一种高效的离线数据流频繁模式挖掘算法

侯 伟[1] 吴晨生[2] 杨炳儒[1] 方炜炜[1]
(北京科技大学信息工程学院 北京 100083)[1] (北京市科学技术情报研究所 北京 100037)[2]

摘 要 数据流频繁模式挖掘是当前数据挖掘领域中的研究热点之一,数据流连续性、无序性、无界性及实时性的特点为挖掘算法在时间及空间性能方面提出了更高的要求。数据流中模式频度的震荡现象,迫使现有算法对概要数据结构频繁维护,致使其时间、空间效率均受到较大影响。构造了具备较高空间性能的概要数据结构SP-tree,同时定义了震荡性因子χ以量化震荡信息,提出了一种高效的离线数据流频繁模式挖掘算法SPDS,有效降低了数据震荡对算法性能的影响;在处理新到数据集时,算法采取分而治之的分离映射策略,进一步提升了时间效率;同时在查询结果方面提高了部分模式的计数精度。

关键词 数据挖掘,数据流,频繁模式,震荡性因子
中图法分类号 TP182 **文献标识码** A

Efficient Algorithm for Mining Frequent Patterns over Offline Data Streams

HOU Wei[1] WU Chen-sheng[2] YANG Bing-ru[1] FANG Wei-wei[1]
(School of Information Engineering, University of Science and Technology Beijing, Beijing 100083, China)[1]
(Beijing Municipal Institute of Science and Technology Information, Beijing 100037, China)[2]

Abstract Mining frequent patterns from data streams is one of the hottest research topics in data mining nowadays. The features of data streams, such as consecution, disorder and real-time, raise requirements for higher time and space performance of mining algorithms. Vibration of pattern frequency in data streams, compels the present algorithms to revise the synopsis structure continually, and leads up to disadvantage impact on both time and space efficiency. A more scalable synopsis structure SP-tree was designed firstly, meanwhile the concept of vibration factor χ was given for maintaining vibrational information. Then an efficient algorithm for mining frequent patterns over offline data streams SPDS was proposed, which relieves the performance from the impact of vibration effectively, and increases the count accuracy of partial patterns. This algorithm adopts a divide-and-conquer mechanism to mine the current dataset, thereby improves itself further.

Keywords Data ming, Data stream, Frequent pattern(FP), Vibration factor

随着信息技术的发展,海量数据库迅速增加,对其有效的分析处理技术的缺乏逐渐显现。在需求的推动下,数据库中知识发现(Knowledge Discovery in Databases,KDD)技术应运而生。数据挖掘(Data Mining,DM)是KDD中的重要过程,在该过程中系统采用智能算法从数据中提取有益的数据模式[1]。频繁模式(Frequent Pattern)挖掘是DM中重要的研究问题,该领域已出现许多高效算法,如FP-growth,Depth First等。目前的研究[2,3]表明,大量数据以数据流(Data Stream)的形式产生、维护与管理,如网络数据、交易数据等。区别于传统的静态数据,数据流具有连续性、无序性、无界性及实时性的特点,通常要求算法能够实时地反映模式变化,为挖掘数据流中的知识带来了研究挑战。数据流处理已经成为当前数据挖掘领域的一个研究热点。发现数据流中的频繁模式是数据流挖掘中最基本的问题之一。

根据数据流特点,文献[4]指出数据流挖掘算法须克服4个问题:1)数据最多访问一次;2)概要维护以往产生的数据;3)高效处理到达数据;4)在线分析挖掘结果。为了获得较高的时空效率,数据流挖掘算法往往需要牺牲部分准确性[5]。依数据产生方式,数据流可分为在线数据流(Online Data Stream)与离线数据流(Offline Data Stream)。前者的频繁模式挖掘研究起步较早,已有一些比较成熟的算法,如Carma[6],estDec[7]以及Manku提出的Lossy Counting算法[8]等。相对在线数据流,离线数据流到达计算终端的数据量更大、更加随机,对瞬时计算能力有更高的要求。C. Giannella等提出的FP-stream算法[9],是目前离线数据流频繁模式挖掘算法中最具代表性的。

在现实数据流中,模式的频繁性质并不是稳定的,即模式的相对支持度是随数据流的到达不断变化的,如交易数据流、股票数据流就存在很强的季节和周期变化,这种震荡现象迫使挖掘算法频繁对概要数据结构(Synopsis Structure)进行维

到稿日期:2008-09-09 返修日期:2008-11-25 本文受国家自然科学基金项目(60675030)资助。
侯 伟(1981-),男,博士研究生,研究方向为数据挖掘,E-mail:dr. houwei@gmail. com;吴晨生(1968-),男,副研究员,研究方向为系统工程;杨炳儒(1943-),男,教授,博士生导师,研究方向为知识发现与智能系统;方炜炜(1979-),女,博士研究生,研究方向为数据挖掘。

