根据式(4)，可知 $U(0)=\{2,3\}$; $U(1)=\{2,3\}$; $U(2)=\{3\}$. 因此得到如下所示的候选间隔约束下界及相应的候选上界集合.



首先，从$\{[0,2],[1,2],[2,3]\}$中选择最紧凑的$[1,2]$作为候选间隔约束. 若$[1,2]$不满足，则继续从$\{[0,2],[1,3],[2,3]\}$中选择最紧凑的间隔约束作为候选，以此类推，直至找到满足条件的间隔约束，或考察完全部候选间隔约束.

算法 3 描述了本小节讨论的基于序列模式实例的间隔约束挖掘算法.

分析算法 3 最坏情况下的时间复杂度，通过扫描序列模式 P 中各个元素在 pos 每条序列中出现的位置，可以得到 $\mathcal{G}(P)$并同时得到 $\mathcal{G}(P).\min$. 因此，算法 3 步 1 和步 2 需要 $O(|pos|\times l_{pos}\times l_{pos})$时间，其中，$|pos|$表示正类序列样本的个数，$l_{pos}$ 表示正类序列样本的最大长度. 步 3～步 5 计算所有的$U(v)$需要 $O(|\mathcal{G}(P).\min|\times l_{pos}\times\log l_{pos})$时间. 步 6～步 15 对$\langle P,\gamma\rangle$进行支持度检测，需要$O(|\mathcal{G}(P).\min|\times l_{pos}\times(|pos|\times l_{pos}+|neg|\times l_{neg}))$时间，其中 l_{neg} 表示负类序列样本的最大长度. 因此，算法 3 的时间复杂度为$O(|pos|\times l_{pos}\times l_{pos}+|\mathcal{G}(P).\min|\times l_{pos}\times\log l_{pos}+|\mathcal{G}(P).\min|\times l_{pos}\times(|pos|\times l_{pos}+|neg|\times l_{neg}))=O(|\mathcal{G}(P).\min|\times l_{pos}\times(|pos|\times l_{pos}+|neg|\times l_{neg}))$.

相比于间隔约束挖掘朴素算法(算法 2)，算法 3 显著地减小了候选间隔约束集合的规模. 同时，通过对候选间隔约束排序，避免了产生冗余结果(非最紧凑间隔约束)，提高了效率. 下一节，我们将通过实验验证算法 3 的执行效率.

算法 3.　基于序列模式实例的间隔约束挖掘.

输入：正类序列样本集合 pos，负类序列样本集合 neg，序列模式 P

输出：间隔约束 γ

1. 根据 pos 产生 $\mathcal{G}(P)$;
2. $\mathcal{G}(P).\min\leftarrow\{r.\min|r\in\mathcal{G}(P)\}$;
3. FOR EACH $v\in\mathcal{G}(P).\min$ DO
4. 　计算 $U(v)$并按升序排列;
5. END FOR
6. REPEAT
7. 　$\gamma\leftarrow\{[v,w]|v\in\mathcal{G}(P).\min, w$ 是 $U(v)$的第一个元素$\}$中最紧凑的间隔约束;
8. 　$U(v)\leftarrow U(v)\backslash\{w\}$;
9. 　IF $sup_{pos}(\langle P,\gamma\rangle)\geq\alpha\wedge sup_{neg}(\langle P,\gamma\rangle)\leq\beta$ THEN
10. 　　RETURN γ; //$\langle P,\gamma\rangle$满足定义 3 条件(i),(ii),(iv)
11. 　END IF
12. 　IF $sup_{neg}(\langle P,\gamma\rangle)>\beta$ THEN //剪枝条件 3
13. 　　移除包含 γ 的候选间隔约束;
14. 　END IF
15. UNTIL $\gamma=\varnothing$;
16. RETURN $\varnothing$;

若由算法 3 得到间隔约束 γ 为非空，则$\langle P,\gamma\rangle$为带紧凑间隔约束的候选最小对比序列模式. 在两种情况下，候选序列模式 P 找不到最合适的间隔约束(算法 3 输出结果为$\varnothing$)：第一，在正类序列样本集合中不存在间隔约束使得 P 的支持度大于等于阈值α；第二，在负类序列样本集合中不存在间隔约束使得 P 的支持度小于等于阈值β. 在这两种情况下，P 将不作为结果输出.

