计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20150804
54(1):94-110,2017

界标窗口下数据流最大规范模式挖掘算法研究

闻英友[1,3]　王少鹏[1,2]　赵　宏[1,3]
[1](东北大学计算机科学与工程学院　沈阳　110819)
[2](内蒙古大学计算机学院　呼和浩特　010021)
[3](医学影像计算教育部重点实验室(东北大学)　沈阳　110819)
(wangshaopeng1984@163.com)

The Maximal Regular Patterns Mining Algorithm Based on Landmark Window over Data Stream

Wen Yingyou[1,3], Wang Shaopeng[1,2], and Zhao Hong[1,3]
[1](*College of Computer Science and Engineering*, *Northeastern University*, *Shenyang* 110819)
[2](*College of Computer Science*, *Inner Mongolia University*, *Huhhot* 010021)
[3](*Key Laboratory of Medical Image Computing* (*Northeastern University*), *Ministry of Education*, *Shenyang* 110819)

Abstract　Mining regular pattern is an emerging area. To the best of our knowledge, no method has been proposed to mine the maximal regular patterns about data stream. In this paper, the problem of mining maximal regular patterns based on the landmark window over data stream is focused at the first time. In order to resolve the issue that the naïve algorithm which is used to handle the maximal regular patterns mining based on the landmark window over data stream does not have the characteristic of incremental computation, the DSMRM-BLW(data stream maximal regular patterns mining based on boundary landmark window) algorithm is proposed. It takes the first window as the boundary landmark window, and handles it with the naïve algorithm. For all other windows, it can obtain the maximal regular patterns over them based on the ones over the adjacent last window incrementally, and can overcome the drawback of the naïve algorithm. It is revealed by the extensive experiments that the DSMRM-BLW algorithm is effective in dealing with the maximal regular patterns mining based on the landmark window over data stream, and outperforms the naïve algorithm in execution time and space consumption.

Key words　data stream; landmark window; maximal regular pattern; incremental calculation; boundary landmark window technology

摘　要　首次对界标窗口下数据流最大规范模式挖掘问题进行了研究.为了克服 naïve 算法在处理该问题时不具有增量计算的缺点,提出了一种基于边界界标窗口技术的数据流最大规范模式挖掘(data

收稿日期:2015-09-06;**修回日期**:2016-02-16
基金项目:国家自然科学基金项目(60903159,61173153,61402096,61163011,61262082,61662054);中央高校基本科研业务费专项资金项目(N110818001,N100218001,N130504007,N120104001);国家"八六三"高技术研究发展计划基金项目(2015AA016005);沈阳市科技计划项目(1091176-1-00);内蒙古自然科学基金项目(2015MS0612)
This work was supported by the National Natural Science Foundation of China (60903159, 61173153, 61402096, 61163011, 61262082, 61662054), the Fundamental Research Funds for the Central Universities (N110818001, N100218001, N130504007, N120104001), the National High Technology Research and Development Program of China (863 Program)(2015AA016005), the Science and Technology Plan of Shenyang of China (1091176-1-00), and the Natural Science Foundation of Inner Mongolia (2015MS0612).

stream maximal regular patterns mining based on boundary landmark window, DSMRM-BLW)算法.该算法将数据流上的第1个待处理窗口定义为边界界标窗口,使用 naïve 算法对其进行处理;之后每个窗口上的最大规范模式都可以基于前一个窗口上的最大规范模式集合增量获得,可以克服 naïve 算法的缺点.实验结果表明:DSMRM-BLW 算法是处理界标窗口下数据流最大规范模式挖掘的有效方法,与 naïve 算法相比,具有相同的执行结果,但时间与空间效率得到了很大的提高.

关键词 数据流;界标窗口;最大规范模式;增量计算;边界界标窗口技术

中图法分类号 TP311

随着数据流应用的不断增多,挖掘数据流中用户感兴趣的模式已然成为数据流挖掘领域中一类非常重要的问题[1].现有相关研究关注较多的是频繁模式的挖掘[1-5],但这些研究存在一个共同问题就是所挖掘模式的判别标准是它们在数据流区间中的出现次数,而并不是模式本身的出现行为(如模式是否周期性、规律性地出现),所以对于很多更注重模式出现行为的实际应用来说,这些研究无法有效解决它们当中所涉及到的问题.比如在超市等销售行业中,管理人员要获得一年中常态情况下货物的销售情况,这种常态通常是指货物的销售具有规律性、周期性且销售的周期不长.如果我们使用挖掘频繁模式的方法来处理这个问题,对于像夏天的电扇和蚊香、冬天的电热宝、雨季的雨具等在一年中某些特定时段内销售频繁的货物,也会因为达到频繁标准而被选择出来,但显然它们并不是一年中满足常态销售这样条件的货物,所以并不是此时我们关注的对象.像实例中这样要求出现具有规律性、周期性的模式通常被称为规范模式(regular pattern)[6-12].而类似应用场景的存在也使挖掘规范模式的研究显得非常必要.

根据模式规范度的度量标准不同,现有规范模式可分为2类:1)以模式发生周期序列中的最大值作为规范度的规范模式;2)以模式发生周期序列的方差作为规范度的规范模式.比较起来前者更有时间优势,而后者精确度更高.Tanbeer 等人在文献[1]中对数据流环境下第1类规范模式的挖掘问题进行了研究,给出了基于 RPS-tree 的处理算法;接着 Tanbeer 等人又在文献[6]中对规范模式的增量挖掘处理机制进行了研究,提出了一种基于 IncRT 树的规范模式挖掘算法,只需单遍扫描所关注的数据内容就可以构建 IncRT 树,接着基于该树以模式增长的方法获得规范模式;Kumar 等人在文献[7]中对于数据流环境下基于窗口的第1类规范模式挖掘算法进行了研究,提出了 VDSRP 算法,该算法采用垂直数据格式组织窗口中的数据,通过类 Aprior 方法搜索可能的结果模式,在这个过程中通过比较这些模式规范度与用户给定的规范阈值来确定该模式是否是规范模式,能够有效地处理滑动窗口下数据流的规范模式挖掘问题;Verma 等人在文献[8]中对 VDSRP 算法进行了改进,主要改进点在于进一步使用了 *bit-sequence* 来组织窗口中的数据,基于窗口中项的 *bit-sequence* 执行位操作来获得相关模式规范度,提高了算法执行效率.很多研究还把这类规范模式与频繁模式二者结合起来.如 Kumar 等人在文献[9]中给出了滑动窗口下基于该规范度的数据流规范频繁模式挖掘方法 RFPDS 算法,该算法采用垂直数据格式组织窗口中的数据,只计算频繁模式的规范度,通过与用户给定的规范阈值进行比较来确定该模式是否是规范频繁模式,能够有效地处理滑动窗口下数据流的规范频繁模式挖掘问题.Sreedevi 等人在文献[10]中首次提出了用于处理滑动窗口下基于该规范度的数据流规范闭模式挖掘算法,该算法将整个规范频繁闭模式的求解过程分为2个阶段:第1个阶段求解规范模式集合;第2个阶段在此基础上得到规范频繁闭模式,可以有效处理这类问题.当前有关第2类规范模式的研究是将规范模式与频繁模式结合起来展开.如 Rashid 等人在文献[11]中对静态数据库下基于这种规范度的规范频繁模式挖掘问题进行了研究,并给出了相应的处理方法,该方法中使用基于成员出现频率递减顺序的 RF-tree 来存放数据库中成员信息,另外使用类 FP-growth 算法的方式挖掘树中的规范频繁模式;Rashid 等人在文献[12]中设计了用于寻找传感网络中规范频繁传感器模式的 RSP-tree 数据结构,使用模式出现周期序列的方差作为模式的规范度,并给出了相应的挖掘算法,能够有效地应用于该领域规范频繁传感模式的挖掘.

由文献[1,13]可知,规范模式具有向下闭包的特点,即如果一个模式是规范的,那么它所有的非空

子模式也都是规范的,所以在挖掘数据集中的规范模式时,如果我们只定性地关注哪些模式是规范的而不在意它们的具体规范度,那么完全可以通过挖掘其中所有长度最大的规范模式来提高挖掘效率.这些长度最大的规范模式间彼此并不包含,但能保证数据集中任何一个规范模式都是这些长度最大规范模式集合中某个成员的子集,我们把这类长度最大的规范模式称为最大规范模式.由规范模式的闭包性可知一个数据集上所有的最大规范模式隐含了这个数据集上所有的规范模式,但数量却比全部规范模式要少,这点与频繁模式和最大频繁模式间的关系类似.但由前面分析可知,当前还没有这个方向的研究.

本文基于这点,首次对以模式周期最大值作为规范度的界标窗口下数据流最大规范模式挖掘问题进行了研究.为了能增量地处理这个问题,本文分析了相邻窗口上最大规范模式间的关系,提出了边界界标窗口技术,接着在此技术基础之上给出了该问题的增量处理方法 DSMRM-BLW(data stream maximal regular patterns mining based on border landmark window)算法,一系列公用数据集上的实验证明了该算法的有效性.

1 相关知识

1.1 IncRT 算法

IncRT 算法是由 Tanbeer 等人在文献[1]中提出的一种目前唯一用于增量挖掘数据库中规范项集(或规范模式)的方法.该方法可以在当前数据库成员规范项集集合的基础上增量获得新数据成员到来后的数据库中规范项集集合.整个算法通过 1 个 IncRT 树来维护数据库中的成员信息,该树以字典顺序存放每一条数据记录.树中每条记录的尾结点存放着该记录在数据库中的索引号,以此标识该记录在数据库中的发生情况.树中还维护着 1 个增量规范表 IncR-table,每个表项分为 5 个域,i 域用于存放树中项的名称,IncR-table 中该域中成员的排列顺序与树中项的排列顺序相同;r 域是该项在树中的规范度;t_l 域是该项最近发生的记录索引;m 域是该项的修改标识域;p 域是 1 个指针域,用于指向树中具有相同项名的结点,树中所有具有相同项名的结点间也有指针彼此连接.在 IncRT 的构造过程中,头表中除了项的 r 域外,其他域值都可以设定.当树构造完成后,借助为每个项分配的临时数组结构,按照由下而上的顺序完成对树的 1 次扫描后得到每个项的 r 域值.当树的构造完成后,按照 FP-growth 模式增长挖掘技术来获得所有规范项集,保留所有规范项集成员.之后重置头表成员的 m 域值.接着把新到来的数据成员插入到 IncRT 中,这个过程中重新设定头表中成员的 m 域值;对于新的 IncRT,只对 IncR-table 中 m 域发生变化的项使用 FP-growth 算法进行挖掘,假设得到的规范项集集合为 A.对于前一个窗口上的规范项集集合,找到其中不包含 IncR-table 中 m 域发生变化项的项集,如果在更新它们规范度后得到的规范项集集合为 B,则 $A \cup B$ 即为当前数据库上的规范项集集合.有关该算法更详细的说明可参见文献[6].