护操作，影响了算法的时空性能。以往的数据流频繁模式挖掘算法，均忽略了震荡现象对性能的影响。针对这一问题，本文首先构造了较高空间性能的概要数据结构 SP-tree，同时定义震荡性因子 χ 以量化震荡信息。在此基础上，提出了一种高效的离线数据流频繁模式挖掘算法 SPDS。算法产生 ε-deficient synopsis 近似结果，在保持较高空间效率的同时，自动适应数据模式的震荡性，减少其对概要数据结构的影响，从而获得了较大程度的性能提高。同时通过减少部分震荡模式的反复删除，使其计数更接近于真实频度，从而提高了模式的查询精度。

1 问题背景与定义

传统频繁模式挖掘任务面向静态事务数据环境，设事务数据库 DB 是由 n 个事务 T_i 构成的集合 $DB=\{T_1,T_2,\cdots,T_n\}$，每个事务包含若干项 I_i，$I_i\in I=\{I_1,I_2,\cdots,I_m\}$，即 $T_i\subseteq I$。设 $Count(P,DB)$ 为项集 P 在 DB 中出现的计数。当且仅当 $Count(P,DB)\geqslant n\times\sigma$，称项集 P 为频繁模式，其中 σ 为预先设置的相对支持度。

近年来，数据流环境下的数据挖掘研究得到了较多关注，已成为数据挖掘领域的热点，并取得了许多成果[3]。相对其它挖掘任务，数据流频繁模式挖掘的研究起步较晚，这主要由于在数据流环境下，模式支持度并不恒定，Depth First 等快速方法不能适用。该环境下，算法只能访问一次事务，且模式的支持度随时间的变化而变化，算法须维护历史概要信息，这对算法的时空复杂度都提出了更高要求。

数据流频繁模式挖掘的目标是在动态的、增量的数据环境下挖掘出所有频繁模式。依据数据流的产生及传输形式，可分为在线流与离线流两类，其主要区别在于：前者在任意时刻只到达一个事务；而后者是以批量的方式传输，即同一时刻到达一组事务。在线流可视为一种特殊的离线流，即该数据流在任意时刻到达的事务集只包含一个事务。

Manku 等在文献[8]提出了基于概率的 Sticky Sampling 与确定性的 Lossy Counting 算法，这两种算法是目前最具代表性的在线流频繁模式挖掘算法；文献[8]还提出了 ε-deficient synopsis 近似结果的概念，即保证输出所有频繁模式的同时，输出部分支持度不低于 $\sigma\varepsilon$ 的模式。在此基础上，C. Giannella 等提出了以字典树为基础的 FP-stream 数据结构[9]以及倾斜窗口机制（Tilted-time windows）。由于 FP-stream 算法对历史数据进行了多时间粒度的维护，因此可以响应针对任何时段的查询。FP-stream 算法仍然是目前最具代表性的离线流频繁模式挖掘算法之一。以 FP-stream 数据结构为基础，张昕等提出了 FPIL-stream 算法[10]。与 FP-stream 算法不同，该算法只针对在线数据流，其执行效率较其它在线流算法有所提高。

通常在线流算法具有较好的实时特性，离线流算法具有较好的时空效率，并且对瞬时计算能力有更高的要求。本文旨在研究离线流环境下的频繁模式挖掘算法。相关概念如下：

定义 1（离线数据流） 与静态事务数据库不同，离线数据流 D 中事务以批量形式传输，是由不断到达的事务集组成的动态增长事务集，即 $D=\{B_0,B_1,\cdots,B_n\}$，其中 $B_i=\{T_1,T_2,\cdots,T_m\}$ 是 t_i 时刻到达的事务集，如果 $j>i$，则 B_i 先于 B_j 到达；如果 $|j-i|=1$，则 B_i 与 B_j 相继到达。

定义 2（窗口） 离线数据流 D 中的窗口集 $W=\{w_0,w_1,\cdots,w_n\}$，是 D 依时间的划分，即 $w_i\subset D$，$w_i\cap w_j=\phi$，$D=\bigcup_{i=1}^{n}w_i$，其中 $i\neq j$。窗口 w_i 的基数 $|w_i|$ 指 w_i 中事务 T_k 的个数。设窗口 w_i 的时长 $long(w_i)$ 指 w_i 中所含事务集个数，若 $long(w_j)>long(w_i)$，其中 $j>i$，则 W 称为倾斜窗口集。

倾斜窗口策略在维护模式历史信息与空间效率间起到平衡的作用。在该策略下，窗口分为多个等级，等级越高包含的事务集数量越多。新事物集到达时，首先将相邻的最低一级窗口中的事务集合并，并将结果转移到上一级窗口。若该级没有空闲窗口，则将该级窗口进行合并处理，并转移至更高一级窗口，直到找到空闲窗口为止。

定义 3（窗口内的支持度） 设 $F(w_i,P)$ 为窗口 w_i 内模式 P 的频度，$S(w_i,P)=\dfrac{F(w_i,P)}{|w_i|}$ 称为模式 P 在窗口 w_i 内的支持度。