根据剪枝策略和算法 3 的设计，容易看出 MDSP-CGC 算法的挖掘结果一定满足定义 3 的条件. 同时，定理 1 保证了 MDSP-CGC 算法的完备性.

定理 1.　给定候选序列模式 P，间隔约束 γ，若$\langle P,\gamma\rangle$为带紧凑间隔约束的最小对比序列模式，则$\langle P,\gamma\rangle$一定在 MDSP-CGC 算法的结果中.

证明.　我们用反证法证明.

假设存在一个$\gamma'\in\Gamma$，使得$\langle P,\gamma'\rangle$为满足带紧凑间隔约束的最小对比序列模式成立，但不在 MDSP-CGC 算法的挖掘结果中，那么有以下情况：

(1)$\langle P,\gamma'\rangle$被剪枝条件 1 移除；

∵ $\langle P,\gamma'\rangle$满足剪枝条件 1；

∴ 存在序列模式 P'满足定义 3 要求，且 $P'\sqsubseteq P$；

$\therefore \langle P,\gamma'\rangle$不满足定义 3 条件(iii);

(2)$\langle P,\gamma'\rangle$被剪枝条件 2 移除;

$\because \langle P,\gamma'\rangle$满足剪枝条件 2;

$\therefore sup_{pos}(\langle P,\gamma'\rangle)<\alpha$;

$\therefore \langle P,\gamma'\rangle$不满足定义 3 条件(i);

(3)$\langle P,\gamma'\rangle$被剪枝条件 3 移除;

$\because \langle P,\gamma'\rangle$满足剪枝条件 3;

$\therefore sup_{neg}(\langle P,\gamma'\rangle)>\beta$;

$\therefore \langle P,\gamma'\rangle$不满足定义 3 条件(ii);

综上,$\langle P,\gamma'\rangle$为满足带紧凑间隔约束的最小对比序列模式不成立,与原假设矛盾. 证毕.

5 实 验

5.1 实验环境

实验采用 UCI 机器学习数据集①提供的 *E.coli* 基因序列数据集和用户行为序列数据集 Activities of Daily Living(ADLs),以及 pfam 数据库(http://pfam.sanger.ac.uk/)提供的蛋白质序列数据集 CbiA/X 和 TatD/C 来验证本文提出的 MDSP-CGC 算法的有效性和执行效率. *E.coli* 中的基因序列包含 G、C、A、T 4 种编码,CbiA/X 和 TatD/C 中的序列包含 41 种编码,ADLs 记录了 A、B 两个用户的日常行为序列(10 种类型). 表 3 列出了实验数据集的特征.

表 3 实验数据集特征

正类序列集合	正类序列个数	正类序列平均长度	负类序列集合	负类序列个数	负类序列平均长度
E.coli+	53	58	*E.coli*−	53	58
CbiA	80	205	CbiX	76	106
TatD	119	205	TatC	75	226
ADLs-A	21	23	ADLs-B	14	17

所有算法采用 Java 语言实现. 为区别 4.2 节和 4.3 节给出的两种不同的候选间隔约束产生方式,将基于序列模式实例产生候选间隔约束(算法 3)的版本记为 MDSP-CGC,将采用间隔约束挖掘朴素算法(4.2 节算法 2)产生候选间隔约束的版本记为 Baseline.

同时,为验证 MDSP-CGC 算法的实用性,我们实现了与本文工作最相关的文献[4]中提出的 ConSGapMiner 算法(为方便表达,用 CSM 指代 ConSGapMiner). 请注意,CSM 算法必须预先给定间隔约束,并且间隔约束下界的取值固定为 0. 为进行对比,我们用 Java 语言重新编写 CSM 算法以支持间隔约束 γ 下界为任一非负整数值.

在本文所有实验中,默认正类样本集的相对支持度阈值(α)设置为 0.6,负类样本数据集的相对支持度阈值(β)设置为 0.4.

所有实验都在配置为 Intel Core i7-3770 3.90 GHz CPU,8 GB 内存,Windows 7 操作系统的 PC 上完成.

5.2 MDSP-CGC 算法有效性验证

为验证 MDSP-CGC 算法的有效性,我们在 *E.coli*、CbiA/X、TatD/C、ADLs 上分别运行 MDSP-CGC 算法,挖掘带紧凑间隔约束的最小对比序列模式. 图 2 给出了在 4 个数据集上挖掘的序列模式最紧凑间隔约束宽度的分布情况. 可以看出,序列模式集合在各个间隔约束宽度上均分布有一定数量的序列模式. 请注意 MDSP-CGC 在 *E.coli*、CbiA/X、TatD/C、ADLs 数据集上挖掘得到的大部分(>66.7%)结果的间隔约束的宽度不超过 4. 我们分析这是因为 MDSP-CGC 优先选择满足支持度阈值的紧凑(宽度小)间隔约束.

