基于 FP-Tree 的最大频繁项目集挖掘及更新算法*

宋余庆[1], 朱玉全[1,2+], 孙志挥[1], 陈 耿[1]

[1](东南大学 计算机科学与工程系,江苏 南京 210096)

[2](河海大学 计算机及信息工程学院,江苏 常州 213022)

An Algorithm and Its Updating Algorithm Based on FP-Tree for Mining Maximum Frequent Itemsets

SONG Yu-Qing[1], ZHU Yu-Quan[1,2+], SUN Zhi-Hui[1], CHEN Geng[1]

[1](Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

[2](College of Computer and Information Engineering, Hohai University, Changzhou 213022, China)

+ Corresponding author: Phn: 86-25-3795451; 86-519-5110090, E-mail: yuquanz@sina.com

http://www.seu.edu.cn

Received 2002-04-15; Accepted 2002-07-02

Song YQ, Zhu YQ, Sun ZH, Chen G. An algorithm and its updating algorithm based on FP-tree for mining maximum frequent itemsets. *Journal of Software*, 2003,14(9):1586~1592.

http://www.jos.org.cn/1000-9825/14/1586.htm

Abstract: Mining maximum frequent itemsets is a key problem in many data mining application. Most of the previous studies adopt an Apriori-like candidate set generation-and-test approach. However, candidate set generation is still costly, especially when there exist prolific patterns and/or long patterns. In this paper, a fast algorithm DMFIA (discover maximum frequent itemsets algorithm) and its updating algorithm UMFIA (update maximum frequent itemsets algorithm) based on frequent pattern tree (FP-tree) for mining maximum frequent itemsets is proposed. The algorithm UMFIA makes use of previous mining result to cut down the cost of finding new maximum frequent itemsets in an updated database.

Key words: data mining; maximum frequent itemset; association rule; frequent pattern tree; incremental updating

摘 要: 挖掘最大频繁项目集是多种数据挖掘应用中的关键问题,之前的很多研究都是采用 Apriori 类的候选项目集生成-检验方法.然而,候选项目集产生的代价是很高的,尤其是在存在大量强模式和/或长模式的时候.提出了一种快速的基于频繁模式树(FP-tree)的最大频繁项目集挖掘 DMFIA(discover maximum frequent itemsets algorithm)及其更新算法 UMFIA(update maximum frequent itemsets algorithm).算法 UMFIA 将充分利用以前的挖掘结果来减少在更新的数据库中发现新的最大频繁项目集的费用.

关键词: 数据挖掘;最大频繁项目集;关联规则;频繁模式树;增量式更新

中图法分类号: TP311 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.79970092 (国家自然科学基金); the National Innovation Fund for Small Technology-Based Firms of China under Grant No.00C26213211014 (国家科技型中小企业技术创新基金)

第一作者简介: 宋余庆(1959－),男,江苏扬州人,博士生,教授,主要研究领域为图像检索,知识发现.

关联规则是由 Agrawal 等人首先提出来的一个重要的 KDD 研究课题[1~3],它反映了大量数据中项目集之间有趣的关联或相关联系.发现频繁项目集是关联规则挖掘应用中的关键技术和步骤.近年来,在频繁项目集的算法研究中先后出现了 Apriori,AIS[4],SETM[5],PARTITION[6],ML_T2L1[7]等数据挖掘算法,在众多算法中,以 Agrawal 等人提出的 Apriori 算法[1]最为著名,其后的数据挖掘算法大多数建立在 Apriori 算法基础之上,或进行改进,或衍生变种,诸如 AprioriTid,AprioriHybird[2~4]等.在如上所述的诸多算法中,计算项目集的支持数是发现频繁项目集中最耗时的工作,占据整个计算量的大部分,因此,降低候选项目集的数量是减小开销的最好手段.由于最大频繁项目集中已经隐含了所有频繁项目集,所以可把发现频繁项目集的问题转化为发现最大频繁项目集的问题.另外,某些数据挖掘应用仅需发现最大频繁项目集,而不必发现所有的频繁项目集,因而发现最大频繁项目集对数据挖掘具有重大意义.