定义 4（频繁模式） 给定窗口 w_i，如果模式 P 的窗口支持度 $S(w_i,P)$ 大于等于指定支持度 σ，$(0<\sigma\leqslant 1)$，即 $F(w_i)\geqslant\sigma|w_i|$，则称 P 是 w_i 内的频繁模式。模式 P 在时间 $T=\bigcup_{k=i}^{j}t_k$ 内频繁，指 $f_p(T)=\sum_{k=i}^{j}F(B_k,P)\geqslant\sigma\sum_{k=i}^{j}|B_k|$，其中 $F(B_k,P)$ 是模式 P 在事务集 B_k 中的频度。

定义 5（次频繁模式） 模式 P 在时间 $T=\bigcup_{k=i}^{j}t_k$ 内不频繁，但 $f_p(T)=\sum_{k=i}^{j}F(B_k,P)\geqslant\varepsilon\sum_{k=i}^{j}|B_k|$ 成立，$F(B_k,P)$ 是模式 P 在事务集 B_k 中的频度，则称模式 P 为该时间内的次频繁模式，其中 $\varepsilon(0<\varepsilon<\dfrac{\sigma}{2})$ 为最大支持度容差。

定义 6（近似频度） 在概要数据结构维护过程中，模式 P 在时间 $T=\bigcup_{k=i}^{j}t_k$ 内的频度由近似频度估计，$\hat{f}_p(T)=\sum_{n}^{k=\min\{m,l\}}F(w_k,P)$，其中 $B_i\in w_n$，$B_j\in w_m$，w_l 是概要数据结构所维护的关于 P 的最久窗口。

引理 1[9] 模式 P 在时间 $T=\bigcup_{k=i}^{j}t_k$ 内的近似频度满足性质：$f_p(T)-\varepsilon W\leqslant\hat{f}_p(T)\leqslant f_p(T)$，$W=\sum_{k=n}^{m}|w_k|$，其中 $B_i\in w_n$，$B_j\in w_m$。

2 SPDS 算法

2.1 SP-tree 概要数据结构

数据流挖掘算法一般由 3 部分组成，即概要数据结构、挖掘算法以及查询算法。其中概要数据结构关系到算法整体的时空效率，其设计对算法性能影响很大。字典树是一种高效的数据结构，因其具有高效的访问效率，广泛应用于频繁模式的维护。然而在数据流环境下，为了保证一定的查询精度，不仅需要维护大量的历史数据，还要保留潜在的频繁模式，因此对数据结构的空间效率提出了更高的要求。本文以 P-tree 结构[11]为基础，提出了 SP-tree 结构，此结构具有高度的空间效率和较强的访问性能。

定义 7（SP-tree） SP-tree 由 3 部分组成：(1) 主体是一个二叉字典树，其中每个树节点表示一个模式。拥有两个节点指针，分别指向孩子节点与兄弟节点。其孩子节点为该项集的超集，兄弟节点与该项集的交集是其共同的子集。项目间

顺序由初始窗口 w_0 中各项目支持度决定。(2)倾斜窗口表用于维护各个节点的历史信息,包括节点处于各个窗口的支持度。每节点拥有一个指针与之对应。(3)VN-hash 散列结构用于维护当前震荡模式的信息,如震荡次数等。

相对于一般字典树,由于 SP-tree 结构采用了二叉树的形式,避免了难以确定节点指针数目的问题,具有很好的空间效率,且实现较为简单。通过减少指针的使用和遍历过程中队列操作,可以保证较高的访问性能。此外,此结构还具有以下性质,为针对单一模式的高效查询算法以及计算负边界算法的构造提供了依据。为简化叙述,下面称表示模式 P 的节点为节点 P。概要数据结构 SP-tree 如图 1 所示。

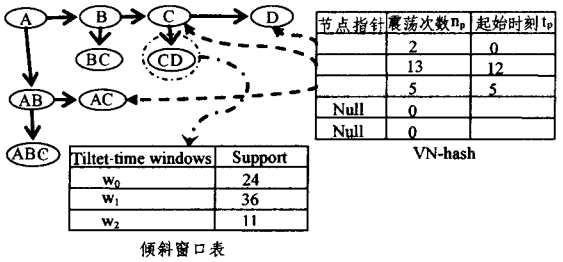

图 1 概要数据结构 SP-tree

性质 1 从 SP-tree 结构的根节点出发,到任意节点 P 有唯一确定的路径,其路径可由该节点表示的模式 P 计算得到。(证明略)

性质 2 SP-tree 结构中任一节点 P,其左子树中任取一节点 P^l,则一定不存在 P^r,P^r 属于 P 的右子树,且 $P^l \subseteq P^r$。