表 4~表 7 列出了在 4 个数据集上,带紧凑间隔约束的最小对比序列模式的平均长度和序列模式

表 4 *E.coli* 挖掘模式统计(β=0.4)

$\alpha\times\lvert E.coli+\rvert$	序列模式平均长度	序列模式个数
32	2.714	7
34	2.750	4
36	6.586	7
38	6.250	8

表 5 CbiA/X 挖掘模式统计(β=0.4)

$\alpha\times\lvert CbiA\rvert$	序列模式平均长度	序列模式个数
48	1.997	359
50	1.997	357
52	1.997	354
54	1.997	351

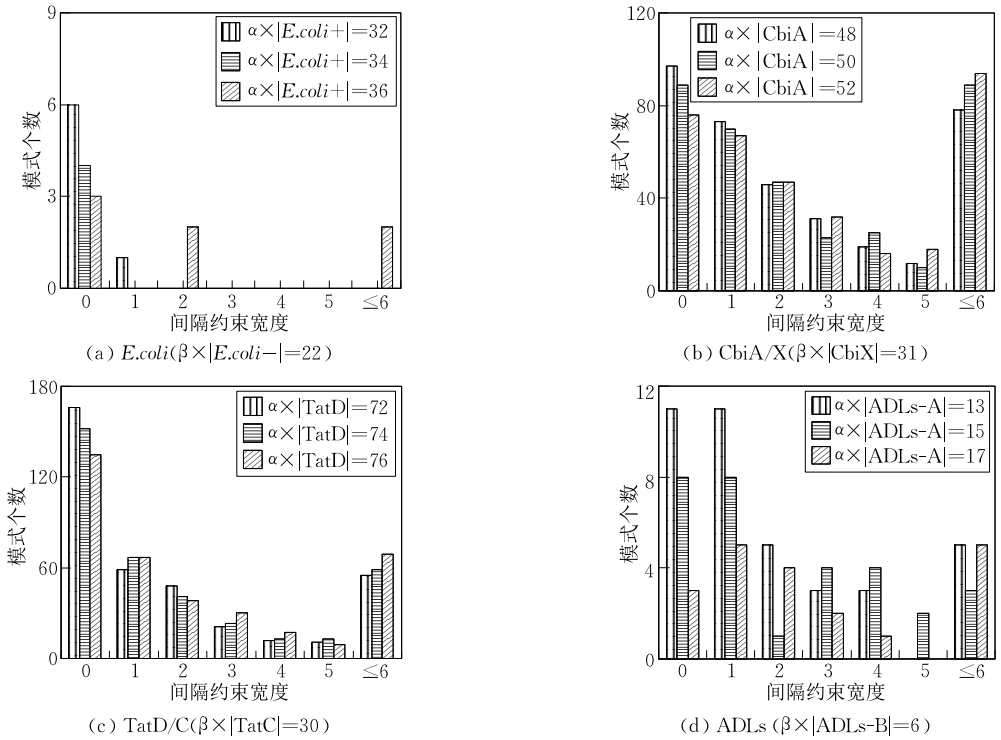
表 6 TatD/C 挖掘模式统计(β=0.4)

$\alpha\times\lvert TatD\rvert$	序列模式平均长度	序列模式个数
72	2.000	372
74	2.003	368
76	2.000	365
78	2.008	363

表 7 ADLs 挖掘模式统计(β=0.4)

$\alpha\times\lvert ADLs\text{-}A\rvert$	序列模式平均长度	序列模式个数
13	2.00	38
15	2.00	30
17	2.00	20
19	2.17	6

① http://archive.ics.uci.edu/ml



(a) *E.coli*(β×|*E.coli*−|=22)　(b) CbiA/X(β×|CbiX|=31)

(c) TatD/C(β×|TatC|=30)　(d) ADLs(β×|ADLs-B|=6)

图 2　带紧凑间隔约束的对比序列模式数量的分布 ($\beta=0.4$)

的个数. 由表 4 可知，当负类样本支持度阈值固定，随着正类样本支持度阈值增大，*E.coli* 数据集(DNA 序列)上发现的对比序列模式的平均长度呈递增趋势. 由于基因序列中只有 G,C,A,T 这 4 种编码，较短的序列模式难以满足支持度阈值条件. 随着正类样本支持度阈值的增大，集合枚举树向下搜索的深度相应增加. 因此，结果集序列模式的平均长度会增加.

