软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(8):1804−1815 [doi: 10.3724/SP.J.1001.2013.04422]

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

带通配符和 One-Off 条件的序列模式挖掘*

吴信东[1,2], 谢 飞[3], 黄咏明[1], 胡学钢[1], 高 隽[1]

[1](合肥工业大学 计算机与信息学院,安徽 合肥 230009)

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

[3](合肥师范学院 计算机科学与技术系,安徽 合肥 230601)

通讯作者: 吴信东, E-mail: xwu@uvm.edu, http://www.hfut.edu.cn

摘 要: 很多应用领域产生大量的序列数据.如何从这些序列数据中挖掘具有重要价值的模式,已成为序列模式挖掘研究的主要任务.研究这样一个问题:给定序列 *S*、支持度阈值和间隔约束,从序列 *S* 中挖掘所有出现次数不小于给定支持度阈值的频繁序列模式,并且要求模式中任意两个相邻元素在序列中的出现位置满足用户定义的间隔约束.设计了一种有效的带有通配符的模式挖掘算法 One-Off Mining,模式在序列中的出现满足 One-Off 条件,即模式的任意两次出现都不共享序列中同一位置的字符.在生物 DNA 序列上的实验结果表明,One-Off Mining 比相关的序列模式挖掘算法具有更好的时间性能和完备性.

关键词: 数据挖掘;序列模式挖掘;频繁模式;通配符;One-Off 条件

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 吴信东,谢飞,黄咏明,胡学钢,高隽.带通配符和 One-Off 条件的序列模式挖掘.软件学报,2013,24(8):1804−1815. http://www.jos.org.cn/1000-9825/4422.htm

英文引用格式: Wu XD, Xie F, Huang YM, Hu XG, Gao J. Mining sequential patterns with wildcards and the One-Off condition. Ruan Jian Xue Bao/Journal of Software, 2013,24(8):1804−1815 (in Chinese). http://www.jos.org.cn/1000-9825/4422.htm

Mining Sequential Patterns with Wildcards and the One-Off Condition

WU Xin-Dong[1,2], XIE Fei[3], HUANG Yong-Ming[1], HU Xue-Gang[1], GAO Jun[1]

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

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

[3](Department of Computer Science and Technology, Hefei Normal University, Hefei 230601, China)

Corresponding author: WU Xin-Dong, E-mail: xwu@uvm.edu, http://www.hfut.edu.cn

Abstract: There is a huge wealth of sequence data available in real-world applications. The task of sequential pattern mining serves to mine important patterns from the sequence data. Given a sequence *S*, a certain threshold, and gap constraints, this paper aims to discover frequent patterns whose supports in *S* are no less than the given threshold value. There are flexible wildcards in pattern *P*, and the number of the wildcards between any two successive elements of *P* fulfills the user-specified gap constraints. The study designs an efficient mining algorithm: One-Off Mining, whose mining process satisfies the One-Off condition under which each character in the given sequence can be used at most once in all occurrences of a pattern. Experiments on DNA sequences show that this method performs better in time and completeness than the related sequential pattern mining algorithms.

Key words: data mining; sequential pattern mining; frequent pattern; wildcard; One-Off condition

多个物种全基因组测序的完成,使得人们对基因序列进行分析,进而发现有趣的模式成为可能.很多植物和

* 基金项目: 国家自然科学基金(61229301, 60828005, 61273292); 美国国家科学基金(CCF-0905337, CCF-0514819); 国家高技术研究发展计划(863)(2012AA011005); 国家重点基础研究发展计划(973)(2013CB329604)

收稿时间: 2011-08-05; 修改时间: 2012-02-21, 2012-09-12; 定稿时间: 2013-04-28

动物的基因组中都含有大量重复出现的模式,某些重复模式具有重要的生物和医学方面的价值.研究表明,人类的很多疾病,如细菌病毒[1]等,都与基因中某部分的重复片段有关.然而,重复模式并不是简单地复制自己,它们在序列中每次出现的形式可能不一样,模式中相邻两个字符之间可能插入或删除较短的序列片段.因此,带有通配符的序列模式挖掘比传统的序列模式挖掘更具有重要的研究价值.

传统的序列模式挖掘是从事务数据集中挖掘出现频繁的模式.它是 Agrawal 等对超市数据进行分析时首先提出的[2].带通配符(wildcards)的序列模式挖掘是基于传统的模式挖掘问题上的提高.它允许挖掘出的序列模式中含有灵活的通配符,如下例所示:

例 1:传统的频繁模式挖掘算法从一个网络日志(该日志中的 *A*,*B*,*C*,*D* 分别指代不同的用户事件)中挖掘出 *ABACD* 模式;用通配符Φ代表 *A*,*B*,*C*,*D* 中任意一个事件,则带通配符的序列模式挖掘可能挖掘出 *A*ΦΦΦ*B*ΦΦ*A*ΦΦ*CD* 模式,该模式的意义是:发生 *A* 后,隔 3 个事件发生 *B*,再隔 2 个事件发生 *A*,再隔 2 个事件发生 *C*,随后发生 *D*.

可见,引入通配符虽然使问题变得复杂,但是模式的形式更加灵活,这使得带通配符的序列模式挖掘不仅具有理论研究价值,而且在生物信息学、文本索引、数据流挖掘等领域具有巨大的应用价值:它可以从生物序列中发现对生物学家有用的信息,也可以从网络日志、入侵检测、购物数据等数据集中发现更加灵活的模式.

我们用下面的例子来演示带通配符的序列模式挖掘的过程:

例 2:

- 问题:给定一个长度为 5 的序列 $S=s_0s_1s_2s_3s_4=ABCAC$,最小支持度阈值 *min_sup*=2,通配符约束范围 *g*=[0,2].也就是说,用户想找出出现次数不小于 2 的序列模式,并且模式中每相邻两个元素之间的通配符个数要在 0~2 之间.
- 过程:
  (1) 候选 1 项集:*A*,*B*,*C*.易见,频繁 1 项集:{*A*,*C*}.
  (2) 由频繁 1 项集生成候选 2 项集:{*AA*,*AC*,*CA*,*CC*}.*A*[0,2]*C* 在 *S* 中的出现位置有(0,2)和(3,4),支持度为 2≥*min_sup*,所以模式 *AC* 是频繁的;同样,可以计算出 *AA*,*CA*,*CC* 是非频繁的.因此,频繁 2 项集为{*AC*}.
  (3) 由频繁 2 项集生成候选 3 项集:{*ACA*,*ACC*},它们的支持度都为 1≤*min_sup*.无频繁 3 项集,结束.
- 结果:频繁模式集:{*A*,*C*,*AC*}.