目前已经提出的可用于发现最大频繁项目集的算法有 Max-Miner[8],Pincer-Search[9]以及 DMFI[10]等.Max-Miner 突破了传统的自底向上的搜索策略,尽可能早地对项目集进行修剪,其缺陷是:① 未利用自顶向下的信息进行修剪;② 未对 MFCS 进行适当的排序,产生了多余的最大频繁候选项目集.Pincer-Search 采用了自底向上和自顶向下的双向搜索策略,但其第 k 次的 MFCS 是由 k–1 次的 MFCS 中的非频繁项目集去掉一个元素来生成的,产生了过多的无用候选项目集,对海量数据库来讲,算法 Pincer-Search 将陷入 NP 难度的陷阱.DMFI 有效地把自底向上和自顶向下的搜索策略进行了合并,该算法为海量数据库中发现最大频繁项目集和仅需要发现最大频繁项目集的数据挖掘应用提供了一种有效而快速的算法,但该算法仍需多次重复扫描数据库 D,计算项目集的支持数.为此,本文提出了一种快速的基于 FP-tree[11]的最大频繁项目集挖掘算法 DMFIA(discover maximum frequent itemsets algorithm),该算法只需扫描事务数据库 D 一次,从而提高了算法的执行效率.另外,在现实世界事务数据库中,数据是随时间的变化而变化的.商业机构中的交易是一个不断进行的过程,交易行为的模式很可能随时间呈现出某种发展趋势,使得当前已发现的最大频繁项目集可能不再生效,而可能存在新的有效的最大频繁项目集有待于进一步去发现.因此,迫切需要设计高效的算法来更新、维护和管理已挖掘出来的最大频繁项目集,然而,对于这一问题,目前国内外尚未有相关研究,对此,本文提出了一种增量式更新最大频繁项目集算法 UMFIA(update maximum frequent itemsets algorithm),该算法将充分利用已有的一切信息(如旧的最大频繁项目集、原来的 FP-tree 等),以高效地发现最新事务数据库中所有的最大频繁项目集.

1 问题描述

1.1 频繁项目集和最大频繁项目集

设 $I=\{i_1,i_2,\ldots,i_m\}$ 是 m 个不同项目的集合.给定事务数据库 D,对于项目集 $X\subseteq I$.X 在 D 中的支持数是指 D 中包含 X 的事务数,记为 $X.count_D$.X 在 D 中的支持度是指 D 中包含 X 事务的百分比,记为 $X.sup_D$.如果 X 的支持度不小于用户给定的最小支持度阈值 s,则称 X 为 D 中的频繁项目集,项目集中项目的个数称为项目集的维数或长度,频繁 1-项目集简称频繁项目.

定义 1. 对于项目集 $X\subseteq I$,如果 $X.sup_D\geq s$,并且对于任意 $Y\supset X$,均有 $Y.sup_D<s$,则称 X 为 D 中的最大频繁项目集.

显然,任何频繁项目集都是某最大频繁项目集的子集,所以可以把发现所有频繁项目集的问题转化为发现所有最大频繁项目集的问题.

1.2 频繁模式树FP-tree[3,4,11]

在 FP-tree 中,每个节点由 4 个域组成:节点名称 node-name、节点计数 node-count、节点链 node-link 及父节点指针 node-parent.另外,为方便树遍历,创建了一个频繁项目头表 Htable,它由两个域组成:项目名称 item-name 和项目链头 item-head,其中项目链头指向 FP-tree 中与之名称相同的第 1 节点.频繁模式树 FP-tree 的构造算法如下:

(1) 扫描 D 一次,产生频繁项目集合 F 及其支持数.按其支持数降序排列 F,生成频繁项目列表 L_{DF};

(2) 创建 FP-tree 的根节点,标号为“null”.对于 D 中的每个事务 Trans 作如下处理:① 按 L_F 中的次序排列

Trans 中的频繁项目,设排列后的结果为[$p|P$],其中 p 是第 1 个项目,而 P 是剩余项目的列表;② 调用 insert_tree([$p|P$],T);③ 如果 P 非空,递归调用 insert_tree(P,N).过程 insert_tree([$p|P$],T)的执行情况如下:如果 T 有子女 N 使得 N.node-name=p,则 N 的计数增加 1;否则创建一个新节点 N,将其名称 node-name、计数 node-count 分别设置为 p,1,由父节点指针 node-parent 链接到它的父节点 T,并通过节点链 node-link 将其链接到具有相同名称 node-name 的节点.