表 5、表 6、表 7 分别列出了 MDSP-CGC 算法在蛋白质序列数据集 CbiA/X、TatD/C 和用户行为序列数据集 ADLs 上所发现的对比序列模式的平均长度和个数. 与表 4 不同，在蛋白质序列数据集中 MDSP-CGC 算法发现的序列模式的平均长度和模式的个数没有随着正类样本支持度阈值的增大而变化. 我们分析其原因为：在蛋白质序列数据集中存在满足条件且长度较短的对比序列模式，根据剪枝条件 1，所以由集合枚举树生成的候选模式集合规模相对固定.

值得一提的是，表 5 中序列模式的平均长度为 1.997，是因为在 CbiA/X 数据集中单元素模式“X”在 CbiA 类中出现了 773 次，而在 CbiX 类中出现次数为 0. 因此，模式“X”满足支持度阈值条件，出现在结果中(请注意定义 3 关于对比序列模式的条件并不限定模式长度必须大于 1).

图 2 和表 4～表 7 说明，如果只考虑某一宽度的间隔约束，将不能发现满足其他间隔约束的最小对比序列模式.

为进一步说明此问题，我们利用与本文工作最相关的文献[4]中提出的 CSM 算法来挖掘指定间隔约束下的最小对比序列模式. 表 8 列出了在支持度阈值 $\alpha=0.6$ 和 $\beta=0.4$ 时，CSM 算法(需要预先指定间隔约束)发现的最小对比序列模式个数，以及 MDSP-CGC 算法挖掘的最小对比序列模式个数(MDSP-CGC 算法不需要预先设定间隔约束). 从表 8 可以看出一般情况下，MDSP-CGC 算法能够发现更多的最小对比序列模式. 但值得注意，在 *E.coli* 数据集中 CSM 算法得到的序列模式个数更多. 这是因为 MDSP-CGC 算法挖掘满足支持度阈值的最小对比序列模式，根据最小模式剪枝，CSM 得到的部分结果在 MDSP-CGC 算法结果中不是最小，因此不会出现在 MDSP-CGC 的挖掘结果中. 因此，在不具备设定间隔约束先验知识的前提下，本文提出

表 8　MDSP-CGC 与 CSM 挖掘模式个数($\alpha=0.6,\beta=0.4$)

数据集	MDSP-CGC	CSM		
		[0,1]	[1,3]	[3,6]
E.coli	7	19	46	20
CbiA/X	359	37	80	156
TatD/C	372	65	121	205
ADLs	38	7	6	13

的 MDSP-CGC 算法能更灵活、有效地发现对比序列模式.