本文中的模式支持度的计算满足 One-Off 条件:模式的任意两次出现都不共享序列中同一位置的字符[3].事实上,采取 One-Off 条件是较为合理的,因为在实际应用中,序列中的每个元素都代表一个事件或行为.上例中,*B* 事件本身只发生了 1 次,但是在统计模式 *BC*(即 *B* 事件发生 0~2 个事件后 *C* 发生)时,如果把 *B* 事件当成了两次发生,显然是不合理的.

综上所述,本文研究的问题是:给定序列 *S*、间隔约束 *g* 和最小支持度阈值 *min_sup*,从序列 *S* 中挖掘满足间隔约束的支持度不小于 *min_sup* 的频繁序列模式,要求模式的任意两次出现都不共享序列中同一位置的字符,即满足 One-Off 条件.挖掘的主要思路是:1) 扫描序列找出频繁 1 项集;2) 由频繁 *k*–1 项集生成候选 *k* 项集;3) 对每一个候选模式 *P*,在满足间隔约束和 One-Off 条件的情况下搜索序列,计算 *P* 的支持度 *sup*(*P*),若 *sup*(*P*)≥*min_sup*,则将 *P* 放入频繁集中;4) 重复步骤 2)和步骤 3),直至候选集为空.

将本文设计的算法应用于生物领域的 DNA 序列,可以发现满足用户要求的频繁模式,为相关研究者提供有价值的信息.我们在实际的 DNA 序列上做了充分的实验,与相关的序列模式挖掘算法比较,One-Off Mining 具有更好的时间性能和解的完备性.

本文第 1 节介绍相关的工作.第 2 节给出问题定义.第 3 节详细描述算法的步骤,并进行时间和空间复杂度分析.第 4 节给出实验结果及算法的对比分析.最后,总结工作.

1 相关工作

序列模式挖掘问题最早由 Agrawal 和 Srikant 在 1995 年提出来[2],随后,一些改进的算法相继被提出来[4−7].邹祥等人对分布式环境下的序列模式挖掘算法进行了研究[8].传统的序列模式挖掘算法是从序列数据库中挖掘频繁出现的模式,没有定义通配符约束条件.模式的支持度定义是:序列数据库中包含该模式的序列个数.

Ji 等人[9]和 Li 等人[10]研究了序列数据库中带有通配符的模式挖掘问题.文献[9]中提出了带有通配符的最小区分模式概念,即在正类别序列(positive sequence)中出现频繁,且在负类别序列(negative sequence)中出现不频繁的最小子序列模式.间隔约束条件定义为 *g*-gap,即允许的最大通配符个数.文献[10]中对带有通配符的闭合模式挖掘问题进行了研究.

序列模式挖掘研究的另一个焦点问题是从单序列中挖掘频繁模式,序列通常比较长,例如 DNA 序列和蛋白质序列等.Zhang 等人对单序列中带有通配符的模式挖掘问题进行了研究,提出了 MPP 算法[11].文献[11]中,模式的支持度定义为模式在序列中出现的次数.

例 3:例 2 中,MPP 算法定义的 $B[0,2]C$ 的出现位置为(1,2)和(1,4),支持度为 2,而 B 的支持度为 1.所以在 MPP 算法中,BC 也是频繁的,而它的子模式 B 不是频繁的.

这样,文献[11]中的问题定义不再满足 Apriori 性质.MPP 算法中虽然使用了一个剪枝定理,但是候选集的个数仍然很多,导致该算法在处理大规模数据时效率不够理想.

Zhu 等人的 MCPaS 算法[12]是从多序列中挖掘频繁模式.它和 MPP 算法一样,也允许模式的重复出现.该算法在模式搜索和候选项剪枝上做了一些改进,但是它仍然没有摆脱存在大量候选模式的问题.

He 等人[13]对单序列中序列模式挖掘问题进行研究,模式的出现满足 One-Off 条件,使用 One-Way Scan 和 Two-Way Scan 两种搜索策略计数模式的支持度,取它们的最大值作为模式的支持度.但是,He 等人[13]问题定义中没有对通配符范围进行限制,导致挖掘出来的频繁模式不满足特定的间隔约束.

Xie 等人[14]和 Huang 等人[15]对带有通配符的序列模式挖掘问题进行了研究,模式中含有灵活的通配符约束和 One-Off 条件.文献[14]中提出了一种基于模式增长的 MAIL 算法,利用前缀模式的出现信息,构造扩展模式的出现空间,同时采用最左优先和最右优先两种剪枝策略,提高算法的时间效率和解的完备性.本文在文献[15]中的模式挖掘框架基础上,提出基于前向搜索和后向寻找解的模式支持度的计算方法,进一步提高了解的完备性.

Ding 等人[16]提出从序列数据库中挖掘带通配符的频繁所有模式和闭合模式,模式的出现满足 Non-Overlap 条件,挖掘的过程满足 Apriori 性质.Non-Overlap 条件与我们定义的 One-Off 条件非常相似,但存在着本质的区别.假设 $occ_1=(l_1,l_2,\ldots,l_m)$ 和 $occ_2=(l'_1,l'_2,...,l'_m)$ 是长度为 m 的模式 P 的两个出现,如果对任意的 $1\leqslant i\leqslant m$,都有 $l_i\neq l'_i$,则称 occ_1 和 occ_2 满足 Non-Overlap 条件.

表 1 是上述相关序列模式挖掘算法问题定义的性质比较.

Table 1 Property comparisons of the related problem definitions

表 1 相关问题定义的性质比较

算法	输入		模式搜索	挖掘过程	挖掘目标
	数据集	是否定义通配符范围	出现位置	是否满足 Apriori 性质	频繁模式
Agrawal, *et al*.[3]	序列数据库	否	含有模式的序列数	是	所有
Li, *et al*.[10]	序列数据库	是	含有模式的序列数	是	闭合
Ji, *et al*.[9]	两类序列	是	含有模式的序列数	否	最小区别
Zhang, *et al*.[11]	单序列	是	所有出现	否	所有
Zhu, *et al*.[12]	多序列	是	所有出现	否	所有
He, *et al*.[13]	单序列	否	One-Off 条件	是	所有
Ding, *et al*.[16]	序列数据库	否	Non-Overlap 条件	是	所有、闭合
本文; Xie, *et al*.[14]; Huang, *et al*.[15]	单序列	是	One-Off 条件	是	所有

2 问题定义

从给定的序列中挖掘带通配符的频繁模式,我们称给定的序列为目标序列.长度为 n 的目标序列 S 记为 $S=s_1s_2\ldots s_n$.字符表Σ包含序列 S 中不相同的字符.例如,DNA 序列的字符表为$\Sigma=\{A,C,G,T\}$.