证明:反证法。设存在 P^r,P^r 属于 P 的右子树,且 $P^l \subseteq P^r$,根据定义 1 可知 $P \subseteq P^l$,且 $P \subseteq P^r$,从而 $P \cap P^r = P$。依定义 1 亦可知 $P \cap P^r$ 为 P 的子集,即 $P \cap P^r \subset P$,与假设矛盾,性质成立。

性质 3 SP-tree 结构中任一节点 P,若其的兄弟节点不在 SP-tree 结构中,则其孩子节点也不在 SP-tree 结构中,特别地,在包含全集的 SP-tree 结构中,性质仍成立。(由 Apriori 性质易证)

SP-tree 的遍历是概要数据结构剪枝、更新、查询等操作的基础。根据 SP-tree 结构的定义与性质可以构造依模式长度优先的遍历算法。

算法 1 SP-tree 遍历 Traverse(pnode p,function f)

输入:SP-tree 结构的根节点指针 p,节点操作函数 f,队列 q;

输出:函数 f 访问后的结果与 SP-tree 结构。

```
1 while(p ! = null){
2     f(p); //对节点 p 执行操作函数 f
3     if(p->pchild ! = null) //将节点 p 的孩子指针入队
4         inQueue(q,p->pchild);
5     p = p->psibling;
6     while(p ! = null) //对节点 p 的所有兄弟执行函数 f
7     { f(p);
8         if(p->pchild ! = null) //将 p 兄弟孩子指针入队
9             inQueue(q,p->pchild);
10        p = p->psibling;
11    }
12    p = deQueue(q);
13 }
```

其中,操作函数 f 可以是查询或更新等操作,算法结果与等价字典树 trie 的宽度优先遍历结果一致。

2.2 震荡性因子 χ

数据流中模式的频繁性不是恒定的。一些模式随着数据的相继到达,其支持度会有较大变化,这种现象称为模式频繁性震荡。震荡是数据流频繁模式挖掘不可回避的问题,震荡现象加大了概要数据结构的维护难度。对于震荡较大的模式,算法必须反复对概要数据结构更新、剪枝,加重了算法的时空复杂性。为了定量分析震荡现象,下面给出震荡性因子 χ 的定义,在此基础上给出剪枝算法。

定义 8(震荡性因子 χ) 在维护概要数据结构的过程中,模式 P 的震荡性因子由两个因素确定,即模式 P 依次符合文献[9]中的剪枝条件与加入条件的次数 n_p,与该模式在概要数据结构中持续时间的比值:

$$\chi_p = \frac{2n_p}{t_n - t_p} \tag{1}$$

其中,t_n,t_p 分别表示当前时刻与模式 P 加入概要数据结构的时刻。

震荡性因子客观地反映了模式频度的震荡现象,当因子趋近于 1 时,模式震荡最为剧烈,从而引起针对该模式的反复更新维护操作。因此,维护 SP-tree 结构不仅要依据支持度,还应参考模式震荡性因子,从而避免震荡现象带来的时空复杂性。然而,维护所有模式震荡信息将引入很大的空间复杂性。同时,从图论的观点,只有处于次频繁模式与不频繁模式之间边界附近的模式才会表现出较大的震荡现象。因此提出一个数据结构 VN-hash,用于维护 SP-tree 结构中震荡最为剧烈的模式的震荡信息,包括其起始时刻与震荡次数等。为了在快速定位查询与空间复杂度之间取得平衡,VN-hash 结构同时采用了散列技术与缓存技术,其散列函数根据项目编码将项目对应于相应存储区。其结构实现简单,不再赘述。

2.3 剪枝算法

SP-tree 结构剪枝过程中,算法删除结构中冗余节点,这些节点在近期不可能成为频繁模式,且震荡现象不明显。当节点满足以下条件时,SPDS 算法将其剪枝,同时清空该节点于 VN-hash 结构所占单元:

$$1)\ \hat{f}_p(T) < \varepsilon \sum_{k=0}^{n} |w_k| \tag{2}$$

$$2)\ \chi_p < \beta, (0 < \beta < 1) \tag{3}$$

其中,β 是震荡性因子阈值。

定理 1 设模式 P 满足删除条件,则对于任何历史频繁模式查询,模式 P 都不是合法结果,即 $f_p(T) < \sigma W$,其中 $W = \sum_{k=i}^{j} |w_k|$。(由引理 1 可证)

算法 2 SP-tree 剪枝 Prune(pnode p)

输入:SP-tree 结构的根节点指针 p,VN-hash 结构 v,震荡性因子阈值 β;

输出:剪枝后的 SP-tree 结构。

```
1 if(p ! = null)
2 {   if(p->pchild ! = null) //左子树剪枝
3     Prune(p->pchild);
4         if(p->psibling ! = null) //右子树剪枝
5     Prune(p->psibling);
6     if(式(2)成立) //判断是否符合删除条件
7     {   Xp=Search(p,v); // 查询 VN-hash 结构
8         if(Xp==0)
9         {   Delete(p);
```

```
10            Add(p,v); //VN-hash 添加模式 P
11        }else if(Xp<β)
12        {    Delete(p);
13            Clear(p,v); //清空 VN-hash 对应单元
14        }else
15            Increase(p,v);} //震荡次数加 1
16        }
17 }
```