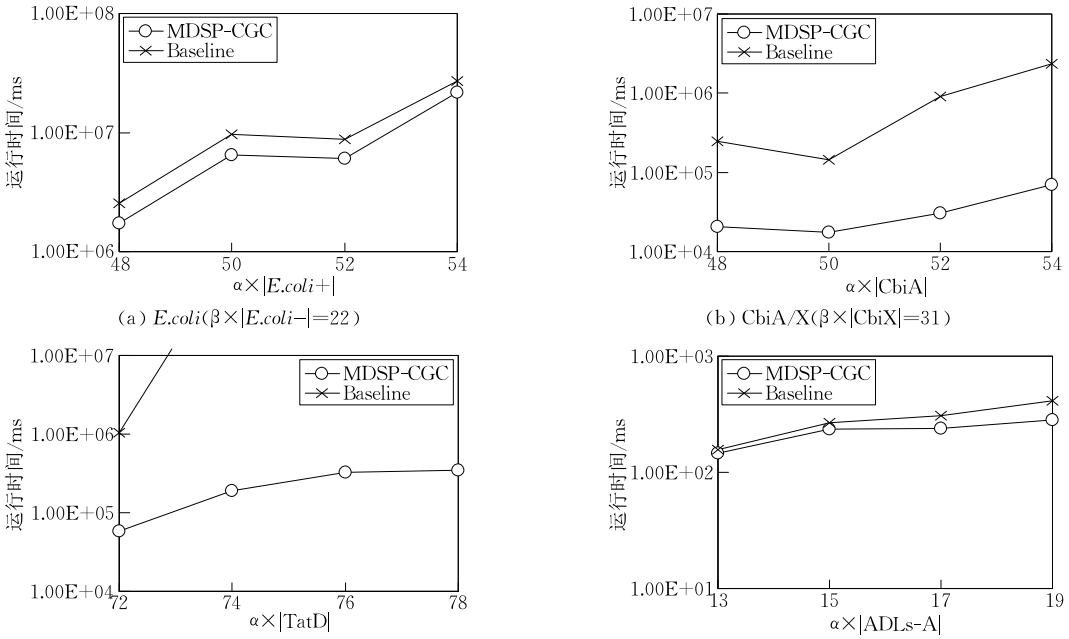
由于篇幅有限,我们仅在表 9 中列出当设定 $\alpha=0.6,\beta=0.4$ 时,MDSP-CGC 算法在 *E.coli* 数据集上的挖掘结果.观察表 9 可知,各个模式的间隔约束范围并不统一.例如,设定间隔约束为[1,3],只能发现部分结果.表 10 中列出当设定 $\alpha=0.6,\beta=0.4$ 时,MDSP-CGC 算法在 ADLs 数据集上间隔约束宽度为 2 的的挖掘结果.由表 10 可知,得到的序列模式的间隔约束在各个区间上均有分布.若先验地给出间隔约束,则可能丢失序列模式.例如:若指定间隔约束为[1,3],则会丢失间隔约束[3,5]、[4,6]、[9,11]和[12,14]所相关的 4 个对比序列模式.

表 9 MDSP-CGC 在 *E.coli* 的挖掘结果($\alpha=0.6,\beta=0.4$)

⟨*aa*,[44,44]⟩	⟨*ag*,[46,46]⟩
⟨*att*,[14,14]⟩	⟨*ctgt*,[6,7]⟩
⟨*tta*,[3,3]⟩	⟨*tc*,[46,46]⟩
⟨*ttt*,[7,7]⟩	

表 10 MDSP-CGC 在 ADLs 的挖掘结果($\alpha=0.6$, $\beta=0.4$,间隔约束宽度为 2)

⟨*Snack Grooming*,[1,3]⟩	⟨*Leaving Snack*,[3,5]⟩
⟨*Breakfast Leaving*,[4,6]⟩	⟨*Toileting Leaving*,[9,11]⟩
⟨*Sleeping Toileting*,[12,14]⟩	



(a) *E.coli*($\beta\times|E.coli-|=22$) (b) CbiA/X($\beta\times|CbiX|=31$)

(c) TatD/C($\beta\times|TatC|=30$) (d) ADLs($\beta\times|ADLs-B|=6$)

图 3 MDSP-CGC 算法与 Baseline 算法运行时间比较($\beta=0.4$)

由此可见,相较于 CSM 需要预先指定间隔约束,MDSP-CGC 算法能自动挖掘对比序列模式并计算最合适的间隔约束则更加实用.

5.3 MDSP-CGC 算法执行效率验证

为验证 MDSP-CGC 算法的执行效率,我们记录了 MDSP-CGC 算法在不同正类样本绝对支持度阈值($\alpha\times|pos|$)下的运行时间.

据我们所知,目前还没有与 MDSP-CGC 挖掘任务完全一致的相关工作,我们将 MDSP-CGC 与 Baseline 算法的运行时间进行对比;容易看出,MDSP-CGC 算法明显快于 Baseline 算法.特别地,随着数据集规模增大,序列平均长度增长,MDSP-CGC 算法相比于 Baseline 算法效率提升更显著.

需要注意的是,随着正类样本支持度阈值的增大,MDSP-CGC 算法耗费的时间逐渐增加.这是因为,随着正类样本绝对支持度阈值增大,大量候选序列模式都不适用于剪枝条件 1,导致集合枚举树上搜索的深度增加,产生更多的候选序列模式,增加算法运行时间.但 MDSP-CGC 算法的执行时间明显少于 Baseline 算法.

表 11 记录了在设置 $\alpha=0.6,\beta=0.4$ 时,MDSP-CGC 算法在不同数据集上经过 β 剪枝后的候选间隔约束(即算法 3 步 9 中考察的候选间隔约束)的宽度分布情况.