通配符可以代表字符集中任意字符,记为Φ.间隔约束 $g=[N,M]$表示通配符个数的范围.$W=M-N+1$ 为间隔灵活度.

模式 $P=p_1g_1p_2g_2\ldots g_{m-1}p_m$ 是由字符表中字符和通配符构成的序列,$p_i(1\leqslant i\leqslant m)$是字符表中的字符,$g_i(1\leqslant i<m)$是字符 p_i 和 p_{i+1}之间的通配符范围,即间隔约束.本文中,所有的间隔约束相同,即 $g_1=g_2=\ldots=g_{m-1}=[N,M]$.模式 P 中含有字符表中字符的个数称为模式 P 的长度,记为$|P|$,不考虑通配符的个数.模式 P 也可简记为 $P=p_1p_2\ldots p_m$.

定义 1. 如果存在一个位置序列 $1\leqslant i_1<i_2<\ldots<i_m\leqslant n$,满足条件,则称$(i_1,i_2,\ldots,i_m)$为模式 P 在序列 S 中满足间隔约束 $g=[N,M]$的一次出现:

- $s_{i_j}=p_j$,其中,$1\leqslant j\leqslant m$;
- $N\leqslant i_j-i_{j-1}-1\leqslant M$,其中,$2\leqslant j\leqslant m$.

定义 2. 假定 $occ_1=(i_1,i_2,\ldots,i_m)$和 $occ_2=(j_1,j_2,\ldots,j_m)$为模式 P 的两次出现,若对任意的 $1\leqslant p,q\leqslant m$,都有 $i_p\neq j_q$,则称 occ_1 和 occ_2 满足 One-Off 条件.

定义 3. 模式 P 在序列 S 中出现的次数称为 P 的支持度,并且任意两次出现都满足 One-Off 条件.若 P 的支持度大于给定的支持度阈值 min_sup,则称 P 为频繁模式.

例 4:$S=AAGCCC$,$P=AC$,间隔约束 $g=[2,3]$,支持度阈值 $min\_sup=3$.满足 One-Off条件的 P 的出现为(0,3),(1,4).P 的支持度为 $Sup(P)=2\leqslant min\_sup$,所以 P 是非频繁的.

定义 4. 设 I 是模式 P 在 S 中满足间隔约束 g 的一组出现的集合,I 中的任意两次出现都满足 One-Off条件,$|I|$是集合中出现的个数,若不存在模式 P 的另一组满足间隔约束和 One-Off 条件的出现集合 I',使得$|I'|>|I|$,则称 I 是模式 P 在 S 中匹配的完备解,$|I|$是匹配的最大出现次数.

给定一个目标序列 S、间隔约束 $g=[N,M]$、最小支持度阈值 min_sup,带有通配符的序列模式挖掘问题是从 S 中挖掘满足最小支持度的频繁模式,若模式的支持度是模式在序列中匹配的最大出现次数,则称挖掘的结果是完备解.模式挖掘算法的完备性是指挖掘的结果接近完备解的程度.

定义 5. 设 $P=p_1p_2\ldots p_m$ 和 $Q=q_1q_2\ldots q_t$ 为两个带有通配符的模式(简写),如果 $m\leqslant t$,并且存在位置序列 $1\leqslant j_1<j_2<\ldots<j_m\leqslant t$,对任意的 $1\leqslant k\leqslant m$,满足条件 $p_k=q_{j_k}$,则称 P 为 Q 的子模式,Q 为 P 的父模式.

定义 6. 在定义 5 中,如果对任意的 $2\leqslant k\leqslant m$ 都有 $j_k=j_{k-1}+1$,则称 P 为 Q 的连续子模式,Q 为 P 的连续父模式.

定理 1(Apriori 性质). 给定序列 S,带有通配符的模式 $P=p_1p_2\ldots p_m$ 和 $Q=q_1q_2\ldots q_t$,$Sup(P)$和 $Sup(Q)$分别是 P 和 Q 在 S 中满足 One-Off 条件和间隔约束 g 的支持度,如果 P 是 Q 的连续子模式,则有 $Sup(P)\geqslant Sup(Q)$.

证明:因为 P 是 Q 的连续子模式,不妨设 $p_1=q_i(2\leqslant i\leqslant t-m+1)$,并且对于任意的 $2\leqslant j\leqslant m$,都有 $p_j=q_{i+j-1}$.假设 $occ'=(l_1,\ldots,l_i,\ldots,l_{i+m-1},\ldots,l_t)$为模式 Q 在 S 中满足间隔约束 g 的一个出现,显然,$occ=(l_i,\ldots,l_{i+m-1})$也是 P 在 S 中满足间隔约束 g 的一个出现.

同时,由于 One-Off 条件限制,模式 Q 的任意两个出现$(l_1,\ldots,l_i,\ldots,l_{i+m-1},\ldots,l_t)$和 $(l'_1,\ldots,l'_i,\ldots,l'_{i+m-1},\ldots,l_t)$ 都不含有相同的位置,则 $(l_i,\ldots,l_{i+m-1})\neq(l'_i,\ldots l'_{i+m-1})$.

所以,$Sup(P)\geqslant Sup(Q)$. □

3 算法设计与分析

3.1 算法描述

算法 1 给出了 One-Off Mining 算法框架.首先,扫描序列 S,找出所有长度为 1 的频繁模式,形成频繁 1 项集 L_1(第 2 行~第 6 行).任何时候,由长度为 $k-1$ 的频繁模式集 L_{k-1} 中的模式进行连接,生成长度为 k 的候选模式集

C_k(第 8 行).模式 $P=p_1p_2\ldots p_{k-1}$ 和 $Q=q_1q_2\ldots q_{k-1}$,若 $p_2=q_1,p_3=q_2,\ldots,p_{k-1}=q_{k-2}$,则由 P 和 Q 进行连接,生成模式 $R=p_1p_2\ldots p_{k-1}q_{k-1}$.计算候选模式集 C_k 中每个模式的支持度(第 10 行),得到 k 项频繁模式集 L_k(第 11 行、第 12 行).依次进行,直到某项频繁模式集为空.