剪枝算法采用递归策略。在分别对左右子树剪枝后，判断该节点的删除条件。由于震荡性因子的作用，震荡性较强的边缘节点及其历史信息并不轻易被删除，不仅提高了算法整体效率，同时在查询相关节点的频繁性时，由于保留了更长时间的历史数据，其估计频度更加接近真实频度。

2.4 更新算法

更新过程在数据流频繁模式挖掘算法中是时空复杂性的瓶颈，其时空效率对算法整体的执行效率具有决定性影响。在这一过程中，算法首先对新到达的数据集进行频繁模式挖掘，获得其中的频繁及次频繁模式。根据这些模式的频度对概要数据结构进行修正，并根据修正结果对概要数据结构进行剪枝。

为提高算法的时空性能，在新数据集到达时，SPDS 算法并不直接对其进行完整挖掘，而是采取分而治之的策略。首先对 SP-tree 结构维护的模式及其负边界[12]计数，利用 SP-tree 结构中模式间子集关系，可以保证计数的高效性。在计数的同时，算法针对负边界将新数据集分离映射(Partition Projection)[13]，形成对应各个负边界模式的映射子库，并酌情对映射数据库进行局部挖掘，以达到高效更新。

定义 9(负边界) 设 S 是由项集构成的集合，负边界 $Bd^-(S)=\{P\in U\backslash S \mid \forall M, M\subset P, M\in S\}$ 是一个模式集合，其中 U 是模式全集。

需要注意的是，由于 SP-tree 结构同时维护频繁模式与次频繁模式，因此这里的集合 S 是两类模式的并集。

定理 2 负边界中各模式间不存在包含关系，即 $\forall P_1, P_2\in Bd^-(S), P_1\not\subset P_2, P_2\not\subset P_1$。

证明：反证法。不失一般性，设 $\forall P_1, P_2\in Bd^-(S), P_1\subset P_2$，则模式 P_1 为 P_2 的子集。根据负边界定义可知 $P_1\in S$，亦即模式 P_1 是模式集 S 的元素，因此模式 P_1 不是负边界 $Bd^-(S)$ 的元素，与假设矛盾，定理得证。

定理 3 设 $P\in Bd^-(S)$，$Super(P)$ 是由模式 P 及其超集组成的集合。任取 $P'\in U\backslash S$，一定存在 $P, P\in Bd^-(S)$，满足 $P'\in Super(P)$。

证明：由 $P'\in U\backslash S$ 得 $P'\notin S$。若存在 P' 的真子集，它不属于 S，设其中不属于 S 且最小的真子集为 P_s'，则 P_s' 的任何真子集均属于 S，即 $P_s'\in Bd^-(S)$，因此 $P'\in Super(P_s')$；反之，$P'\in Bd^-(S)$，显然 $P'\in Super(P_s')$，得证。

定理 4 集合 $W=\{Super(P_1), Super(P_2), \cdots, Super(P_n)\}$ 是集合 $U\backslash S$ 的一个覆盖，即 $U\backslash S=\bigcup_{i=1}^{n} Super(P_i)$，其中 $P_i\in Bd^-(S)$。(由定理 3 可证)

定义 10(映射数据库) 模式 P 在数据库 DB 中的映射数据库 $Sub(P,DB)=\{T\in DB \mid T\supseteq P\}$，它具有性质 $Count(P,DB)=Count(P,Sub(P,DB))$。

推论 1 当新的数据集 B_i 到达时，对于任意项集 $P\in U\backslash S$，S 为 SP-tree 所维护的模式集，则 P 在 B_i 中的频度可通过挖掘 $Sub(P', B_i)$ 得到，其中 $P'\in Bd^-(S), P'\subseteq P$。(证明略)

根据以上原理，构造更新算法如下所示：

算法 3 SP-tree 更新算法 Update(pnode p, DB d)

输入：SP-tree 结构的根节点指针 p，负边界模式数组 NBd，新到达数据集 d；

输出：更新后的 SP-tree 结构。

```
1 Traverse(p,FindNBd);//计算负边界,更新数组 NBd
2 Create(NBdSubDB,n);//构造映射子库
3 for each t in d
4 { Increase(t,p); //递增 SP-tree 与 NBd 相关模式计数
5   for(i=0;i<n;i++)
6   { if(t contains NBd[i]) //遍历 NBd,建立子库
7     { Add(t,NBdSubDB[i]);
8       return;}
9   }
10 }
11 for(i=0;i<n;i++) //按顺序遍历 NbdSubDB
12 { if(NBd[i] is subfrequent)
13    { frequent_itemset = Fpgrowth(NBdSubDB[i]);
14      update frequent_itemset into SP-tree;
15   }
16   move NBdSubDB[i] to latter NBdSubDBs;
17 }
```