表 11 MDSP-CGC候选间隔约束的宽度分布($\alpha=0.6,\beta=0.4$)

数据集	候选间隔约束宽度						
	0	1	2	3	4	5	≥6
E.coli	48	42	31	32	27	22	141
CbiA/X	53	46	56	59	57	50	1129
TatD/C	128	135	153	140	133	133	3358
ADLs	10	15	12	10	8	7	13

根据表 11 所涉及的候选间隔约束，图 4 示例了在 $\alpha=0.6$，$\beta=0.4$ 时，CSM 算法在选取不同间隔约束宽度的情况下，考察同等规模的候选间隔约束所需的执行时间. 为便于对比，图 4 也示例了 MDSP-CGC 在同样支持度阈值下，挖掘带紧凑间隔约束的运行时间. 由于执行 CSM 算法需要预先指定间隔约束，因此，我们计算 CSM 执行所有给定间隔约束的运行时间之和. 例如，图 4 中，CSM-3 表示根据表 11 所涉及的候选间隔约束，选择所有宽度不超过 3 的候选间隔约束来执行 CSM 算法. 可以看出，随着候选间隔约束宽度的增长，利用 CSM 算法发现带紧凑间隔约束的最小对比序列模式需要更多的运行时间. 值得注意的是，在图 4(b)和图 4(c)中，虽然 CSM 算法在间隔约束宽度规模小于等于 4 时执行时间比 MDSP-CGC 算法少，但是 CSM 算法在间隔约束宽度规模小于等于 4 时不能找到所有带紧凑间隔约束的最小对比序列模式.

最后，我们验证了 MDSP-CGC 算法的伸缩性，考虑了 MDSP-CGC 算法执行时间与数据集的选择相关. 我们通过复制的方法，将实验数据集规模依次增大 k 倍. 这样，在保持支持度阈值相同情况下，算法挖掘结果保持不变. 图 5 示例了 MDSP-CGC 算法

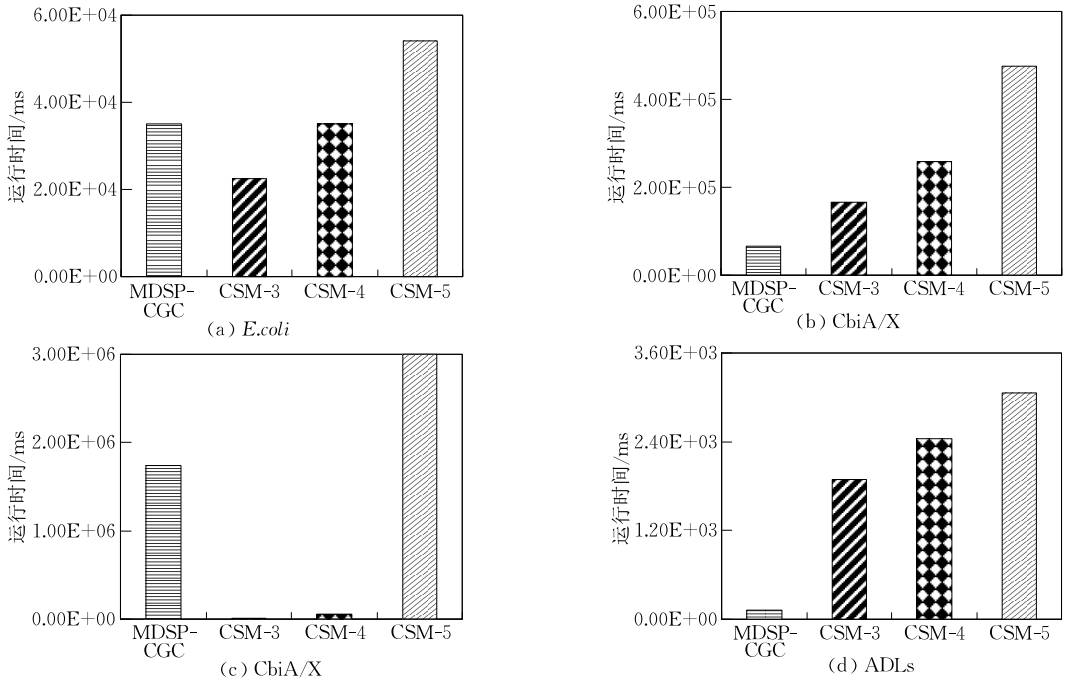


图 4　MDSP-CGC 算法与 CSM 算法运行时间比较($\alpha=0.6$，$\beta=0.4$)