1.2 PA 算法

PA 算法是由 Verma 等人在文献[8]中提出的一种对数据流滑动窗口下以项集周期最大值作为规范度的规范项集(或规范模式)进行挖掘的算法.该算法使用垂直格式组织窗口中的数据流成员,不仅如此,该算法还基于当前窗口中每个项在窗口数据流成员中的出现状况,定义了这些项在当前窗口中的 *bit-sequence*.基于这些 *bit-sequence*,PA 算法采用 Aprior 算法思想可以求得当前窗口中所有项集的 *bit-sequence*.因为基于 *bit-sequence* 中非 0 成员的位置情况,可以得到对应项集在当前窗口中的规范度,所以该算法可以求得当前窗口中所有项集的规范度,进而得到所有规范项集.关于该算法更详细的情况,可参考文献[8].

2 基本概念和问题说明

2.1 基本概念

设 $S_I=\{I_1, I_2, \cdots, I_m\}$ 是 m 个项的集合,I_i 是第 i 个项,I 中的所有成员按全序 $\prec$(字典序)排列.任给集合 P,如果 $P \subseteq S_I$ 且 $l=|P|$(即 P 中含有 l 个 S_I 的成员),则称 P 为 l 模式,P 的长度为 l.$S_T=\{T_1, T_2, \cdots, T_n\}$ 是事务集,其中 $T_i \subseteq S_I$,也即 T_i 为 S_I 的子集.

定义 1. 模式发生周期[1].任给模式 P,令 T_i,T_j 是事务集 S_T 中 2 个包含 P 的事务,且在 T_i 与 T_j 间没有任何包含 P 的其他事务存在,则 i 与 j 的差值被称为 P 的一个发生周期.

定义 2. 模式发生周期集合[1].令 S_P 是 S_T 中所有包含模式 P 的事务集合,$|S_P|$ 是 S_P 中的事务总数,这里规定 null 事务(该事务为 S_T 中第 0 个事

务,索引值为 S_T 中第 1 个事务的索引值−1)和 S_T 中最后一个事务分别是 S_P 中的第 1 个和最后 1 个事务,它们和 S_T 中其他所有包含 P 的事务一起构成了 S_P,S_P 中所有事务按它们在 S_T 中的索引递增排列. 如果设 $Tran_k$ 与 $Tran_{k+1}$分别是 S_P 中任意 2 个相邻事务,它们在 S_T 中的索引分别为 $Tran_k.index$ 与 $Tran_{k+1}.index$,则 $C_P=\{Tran_{k+1}.index-Tran_k.index|1\leqslant k\leqslant|S_P|-1,k\in\mathbb{N}_+\}$ 是 P 在 S_T 中的发生周期集合.

定义 3. 规范模式[1]. 模式 P 在 S_T 中的规范度被定义为 C_P 中成员的最大值,表示为 R_P,即有 $R_P=\max(C_P)$. 对于用户给定的最大规范阈值 $\lambda(1\leqslant\lambda\leqslant|S_T|)$,如果 $R_P\leqslant\lambda$,则 P 是 S_T 中一个规范模式;如果 P 的长度为 1,则 P 又被称为 S_T 中的一个规范项.

定义 4. 最大规范模式. 对于模式 P,如果满足条件 $R_p\leqslant\lambda$,同时对于任意满足条件 $Y\subseteq S_I$ 且 $P\subset Y$ 的模式 Y,均有 $R_Y>\lambda$ 成立,则称 P 为 S_T 上的最大规范模式.

定义 5. 数据流 DS[1]. 数据流 DS 是一个由连续到来的事务组成的事务序列,表示为 $DS=\{T_1,T_2,\cdots,T_i,\cdots\}$. T_i 为第 i 个到来的事务.

定义 6. 基本窗口(element window, EW)与界标窗口(landmark window, LW). EW 是一个确定时间内连续到达的数据流成员,即 $EW=\{T_i,T_{i+1},\cdots,T_j\}(0<i\leqslant j)$. LW 对应一个连续相邻的 EW 序列,即 $LW=\{EW_1,EW_2,\cdots,EW_u,EW_{u+1},\cdots\}$. 随着新数据流成员的到来,界标窗口起始位置不变,终止位置会以基本窗口为单位不断滑动. LW 的大小为其中 EW 的个数,表示为 $|LW|$;$|EW|$ 是基本窗口中数据流成员的个数.

2.2 问题说明

参照文献[6]中关于增量挖掘规范模式的说明,我们可以给出本文要处理问题的说明,即给定数据流 DS、基本窗口大小 $|EW|$、最大规范阈值 $\lambda(1\leqslant\lambda\leqslant|EW|)$、界标窗口起始位置 sp,本文要解决的问题就是获得起始位置为 sp 的每个界标窗口上所有的最大规范模式.

3 基于边界界标窗口技术的数据流最大规范模式挖掘算法

当前并没有针对以模式周期最大值为规范度的界标窗口下数据流最大规范模式挖掘的专门研究,所以最直接的处理方式就是从定义出发来解决这个问题. 文献[6]中的 IncRT 算法是目前唯一可用于解决以模式周期最大值为规范度的界标窗口下数据流规范模式增量挖掘的算法,基于该算法与最大规范模式的定义可以得到解决界标窗口下数据流最大规范模式挖掘的算法. 文献[8]中的 PA 算法使用了另外一种不同方法来挖掘滑动窗口中数据流的规范模式. 该算法使用垂直格式来组织当前窗口上的数据流成员,并为窗口中的每个项定义 *bit-sequence*,然后采用类 Aprior 方法来得到当前窗口上的所有规范模式. 如果在 PA 算法的滑动窗口成员更新步骤中只执行加入新数据成员的操作,之后对新窗口中成员再使用 PA 算法进行处理,那么基于这个改动的算法和最大规范模式的定义,可以得到另外一种挖掘界标窗口下数据流最大规范模式的算法.

以上 2 种算法在挖掘界标窗口上数据流最大规范模式的过程中,都会执行 2 个基本操作:1)得到每个界标窗口上数据流的规范模式集合;2)消除集合中成员间存在的超集关系. 我们把这 2 种算法都归类于 naïve 算法,其中基于 IncRT 算法的方法被称为 NI(naïve based on IncRT)算法,而基于 PA 算法的方法被称为 NP(naïve based on PA)算法. 但这 2 种算法存在明显问题,就是每个界标窗口上(以下简称窗口)最大规范模式集合都需要重新挖掘,而且必须是在得到了所有规范模式之后,才能得到最大规范模式集合,所以时间开销会很大. 如果我们能得到相邻 2 个窗口上最大规范模式间的关系,每次窗口上最大规范模式的求解都是基于前一个窗口上的结果来执行,那么一方面我们无需在求得数量庞大的所有规范模式后再得到最大规范模式集合;另一方面也可以省去对于相邻窗口上相同最大规范模式的重复挖掘,能够减小时间开销. 基于这个思想,在这部分我们首先分析了相邻窗口上规范模式间的关系;接着在此基础上得到了相邻窗口上最大规范模式间的关系(我们称其为边界界标窗口技术);随后又给出了便于该技术实现的数据结构;最后在该数据结构的基础上得到了能够增量挖掘界标窗口下数据流最大规范模式的 DSMRM-BLW 算法.

3.1 相邻窗口上规范模式间的关系

设 LW_i 是第 i 个界标窗口,RS_i 与 US_i 分别是 LW_i 上的规范模式与非规范模式集合,$|RS_i|$ 与 $|US_i|$ 分别是这 2 个集合中的成员个数,$r_{i,j}$ 与 $u_{i,j}$ 分别是 RS_i 与 US_i 中第 j 个成员.

性质 1. 设 $T_{\text{new},k}$ 是 LW_i 后的第 $k(k>0)$ 个新到

事务,则有 US_i 中任意成员 $u_{i,j}(1\leqslant j\leqslant|US_i|)$一定也是 $LW_i\cup T_{new,1}\cup T_{new,2}\cup\cdots\cup T_{new,k}$上非规范模式集合中的成员.

证明. 设 $LW_i\cup T_{new,1}\cup T_{new,2}\cup\cdots\cup T_{new,k}$中 $u_{i,j}$的规范度为 $Ru'_{i,j}$. 因为 $u_{i,j}\in US_i$,所以由规范模式定义可知 $u_{i,j}$在 LW_i 中的规范度 $Ru_{i,j}$满足条件 $Ru_{i,j}>\lambda$. 又因为 LW_i 中成员是 $LW_i\cup T_{new,1}\cup T_{new,2}\cup\cdots\cup T_{new,k}$中成员的子集,所以由模式规范度定义可知 $Ru'_{i,j}\geqslant Ru_{i,j}>\lambda$ 成立,即 $u_{i,j}$也是 $LW_i\cup T_{new,1}\cup T_{new,2}\cup\cdots\cup T_{new,k}$中的非规范模式.

证毕.

性质 2. 设 $T_{new,k}$是 LW_i 后第 $k(k\geqslant1)$个新到事务,$T_{new,k}.index$ 是 $T_{new,k}$在数据流中的索引,$LW_i.sp$ 是界标窗口起始位置的索引. 当 $T_{new,k}.index-(LW_i.sp-1)>\lambda$ 时,$LW_i\cup T_{new,1}\cup T_{new,2}\cup\cdots\cup T_{new,k}$上的规范模式集合相对于 $LW_i\cup T_{new,1}\cup T_{new,2}\cup\cdots\cup T_{new,k-1}$上的规范模式集合,没有新模式加入.

证明. 基于规范模式定义可以这样理解 LW_i 上规范模式的挖掘过程. 首先针对 LW_i 中每个事务求得该事务包含的所有子模式(即由构成该事务的项形成的所有子集),每个子模式的索引都为该事务的索引,这样由产生于每个事务的所有子模式构成了候选规范模式集合 CS_i;接着求 CS_i 中每个模式的规范度;最后通过规范度与最大规范阈值 λ 的比较判断该模式是否为规范模式. 当 $k=1$ 时,令由 $T_{new,1}$产生的所有子模式所构成的集合为 $Subset_{new,1}$. 则 $Subset_{new,1}$与 LW_i 上的候选规范模式集合 CS_i 构成了 $LW_i\cup T_{new,1}$上的候选规范模式集合. 假设在 $T_{new,1}$到来后有新规范模式 r_1 产生,则该 r_1 一定位于 $Subset_{new,1}$中,而并不位于 CS_i 中. 这是因为如果 r_1 也存在于 CS_i 中,由题设可知,r_1 在 CS_i 中并不是规范模式,也即 r_1 在 LW_i 上的规范度大于 λ;当 $T_{new,1}$到来后,因为 r_1 变成了规范模式,所以 r_1 在 $LW_i\cup T_{new,1}$上的规范度小于等于 λ. 但因为 $LW_i\subset(LW_i\cup T_{new,1})$,再结合定义 3 可知,$r_1$ 在 $LW_i\cup T_{new,1}$上的规范度一定大于等于其在 LW_i 上的规范度,也即大于 λ,这与 r_1 是 $LW_i\cup T_{new,1}$上的规范模式相矛盾,所以 r_1 不存在于 CS_i 中. 这样 r_1 的索引一定为 $T_{new,1}.index$. 当 $T_{new,1}.index-(LW_i.sp-1)>\lambda$ 时,由规范度定义可知,此时有 $Rr_1>\lambda$,故 r_1 并非规范模式,这也与假设矛盾,即在 $T_{new,1}$到来后没有新规范模式产生. 与 $k=1$ 的证明相类似,容易推得当 $k\geqslant2$ 时,结论也成立. 证毕.