2 挖掘最大频繁项目集算法 DMFIA

性质 1. 如果 X 为最大频繁项目集,那么 X 的任何子集都是频繁项目集,并且任何最大频繁项目集均为 L_{DF}(L_{DF} 见第 1.2 节)的子集.

性质 2. 设 $X=\{x_1,x_2,\ldots,x_{m-1},x_m\}$为非频繁项目集,那么 $X_i=\{x_j|x_j\in X_j j\neq i\}$均有可能成为频繁项目集.

设 D 的频繁模式树 FP-tree$_D$,对于任何最大频繁候选项目集 X,其支持数即为事务数据库 D 中包含 X 的事务数,由 FP-tree$_D$ 的构造原理可知,X 中的频繁项目已映射到 FP-tree$_D$ 的某惟一一条路径上,因此,求 X 在 D 中的支持数可以转换成求 FP-tree$_D$ 中包含 X 的路径数.

算法 1. 挖掘 D 中的最大频繁项目集算法 DMFIA.

输入:D 的频繁模式树 FP-tree$_D$;频繁项目头表 Htable$_D$;最小支持度阈值 s;D 中的频繁项目列表 L_{DF}(为方便起见,记 $L_{DF}=\{1,2,3,\ldots,k\}$,$k=|L_{DF}|$).

输出:D 中的最大频繁项目集 MFS$_D$.

方法:

```
(1)  MFS_D=∅;
(2)  MFCS_D=L_DF={1,2,3,…,k};//MFCS_D 为 D 中的最大频繁候选项目集
(3)  while (MFCS_D≠∅) do begin
(4)    for (i=k; i>0;i--) do begin
(5)      MFCS_i={c|c∈MFCS_D and c 的最后一个项目为 i};
(6)      MFCS_D=MFCS_D-MFCS_i;
(7)      调用过程 ComputeCount (FP-tree_D,Htable_D,MFCS_i);   //计算项目集在 D 中的支持数
(8)      for all m∈MFCS_i do begin
(9)        if m.sup_D≥s then
(10)         MFS_D=MFS_D∪m
(11)       else
(12)         for all item e∈m do
(13)           if m-{e}不是 MFS_D 或 MFCS_D 中某元素的子集 then
(14)             MFCS_D=MFCS_D∪{m-{e}};//根据性质 2
(15)     end
(16)   end
(17) end
```

Procedure ComputeCount(tree,Htable,MFCS$_i$)

/*计算 MFCS$_i$ 中的项目集在 D 中的支持数,由于在 MFCS$_i$ 中项目集中的项目已按其支持数降序排列,因此,确定包含它的路径数是很方便的,只需根据父指针向根节点搜索而无须向叶节点搜索*/

```
begin
(1) 搜索项目头表 Htable 的项目名称域 item-name,假设 Htable[q_1].item-name=i;
(2) 根据 Htable[q_1].head 找到 tree 中节点名称为 i 的节点 nd_1,…,nd_h;
(3) 根据 nd_1,…,nd_h 及其前缀节点的父节点指针域找到包含 i 的所有路径 P_1,P_2,…,P_h;
(4) for all m∈MFCS_i do
```

(5) for (j=1;$j\le h$;j++) do

(6) 如果路径 P_j 包含 m,那么 m 的支持数增加 nd_j.node-count;

end

3 最大频繁项目集更新算法 UMFIA

性质 3. 设新增事务数据集为 d,MFS_D,MFS_d 分别为 D,d 中最大频繁项目集的集合,X 为 $D\cup d$ 中的最大频繁项目集,则存在 $Y\in MFS_D\cup MFS_d$,使得 $X\subseteq Y$.