首先，通过算法 1 定义的遍历算法计算负边界模式，其中 FindNBd 函数由算法 4 给出；依据新到数据集更新当前 SP-tree 中模式的估计频度，并统计负边界模式的频度，同时根据包含关系，将新到数据集划分到映射子库中，每条事务只存放于一个子库，以保证空间效率；在此基础上，算法检查所有负边界模式的支持度，只有该模式在此时刻为频繁或次频繁模式时才对其子库进行挖掘，并将挖掘结果插入 SP-tree 结构；最后，将该子库中的事务映射到后续子库中。

文献[14]中提出利用 Apriori 算法生成候选集的方法计算负边界。然而利用 SP-tree 的结构性质可设计性能更好的方法，只需一次遍历 SP-tree 结构。

算法 4 计算负边界函数 FindNBd(pnode p)

输入：SP-tree 结构的根节点指针 p，负边界模式数组 NBd；

输出：计算出所有与当前节点相关的负边界模式，并存入 NBd。

```
1 if(p->psibling ! = null)
2 { pattern=p->pattern;
3   pattern=CreateNextSibling(pattern);// p 的兄弟模式
4   while(p->psibling->pattern ! = pattern
5         && pattern is not an empty set)
6   { if(there is no subset of pattern in NBd)
7     { Add(pattern,NBd);
8       Delete(pattern,NBd);//删除 pattern 超集
9     }
10    pattern=CreateSibling(pattern);
11  }
12 }
13 if(p->child == null)
14 { for each child of p
15   { if(there is no subset of child in NBd)
16       Add(child,NBd);
17  }
18 }
```

函数 FindNBd 由两部分组成：搜索模式 P 的兄弟负边界模式以及其孩子负边界模式。其中第 3 步计算该模式在全集中临近的右边兄弟，项目顺序由初始窗口 w_0 中各项目支持

度决定，只需更换该模式最后一个项目即可实现；步骤 14 中产生模式 P 的子模式原理相同，不再赘述。

2.5 SPDS 算法流程

SPDS 算法的基本流程是处理不断到达的新数据集的过程，包括 SP-tree 结构的更新、剪枝，以及更新节点震荡信息等步骤。挖掘过程中，算法确保 SP-tree 结构中维护了任意时间内的频繁模式，并支持对历史频繁模式查询。

算法 5 SPDS

输入：初始数据集 B_0，数据流；

输出：维护所有当前频繁模式的 SP-tree 结构，p 为根节点指针。

```
1 Moveto(B0,w0);
2 itemsets = Fpgrowth(w0);
3 p = Create(itemsets); //构造 SP-tree,确定项目序
4 CreateVNhash(); //构建 VN-hash
5 while(new DB d arrived) { //新数据集到达时触发
6         Decay(p); //倾斜窗口合并转移,参照文献[9]
7         Update(p,d); //算法 3
8         Prune(p); //算法 2
9 }
```

首先，算法根据初始数据集确定项目顺序，同时构建 SP-tree 结构。新数据集到达时，算法先对倾斜窗口进行合并转移；计算负边界，并依据新数据集对现有模式与负边界计数，构造映射子库；根据负边界模式的频度决定各个子库挖掘的必要性；依据子库挖掘结果，更新 SP-tree 结构；最后对其进行剪枝。该过程中维护 VN-hash 结构，至此完成此数据集的挖掘工作。

3 实验及分析

相对其它方法，SPDS 算法的优势在于针对震荡数据流的处理，因此实验在平稳数据流与震荡数据流两种环境下进行，数据由 IBM 数据生成器[15]产生。其中平稳数据流由连续到达的 60 个数据集构成，每个数据集含有 5 万个事务，数据集共涉及 1000 个项目，平均事务长度取 5，其他参数取缺省值。在平稳流的基础上，以 5 为周期对数据集中的项目进行随机映射，以破坏其平稳性，从而获得震荡数据流。实验环境：PC 主频 2.0GHz，内存 1GB，操作系统 Windows 2003。SPDS 算法采用标准 C++实现，编译器为 MinGW。在现有离线流频繁模式挖掘算法中，FP-stream 是最具代表性且计算性能领先的算法之一，因此实验以其作为比较对象。实验中，两种算法支持度取 $\sigma=0.0075$，容差 $\varepsilon=0.00075$。

实验主要检验 SPDS 算法在时间与空间效率两个方面的表现。首先，实验采用平稳流作为输入，考察 SPDS 与 FP-stream 算法的处理时间与内存消耗情况，结果如图 2 所示。