下面我们通过实例说明性质 1~2 的内容. 图 1 给出了 1 个数据流图示,其中$|EW|=3$.

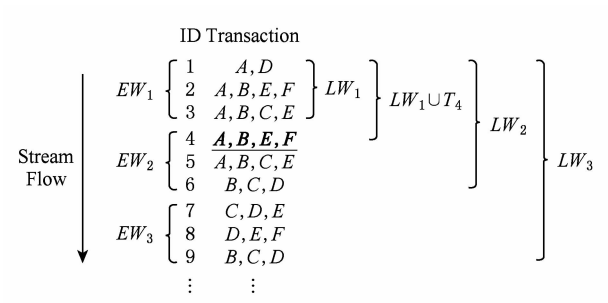


Fig. 1 Illustration of the property 1 and property 2

图 1 性质 1 和 2 的说明

图 1 中 ID 为 4 的 Transaction(图 1 中加重且有下划线的事务,记为 T_4)是界标窗口 LW_1 后的第 1 个事务,由定义 2~3 可知,E 在 LW_1 中的规范度 $R_E=2$. 当 $\lambda=1$ 时,E 是 LW_1 上的非规范模式,这时考虑 E 在 $LW_1\cup T_4$ 上的规范性,由性质 1 可知,E 一定是 $LW_1\cup T_4$ 上的非规范模式;当 $\lambda=3$ 时,考虑 $LW_1\cup T_4$ 上的规范模式,因为此时有 $T_4.index-(LW_1.sp-1)=4-(1-1)=4>\lambda$,所以由性质 2 可知,$LW_1\cup T_4$ 上的规范模式集合与 LW_1 上的规范模式集合相比,没有新模式加入. 同样由性质 2 可知,$LW_1\cup T_4\cup T_5\cup T_6\cdots$ 上的规范模式集合与 LW_1 上的规范模式集合相比,没有新模式加入.

3.2 边界界标窗口技术

定义 7. 边界基本窗口(border element window, BEW)与边界界标窗口(border landmark window, BLW). 令界标窗口 LW 的起始位置为 $LW.sp$,$Tran$ 是起始位置后的 1 个事务,$Tran.index$ 是该事务在数据流中的索引. 如果 $Tran$ 满足条件 $Tran.index-(LW_i.sp-1)=\lambda$,我们就把 $Tran$ 所在的基本窗口定义为边界基本窗口 BEW,而以 BEW 为最后一个基本窗口的界标窗口定义为边界界标窗口 BLW.

性质 3. BLW 后的任意一个界标窗口,其上的规范模式集合与相邻前一个界标窗口上的规范模式集合相比,没有新模式加入.

证明. 由性质 2 容易推得该性质成立. 证毕.

性质 4. 假设 ms 是 BLW 上的最大规范模式. 考虑 BLW 后第 1 个界标窗口 NLW_1,有 3 个结论成立:

1) 如果 ms 是该窗口上的规范模式,则 ms 一定也是该窗口上的最大规范模式.

2) 如果 ms 不是该窗口上的规范模式,且其中每个成员都是规范项,设 ms 长度为 len,将 ms 所有 $len-1$ 模式中满足规范条件的成员放入结果集合;

然后针对每个不满足规范条件的 $len-1$ 模式,求它们的 $len-2$ 模式,将所有 $len-2$ 模式中满足规范条件的成员放入结果集合,这个过程一直到产生的每个模式都满足规范条件或者是执行到所有 1 模式为止,如果对此时结果集合执行超集检测后得到的集合为 NS,则 NS 是当前窗口上 ms 所有子模式的最大规范模式集合.

3) 如果 ms 不是该窗口上的规范模式,且其中存在非规范项.这时令执行了非规范项删除操作后的 ms 为 ms_1,当 ms_1 是规范模式时,该模式也是当前窗口中 ms 子模式的最大规范模式;否则按结论 2 对其进行处理,能得到当前窗口上 ms 所有子模式的最大规范模式集合.

证明. 1) 设 BLW 与 NLW_1 上的规范模式集合分别是 RS_1 与 RS_2.因为 ms 是 BLW 上的最大规范模式,所以由定义 4 可知我们无法在 RS_1 中找到 1 个成员 rs 满足条件 $ms \subset rs$.考虑 NLW_1 上情况,如果 ms 同时也是 NLW_1 上的规范模式,也即 ms 满足条件 $ms \in RS_2$,这时因为无法在 RS_1 中找到 1 个成员 rs 满足条件 $ms \subset rs$,而且由性质 3 可知 $RS_2 \subseteq RS_1$,所以也无法在 RS_2 中找到成员 rs_1 满足条件 $ms \subset rs_1$,也即 ms 是 NLW_1 上的最大规范模式.

2) 因为 ms 是 BLW 上的最大规范模式,由规范模式闭包性可知,ms 所有子模式也都是规范模式,而且 ms 能覆盖(覆盖是指子模式中的所有项都在 ms 中)这些子模式.另外由性质 3 可知 $RS_2 \subseteq RS_1$,所以 ms 也能覆盖存在于 RS_2 中 ms 的所有子模式.但因为 $ms \notin RS_2$,所以 ms 并不是存在于 RS_2 中 ms 子模式的最大规范模式.设 ms 长度为 len,由 Aprior 算法可知,ms 可由 ms 的 $len-1$ 模式执行连接操作来得到,而且 ms 可以覆盖它的所有 $len-1$ 模式.依次类推,可得 ms 的所有 $len-1$ 模式覆盖了除 ms 外所有的 ms 子模式.所以当 ms 不是 NLW_1 上的规范模式时,这时判断 ms 上的 $len-1$ 模式,如果它们都是规范模式,也即它们都是 RS_2 中的成员,那么因为此时 $ms \notin RS_2$,所以由这些规范模式构成了能够覆盖存在于 RS_2 中 ms 子模式的最大规范模式集合.如果其中存在某个 $len-1$ 模式不满足规范条件,则使用相同方法判断该模式的 $len-2$ 模式,一直到关注的每个模式都满足规范条件,则由所有获得的规范模式构成了结果集合.由定义 4 可知,在消除了集合中成员间存在的超集关系后,该集合即为能够覆盖存在于 RS_2 中 ms 子模式的最大规范模式集合.

3) 假设 F 是 ms 在 NLW_1 中任意非规范项,由规范模式闭包性可知,F 在 ms 中的任何超集都不可能是 NLW_1 上的规范模式,所以 ms 上含有 F 的子模式都不可能是 NLW_1 中的规范模式,即 RS_2 中不存在 ms 上含有 F 的子模式.假设 $subset_{ms}$ 是 ms 的子模式集合,则 ms 能覆盖 $subset_{ms}$ 中每个成员;考虑存在于 RS_2 中的 ms 子模式集合 $subset1_{ms}$,由前面分析可知 $subset1_{ms} \subseteq subset_{ms}$ 成立,所以 ms 同样可以覆盖 $subset1_{ms}$ 中的每个成员;从 $subset_{ms}$ 中删除所有含有 ms 在 NLW_1 中非规范项的子模式,我们把新得到的集合记为 $subset2_{ms}$,则有 $subset1_{ms} \subseteq subset2_{ms}$.另外假设 ms 在执行了删除所有 NLW_1 中非规范项的操作后变为 ms_1,则 ms_1 能够覆盖 $subset2_{ms}$ 中的所有成员,进一步可以覆盖 $subset1_{ms}$ 中的所有成员.当 ms_1 为 NLW_1 上的规范模式时,结合定义 4 可知,ms_1 是 $subset1_{ms}$ 中的最大规范模式;当 ms_1 是 NLW_1 上的非规范模式时,与证明 2 情况类似,易得结论成立. 证毕.

性质 5. 假设 BLW 上最大规范模式集合 $mset=\{ms_1, ms_2, \cdots, ms_n\}$,$NLW_1$ 上最大规范模式集合为 $mset_1$.对于 $mset$ 与 $mset_1$ 有 2 个结论成立:

1) 如果 $mset$ 中每个成员仍是 NLW_1 上的规范模式,则 $mset=mset_1$.

2) 如果 $mset$ 中有些成员不是 NLW_1 上的规范模式,此时 $mset$ 中的成员可分为 3 类:①在 NLW_1 上仍然是规范模式的成员;②在 NLW_1 上不是规范模式,但模式的构成项中都是规范项的成员;③在 NLW_1 上不是规范模式,但模式构成项中存在非规范项的成员.分别使用性质 4 的结论 1~3 来处理上述 3 类成员,假设由这 3 类成员的处理结果构成的集合为 RS,则执行了超集检验后的 RS 即为当前窗口上的最大规范模式集合.

证明. 1) 设 RS_1 与 RS_2 分别是 BLW 与 NLW_1 上的规范模式集合.因为 $mset$ 是 BLW 上的最大规范模式集合,所以可知:①$mset$ 中的成员覆盖了 RS_1 中的所有规范模式;②$mset$ 中任意 2 个成员 ms_k 与 ms_l 都有如下关系 $ms_k \cap ms_l \neq ms_k$,且 $ms_k \cap ms_l \neq ms_l$ 成立;③对于 $mset$ 中任意成员 ms_t,我们在 RS_1 中无法找到 1 个成员 rs,使得条件 $ms_t \subset rs$ 成立.如果 $mset$ 中每个成员仍是 NLW_1 上的规范模式,由性质 4 的结论 1 可知,这些成员也是 NLW_1 上的最大规范模式,即 $mset$ 也是 NLW_1 上最大规范模式的集合.又由性质 3 可知 $RS_2 \subseteq RS_1$,而且再由前面① 可推得 $mset$ 中的成员也覆盖了

RS_2 中的所有规范模式,另外前面的②和③显然在 NLW_1 上也依然成立,即 $mset$ 也是 NLW_1 上的最大规范模式集合.

2) 由证明 1 可知,如果 $mset$ 中每个成员仍是 NLW_1 上的规范模式,则 $mset=mset_1$,结合定义 4 可知,对于 NLW_1 上任意 1 个规范模式,在 $mset$(此时也即 $mset_1$)中至少有 1 个成员会完成对该模式的覆盖. 假设此时 $mset$ 中有 M 个成员,我们可以根据 RS_2 中成员(即 NLW_1 中规范模式)被这 M 个成员所覆盖情况,将其中成员分为 M 类,当然这 M 类成员彼此间可能会有重合. 每一类中成员都被对应 $mset$ 中的成员所覆盖. 当 $mset$ 中某些成员并不是 NLW_1 上的规范模式时,NLW_1 上本来应该由这些成员覆盖的所有规范模式,此时可能就没有最大规范模式覆盖了,所以需要求这部分规范模式的最大规范模式. 假设将 $mset$ 中不是 NLW_1 上规范模式的成员组合成集合 $nset$,对于 $nset$ 中任意成员 $nset_i$,如果 $nset_i$ 的构成项都是规范项,由性质 4 的结论 2 可知,使用该结论处理 $nset_i$ 后可以得到集合 NS_i,该集合是 NLW_1 上 $nset_i$ 所有子模式的最大规范模式集合,覆盖了存在于 RS_2 中 $nset_i$ 的所有子模式;如果 $nset_i$ 的构成项中存在非规范项,由性质 4 的结论 3 可知,在使用该结论处理 $nset_i$ 后,也可以得到该成员在 NLW_1 上所有子模式的最大规范模式集合,覆盖了存在于 RS_2 中 $nset_i$ 的所有子模式;设从 $mset$ 中除去 $nset$ 中的成员后得到的集合为 $mset|_{nset}$,由前面分析及由性质 4 的结论 1 可知,这些成员也是 NLW_1 上的最大规范模式,可以覆盖存在于 RS_2 中每个 $mset|_{nset}$ 中成员的所有子模式,所以由这 3 部分构成的 RS 覆盖了 RS_2 中所有规范模式,由定义 4 可知,RS 集合在执行超集检测后,即成为 NLW_1 上的最大规范模式集合. 证毕.