证明:由于 X 为 $D\cup d$ 中的最大频繁项目集,因此 X 或为 D 中的频繁项目集,或为 d 中的频繁项目集,又因最大频繁项目集中已经隐含了所有频繁项目集,故 X 或为 D 中某最大频繁项目集的子集,或为 d 中某最大频繁项目集的子集,即性质 3 成立. □

由性质 3 可知,我们可将 $MFS_D\cup MFS_d$ 作为 $D\cup d$ 中的初始最大频繁候选项目集.由于 MFS_D 已在第 2 节中求得,而求 MFS_d 也很方便,故余下的问题是如何求 $MFCS_{Dd}$($D\cup d$ 的最大频繁候选项目集)中各元素在 D 及 d 中的支持数,由于 d 较小,所以更新问题的核心是如何利用原有的 FP-tree$_D$ 来求 $MFCS_{Dd}$ 中各元素在 D 中的支持数.

性质 4. 设 FP-tree$_{Dd}$ 为 $D\cup d$ 的频繁模式树,L_1 为 D 中的频繁项目,L_{n1} 为 $D\cup d$ 中新增加的频繁项目,则 L_{n1} 中各项目在 D 中的支持数均小于 L_1 中任意项目在 D 中的支持数.

证明:由于 L_1 中各项目集的支持数均不小于 $s\times|D|$,而 L_{n1} 中各项目在 D 中的支持数均小于 $s\times|D|$,故性质 4 成立. □

由于事务数据库发生了变化,因此,项目集的支持数有可能发生变化,很难实现 FP-tree$_D$ 与 FP-tree$_{Dd}$ 之间的转换.事实上,对于 D 中的任何一项事务 t,设 $t_1=t\cap L_1$,$t_2=t\cap L_{n1}$,因此,$t_1\cap t_2=\varnothing$.由 FP-tree$_D$ 的构造原理可知,t_1 中的各项目必将同时出现在 FP-tree$_D$ 的某条惟一路径 p 上,因此如果能将 t_2 中的各项目添加到 FP-tree$_D$ 的同一条路径 p 上,那么 t 中 $D\cup d$ 的频繁项目将映射到 P-tree$_{Dd}$(P-tree$_{Dd}$ 为 FP-tree$_D$ 中添加了 L_{n1} 中各项目后的树)的某条惟一路径上,$MFCS_{Dd}$ 中各元素在 D 中的支持数即为 P-tree$_{Dd}$ 中包含此元素的路径数.由于 L_{n1} 中各项目在 D 中的支持数均小于 L_1 中任意项目的支持数,因此 L_{n1} 中的项目所对应的节点均为 P-tree$_{Dd}$ 的叶子节点或叶子节点的前缀节点,且无 L_1 中的项目所对应的节点介于这些节点之间, 故只需在 FP-tree$_D$ 中添加叶子节点即可.据此可得由 FP-tree$_D$ 来构造 D 中的模式树 P-tree$_{Dd}$(P-tree$_{Dd}$ 不同于频繁模式树)的方法,P-tree$_{Dd}$ 的构造算法描述如下:

(1) if ($L_{n1}\neq\varnothing$) then

(2) begin

(3) 将 L_{n1} 中的各项目添加到频繁项目头表 Htable$_D$ 中,得到新的项目头表 Htable$_{Dd}$;

(4) $L_{DdF}=L_1\cup L_{n1}$; //将 $L_1\cup L_{n1}$ 中的各项目按其支持度降序排列并存放在 L_{DdF} 中

(5) for each transactions $t\in D$ do begin

(6) $t'=L_{DdF}\cap t$; //取出 t 中属于 L_{DdF} 的项目

(7) search FP-tree$_D$ and insert $t'\cap L_{n1}$ into FP-tree$_D$; //FP-tree$_D$ 中已包含 t'中非 L_{DdF} 中的项目

(8) end

(9) end

(10) else do nothing;//FP-tree$_D$ 不变

综上所述,最大频繁项目集更新算法可描述为:

算法 2. 最大频繁项目集更新算法 UMFIA.