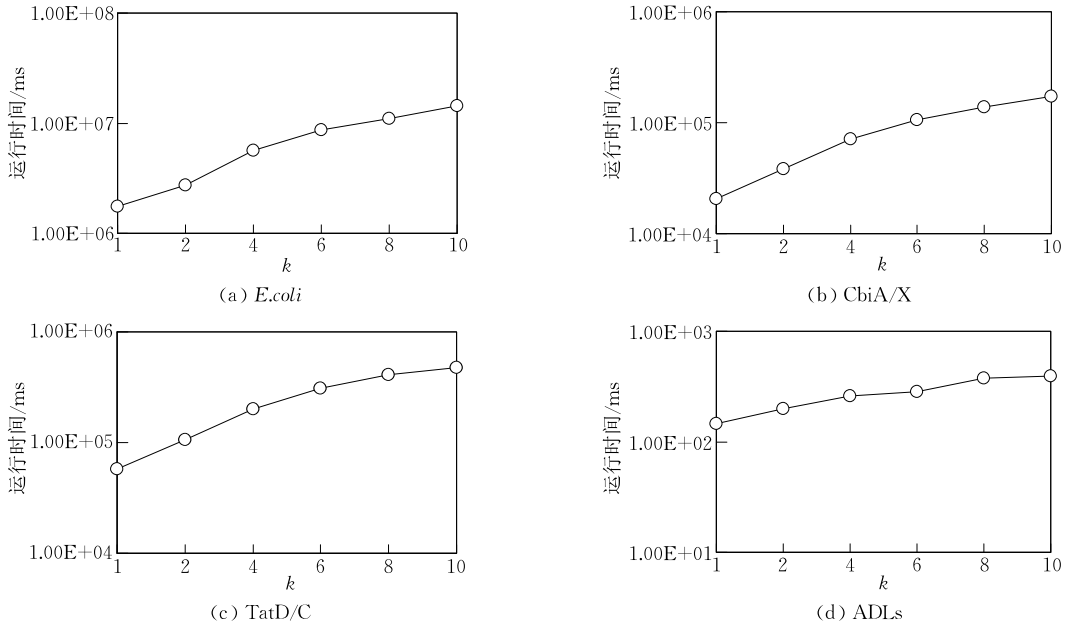


图 5　MDSP-CGC 算法伸缩性分析($\alpha=0.6$，$\beta=0.4$)

在实验数据集 *E.coli*、CbiA/X、TatD/C、ADLs 规模增大 k 倍，正类样本数据集的相对支持度为 0.6，负类样本数据集的相对支持度为 0.4 的条件下，执行时间的变化情况. 观察图 5 可以看出，MDSP-CGC 执行时间随着数据集规模增大而线性增长. 可见 MDSP-CGC 算法具有较好的伸缩性.

6 结 论

对比序列模式在识别不同类别序列样本集合的特征上有着重要的作用. 已有对比序列模式挖掘算法需要用户预设间隔约束. 在不具备充分先验知识的情况下，用户不易准确地预设恰当的间隔约束，进而导致不能发现有用的模式. 针对这个问题，本文设计了带紧凑间隔约束的最小对比序列模式挖掘 MDSP-CGC 算法，实现免用户设置间隔约束，并对候选模式自动计算最适合的间隔约束，避免了用户在不具备先验知识的情况下因为设置参数不恰当而丢失结果的情况，因此对推动对比序列模式挖掘的实际应用有积极作用. 我们设计了 3 种剪枝策略和相应的数据结构，来保证算法 MDSP-CGC 具有较高的执行效率. 在蛋白质序列、DNA 序列以及行为序列数据集上的实验验证了 MDSP-CGC 算法的有效性和执行效率.

下一步，我们将把免间隔约束对比序列模式挖掘的思想应用于其他类型的序列模式挖掘. 此外，我们将设计 MDSP-CGC 算法的分布式实现版本，以及基于免间隔约束对比序列模式的分类器训练，并尝试将其应用于金融、医学等领域的数据分析.