我们仍以图 1 为例说明这些性质内容. 设 LW_1 上 $\lambda=3$ 时的规范模式集合为 RS,由图 1 可知 $RS=\{A,B,C,D,E,F,AC,AD,AB,AE,AF,BC,BE,BF,EF,CE,ABC,ABE,ABF,BCE,BEF,AEC,AEF,ABCE,ABEF\}$;显然最大规范模式集合 $MRS=\{ABCE,ABEF,AD\}$;在求 LW_2 上最大规范模式时,如果 $ABCE$,$ABEF$ 与 AD 仍是 LW_2 上的规范模式,则 LW_2 上的最大规范模式集合仍然为 $\{ABCE,ABEF,AD\}$(性质 5 的结论 1). 因为 $ABCE$ 与 $ABEF$ 都是 LW_2 上的规范模式,所以它们仍然是 LW_2 上的最大规范模式(性质 4 的结论 1);又因为 AD 不是 LW_2 上的规范模式,而且这时 D 不是 LW_2 上的规范模式,所以这时只考虑 A. 因为 A 是规范模式,所以 A 是 AD 在 LW_2 上所有子模式的最大规范模式(性质 4 的结论 3). 令 A 与 $ABEF$ 以及 $ABCE$ 构成集合 S_{tem},由性质 5 的结论 2 可知,在对该集合执行超集检测后,可得 LW_2 上 $\lambda=3$ 时的 $MRS=\{ABCE,ABEF\}$.

由性质 4～5 不难推得 BLW 后任意 2 个相邻界标窗口上最大规范模式集合间都有相同结论成立. 另外由 2.2 节可知,本文中 λ 的取值范围为 $1\leqslant\lambda\leqslant|EW|$,所以本文中 BLW 为最大规范模式挖掘过程中的第 1 个界标窗口. 由性质 5 可知,从第 2 个界标窗口开始,可以基于前一个窗口上最大规范模式集合来获得当前窗口上最大规范模式集合. 具体方法为首先考虑前一个窗口上最大规范模式集合中每个成员模式的规范度,如果这些模式规范度都满足规范条件,则它们构成了新窗口上的最大规范模式集合,否则使用性质 5 中的结论 2 对其进行处理,则可以得到当前窗口上的全部最大规范模式. 这种从边界界标窗口的下一个窗口开始,基于前一个窗口最大规范模式集合求得当前窗口最大规范模式集合的方法被称为边界界标窗口技术.

3.3 数据结构

通过前面对边界界标窗口技术的分析可知,该技术执行的 1 个重要前提就是需要一种便于计算前一个窗口上最大规范模式在当前窗口上规范度的方法,而这种方法还需要便于计算当前窗口上由规范项所构成的任意模式的规范度. 本部分主要给出了相关的数据结构,而这些数据结构的使用可以满足边界界标窗口技术使用的前提条件.

3.3.1 *BV*(*bit-vector*) table

参照文献[8]中的 *bit-sequence* 思想,我们对于边界界标窗口中每个项维护一个 *bit-vector* 结构. 窗口中所有项的 *bit-vector* 构成了 *bit-vector* table(简称 *BV* table). 因为通过性质 3 可以知道,BLW 后的每个窗口上的规范模式都是 BLW 上规范模式的子集,进一步可知,BLW 后每个窗口上所有规范项的集合都是 BLW 上所有规范项集合的子集,所以我们只要维护好边界界标窗口上的 *BV* table 在每个新窗口上的状态,无论计算前一个窗口上的最大规范模式在当前窗口上的规范度,还是计算当前窗口上由规范项所构成模式的规范度,只需在当前窗口的 *BV* table 中找到模式中项对应的 *bit-vector*,然后基于它们执行与操作,最后对得到的结果计算规范度即可.

1) BV table 结构

BV table 包含 *name*,*bit-sequence*,*reg*,*lastpos*,*sign* 这 5 个域,分别对应项的名称、项在当前窗口中的 bit 序列、项的规范度 *reg*、*bit-sequence* 后 $|EW|$ 个 bit 序列中最后一个非 0 位置以及该项在当前窗口上的规范度是否不大于规范阈值 λ 的结果. 其中 *bit-sequence* 是 1 个长度为 $|LW|\times|EW|$ 的位序列,每一位与界标窗口中的每个数据流成员相对应. 如果该项在这个数据流成员中出现,则对应的 *bit-sequence* 位设为 1,否则设为 0. BV table 中每条记录与当前窗口中的每个项对应,该记录也是对应项的 *bit-vector*. BV table 中所有项按字典顺序排列. 表 1 给出了图 1 中 LW_1 上当 $\lambda=3$ 时的 BV table.

Table 1 BV table over the LW_1

表 1 LW_1 上的 BV table

name	*bit-sequence*	*reg*	*lastpos*	*sign*
A	111	1	3	true
B	011	2	3	true
C	001	3	3	true
D	100	2	1	true
E	011	2	3	true
F	010	2	2	true

2) 构建 BV table

由定义 7 分析可知,本文中 BLW 是界标窗口中第 1 个窗口,所以 BV table 在第 1 个窗口中成员到来后开始构建. ①首先将 BLW 中的数据流成员按字典顺序转换成垂直格式,完成 BV table 中每个项 *bit-sequence* 的构造;②将 *bit-sequence* 中非 0 成员位置记录下来形成 1 个序列,该序列在增加头成员 0 和尾成员 $|EW|$ 后形成 1 个新序列,求该序列中相邻成员差值的最大值,以此作为该项的 *reg* 值,同时将 *bit-sequence* 中最后一个非 0 成员的位置作为该项的 *lastpos*;③根据该项 *reg* 值与规范阈值 λ 间的关系,完成 *sign* 域的赋值. 当完成这些操作后,BLW 上的 BV table 构造完成. 具体如表 1 所示.

3) 更新 BV table

首先,结合规范模式的性质与 BV table 结构特点,可以得到性质 6:

性质 6. BLW 上的 BV table 中如果有项 A 的 *sign* 域值为 false,则有 2 个性质成立:①之后每个界标窗口上的 BV table 中,该项的 *sign* 域值都为 false;②BLW 上及其之后的每个窗口上的规范模式集合中,都不会有含有该项的模式出现.

证明. ① 令 BLW 上的 BV table 为 BV,假设 BV 中 I 项的 *sign* 域值为 false,这说明 I 项在 BLW 上的规范度 $R_I>\lambda$. 这里令 BLW 后新界标窗口上的 BV table 为 NV,由界标窗口本身特点可知,NV 中含有 BV 中的全部项,且 NV 中项 I 的 *sign* 域值是该项在新界标窗口上的规范度 R'_I 与 λ 之间不等式 $R'_I\leqslant\lambda$ 的结果. 同样由界标窗口特点及模式规范度定义可知,有 $R'_I\geqslant R_I>\lambda$ 成立,所以该项在 NV 中的 *sign* 域值为 false. 依次类推可得,性质 6 的结论①成立.

② 由规范模式向下闭包性的性质可知,每个界标窗口上都不会有含该项的规范模式出现. 性质 6 中结论②成立. 证毕.

由性质 6 可知,在我们得到边界界标窗口上的 BV table 在每个新窗口上的状态时,我们只需对前一窗口上 BV table 中 *sign* 域值为 true 的项执行更新即可.

具体更新过程在新 EW 中所有成员到达后执行,设前一个界标窗口上的 BV table 为 PV. 首先我们会针对 PV 中所有 *sign* 域为 true 的项,按照它们在新 EW 的数据流成员中是否出现的情况,构建这些项在新 EW 中的 *bit-sequence*(记为 *item*. *bit-sequence*). 同时记下 *item*. *bit-sequence* 中第 1 个非 0 成员位置 fp 和最后一个非 0 成员位置 lp. 如果 *item*. *bit-sequence*$=0$,我们令 $fp=|EW|$,$lp=0$;接着按照定义 1~3 对 *item*. *bit-sequence* 进行处理可以得到该项的 *reg* 值,记为 *item*. *reg*. 为便于说明,假设 PV 中 *sign* 域值为 true 的项 I 在新 EW 上的信息组成的临时表为 Tem_I,另设项 I 在 PV 中对应的 *bit-vector* 为 PV_I,则可通过 3 个步骤更新 PV_I 中信息:① $PV_I.bit-sequence=PV_I.bit-sequence\cup Tem_I.bit-sequence$;② $PV_I.reg=\max(|EW|-PV_I.lastpos+Tem_I.fp,PV_I.reg,Tem_I.reg)$;③ $PV_I.lastpos=Tem_I.lp$. 表 2 给出了图 1 中 LW_2 上当 $\lambda=3$ 时的 BV table.

Table 2 BV table over the LW_2

表 2 LW_2 上的 BV table

name	*bit-sequence*	*reg*	*lastpos*	*sign*
A	111110	1	2	true
B	011111	2	3	true
C	001011	3	3	true
D	100001	5	3	false
E	011110	2	2	true
F	010100	2	1	true

4) *BV* table 的结构分析

令 *item* 是 BLW 上任意规范项. 由界标窗口特点可知,相邻下一个界标窗口是在当前窗口上加 1 个基本窗口构成的,也就是任何项在下一个界标窗口上的 *bit-sequence* 是由该项在当前窗口上的 *bit-sequence* 与其在下一个基本窗口上的 *bit-sequence* 构成. 设 BS_{cur} 是 *item* 在 BLW 上的 *bit-sequence*,而 BS_{next} 是 *item* 在 NLW_1 上的 *bit-sequence*,则 BS_{cur} 与 BS_{next} 之间除了最后 $|EW|$ 个位序列,其余完全相同. 因为我们要求 *item* 在下一个窗口上的规范度,也即基于 BS_{next} 求 *item* 的规范度,而在处理 BLW 时我们基于 BS_{cur} 已经得到了 *item* 在该窗口上的规范度,所以在基于 BS_{next} 求 *item* 的规范度时,我们可以先求得 BS_{next} 中后 $|EW|$ 个位序列上的规范度,把这个作为 1 个最终可能规范度. 另外需要注意的是,BS_{next} 的后 $|EW|$ 中的第 1 个非 0 位与前面序列中最后一个非 0 位之间的差值,也可能是规范度,所以我们设立了 *lastpos* 域,我们把这个作为第 2 个最终的可能规范度. 由定义 3 可知,这 2 个可能规范度和 BS_{cur} 上规范度 3 者中的最大值为最终 *item* 的 *reg* 值. 这样就使我们在只关注 *item* 在新界标窗口上 *bit-sequence* 中的后 $|EW|$ 个位序列的情况下,能够得到该项在新界标窗口上的规范度. 类似地,任意界标窗口上规范项都有这样的规律. 而和基于新界标窗口上该项的整个 *bit-sequence* 来计算该项的规范度相比,显然现有的方法可以更好地减少运算量.