输入:D,d 中的最大频繁项目集 MFS_D,MFS_d;新的项目头表 Htable$_{Dd}$;D 中的模式树 P-tree$_D$;最小支持度阈值 s(为方便起见,记 $L_1\cup L_{n1}=\{1,2,3,\ldots,k_1\}$,$k_1=|L_1\cup L_{n1}|$).

输出:$D\cup d$ 中的最大频繁项目集 MFS_{Dd}.

方法:

(1) $MFS_{Dd}=\varnothing$;

(2) $MFCS_{Dd}=MFS_D\cup MFS_d$;

(3) while ($MFCS_{Dd}\neq\varnothing$) do begin
(4) for ($i=k_1;i>0;i--$) do begin
(5) $MFCS_i=\{c|c\in MFCS_{Dd}$ and c 的最后一个项目为 $i\}$;
(6) $MFCS_{Dd}=MFCS_{Dd}-MFCS_i$;
(7) 调用过程 ComputeCount($P\text{-}tree_{Dd}$,$Htable_{Dd}$,$MFCS_i$);
(8) 计算 $MFCS_i$ 中各项目集在 d 中的支持度;
(9) for all $m\in MFCS_i$ do begin
(10) if $m.sup_{Dd}\geq s$ then
(11) $MFS_{Dd}=MFS_{Dd}\cup m$
(12) else
(13) for all item $e\in m$ do
(14) if $m-\{e\}$不是 MFS_{Dd} 或 $MFCS_{Dd}$ 中某元素的子集 then
(15) $MFCS_{Dd}=MFCS_{Dd}\cup\{m-\{e\}\}$;//根据性质 2
(16) end
(17) end
(18) end

例 1:设事务数据库 D 如图 1 所示,事务数据集如图 2 所示,最小支持度为 50%,数据库 D 的 FP-tree 如图 3(节点 o 除外)所示.

TID	itemset
100	f,a,c,d,g,i,m,p
200	f,a,b,c, l,m,o,p
300	b,f,h,j,o
400	b,c,k,s
500	a,f,c,e,l,p,m,n
600	a,b,c,f,m,p

Fig.1 Transaction database D
图 1 事务数据库 D

TID	itemset
700	f,l,m,o,p
800	f,h,j,o

Fig.2 Transaction data set d
图 2 事务数据集 d

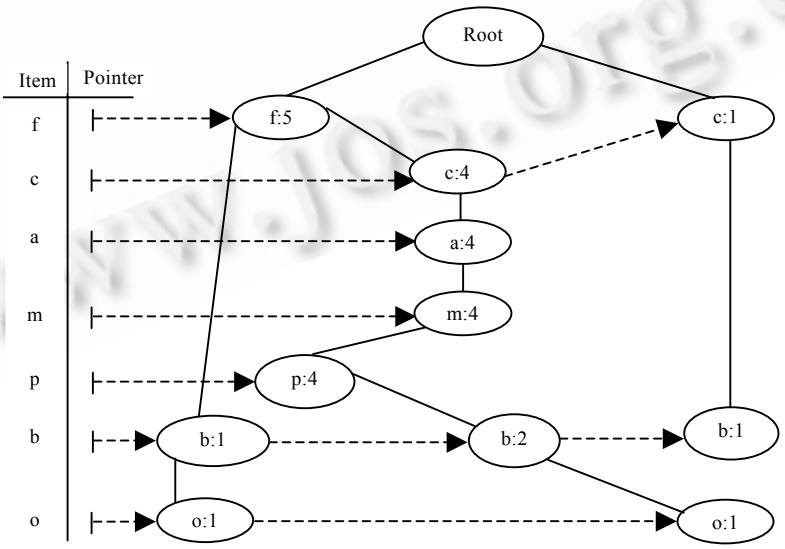


Fig.3 $Htable_D$ (except node o), $Htable_{Dd}$, $FP\text{-}tree_D$ (except node o) and $P\text{-}tree_{Dd}$ of example 1
图 3 例 1 的 $Htable_D$(节点 o 除外),$Htable_{Dd}$,$FP\text{-}tree_D$(节点 o 除外)及 $P\text{-}tree_{Dd}$