算法 1. *One-Off Mining*(*S*,*N*,*M*,*min_sup*).

输入:目标序列 *S*、最小间隔 *N*、最大间隔 *M*、最小支持度 *min_sup*.

输出:频繁模式集 *FP*.

```
1:   FP=∅;
2:   C1=长度为 1 的模式集;
3:   for each P∈C1 do
4:      support=CalcSup(S,P,N,M);
5:      if (support≥min_sup)
6:          L1=L1∪{P};
7:   for (k=2; Lk−1≠null; k++)
8:          Ck=Gen(Lk−1);   //长度为 k–1 的频繁模式连接,生成长度为 k 的候选模式
9:      for each P∈Ck do
10:         support=CalcSup(S,P,N,M);
11:     if (support≥min_sup)
12:         Lk=Lk∪{P};
13:  return (FP=L1∪L2∪…)
```

算法 2 是计算模式支持度的算法描述.初始化使用标记数组 *used*,序列中每个位置使用标记设为 *false*(第 2 行).寻找模式的第 1 个字符匹配的位置(第 4 行).任何时候,根据模式中前一个字符匹配的位置以及间隔约束,寻找模式中当前字符匹配的位置,直至找到模式中最后一个字符匹配的位置(第 7 行~第 15 行),当前找到的模式出现的匹配位置保存在数组 *occ* 中.

算法 2. *CalcSup*(*S*,*P*,*N*,*M*).

输入:序列 $S=s[0]s[1]\ldots s[n-1]$、模式 $P=p[0]p[1]\ldots p[m-1]$、最小间隔 *N*、最大间隔 *M*.

输出:模式 *P* 的支持度.

```
1:  sup=0;
2:  初始化 used 数组,S 中每个位置标记为 false;
3:  for i=0 to n−1 do
4:     if (s[i]==p[0] && used[i]==false)
5:         pos=i;   //出现的起始位置
6:         初始化 occ 数组,移除数组中所有元素;
7:         occ.add(pos);
8:         for j=1 to m−1 do
9:            flag=false;
10:           for k=pos+N+1 to pos+M+1
11:              if (s[k]==p[j] && used[k]==false)
12:                    occ.add(k);
13:                    flag=true;
14:                    pos=k;
15:                    break;
16:        if (!flag) break;
```

```
17:     if (flag)
18:         sup++;
19:         将 occ 中所有位置的使用标记设为 true;
20:   return sup;
```

下面举例说明计算模式的支持度的过程.

例 5:目标序列 S=ccccbabbbfadacbbabeacc(n=22),模式 P=babac(m=5),通配符约束 g=[0,3].算法 2 的搜索过程见表 2:p[0]=s[5],则 $start$=5.根据模式中前一个字符匹配的位置和间隔约束,寻找当前字符匹配的位置.例如,p[1]=s[6],S 中满足通配符约束的位置 7~位置 9 都能和 p[2]匹配,选择最左边的位置 7 作为 p[2]的位置.当搜索到 s[14]时,模式 P 的最后一个字符匹配成功.此时,找到了模式 P 的一个出现 occ_1=(5,6,7,11,14),将 occ_1 中的所有位置使用标记设为 $true$,继续寻找下一个 $start$=15,可以找到另一个出现 occ_2=(15,17,18,20,21).所以,模式 P 的支持度为 2.

Table 2 Search table of algorithm 2

表 2 算法 2 的搜索表

		5	6	7	8	9	10	11	12	13	14
		b	*a*	*b*	*b*	*b*	*f*	*a*	*d*	*a*	*c*
0	*b*	√									
1	*a*		√								
2	*b*			√							
3	*a*							√			
4	*c*										√

在上述的例子中,若将间隔约束改为 g=[0,2],则算法 2 只能计算得到模式 P 的一个出现(15,17,18,20,21),丢失了解(5,6,7,11,14).这是因为当搜索到 s[7]时,和 p[2]进行匹配;但是继续向前搜索时,由于间隔约束是[0,2],在扩展区间[8,10]内不存在和 p[3]匹配的位置.因此,我们对算法 2 进行了改进,模式 P 的支持度计算过程分为两个阶段:前向搜索匹配位置和后向寻找模式的解.在前向搜索过程中,根据前一个字符匹配的位置以及间隔约束,寻找当前字符匹配的所有可能位置,并保存在二维数组 $posArr$ 中,直至找到模式的最后一个字符匹配的可能位置;在后向寻找解的过程中,从模式的最后一个字符开始,采用最左优先策略,在二维数组 $posArr$ 中选择模式的每个字符匹配的位置.改进后的模式支持度的计算过程如算法 3 所示.

算法 3. *i-CalcSup*(*S*,*P*,*N*,*M*).

```
…
6:              posArr[0].add(start);
7:              for j=0 to m−2 do
8:                  flag=false;
9:                  for k=0 to posArr[j].size−1 do
10:                     pos=posArr[j].get(k);
11:                     for kk=pos+N+1 to pos+M+1
12:                         if (s[kk]==p[j+1] && used[kk]==false)
13:                             posArr[j+1].add(kk);
14:                             flag=true;
15:                 if (!flag) break;
16:       if (flag)
17:           sup++;
18:           采用最左优先策略,在二维数组 posArr 中选择 P 的一个出现,并更新 used 数组;
19:   return sup;
```

表 3 是改进后的算法 3 的搜索过程表,间隔约束 g=[0,2].可以看出,在反向寻找解的过程中,找到了模式 P 的

出现(5,6,8,11,14)(在表 3 中用下划线对找出的出现位置做了标记).