3.3.2 Hash 索引表 HIT(Hash index table)

在基于 *BV* table 计算前一个窗口上的最大规范模式在当前窗口上的规范度,或者是求得当前窗口上由规范项所构成的任意模式的规范度时,我们需要随时能得到模式的每个项在当前窗口 *BV* table 中的 *bit-sequence*. 如果直接使用 *BV* table,因为求得每个项在当前窗口中的 *bit-sequence*,实际上都需要对 *BV* table 执行一次扫描操作,所以效率较低. 为解决这个问题,我们以项名为 *key*,以项在 *BV* table 中索引位置 *index* 为 *value* 设计了 Hash 索引表 HIT. 通过这个数据结构,我们能以 $O(1)$ 的时间开销找到当前窗口中任何项在 *BV* table 中的位置,进而得到该项的 *bit-sequence*. 图 2 说明了 LW_1 上 $\lambda=3$ 时的 HIT 与 *BV* table 间的关系.

性质 7. 每个窗口上 *BV* table 的 HIT 都是相同的,无需随着窗口成员的更新而更新.

key	*value*		*index*	*name*	*bit-sequence*	*rge*	*lastpos*	*sign*
A	1	→	1	*A*	111	1	3	true
B	2	→	2	*B*	011	2	3	true
C	3	→	3	*C*	001	3	3	true
D	4	→	4	*D*	100	2	1	true
E	5	→	5	*E*	011	2	3	true
F	6	→	6	*F*	010	2	2	true

Fig. 2 The relationship between the HIT and the *BV* table over the LW_1

图 2 LW_1 上的 HIT 与 *BV* table 间的关系

证明. 由 *BV* table 的更新过程可知,每个窗口上的 *BV* table 中项的个数与位置并不会发生变化,所以与每个窗口上 *BV* table 相关的 HIT 也不会发生变化,无需随着窗口成员的更新而更新. 证毕.

3.3.3 优化策略

性质 8. 在按照边界界标窗口技术求得当前窗口上的最大规范模式集合时,我们需要计算前一窗口上的所有最大规范模式在当前窗口上的规范度. 在这个过程中,假设前一个窗口上的最大规范模式中存在着某些成员项,这些项在当前窗口 *BV* table 中的 *sign* 域值为 false,则我们在处理该最大规范模式时可以忽略对于这些项的处理.

证明. 由规范模式向下闭包性特点,性质 4 的结论 3 以及性质 5 的结论 2 可得性质 8 成立. 证毕.

该优化策略可以减少我们在使用边界界标窗口技术求得界标窗口上最大规范模式过程中对于很多不必要成员项的处理,能够提高整个执行过程的时间效率.

我们以 LW_3 上最大规范模式的求解过程为例来说明优化策略的内容,表 3 是图 1 中 LW_3 上当 $\lambda=3$ 时的 *BV* table.

Table 3 *BV* table over the LW_3

表 3 LW_3 上的 *BV* table

name	*bit-sequence*	*reg*	*lastpos*	*sign*
A	111110000	4	0	false
B	011111001	3	3	true
C	001011101	3	3	true
D	100001	5	3	false
E	011110110	2	2	true
F	010100010	4	2	false

由前面可知 LW_2 上 $\lambda=3$ 的最大规范模式集合为 $RS_2=\{ABCE, ABEF\}$. 又由表 3 可知 *A*, *F* 在 LW_3 上 *BV* table 中的 *sign* 域值为 false,所以按照优化策略,当重新计算 *ABCE* 与 *ABEF* 在 LW_3 上

的规范度时,可以忽略对其中 A,F 的处理.也就是说,只需计算 BCE 和 BE 的规范度.对于 BCE 来说,因为它在 LW_3 上的规范度为 4,所以不满足规范条件.按照边界界标窗口技术考虑 BCE 长度为 2 的子模式:BC,BE,CE.因为 BC 的规范度为 3,CE 的规范度为 2,都满足规范条件,所以它们仍是当前窗口上 BCE 所有子模式的最大规范模式集合中的成员;对于 BCE 的另一个长度为 2 的子模式 BE,因为它在 LW_3 上的规范度为 4,所以不满足规范条件.按照边界界标窗口技术考虑 BE 的长度为 1 的项子模式:B,E.显然 B,E 此时都满足规范条件.令 $S_{\text{tem}}=\{BC,CE,B,E\}$,按边界界标窗口技术可知,当消除了 S_{tem} 中成员之间存在的超集关系后,$S_{\text{tem}}=\{BC,CE\}$即为当前窗口上 BCE 所有子模式的最大规范模式集合;对于 RS_2 中此时需要处理的另外一个模式 BE,类似地可以得到 $S1_{\text{tem}}=\{B,E\}$为当前窗口上 BE 所有子模式的最大规范模式集合.同样由边界界标窗口技术可知,假设 S_{tem} 与 $S1_{\text{tem}}$ 中成员构成了新集合 S_{final},即 $S_{\text{final}}=\{BC,CE,B,E\}$,那么当消除了该集合中成员间存在的超集关系后,该集合便是 LW_3 上当 $\lambda=3$ 的最大规范模式集合 RS_3,也即 $RS_3=\{BC,CE\}$.

3.4 DSMRM-BLW 算法

由边界界标窗口技术可知,在使用 naïve 算法求得第 1 个界标窗口上最大规范模式集合后,我们可以基于该集合求得相邻下一个窗口上的最大规范模式集合.依次类推,可以得到之后每个界标窗口上的最大规范模式集合.另外通过文献[1,6]可知,NI 算法在第 1 个窗口上的执行效果与文献[1]中的 RPS-tree 算法相同,又由文献[8]可知,PA 算法求得滑动窗口上规范模式的时间开销小于 RPS-tree 算法.所以我们可以在第 1 个窗口上执行 NP 算法,即使用 PA 算法求得该窗口上的规范模式集合,然后消除该集合成员间存在的超集关系,以此来得到该窗口上的最大规范模式集合,对于之后每一个界标窗口上的最大规范模式集合,使用边界界标窗口技术来获得,我们把这种算法称为 DSMRM-BLW 算法.另外由前面 BV table 结构可知,该数据结构中含有 PA 算法执行所需的 *bit-sequence*,所以 DSMRM-BLW 算法在第 1 个窗口上的执行无需再构造其他数据结构,直接使用 BV table 中的 *bit-sequence* 即可.

DSMRM-BLW 算法在前面数据结构的基础上,完成了基于边界界标窗口技术挖掘每个界标窗口上数据流最大规范模式的操作.算法具体描述如下:

算法 1. DSMRM-BLW.

输入:数据流 DS、界标窗口 LW 的起始位置 sp、规范阈值 λ、基本窗口大小 $|EW|$;

输出:每个界标窗口上的最大规范模式集合.

① 从数据流中索引为 sp 的成员开始,对每个界标窗 LW 执行如下操作;
② { if (the LW is BLW)
③ {构建 BLW 上的 BV table *curtable* 与 HIT;
④ $S_{\text{regular}}=PA(curtable)$; /* 使用 PA 算法得到当前窗口上完整的规范模式集合 */
⑤ 消除 S_{regular} 中任意 2 个成员间存在的超集关系;
⑥ $S_{\text{maxregular}}=S_{\text{regular}}$;
⑦ output $S_{\text{maxregular}}$; /* 输出 BLW 上的最大规范模式集合 */
⑧ }
⑨ else
⑩ {基于新 EW 的成员,按照 3.3.1 节 3)的方法更新 *curtable*;
⑪ $S'_{\text{maxregular}}=\text{null}$;
⑫ for ($S_{\text{maxregular}}$ 中的每个元素 *eset*)
⑬ {计算 *eset* 的规范度 R_{eset};
⑭ if($R_{eset}<\lambda$)
⑮ $S'_{\text{maxregular}}\leftarrow eset$;
⑯ else
⑰ $S'_{\text{maxregular}}\leftarrow$GetMRM-subset($\lambda$, *eset*, *curtable*); /* 得到 *eset* 所有子规范模式的最大规范模式,并将其放入 $S'_{\text{maxregular}}$ */
⑱ }
⑲ 消除 $S'_{\text{maxregular}}$ 中任意 2 个成员间存在的超集关系;
⑳ output $S'_{\text{maxregular}}$; /* 输出新界标窗口上的最大规范模式集合 */
㉑ $S_{\text{maxregular}}\leftarrow S'_{\text{maxregular}}$;
㉒ }
㉓ }

该算法会首先构建 BLW 中数据流成员的 BV table 和 HIT 结构(行③);接着基于该结构按文献[8]中规范模式的获取方法,求得该窗口上的规范模式集合(行④);当消除了该集合中规范模式间的超集关系后,即可得到该窗口上的最大规范模式集合 $S_{\text{maxregular}}$(行⑤~⑦).对于 BLW 后的窗口,首先使用新 EW 中成员信息按照 3.3.1 节 3)中所述的

方法更新前一窗口上的 BV table,由此得到当前界标窗口上的 BV table(行⑩);然后对 $S_{\text{maxregular}}$ 中的每个模式,按照前面的边界界标窗口技术与优化策略对其进行处理,如果该模式被处理后的规范度小于 λ.则将该模式放入当前窗口的 $S_{\text{maxregular}}$ 即 $S'_{\text{maxregular}}$ 中(行⑫～⑮);否则求得该模式中所有子规范模式的最大规范模式,并将它们放入 $S'_{\text{maxregular}}$ 中(行⑯～⑰);当处理完 $S_{\text{maxregular}}$ 中每个模式后,只要消除 $S'_{\text{maxregular}}$ 中所有模式间的超集关系,就可以得到当前窗口上最大规范模式集合(行⑲～⑳).

DSMRM-BLW 算法在使用边界界标窗口技术求得当前窗口上最大规范模式集合时,如果前一窗口上最大规范模式在当前窗口上不满足规范条件,则需要求得该模式在当前窗口上所有子规范模式的最大规范模式.下面的 GetMRM-subset 算法给出了具体求解过程.

算法 2. GetMRM-subset.

输入:规范阈值 λ、在当前窗口上不满足规范条件的前一窗口上最大规范模式 *eset*、当前窗口上的 BV table *curtable*;

输出:*eset* 所有子规范模式的最大规范模式集合 MRS_{eset}.

① for (*eset* 每个长 $len_{eset}-1$ 的子集 *esubset*)
② { 计算 *esubset* 的规范度 $reg_{esubset}$;
③ if ($reg_{esubset}<\lambda$) /* 判断 *esubset* 的规范度 $reg_{esubset}$ 是否小于 λ */
④ $MRS_{eset}\leftarrow esubset$;
⑤ else
⑥ $MRS_{eset}\leftarrow$GetMRM-subset (λ, *esubset*, *curtable*);
⑦ }
⑧ 消除 MRS_{eset} 中任意 2 个成员间的超集关系;
⑨ return MRS_{eset}.