参 考 文 献

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

[2] Zaki M J. SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, 2001, 42(1-2): 31-60

[3] Yan X, Han J, Afshar R. CloSpan: Mining closed sequential patterns in large databases//Proceedings of the 3rd SIAM International Conference on Data Mining. San Francisco, USA, 2003: 166-177

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

[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): 7

[6] Pei J, Wang H, Liu J, et al. Discovering frequent closed partial orders from strings. IEEE Transaction on Knowledge and Data Engineering, 2006, 18(11): 1467-1481

[7] Shah C C, Zhu X, Khoshgoftaar T M, Beyer J. Contrast pattern mining with gap constraints for peptide folding prediction//Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference. Coconut Grove, USA, 2008: 95-100

[8] Wang X, Duan L, Dong G, et al. Efficient mining of density-aware distinguishing sequential patterns with gap constraints //Proceedings of the 19th International Conference on Database Systems for Advanced Applications. Bali, Indonesia, 2014: 372-387

[9] Dong G, Pei J. Sequence Data Mining. Heidelberg: Springer, 2007

[10] Mooney C H, Roddick J F. Sequential pattern mining: Approaches and algorithms. ACM Computing Surveys, 2013, 45(2): 19

[11] Kumar P, Krishna P R, Raju S B. Pattern Discovery Using Sequence Data Mining: Applications and Studies. USA: IGI Global, 2011

[12] Antunes C, Oliveira A L. Generalization of pattern-growth methods for sequential pattern mining with gap constraints// Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition. Leipzig, Germany, 2003: 239-251

[13] Pei J, Han J, Mortazavi-Asl B, et al. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth//Proceedings of the 1st International Conference on Data Engineering. Heidelberg, Germany, 2001: 0215-0215

[14] Xie F, Wu X, Hu X, et al. MAIL: Mining sequential patterns with wildcards. International Journal of Data Mining & Bioinformatics, 2013, 8(1): 1-23

[15] 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): 2

[16] Zeng Q, Chen Y, Han G, Ren J. Sequential pattern mining with gap constraints for discovery of the software bug features. Journal of Computational Information Systems, 2014, 10(2): 673-680

[17] Schneider T D, Stephens R M. Sequence logos: A new way to display consensus sequences. Nucleic Acids Research, 1990, 18(20): 6097-6100

[18] Wu X, Zhu X, He Y, Arslan A N. PMBC: Pattern mining from biological sequences with wildcard constraints. Computers in Biology & Medicine, 2013, 43(5): 481-492

[19] Lo D, Li J, Wong L, Khoo S. Mining iterative generators and representative rules for software specification discovery. IEEE Transaction on Knowledge and Data Engineering, 2011, 23(2): 282-296

[20] Rymon R. Search through systematic set enumeration// Proceedings of the 3rd International Conference on Principle of Knowledge Representation and Reasoning. Cambridge, USA, 1992: 539-550

[21] Thomas H C, Charles E L, Ronald L R, Clifford S. Introduction to Algorithms. 3rd Edition. London: The MIT Press, 2009

WANG Hui-Feng, born in 1987, M. S. candidate. His research interest is data mining.

DUAN Lei, born in 1981, Ph. D., associate professor. His research interests include data mining and health-informatics.

ZUO Jie, born in 1977, Ph. D., associate professor. His research interests include database and knowledge engineering.

WANG Wen-Tao, born in 1991, B. S. His research interest is data mining.

LI Zhong-Qi, born in 1991, Ph. D. candidate. His research interest is data mining.

TANG Chang-Jie, born in 1946, professor, Ph. D. supervisor. His research interests include database and knowledge engineering.

Background

Sequential pattern mining, which focuses on extracting knowledge from sequence data, has wide applications, such as outlying user behavior detection, medical diagnosis and economic prediction. Distinguishing sequential patterns describe differences between two or more sequence sets or classes. Mining of distinguishing sequential patterns with gap constraints is useful for identifying important features for discriminating one class of sequences from sequences of other classes.

In previous studies on distinguishing sequential patterns mining, a gap constraint indicating the maximum number and minimum number of wild-cards between elements in a pattern should be predefined by users. It is difficult for users to set suitable gap constraints without enough priori knowledge. As a result, useful patterns may be missed.

In this work, the authors design an algorithm for mining distinguishing sequential patterns without a predefined gap constraint. Specifically, the method enumerates all candidate patterns by a set enumeration tree. For each candidate pattern, the algorithm discovers the most suitable gap constraint such that the pattern satisfies the support thresholds. Experiments on real world datasets demonstrate that the proposed method is effective and efficient.