Table 3 Search table of algorithm 3

表 3 算法 3 的搜索表

		5	6	7	8	9	10	11	12	13	14
		b	*a*	*b*	*b*	*b*	*f*	*a*	*d*	*a*	*c*
0	*b*	√									
1	*a*		√								
2	*b*			√	√	√					
3	*a*							√			
4	*c*										√

3.2 算法复杂度分析

(1) 时间复杂度分析

假设序列 S 的长度$|S|=n$,模式 P 的长度$|P|=m$,模式的首字符在序列中出现次数为 k,模式中的字符在序列中出现的最大次数为 l,间隔约束 $g=[N,M]$,间隔灵活度 $w=M-N+1$.首先分析计算模式支持度的时间复杂度.在算法 2 中,从模式的首字符的一个匹配位置开始,在其扩展区间内搜索模式的每个字符匹配的最左边位置,依次进行,直至找到模式的最后一个字符匹配的位置,需要的时间复杂度是 $O(mw)$.因此,算法 2 的时间复杂度为 $O(kmw)$.在算法 3 中,从模式的首字符的一个匹配位置开始,寻找模式的一次出现分为前向搜索匹配位置和后向寻找解两个阶段:前向搜索过程需要保留模式中每个字符的所有可能的匹配位置,则前向搜索的时间复杂度为 $O(lmw)$;后向寻找解过程需要判断每个可能的匹配位置,时间复杂度为 $O(lm)$.因此,算法 3 的时间复杂度为 $O(klmw)$.

所以,One-Off Mining 在采用算法 2 和算法 3 计算模式的支持度时的时间复杂度分别为 $O(gkmw)$和 $O(gklmw)$.其中,g 是频繁模式的个数.

(2) 空间复杂度分析

采用算法 2 计算模式的支持度时,需要 $O(n)$的空间存储序列中字符的使用标记和 $O(m)$的空间记录模式中每个字符匹配的最左位置.采用算法 3 计算模式的支持度时,需要存储模式中每个字符的所有可能的匹配位置,存储所有字符的匹配位置需要 $O(ml)$的空间;另外,需要 $O(n)$的空间存储序列中每个字符的使用标记.因此,One-Off Mining 在采用算法 2 和算法 3 计算模式的支持度时的空间复杂度分别为 $O(n+m)$和 $O(n+ml)$.

4 实验及分析

实验数据是从生物信息中心网站[17]下载的人类 DNA 序列(AX829170,AX829174,AX829178),其中,AX829170 的长度为 3 737,AX829174 的长度为 10 011,AX829178 的长度为 5 393,随机地选取长度为 L 的子序列作为目标序列.所有算法是在 Pentium Dual CPU 3.0GHz,内存 2GB 的计算机上,使用 JAVA 开发环境实现的.One-Off Mining 算法在采用算法 2 和算法 3 计算模式的支持度时,分别记为 OFMI 和 i-OFMI.

实验 1. 在本实验中,比较 OFMI 和 i-OFMI 挖掘的模式个数和时间性能.使用的 DNA 序列是 AX829170,间隔约束 $g=[9,12]$.目标序列的长度从 1 000 变化至 3 737,支持度阈值为 20,40,60,70.

图 1(a)是 OFMI 和 i-OFMI 挖掘的模式个数比较.可以看出,i-OFMI 能够比 OFMI 挖掘更多的模式,平均情况下,i-OFMI 挖掘的模式个数是 OFMI 挖掘模式个数的 3.45 倍.这说明 i-OFMI 算法的完备性能优于 OFMI.在 i-OFMI 算法中,模式的支持度计算过程包括两个阶段:在前向搜索匹配位置时,保留了模式中每个字符的多个可能的匹配位置;在后向寻找模式的解时,采用最左优先策略,在满足间隔约束的多个可能的匹配位置中选择最左边的那个.这样,可以把后面的匹配位置让给后续的解匹配.OFMI 算法采用一遍扫描计算模式的支持度,在搜索过程中,只选择最左边的匹配位置,不考虑其他可能的匹配位置.如果最左边的匹配位置不能继续向后扩展,而其他匹配位置能够扩展,在这种情况下,OFMI 就会丢失解.图 1(a)中,i-OFMI 的曲线在序列长度是 3 000 时出现了折点,而 OFMI 则比较平缓.通过查看 AX829170 序列发现,在字符 3 000~3 737 这段重复出现很密集,这说明

i-OFMI 比 OFMI 更适合于找出嵌套出现的解.

由于 i-OFMI 能够比 OFMI 挖掘更多的模式,因此,i-OFMI 的时间性能差于 OFMI,如图 1(b)所示,在平均情况下,i-OFMI 算法运行的时间是 OFMI 算法的 11.38 倍.从图 1(b)中还可以看出,随着目标序列长度的增加,i-OFMI 和 OFMI 的时间性能都会随之下降.

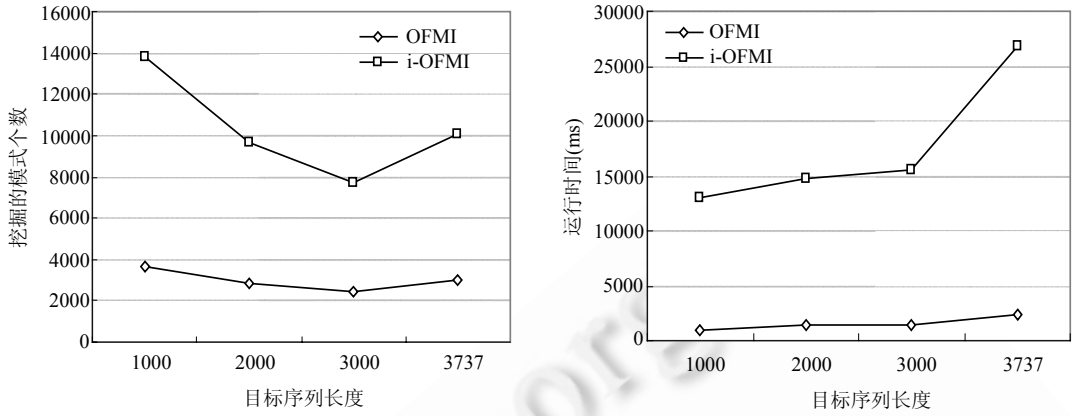

Fig.1 Mining pattern number and running time performance comparisons of OFMI and i-OFMI

图 1 OFMI 和 i-OFMI 挖掘模式个数和运行时间性能比较

实验 2. 在本实验中,研究不同的间隔灵活度对算法挖掘的模式个数和时间性能的影响.从 DNA 序列 AX829178 中选取长度为 1 000 的子序列作为目标序列,最小支持度阈值 *min_sup* 设置为 20,最大间隔 *M* 固定为 12,间隔灵活度 *W* 从 4 变化至 7,最小间隔 *N*=*M*–*W*+1.图 2 是在不同间隔灵活度下,OFMI 算法(采用算法 2 计算模式的支持度)挖掘的模式个数和需要运行的时间.可以看出,随着间隔宽度的增加,OFMI 挖掘出更多的模式,同时也消耗更多的时间.这是因为,间隔灵活度实际上反映了允许模式中的字符在序列中的匹配范围,增加了间隔的灵活度,等于加大了模式中字符匹配的可能性,这样,模式出现的次数也会增加,在最小支持度阈值保持不变的情况下,挖掘模式的个数变多了;另一方面,随着间隔宽度的增加,需要更多的时间在更大的间隔约束范围内寻找模式中字符的匹配位置,所以算法的运行时间也会增加.