设 *eset* 的长度为 len_{eset},对于 *eset* 中每个长为 $len_{eset}-1$ 的子集 *esubset* 执行如下步骤:如果 *esubset* 满足规范条件,则将该子集放入 MRS_{eset} 中(行③～④);如果 *esubset* 的规范度不满足规范条件,设 $len_{esubset}$ 是 *esubset* 的长度,则使用相同方法对于 *esubset* 的长度为 $len_{esubset}-1$ 的子集进行处理(行⑥);当处理完 *eset* 中每个长为 $len_{eset}-1$ 的子集后,只需消除 MRS_{eset} 中成员间的超集关系,便可得到 *eset* 中所有子规范模式的最大规范模式集合 MRS_{eset}(行⑧～⑨).

4 实验及结果分析

本部分实验主要分 3 部分:1)实验验证最大规范模式相对于规范模式的优势,这也是本文的研究意义所在;2)验证本文提出的 DSMRM-BLW 算法在挖掘界标窗口上数据流最大规范模式时的有效性;3)对 DSMRM-BLW 算法性能进行了评价.

4.1 实验环境

本文全部实验都在一台配置为 Intel® core™ i5 CPU 650 @3.20 GHz CPU、1.12 GHz 主频、4 GB 内存、Windows XP 的 PC 机上执行.算法由 VC++6.0 实现.实验使用了频繁模式挖掘公用数据集中的 mushroom 和 retail 数据集.其中 mushroom 是 1 个稠密数据集,共有 8 124 条记录,大小为 0.54 MB,*item* 的个数为 120,平均记录长度为 23(即含有 23 个 *item*);retail 是 1 个稀疏数据集,共有 88 162 条记录,大小为 3.97 MB,16 470 个 *item*,平均每条记录的大小为 13;另外在实验中,我们设定界标窗口的起始位置都为数据样本中第 1 个成员的位置.因为当前并没有关于这个方向的研究,所以实验中的对比算法为前面第 3 节提到的 NI 与 NP2 类 naïve 算法(其中基于 IncRT 算法的方法被称为 NI 算法;而基于 PA 算法的方法被称为 NP 算法).

4.2 最大规范模式相对于规范模式的优点

本节就最大规范模式相对于规范模式的优势作了实验验证.具体实验方法是首先对数据样本分别执行 NI 与 IncRT 算法以得到最大规范模式集合 A 与规范模式集合 B;接着比较这 2 个结果集合中成员的内容与数目.为便于说明,我们定义了 2 个参数:1)$R_{\text{accommodate}}$,该参数用于表征集合 A 对于集合 B 的容纳程度;2)$R_{\text{reduction}}$,用于描述集合 A 相对于集合 B 的约减程度.设 $|A|$ 与 $|B|$ 分别表示集合 A,B 中的成员个数,又设变量 $N_{\text{accommodate}}$ 表示集合 A 所容纳集合 B 中成员的数目,该变量的更新变化规则是,对于集合 B 中的每个成员 $element_b$,如果能在 A 中找到一个成员 $element_a$,使得 $element_b$ 是 $element_a$ 的子集,则令 $N_{\text{accommodate}}$ 加 1.基于这样的规定,我们可以将集合 A 相对于集合 B 的 $R_{\text{accommodate}}$ 与 $R_{\text{reduction}}$ 具体定义描述如下:

$$R_{\text{accommodate}}=|N_{\text{accommodate}}|/|B|, \tag{1}$$

$$R_{\text{reduction}}=|A|/|B|. \tag{2}$$

由上面的定义描述可知,如果实验结果中能够

保持 $R_{accommodate}=1$,且 $R_{reduction}<1$,则可以说明最大规范模式集合具有全部规范模式集合的信息,同时规模比规范模式集合要小.具体针对不同真实数据样本的实验结果,如图 3 所示:

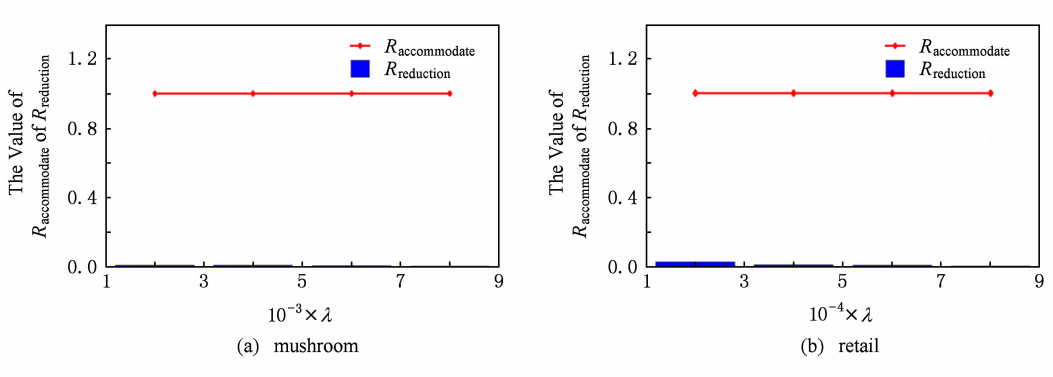


Fig. 3 The $R_{accommodate}$ and $R_{reduction}$ about maximal regular patterns over the regular ones when different data samples are taken

图 3 不同数据样本下最大规范模式相对于规范模式的 $R_{accommodate}$ 与 $R_{reduction}$

由图 3 可以看到,在不同数据样本下,最大规范模式集合相对于规范模式集合的 $R_{accommodate}$ 始终为 1,而 $R_{reduction}$ 取值也远小于 1.由前面分析可知,这样的实验结果足以说明最大规范模式集合含有规范模式集合所有的信息,同时数量规模上也小于规范模式集合,所以,当我们只想定性地确定数据集中哪些是规范模式,而不关注具体每个规范模式规范度时,挖掘最大规范模式更有效率,这也是本文的研究意义所在.

4.3 DSMRM-BLW 算法的有效性

本节实验主要用于验证 DSMRM-BLW 算法在挖掘界标窗口上数据流最大规范模式时的有效性.这里我们使用的方法是将 DSMRM-BLW 算法的执行结果与最基本的 naïve 算法执行结果进行比较.为了便于比较,我们定义了 DSMRM-BLW 算法相对于其他算法的查全率与查准率.具体定义如下.设 DS_{result} 与 N_{result} 分别为相同参数下 DSMRM-BLW 和 naïve 算法的执行结果集合,$|DS_{result}|$ 与 $|N_{result}|$ 分别是这 2 个集合中成员数目.令 V_{equal} 是这 2 个集合中相等的成员个数.这里定义 DS_{result} 中任一成员 A 与 N_{result} 中成员 B,如果它们对应的窗口起始位置相同,且 A 与 B 也相同,则 A,B 相等;如果二者中有 1 项不同,则 A,B 不相等.基于这些我们定义了 DSMRM-BLW 相对于 naïve 算法的查全率(recall)与查准率(precision),分别用 R_{recall} 与 $R_{precision}$ 表示.使用它们来评价 DSMRM-BLW 算法的有效性.

$$R_{recall}=V_{equal}/|N_{result}|, \tag{3}$$

$$R_{precision}=V_{equal}/|DS_{result}|. \tag{4}$$

表 4 给出了 mushroom 与 retail 样本下 DSMRM-BLW 算法分别相对于 NI 与 NP 算法的查全率与查准率.表 4 中我们用 $R_{recall,NI}$ 和 $R_{precision,NI}$ 分别表示 DSMRM-BLW 算法相对于 NI 算法的查全率与查准率,而用 $R_{recall,NP}$ 和 $R_{precision,NP}$ 分别表示 DSMRM-BLW 算法相对于 NP 算法的查全率与查准率.

Table 4 Parameters and Results in Experiment 2

表 4 实验 2 中的参数及执行结果

Datasample	$\|EW\|$	λ	$R_{precision,NI}$	$R_{precision,NP}$	$R_{recall,NI}$	$R_{recall,NP}$
mushroom	500	100	1	1	1	1
mushroom	1 000	200	1	1	1	1
mushroom	1 500	300	1	1	1	1
retail	500	100	1	1	1	1
retail	1 000	200	1	1	1	1
retail	1 500	300	1	1	1	1

由表 4 可见,在不同数据样本及参数设置下,DSMRM-BLW 算法相对于 naïve 算法的 R_{recall} 与 $R_{precision}$ 的取值都为 1,也即 DSMRM-BLW 算法与 naïve 算法的执行结果相同,即 DSMRM-BLW 算法能正确地得到界标窗口下数据流的最大规范模式.

4.4 DSMRM-BLW 算法的性能评价

本节对 DSMRM-BLW 算法的时间与空间性能作了评测.因为目前并没有关于这方面的研究,所以使用的对比算法为前面提到的 2 类 naïve 算法.具体来说,我们在 4.4.1 节讨论了不同 $|EW|$,λ 以及样本大小 L_{sample} 与算法执行时间间的关系;而在 4.4.2 节讨论了不同 $|EW|$,λ 以及样本大小 L_{sample}

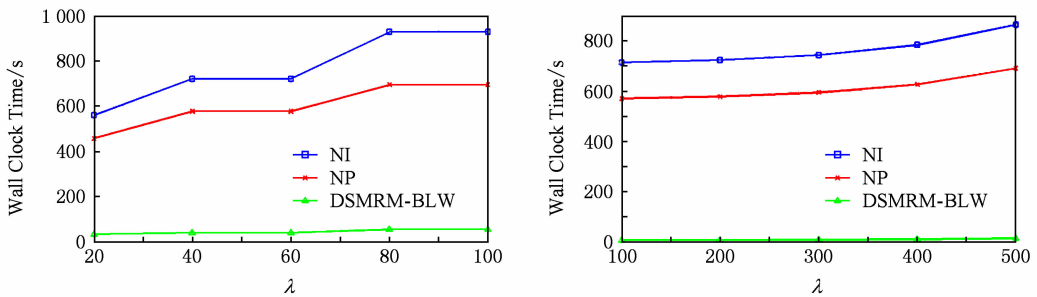
与算法空间开销间的关系.

4.4.1 DSMRM-BLW 算法的时间开销分析

首先分析不同 λ 大小与算法执行时间之间的关系. 实验效果如图 4 所示.

由图 4 可见,2 类样本下 DSMRM-BLW 算法执行时间远小于其他 2 类算法. 这说明边界界标窗口技术是有效的. 另外还可以看到,2 类样本下 3 种算法执行时间都随 λ 大小的增加而增加. 这是因为 λ 越大,每个界标窗口中满足规范度小于 λ 条件的成员就越多. 也即在同一窗口中,λ 变大后满足规范度小于 λ 的成员个数大于等于 λ 变大前满足规范度小于 λ 的成员个数. 对于 NI 算法,满足规范度小于 λ 的成员个数越多,算法需要处理的成员就越多,得到的规范模式数目也会增加. 而因为我们需要消除所有规范模式间存在的超集关系,所以这时需要执行该操作的成员数目变得更多,算法时间会增加. 同理对于 NP 算法,当满足规范度小于 λ 的成员个数增加时,算法中也增加了基于 *bit-sequence* 执行逻辑与操作的成员个数,得到的规范模式数目也会增加,而对于所有这些规范模式,NP 算法也会通过消除它们间存在的超集关系来得到最大规范模式集合,所以算法时间会增加;DSMRM-BLW 算法在第 1 个窗口上的情况与 NP 算法一样,之后每个窗口上会使用边界界标窗口技术来处理,所以算法时间开销主要决定于第 1 个窗口. 由前面分析可知,该窗口上算法时间会随着满足规范度小于 λ 的成员个数增加而增加. 另外当 λ 取值的增加并没有使窗口中满足规范度小于 λ 的项成员个数增加时,算法在这些窗口上的执行情况在 λ 取值改变前后变化不大(如图 4(a)中的 λ 分别在 40～60,80～100 中取值);当 λ 取值的增加使窗口中满足规范度小于 λ 的项成员个数增加时,算法在这些窗口上执行时间会增加.