根据算法 DMFIA 和图 3 可求得 D 中的最大频繁项目集 MFS_D={{f,c,a,m,p},{f,b},{c,b}},根据 d 可求得 d 中的最大频繁项目集 MFS_d={{f,m,p,o}}(已删除了 $D\cup d$ 的非频繁项目 l,h,j),因此,$MFCS_{Dd}$ 的初始值为 $MFCS_{Dd}=MFS_D\cup MFS_d$={{f,c,a,m,p},{f,b},{c,b},{f,m,p,o}},算法 UMFIA 的执行过程如下:

(1) 将 L_{n1}={o}添加到频繁项目头表 $Htable_D$ 中,得到新的项目头表 $Htable_{Dd}$;

(2) 扫描 d 求项目集{f,c,a,m,p},{f,b},{c,b}在 d 中的支持数,其值分别为 0,0,0,故项目集{f,c,a,m,p},{f,b},{c,b}在 $D\cup d$ 中的支持数分别为 4,3,3;

(3) 搜索项目头表 $Htable_{Dd}$ 及模式树 $P\text{-}tree_{Dd}$ 求得项目集{f,m,p,o}在 D 中的支持数,其值为 1,因此项目集{f,m,p,o}在 $D\cup d$ 中的支持数为 2;

(4) 由上面的(2),(3)可得:MFS_{Dd}={{f,c,a,m,p}},$MFCS_{Dd}$={{b},{f,m,o},{f,p,o},{m,p,o}}.扫描 d 求得项目集{b},{f,m,o},{f,p,o},{m,p,o}在 d 中的支持数,其值分别为 0,1,1,1;

(5) 搜索项目头表 $Htable_{Dd}$ 及模式树 $P\text{-}tree_{Dd}$ 求得项目集{b},{f,m,o},{f,p,o},{m,p,o}在 D 中的支持数,其值分别为 4,1,1,1,因此项目集{b},{f,m,o},{f,p,o},{m,p,o}在 $D\cup d$ 中的支持数为 4,2,2,2;

(6) 由上面的(4),(5)可得:MFS_{Dd}={{f,c,a,m,p},{b}},$MFCS_{Dd}$={{f,o},{m,o},{p,o}}.扫描 d 求得项目集{f,o},{m,o},{p,o}在 d 中的支持数,其值分别为 2,1,1;

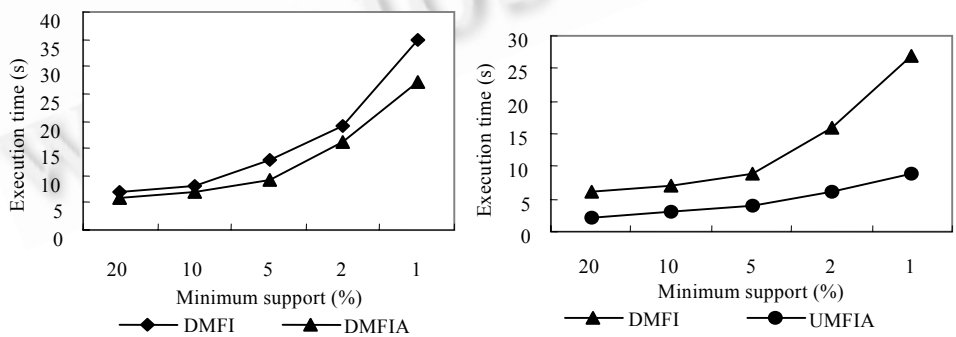
(7) 搜索项目头表 $Htable_{Dd}$ 及模式树 $P\text{-}tree_{Dd}$ 求得项目集{f,o},{m,o},{p,o}在 D 中的支持数,其值分别为 2,1,1,因此项目集{f,o},{m,o},{p,o}在 $D\cup d$ 中的支持数为 4,2,2;

(8) 由上面的(6),(7)可得:MFS_{Dd}={{f,c,a,m,p},{b},{f,o}},$MFCS_{Dd}=\varnothing$;

(9) 结束.