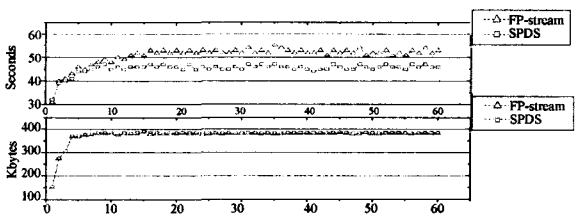

图 2 平稳数据流中的性能比较

如图 2 所示，在平稳流环境下，SPDS 较 FP-stream 在数据集处理时间上具有较为明显的优势，其主要原因是 SPDS 算法针对平稳流的特点，改进了数据集的处理方法。即在负边界上做分离映射，把完整的挖掘划分成多个子库的子挖掘任务。值得注意的是，基于负边界的性质，其中许多子挖掘任务可以省略，这在平稳流中更为明显。在内存消耗方面，虽然 SPDS 算法采用了空间性能较高的 SP-tree 结构，但由于需要维护部分节点的震荡信息，因此其内存消耗略大于 FP-stream 算法，然而较之性能差异并不明显。在此基础上，实验考察两种算法在震荡流下的时空效率，结果如图 3 所示。

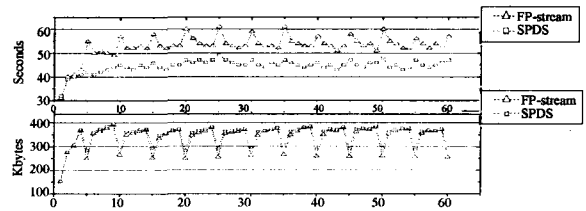

图 3 震荡数据流中的性能比较

从图 3 可见，在震荡位置 FP-stream 算法在时间与空间效率都发生了较大变化，这是由于项目随机映射使得一些频繁模式的支持度突然降低，迫使算法对概要数据结构进行较大调整。SPDS 算法在震荡位置的变化相对更平稳。虽然由于对倾斜窗口的操作使时空效率也有所变化，但总体上能够保证稳定执行。并通过识别震荡节点减少了剪枝操作，使程序获得更高的时间性能。在内存消耗方面，VN-hash 结构的震荡信息同样带来一些消耗，但总体来看 SPDS 空间效率与 FP-stream 相差并不明显。综合两方面考虑，SPDS 算法的性能优于 FP-stream 算法。

结束语 在数据挖掘领域，数据流频繁模式的挖掘问题是目前的研究热点之一，数据流的特点对挖掘算法在时空效率上提出了很高的要求。以往的数据流频繁模式挖掘算法，其构造忽略了数据震荡现象对性能的影响。然而现实情况中的数据流往往不是稳定的。针对这一问题，本文提出了一种高效的离线数据流频繁模式挖掘算法 SPDS，其具有基于较高空间性能的概要数据结构 SP-tree，并结合其中的节点震荡信息，有效降低了数据震荡对算法性能的影响；其数据集处理采取分而治之的分离映射策略，进一步提高了时间性能；同时在查询结果方面提高了部分模式的计数精度。实验结果与理论分析说明，此算法的总体性能超过以往离线数据流频繁模式的挖掘算法。

参 考 文 献

[1] Han J, Kamber M. Data Mining: Concepts and Techniques[M]. San Francisco: Morgan Kaufmann, 2000

[2] Babcock B, Babu S, Datar M, et al. Models and issues in data stream systems[C]// Proc. of the 21th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. Wisconsin: Madison, 2002: 1-16

[3] Gaber M M, Zaslavsky A, Krishnaswamy S. Mining Data Streams: A Review[J]. ACM SIGMOD Record, 2005, 34(2): 18-26

[4] Garofalakis M N, Gehrke J, Rastogi R. Querying and mining data streams: you only get one look[C]// Proceedings of the 28th Int. Conf. on Very Large Data Bases (VLDB'02). Hong Kong, 2002

（下转第 291 页）

要优于传统方法的结果。

(3)通常情况下,焊点的三维形态的差异也就是焊点质量的原始信息,焊点三维形态参数的提取是进行焊点质量检测、比较、分析和评价的前提。从焊点三维图像中获取焊点形态参数的过程,实质是一个图像测量的过程,主要涉及到图像的面积、长度、高度等基本内容的计算,一般可以通过对焊点三维重构中的计算公式计算获得。