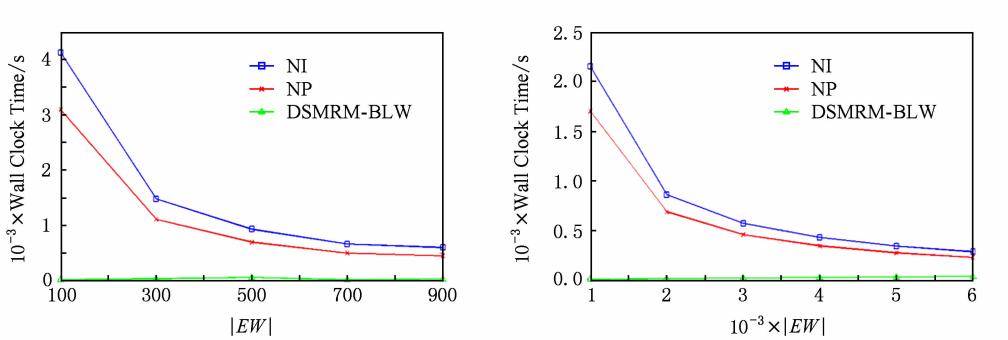
其次分析不同 $|EW|$ 大小与算法执行时间之间的关系. 实验效果如图 5 所示.



(a) mushroom ($|EW|=500$) (b) retail ($|EW|=2\,000$)

Fig. 4 The comparison of the execution time about different algorithms when λ changes

图 4 算法在 λ 变化时的执行时间比较



(a) mushroom ($\lambda=100$) (b) retail ($\lambda=500$)

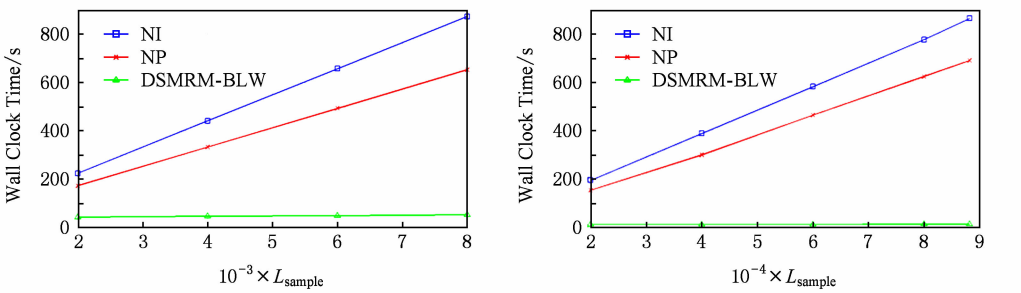
Fig. 5 The comparison of the execution time about different algorithms when $|EW|$ changes

图 5 算法在 $|EW|$ 变化时的执行时间比较

由图 5 可见,2 类样本下 DSMRM-BLW 算法执行时间小于其他 2 类算法,这说明边界界标窗口技术是有效的. 另外可以看到 NI 与 NP 算法时间都随 $|EW|$ 的增加而减少,这是因为对于同一个数据流来说,$|EW|$ 越大,算法在该数据流上执行最大规范模式挖掘的次数就越少,而且对于数据流上起始位置不变、终止位置以 $|EW|$ 为单位不断滑动的界标窗口来说,当 $|EW|$ 小时,naïve 算法在数据流上执

行挖掘的总次数以及所处理数据流成员的总量,大于$|EW|$变大后算法所执行挖掘的总次数以及所处理的数据流成员的总量,所以它们的时间都随窗口大小的增加而减少. DSMRM-BLW 算法在第 1 个窗口上与 NP 算法一样,之后每个窗口上会使用边界界标窗口技术来处理,所以算法时间开销主要决定于第 1 个窗口. 因为当$|EW|$变大时,变化后第 1 个窗口上的规范项数目小于等于变化前该窗口上的规范模式数目,所以对于稠密样本下第 1 个窗口上的 DSMRM-BLW 算法来说,当该窗口上规范模式数目不变时,因为窗口大小的增加会使该窗口上参与运算的 *bit-sequene* 规模变大,所以执行时间会增加(图 5(a)中$|EW|$大小在 100～500,700～900);当该窗口上的规范模式数目减少时,特别是当窗口大小的增加使得原第 1 个窗口上本来规范的项变得不规范时,界于样本自身的稠密特征,该算法在新窗口上的规范模式集合与原窗口上的规范模式集合相比,会少了很多与被删除项有关的规范模式,这样后期再执行消除规范模式集合中成员间超集关系的操作,时间开销会少很多,所以此时算法时间开销会减少(图 5(a)中$|EW|$大小在 500～700). 对于稀疏样本下第 1 个窗口上的 DSMRM-BLW 算法,样本稀疏的特点会使得该窗口上的规范模式不多,同时数据结构的规模会很大,所以该窗口上的时间开销更多的是在于挖掘规范模式本身所花费的时间,消除所有规范模式间超集关系所花费的时间反而不多. 当$|EW|$增加时,因为第 1 个窗口上数据结构的规模会变大,所以执行时间会增加. 另外我们可以看到 3 种算法彼此间的时间差异会随$|EW|$的减少而增加,这是因为$|EW|$越小,算法执行挖掘的窗口及次数就越多,每个窗口上这 3 类算法执行时间都会有差异,随着窗口数目的增加,这种差异的累积就会越来越大,所以它们间执行时间的差别也会越来越大.

最后分析样本大小 L_{sample} 与算法执行时间之间的关系. 实验效果如图 6 所示:



(a) mushroom($\lambda=100,|EW|=500$) (b) retail($\lambda=500,|EW|=2\,000$)

Fig. 6 The comparison of the execution time about different algorithms when L_{sample} changes

图 6 算法在 L_{sample} 变化时的执行时间比较

由图 6 可见,2 类样本下 DSMRM-BLW 算法执行时间小于其他 2 类算法,这说明边界界标窗口技术是有效的. 另外可以看到这 3 类算法时间都随样本大小的增加而增加,这是因为对于同一个数据流来说,样本越大,算法在该样本上需要处理的数据流成员就越多,所以执行时间会随着样本大小的增加而增加;另外由图 6 中还可以看到,3 类算法彼此间的时间差异会随着样本大小的增加而增加. 这是因为样本越大,算法执行挖掘的窗口及次数就越多,每个窗口上这 3 类算法执行时间都会有差异,随着窗口数目的增加,这种差异的累积就会越来越大,所以它们间执行时间的差别也会越来越大.

4.4.2 DSMRM-BLW 算法的空间开销分析

1) 分析不同 λ 大小与算法空间开销之间的关系. 实验效果如图 7 所示.

由图 7 可以看到,在空间开销上 DSMRM-BLW<NP<NI. 这是因为 NI 算法在每个窗口上需要维护一棵前缀树,另外还需要保留该窗口上的完整规范模式集合,除此之外,NI 算法还需要开辟额外空间用于在当前窗口完整规范模式集合基础上求得最大规范模式集合,所以空间开销最大;NP 算法在每个窗口上除了需要保留该窗口上所有项的 *BV* table,还需要开辟空间用于在当前窗口所有规范模式的基础上得到最大规范模式集合,但它在获得每个窗口上规范模式时,无需像 NI 算法一样保留前一窗口上的所有规范模式,所以空间开销比 NI 算法小;DSMRM-BLW 算法在第 1 个窗口上执行情况与 NP 算法相同,所以该窗口上的空间开销除了 *BV* table

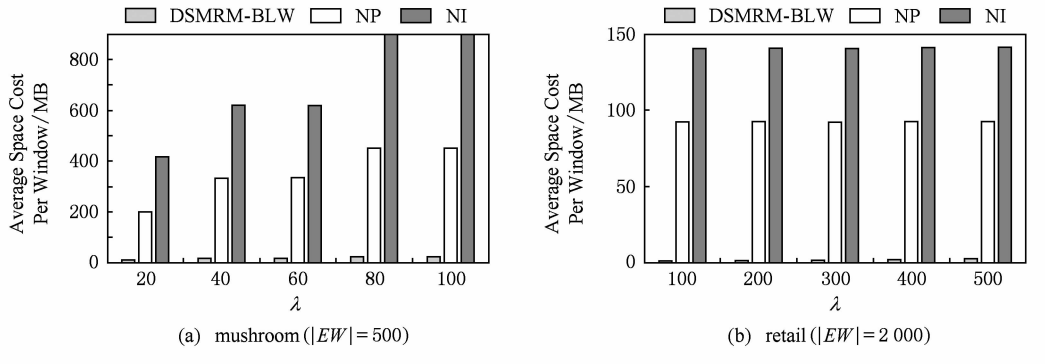


Fig. 7 The comparison of the space consumption about different algorithms when λ changes

图 7 算法在 λ 变化时的空间开销比较

以及与之相关的 HIT 外,还包括容纳所有规范模式的空间,此后每个窗口上的空间开销只有该窗口上的最大规范模式集合与 *BV* table 以及与之相关的 HIT,又因为边界界标窗口技术的原因,这些窗口上 *BV* table 中项的个数和类型都和第 1 个窗口上 *BV* table 中项成员相同,而且最大规范模式集合的规模小于规范模式集合的规模,所以平均每个窗口的开销小于 NP 算法.

另外从图 7 可以看到,3 种算法空间开销随 λ 大小增加而单调增加. 对于稠密样本下的 NI 与 NP 算法,样本的稠密性会使算法的空间开销决定于每个窗口上所有规范模式的规模. 而当 λ 增加时,每个窗口中满足规范度小于 λ 的项成员数目可能会有 2 种变化:①数量保持不变;②数量增加. NI 算法需要保留每个窗口上完整的规范模式集合,同时还需要开辟空间用于在该集合基础上得到最大规范模式集合,所以如果 λ 的增加没有使窗口中满足规范条件的成员数目发生变化,则 NI 算法用于存放规范模式集合的空间保持不变(如图 7(a)中 λ 取值在 40～60 和80～100 区间);如果 λ 的增加使得窗口中满足规范条件的成员数目增加,则 NI 算法用于存放规范模式的空间就会增加. 对于 NP 算法,因为 NP 算法需要对每个窗口上所有项构建 *bit-sequence* 序列,另外 NP 算法也要开辟空间基于每个窗口上的完整规范模式集合来得到最大规范模式集合,所以 NP 算法与 NI 算法一样,空间开销都会随着 λ 取值的增加而单调递增. DSMRM-BLW 算法在第 1 个窗口上执行情况与 NP 算法一样,所以该窗口上空间开销会随着 λ 的增加而单调递增. 又因为该算法在其他窗口上使用边界界标窗口技术,所以空间开销并不大,也即此时整个算法的空间开销集中在第 1 个窗口上,于是空间开销会增加. 总体来看,DSMRM-BLW 算法在每个窗口上的空间开销会随着 λ 取值的增加而单调递增. 对于稀疏样本下的算法,样本的稀疏性会使得每个窗口上规范模式的数目较少,且数据结构规模较大. 这种情况下算法的空间开销决定于每个窗口上数据结构的规模. 又因为 3 类算法在每个窗口上的数据结构与 λ 无关,所以可以看到它们在每个窗口上的空间开销基本相同.