4 算法实现与比较

我们用 Vc++6.0 在内存 128M,CPU 为 Pentium III-733MHz,操作系统为 Windows 2000 的机上实现了 DMFI,DMFIA 和 UMFIA 算法,使用了与文献[10]同样的测试数据库,该数据库有 8 124 条记录,记录了蘑菇的 23 种属性.图 4(a)显示了在不同的最小支持度(分 5 档:20%,10%,5%,2%,1%)下算法 DMFI,DMFIA 的性能比较,由于算法 DMFIA 只需扫描数据库 D 一次,因此 DMFIA 算法的执行时间比 DMFI 算法要少,图 4(a)也表明了这一点.另外,为了验证 UMFIA 算法的有效性,我们随机从范例数据库中抽取了 8 000 条记录作为原始数据库 D,100 条记录作为新增加的数据集 d,在不同的最小支持度(分 5 档:20%,10%,5%,2%,1%)下分别对算法 UMFIA,DMFIA(对新数据库重新运行一遍)进行了测试,测试结果如图 4(b)所示.由于算法 UMFIA 充分利用了已有的一切信息(旧的最大频繁项目集、原来的 FP-tree 等)来高效地发现最新事务数据库中所有的最大频繁项目集,因此其效率是较高的.



(a) Execution time of algorithm DMFI and DMFIA (b) Execution time of algorithm DMFI and UMFIA

(a) 算法 DMFI 和 DMFIA 的执行时间 (b) 算法 DMFI 和 UMFIA 的执行时间

Fig.4 Execution time of algorithm DMFI, DMFIA and UMFIA

图 4 算法 DMFI,DMFIA 和 UMFIA 的执行时间

5 结束语

发现最大频繁项目集是多种数据挖掘应用中的关键问题.本文提出了一种快速的基于 FP-tree 的最大频繁项目集挖掘算法 DMFIA.算法 DMFIA 只需扫描数据库 D 一次,从而提高了算法的执行效率.另外,本文还对其更新算法进行研究,提出了一种增量式更新最大频繁项目集算法 UMFIA,并举例说明了算法的执行过程.

References:

[1] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD International Conference Management of Date. Washington, 1993. 207~216.

[2] Han J, Kamber M. Data Mining: Concepts and Techniques. Beijing: High Education Press, 2001.

[3] Fan M, Meng XF, *et al*. Data Mining: Concepts and Techniques. Beijing: Mechanical Industrial Press, 2001 (in Chinese).

[4] Agrawal R, Srikant R. Fast algorithm for mining association rules. In: Proceedings of the 20th International Conference on VLDB. Santiago, 1994. 487~499.

[5] Houtsma M, Swami A. Set-Oriented mining for association rules in relational databases. In: Yu PS, ed. Proceedings of the International Conference on Data Engineering. Los Alamitos: IEEE Computer Society Press, 1995. 25~33.

[6] Savasere A, Omiecinski E, Navathe SM. An efficient algorithm for mining association rules. In: Proceedings of the 21st International Conference on VLDB. Zurich, 1995. 432~444.

[7] Han J, Fu Y. Discovery of multiple-level association rules from large databases. In: Proceedings of the 21st International Conference on VLDB. Zurich, 1995. 420~431.

[8] Bayardo R. Efficiently mining long patterns from databases. In: Haas LM, ed. Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1998. 85~93.

[9] Lin DI, Kedem ZM. Pincer-Search: A new algorithm for discovering the maximum frequent set. In: Schek HJ, ed. Proceedings of the 6th European Conference on Extending Database Technology. Heidelberg: Springer-Verlag, 1998. 105~119.

[10] Lu SF, Lu ZD. Fast mining maximum frequent itemsets. Journal of Software, 2001,12(2):293~297 (in Chinese with English abstract).

[11] Han J, Jian P, Yiwen Y. Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD International Conference Management of Data. Dallas, 2000. 1~12.

附中文参考文献:

[3] 范明,孟小峰,等.数据挖掘:概念与技术.北京:机械工业出版社,2001.

[10] 路松峰,卢正鼎.快速开采最大频繁项目集.软件学报,2001,12(2):293~297.