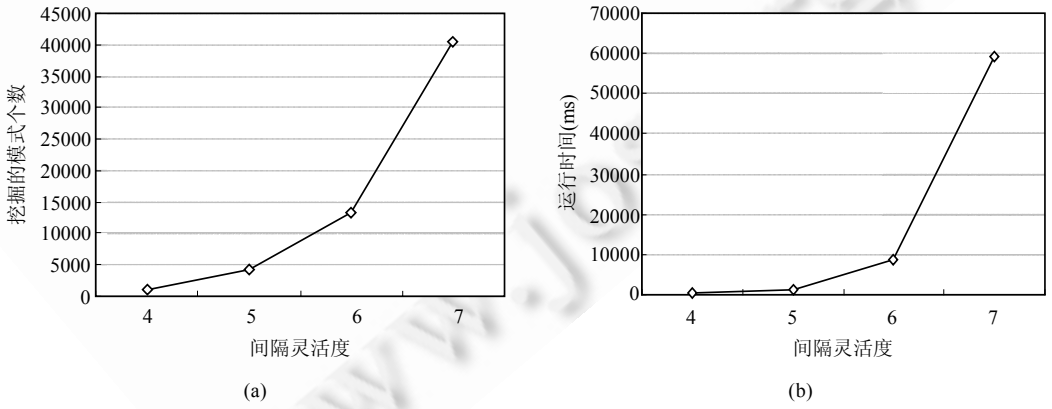

Fig.2 Mining pattern numbers and running time of OFMI under different gap constraints

图 2 不同间隔灵活度下,OFMI 挖掘的模式个数和运行的时间

实验 3. 在本实验中,比较本文提出的 One-Off Mining 算法和单序列中带有通配符的模式挖掘算法 MPP[11].

从 AX829174 序列中任意选取长度为 1 000 的子序列作为目标序列,通配符间隔约束 g=[9,12],支持度阈值 min_sup=0.013×|S|=13,其中,|S|为目标序列的长度.MPP 算法的支持度阈值ρ=0.03%(在 MPP 算法中,支持度定义为模式在序列中的实际出现次数和可能的出现次数的比例).表 4 分别列出了 OFMI(采用算法 2 计算模式的支持度)和 MPP 挖掘的频繁模式个数和算法运行的时间.

Table 4 Mining pattern number and running time comparisons of OFMI and MPP

表 4 OFMI 与 MPP 挖掘的模式个数和运行时间比较

|P|	3	4	5	6	7	8	9	All	Time (ms)
MPP	64	256	1 021	955	86	15	0	2 397	7 703
OFMI	64	256	1 016	2 752	1 460	136	3	5 687	2 185

从表 4 可以看出,OFMI 算法在挖掘的频繁模式的个数是 MPP 算法挖掘的频繁模式个数的 2.37 倍的情况下,时间性能比 MPP 提高了 3.53 倍.

为了进一步比较 One-Off Mining 和 MPP 挖掘出的模式,我们用模式相似度比较两种算法挖掘出的模式的一致性情况:

$$similarity = \frac{\#common}{\#average} \tag{1}$$

其中,*#common* 表示两种算法挖掘出相同模式的个数,*#average* 表示两种算法挖掘的模式个数的平均值.

选取 AX829174 的长度为 1 000~5 000 的子序列作为目标序列,通配符约束为[9,12].One-Off Mining 中采用算法 3 计算模式的支持度,通过调整 i-OFMI 的支持度阈值,使得 i-OFMI 和 MPP 挖掘的模式个数相近.

表 5 是参数设置以及挖掘的模式个数和模式相似性情况,其中,min_sup 是 i-OFMI 算法设置的支持度阈值,ρ 是 MPP 算法设置的支持度阈值.可以看出,i-OFMI 算法和 MPP 算法挖掘模式相似性达到 75%以上,并且随着序列长度的增加,模式的相似性也增加.

Table 5 Mining pattern numbers and similarities of i-OFMI and MPP

表 5 i-OFMI 和 MPP 挖掘的模式个数和相似性

	|S|=1000, min_sup=21, ρ=0.03%	|S|=2000, min_sup=47, ρ=0.03%	|S|=3000, min_sup=76, ρ=0.03%	|S|=4000, min_sup=101, ρ=0.03%	|S|=5000, min_sup=130, ρ=0.03%
MPP	2 397	2 267	2 251	2 159	2 160
i-OFMI	2 448	2 327	2 233	2 204	2 153
Similarity (%)	74.84	76.00	76.51	79.39	80.15

图 3 是 i-OFMI 和 MPP 的时间性能比较.

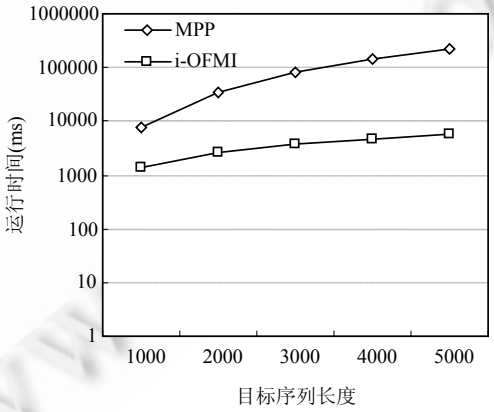

Fig.3 Running time comparisons of i-OFMI and MPP

图 3 i-OFMI 与 MPP 运行时间比较

可以看出:在挖掘模式数目相接近的情况下,i-OFMI 算法的时间性能比 MPP 算法平均提高了 22 倍;并且随着序列长度的增加,MPP 消耗的时间呈指数增长,而 i-OFMI 算法消耗的时间增长非常缓慢.主要原因是:MPP 算法中对“预候选项集”的判断、生成以及计算模式的所有出现,导致其采用的 PIL 数据结构的长度很大,增加了查找模式位置和计算支持度的时间消耗;而 i-OFMI 算法挖掘的模式的出现满足 One-Off 条件,使得挖掘过程满足 Apriori 性质,并且每个模式的出现次数在很大程度上减少了.