2) 分析不同 $|EW|$ 大小与算法空间开销间的关系. 实验效果如图 8 所示.

由图 8 可以看到,DSMRM-BLW 算法具有最

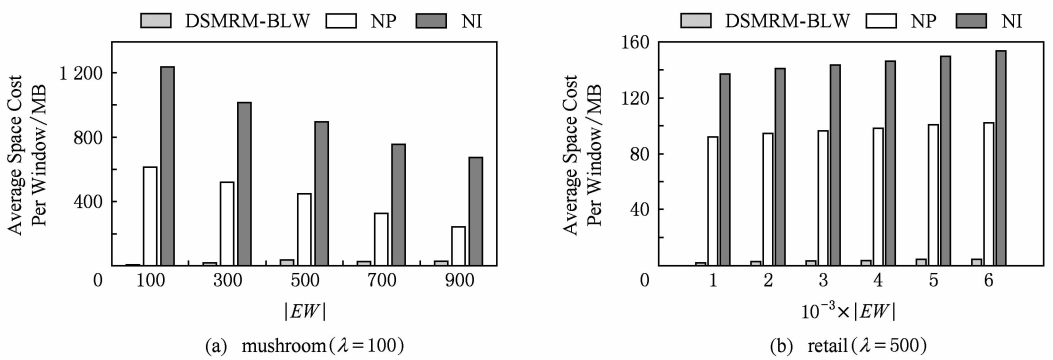
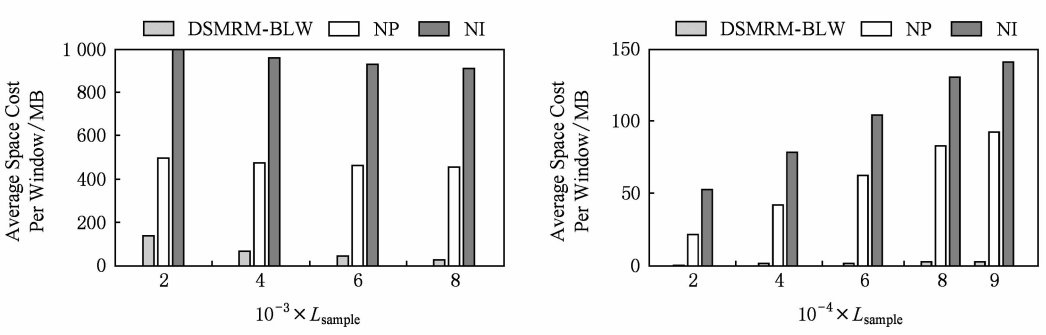


Fig. 8 The comparison of the space consumption about different algorithms when $|EW|$ changes

图 8 算法在 $|EW|$ 变化时的空间开销比较

好的空间开销,而且不同算法间空间开销的关系与 λ 变化时所呈现的状态一样. 具体原因与在 λ 变化时不同算法空间开销间关系分析中已有说明,篇幅原因,这里不再赘述. 稠密样本下 2 种 naïve 算法在每个窗口上的空间开销,都随 $|EW|$ 增加而减小. 这是因为对于界标窗口而言,当 λ 确定时,基本窗口较大情况下界标窗口中规范模式数目会小于等于基本窗口较小情况下界标窗口中的规范模式数目,所以用于存放小基本窗口下界标窗口中所有规范模式的空间开销大于等于大基本窗口下界标窗口中用于存放所有规范模式的空间开销. 对于 DSMRM-BLW 算法,因为 mushroom 样本稠密性的特点,该算法在第 1 个窗口上的空间开销远远大于其他窗口上的空间开销,这时该算法的空间开销主要决定于这个窗口,当 $|EW|$ 增加时,需要处理的界标窗口数会变小,这样平均每个窗口上的空间开销就会增加;retail 样本下算法的空间开销都随 $|EW|$ 增加而单调增加. 这是因为样本稀疏的特点会使得每个界标窗口上满足规范条件的项变得很少,而且同时又会使窗口上的数据结构规模增加,所以这种情况下算法空间开销决定于每个窗口的数据结构规模. 对于 NI 算法,因为窗口越大,其中成员越多,每个窗口上的前缀树规模就越大,所以空间开销会增加;对于 NP 算法与 DSMRM-BLW 算法,因为窗口越大,每个窗口上 BV table 规模就越大,所以空间开销单调增加.

3) 分析样本大小 L_{sample} 与算法空间开销之间的关系. 实验效果如图 9 所示:



(a) mushroom($\lambda=100, |EW|=500$) (b) retail($\lambda=500, |EW|=2\,000$)

Fig. 9 The comparison of the space consumption on different algorithms when L_{sample} changes

图 9 算法在 L_{sample} 变化时的空间开销比较

由图 9 可见,DSMRM-BLW 算法具有最好的空间开销;另外还可以看到稠密样本下 3 种算法空间开销都随样本大小增加而单调递减. 对于 2 类 naïve 算法,因为随着样本大小的增加,会出现更多需要处理的界标窗口,而且这些窗口大小会越来越大,窗口上规范模式的数目会小于等于之前出现的窗口上规范模式的数目,所以平均每个窗口上的空间开销会单调减少;对于 DSMRM-BLW 算法,因为该样本下的空间开销决定于第 1 个界标窗口,所以随着窗口数目的增加,平均每个窗口的空间开销会越来越小. 另外还可以看到稀疏样本下算法的空间开销都会随着样本大小的增加而单调增加. 这是因为这类样本下算法的空间开销主要决定于每个窗口上的数据结构规模. 对于 NI 算法,因为每个新增加窗口上的前缀树规模大于等于相邻前一个窗口上的前缀树规模,所以在样本大小增加过程中,每个窗口上平均空间开销会增加;对于 NP 算法,因为每个新增加窗口上的 BV table 规模会大于相邻前一个窗口上的 BV table 规模,所以在样本大小增加过程中,每个窗口上的平均空间开销也会增加;同理 DSMRM-BLW 算法在样本大小增加过程中,每个窗口上平均空间开销也会增加.

总之,由本节实验可以看到:①在需要定性确定数据集上哪些成员是规范模式时,挖掘最大规范模式有更好的效率,能在不减少规范模式所蕴含信息的同时,极大减少需要挖掘的模式数量;②DSMRM-BLW 算法在挖掘界标窗口下数据流最大规范模式时,与 naïve 算法执行结果相同;③同 2 类 naïve 算法相比,DSMRM-BLW 算法具有更好的时间与空间效率.

5 总结与展望

本文首次对于界标窗口下数据流最大规范模式挖掘问题进行了研究. 首先给出了该问题的形式化描述;接着对处理该问题的 naïve 算法中相邻窗口上最大规范模式间的关系进行了分析,得到了边界

界标窗口技术,并由此提出了 DSMRM-BLW 算法,与 naïve 算法相比,该算法在保持查询结果不变的同时,很好地减少了时间与空间开销;最后通过对于公有数据样本的实验证明了该算法的有效性. 因为规范模式与频繁模式二者都具有向下闭包的特点,所以很多与数据流最大频繁模式挖掘相关的技术应该可以直接或间接地应用于数据流最大规范模式的挖掘中,所以未来研究中我们会着重分析这二者间的关系,争取进一步提高 DSMRM-BLW 算法的执行效率;另外,界标窗口中历史数据的重要性会随着时间的推移而减小. 为了能够体现出这个特点,可以将衰减模型与 DSMRM-BLW 算法相结合,这也是未来应该着手做的一个研究方向.

参 考 文 献

[1] Tanbeer S K, Ahmed C F, Jeong B S. Mining regular patterns in data streams [C] //Proc of the 15th Int Conf on Data Systems for Advanced Applications. Berlin: Springer, 2010: 399-413

[2] Li Guohui, Chen Hui. Mining the frequent patterns in an arbitrary sliding window over online data stream [J]. Journal of Software, 2008, 19(10): 2585-2596 (in Chinese)
(李国徽, 陈辉. 挖掘数据流任意滑动窗口内频繁模式[J]. 软件学报, 2008, 19(10): 2585-2596)

[3] Yun U, Lee G, Ryu K H. Mining maximal frequent patterns by considering weight conditions over data streams [J]. Knowledge-Based Systems, 2014, 55: 49-65

[4] Lee G, Yun U, Ryu K H. Sliding window based weighted maximal frequent pattern mining over data streams [J]. Expert Systems with Applications, 2014, 41(2): 694-708

[5] Hang Meng, Wang Zhihai, Yuan Jidong. Efficient method for mining closed frequent patterns from data streams based on time decay model [J]. Chinese Journal of Computers, 2015, 38(7): 1473-1483 (in Chinese)
(韩萌, 王志海, 原继东. 一种基于时间衰减模型的数据流闭合模式挖掘方法[J]. 计算机学报, 2015, 38(7): 1473-1483)

[6] Tanbeer S K, Ahmed C F, Jeong B S. Mining regular patterns in incremental transactional databases [C] //Proc of the 12th Asia-Pacific Web Conf. Piscataway, NJ: IEEE, 2010: 375-377

[7] Kumar G V, Sreedevi M, Kumar N P. Mining regular patterns in data streams using vertical format [J]. International Journal of Computer Science and Security, 2012, 6(2): 142-149

[8] Verma V K, Saxena K. Mining regular pattern over dynamic data stream using bit stream sequence [J]. International Journal of Innovative Technology and Exploring Engineering, 2013, 3(7): 7-10

[9] Kumar G V, Kumari V V. Sliding window technique to mine regular frequent patterns in data streams using vertical format [C] //Proc of the Int Conf on Computational Intelligence and Computing Research. Piscataway, NJ: IEEE, 2012: 1-4

[10] Sreedevi M, Reddy L S S. Mining closed regular patterns in data streams [J]. International Journal of Computer Science & Information Technology, 2013, 5(1): 171-179

[11] Rashid M M, Karim M R, Jeong B S, et al. Efficient mining regularly frequent patterns in transactional databases [C] //Proc of the 17th Int Conf on Database Systems for Advanced Applications. Berlin: Springer, 2012: 258-271

[12] Rashid M M, Gondal I, Kamruzzaman J. Regularly frequent patterns mining from sensor data stream [C] //Proc of the 20th Int Conf on Neural Information Processing. Berlin: Springer, 2013: 417-424

[13] Agrawal R, Srikant R. Fast algorithms for mining association rules in large database [C] //Proc of the 20th Int Conf on Very Large Data Bases. San Francisco, CA: Morgan Kaufmann, 1994: 1-32

Wen Yingyou, born in 1974. PhD and post doctor of Northeastern University. Member of CCF. His main research interests include next generation network, wireless sensor network, network security and network management (wenyy@neusoft.com).

Wang Shaopeng, born in 1984. PhD from Northeastern University. Member of CCF. His main research interests include data stream and wireless sensor network.

Zhao Hong, born in 1954. Professor and PhD supervisor of Northeastern University. Senior member of CCF. His main research interests include computer network security and information security, distributed multimedia, and network management (zhaoh@neusoft.com).