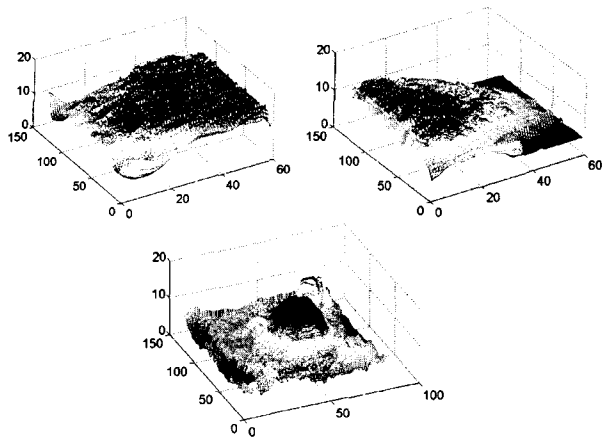

图 8 SMT 焊点图像三维恢复结果

结束语 比较全面地介绍了基于 SFS 原理的 SMT 焊点三维重构技术原理及其实现方法和步骤。同时应用该技术和方法对多种类型焊点进行了检测和分析实验,结果证明了其有效性。根据重构后的 SMT 焊点 3D 图像,可通过数学形态学的方法得到焊点的面积、周长和焊点边缘等一些信息,可以对 SMT 焊点做更多的研究。同时将重构后的图像引入 3D 测量技术,提取 SMT 焊点质量信息,对改善 SMT 焊点质量及其焊接工艺有一定的指导意义和参考价值。由于 SMT 产品组装过程中质量影响因素多、焊点类型多,要在上述理论和方法基础上形成实用系统,还有很多工作要做。焊点三维重构技术和焊点图像处理技术也还有很多改进的空间。

参 考 文 献

[1] 周德俭,黄春跃,吴兆华,等. SMT 焊点虚拟成形和 SMT 产品虚拟组装技术研究[J]. 计算机集成制造系统,2006,12(8):1267-1272

[2] Gao Yuefang, Luo Fei, Gao Jianzhong. A Shape from Shading Algorithm and Its Application[C]// 2007 IEEE International Conference on Control and Automation Guangzhou. June 2007: 2133-2135

[3] Ghayourmanesh S, Zahng Y. Shape from shading of SAR imagery in fourier space[C]// IGARSS 2007. Barcelona, July 2007: 835-837

[4] Zhang Li. Restoring warped document images using shape-from-shading and surface interpolation[C]// 2006 18th International Conference on Pattern Recognition. Sept. 2006: 20-24

[5] Zhang Ruo, Tsai Ping-Sing, Cryer J E, et al. Shape from Shading: A Survey [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999, 21(8): 119-131

[6] Kong Fanhui , Wang Yongxin . Reconstruction of Solder Joint Surface Based on Shape from Shading [C]// IEEE Third International Conference on Natural Computation (ICNC 2007). Aug. 2007: 58-62

[7] Ivan Fanany M, Kumazawa I. Analysis of Shape from Shading Algorithms for Fast and Realistic 3D Face Reconstruction[C]// APCCAS2002. 2002: 181-185

[8] 廖熠,赵荣椿. 从明暗恢复形状(SFS)的几类典型算法分析与评价[J]. 中图像图形学报,2001,6(10):953-961

[9] 赵颜利,郭成,吴王湛,等. 基于 CB 样条曲线的空间物体三维重建[J]. 计算机科学,2006,33(5):247-249

[10] 鲁瑞华,杨明. 一种基于中值滤波的非线性图像处理优化算法[J]. 计算机科学,2004,31(11):224-226

[11] IPC-A-610C 印制板组装件验收条件. 2000

[12] Prados E, Faugeras O. Shape form shading: a well posed problem[C]// International Conference on Computer Vision and Pattern Recognition CVPR05. June 2005: 870-877

(上接第 251 页)

[5] 潘云鹤,王金龙,徐从富. 数据流频繁模式挖掘研究进展[J]. 自动化学报,2006,32(4):594-602

[6] Hidber C. Online association rule mining[C]// Proc. of the 1999 ACM SIGMOD International Conference on Management of Data. Philadelphia, Pennsylvania, 1999

[7] Chang J H, Lee W S. Finding recently frequent itemsets adaptively over online transactional data streams[J]. Information Systems, 2006, 31(8): 849-869

[8] Manku G S, Motwani R. Approximate frequency counts over data streams[C]// Proceedings of the 28th Int. Conf. on Very Large Data Bases(VLDB'02). Hong Kong, 2002

[9] Giannella C , Han J , Pei J . Mining frequent patterns in data streams at multiple time granularities[M]// Kargupta H, Joshi A, Sivakumar K, eds. Next Generation Data Mining. Cambridge, Massachusetts: MIT Press, 2003: 191-212

[10] 张昕,李晓光,王大玲,等. 数据流中一种快速启发式频繁模式挖掘方法[J]. 软件学报,2005,16(12):2099-2105

[11] Coenen F, Goulbourne G, Leng P. Tree Structures for Mining Association Rules[J]. Data Mining and Knowledge Discovery, 2004, 8(1): 25-51

[12] Mannila H , Toivonen H . On an algorithm for finding all interesting sentences[C]// Cybernetics and Systems, Volume 2, The 13th European Meeting on Cybernetics and Systems Research. Vienna, Austria, April 1996

[13] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]// Proc. of the ACM SIGMOD International Conference on Management of Data. New York: ACM, 2000: 1-12

[14] Thomas S, Bodagala S, Alsabti K, et al. An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases[C]// Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining(KDD '97). New Port Beach, California, August 1997

[15] IBM Almaden Research Center. Synthetic Data Generation Code for Associations and Sequential Patterns [EB/OL]. http: // www. almaden. ibm. com/software/quest/Resources/index. shtml Intelligent Information Systems