实验 4. 在本实验中,比较本文提出的 One-Off Mining 算法和 MAIL 算法[14].MAIL 算法中采用的是最右优先的剪枝策略.从 AX829174 中分别选取长度为 1 000~5 000 的子序列作为目标序列,通配符约束 g=[9,12],支持度阈值 min_sup=0.015×|S|.图 4(a)是 i-OFMI,OFMI 和 MAIL 挖掘的模式个数比较,可以看出,i-OFMI 挖掘的模式个数最多,其次是 MAIL,OFMI 挖掘的模式个数最少.这说明 i-OFMI 的完备性优于 MAIL,MAIL 的完备性优于 OFMI.主要原因是:OFMI 在寻找模式中字符的匹配位置时,仅在扩展区间中选择最左边的那个匹配位置;MAIL 算法在保留尽可能多的候选解的同时,采用剪枝策略来提高算法的时间效率,因此损失了一部分解;i-MAIL 在扩展过程中保留了模式中字符的所有可能的匹配位置,在寻找解的过程中采用最左优先策略获取模式的优解,因此完备性最好.i-OFMI,OFMI 和 MAIL 的时间性能比较正好相反,OFMI 的时间性能最好,其次是 MAIL,i-OFMI 消耗的时间最多,如图 4(b)所示.

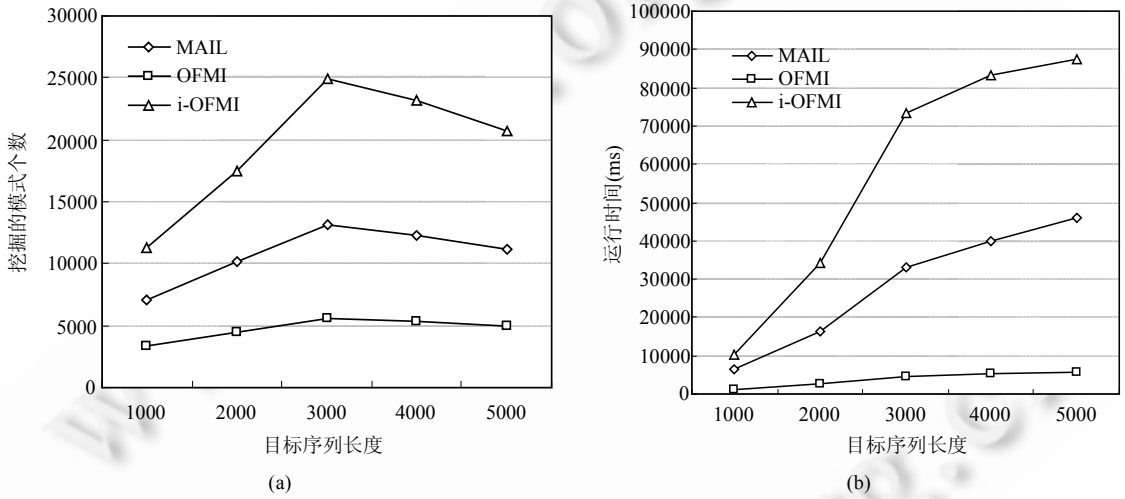

Fig.4 Performance comparisons of i-OFMI, OFMI and MAIL

图 4 i-OFMI,OFMI 和 MAIL 算法性能比较

实验 5. He 等人[13]的工作是首次将 One-Off 条件引入带有通配符的模式挖掘中,因此,我们也将 One-Off Mining 算法和文献[13]中提出的 One-Way Scan 算法和 Two-Way Scan 算法进行了比较.文献[13]中没有对通配符的范围进行具体的限制,为了能够更好地进行比较,我们对 One-Way Scan 和 Two-Way Scan 进行了修改:在计算模式 P 的支持度时,检测模式的每个出现的间隔约束,只考虑那些满足设定的间隔约束的出现.表 6 是 One-Way Scan,Two-Way Scan 和 OFMI(采用算法 2 计算模式的支持度)这 3 种算法挖掘的模式个数比较,目标序列选取的是 AX829174 的长度|S|为 1 000,2 000,3 000,4 000 和 5 000 的子序列,间隔宽度 g=[9,12],最小支持度 min_sup=0.013×|S|.

Table 6 Mining pattern number comparisons of OFMI, One-Way Scan and Two-Way Scan

表 6 OFMI,One-Way Scan 和 Two-Way Scan 挖掘的模式个数比较

	|S|=1000	|S|=2000	|S|=3000	|S|=4000	|S|=5000
One-Way Scan	749	951	1 034	923	766
Two-Way Scan	663	937	964	736	646
OFMI	5 687	7 436	9 623	8 882	8 168

从表 6 可以看出,平均情况下,One-Way Scan 和 Two-Way Scan 挖掘的带有间隔约束的模式个数只有 OFMI 挖掘的模式个数的 10%左右.这说明在挖掘带有具体通配符限定的模式时,不能在原有的带有任意通配符的模式挖掘算法的基础上,通过检测模式每个出现的间隔约束得到带有具体通配符的模式,而必须设计一种全新的算法,这正是本文工作的重点.从表 6 还可以看出,One-Way Scan 挖掘的模式个数比 Two-Way Scan 挖掘的模式个数多.这是因为在 Two-Way Scan 算法中,采用的最左优先和最右优先结合的扫描方式,使得模式的出现间隔过大,不满足用户定义的间隔约束要求.图 5 是 One-Way Scan 和 OFMI 的时间性能比较,平均情况下,OFMI 消耗的时间是 One-Way Scan 的 2 倍.

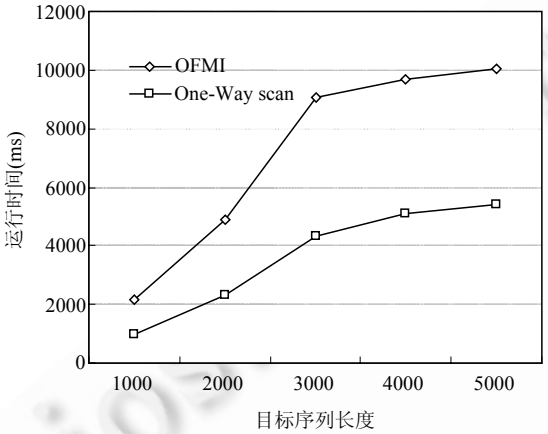

Fig.5 Running time comparisons of OFMI and One-Way Scan

图 5 OFMI 和 One-Way Scan 运行时间比较

5 结 论

本文研究了带有通配符约束的序列模式挖掘问题,用户可以定义灵活的通配符约束,模式的任意两个出现都不共享序列中同一位置的字符,使得问题定义在实际应用中更加合理.提出了一种快速、有效的带有通配符的序列模式挖掘算法 One-Off Mining,设计了两种模式支持度的计算方法,讨论了不同的支持度计算方法对算法的时间性能和解的完备性的影响.实验结果表明,与相关的序列模式挖掘算法相比,One-Off Mining 具有更好的时间性能和解的完备性.

References:

[1] van Belkum A, Scherer S, van Leeuwen W, Willemse W, van Alphen L, Verbrugh H. Variable number of tandem repeats in clinical strains of haemophilus influenzae. Infection and Immunity, 1997,65(12):5017−5027.

[2] Agrawal R, Srikant R. Mining sequential patterns. In: Yu PS, Chen ALP, eds. Proc. of the Int'l Conf. on Data Engineering (ICDE'95). Los Alamitos: IEEE Computer Society, 1995. 3−14. [doi: 10.1109/ICDE.1995.380415]

[3] Chen G, Wu X, Zhu X, Arslan A, He Y. Efficient string matching with wildcards and length constraints. Knowledge and Information Systems, 2006,10(4):399−419. [doi: 10.1007/s10115-006-0016-8]

[4] Srikant R, Agrawal R. Mining sequential patterns: Generalizations and performance improvements. In: Apers P, Bouzeghoub M, Gardarin G, eds. Proc. of the Int'l Conf. on Extending Database Technology (EDBT'96). LNCS 1057, Heidelberg: Springer-Verlag, 1996. 13−17. [doi: 10.1007/BFb0014140]

[5] Pei J, Han J, Mortazavi-Asl B, Chen Q, Dayal U, Hsu MC. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Georgakopoulos D, Buchmann A, eds. Proc. of the Int'l Conf. on Data Engineering (ICDE 2001). Los Alamitos: IEEE Computer Society, 2001. 215−224. [doi: 10.1109/ICDE.2001.914830]

[6] Zaki MJ. SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, 2001,42(1):31−60. [doi: 10.1023/A:1007652502315]

[7] Ayres J, Gehrke J, Yiu T, Flannick J. Sequential pattern mining using a bitmap representation. In: Simoff SJ, Za-ane OR, eds. Proc. of the ACM SIGKDD. New York: ACM Press, 2002. 429−435. [doi: 10.1145/775047.775109]

[8] Zou X, Zhang W, Liu Y, Cai QS. Study on distributed sequential pattern discovery algorithm. Ruan Jian Xue Bao/Journal of Software, 2005,16(7):1262−1269 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1262.htm [doi: 10.1360/jos161262]

[9] Ji XN, Bailey J, Dong GZ. Mining minimal distinguishing subsequence patterns with gap constraints. In: Han JW, Wah BW, Raghavan V, Wu XD, Rastogi R, eds. Proc. of the IEEE Int'l Conf. on Data Mining (ICDM 2005). Los Alamitos: IEEE Computer Society, 2005. 194−201. [doi: 10.1109/ICDM.2005.96]

[10] Li C, Wang J. Efficiently mining closed subsequences with gap constraints. In: Proc. of the SIAM Int'l Conf. on Data Mining. Philadelphia: Society for Industrial Mathematics, 2008. 313−322.

[11] Zhang M, Kao B, Cheung D, Yip K. Mining periodic patterns with gap requirement from sequences. In: Ozcan F, ed. Proc. of the ACM SIGMOD. New York: ACM Press, 2005. 623−633.

[12] Zhu XQ, Wu XD. Mining complex patterns across sequences with gap requirements. In: Veloso MM, ed. Proc. of the Int'l Joint Conf. on Artificial Intelligence (IJCAI 2007). Menlo Park: AAAI Press, 2007. 726−735.

[13] He Y, Wu X, Zhu X, Arslan AN. Mining frequent patterns with wildcards from biological sequences. In: Chang, W, Joshi, JBD, eds. Proc. of the IEEE Int'l Conf. on Information Reuse and Integration (IRI 2007). Los Alamitos: IEEE Computer Society, 2007. 329−334. [doi: 10.1109/IRI.2007.4296642]

[14] Xie F, Wu X, Hu X, Gao J, Guo D, Fei Y, Hua E. Sequential pattern mining with wildcards. In: Gregoire E, ed. Proc. of the 22nd Int'l Conf. on Tools with Artificial Intelligence (ICTAI 2010). Los Alamitos: IEEE Computer Society, 2010. 241−247. [doi: 10.1109/ICTAI.2010.42]

[15] Huang Y, Wu X, Hu X, Xie F, Gao J, Wu G. Mining frequent patterns with gaps and One-Off condition. In: Muzio JC, Brent RP, eds. Proc. of the 12th IEEE Int'l Conf. on Computational Science and Engineering (CSE 2009). New York: IEEE Press, 2009. 180−186. [doi: 10.1109/CSE.2009.160]

[16] Ding B, Lo D, Han J, Khoo S C. Efficient mining of closed repetitive gapped subsequences from a sequence database. In: Ioannidis YE, Lee DL, Ng RT, eds. Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Los Alamitos: IEEE Computer Society, 2009. 1024−1035. [doi: 10.1109/ICDE.2009.104]

[17] NCBI. http://www.ncbi.nlm.nih.gov

附中文参考文献:

[8] 邹祥,张巍,刘洋,蔡庆生.分布式序列模式发现算法的研究.软件学报,2005,16(7):1262−1269. http://www.jos.org.cn/1000-9825/16/1262.htm [doi: 10.1360/jos161262]

吴信东(1963－),男,安徽枞阳人,博士,教授,博士生导师,主要研究领域为数据挖掘,专家系统,万维网信息处理.
E-mail: xwu@uvm.edu

胡学钢(1961－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为知识工程,数据挖掘.
E-mail: jsjxhuxg@hfut.edu.cn

谢飞(1980－),男,博士,副教授,主要研究领域为数据挖掘,文本信息处理.
E-mail: xiefei9815057@sina.com

高隽(1963－),男,博士,教授,博士生导师,主要研究领域为图像处理,模式识别.
E-mail: gaojun@hfut.edu.cn

黄咏明(1986－),女,硕士,主要研究领域为序列模式挖掘.
E-mail: abfom101@gmail.com