信息系统学报
第2卷第1辑(总第2辑)：12－18

China Journal of Information Systems
Vol.2 No.1,May 2008：12－18

基于双向混合搜索的最大频繁项集发现算法*

陈富赞，李敏强
（天津大学管理学院，天津　300072）

摘　要　本文给出了一种基于层次搜索空间的、适应性更广的双向混合(Two-Way Hybrid)搜索方法，能够在迭代早期发现尽可能多的最大频繁项集，还给出了有效的搜索空间分解及剪枝策略，使得搜索空间在迭代过程中能够最大限度地快速缩小。文中对所提出的算法进行了计算实验与分析。

关键词　数据挖掘，关联规则，最大频繁项目集，双向混合搜索

中图分类号　TP311

关联规则是数据挖掘研究中极其重要的一个问题，而发现频繁项集不仅是关联规则挖掘的关键技术和步骤[1,2]，同时对于序列模式挖掘、相关性挖掘、多层模式挖掘等数据挖掘问题也是必不可少的关键步骤。关联规则的挖掘是一个两阶段的过程。第一阶段必须找到所有频繁项集的集合 FI(the set of Frequent Itemtests)；第二阶段由频繁项集产生强关联规则。其中最耗时的是第一阶段。因此，有人提出采用比 FI 更小，同时对第二阶段来说具有足够信息的替换，其中最为典型的是频繁项闭集的集合 FCI(the set of Frequent Closed Itemsets)与最大频繁项集的集合 MFI(the set of Maximal Frequent Itemsets)。显然有下列关系存在：MFI⊆FCI⊆FI。这样，挖掘 MFI 就成为关联规则挖掘的核心问题。

现有的最大频繁项集挖掘算法可以按照搜索空间树的遍历策略分为宽度优先算法和深度优先算法两大类。宽度优先算法中比较典型的有 MaxMiner 和 DMFIA 等算法。MaxMiner[3]算法采用动态排序的方法来保证高效的前瞻剪枝，减少了遍历的结点数，缺陷在于没有利用自顶向下的信息和未对 MFCS (Maximal Frequent Candidate Itemsets)进行适当的排序。DMFIA[4]算法通过两次扫描将数据库投影到一棵频繁模式树 FP-Tree 上，然后通过扫描 FP-Tree 中路径节点的计数来搜索频繁项集。DepthProject[5]算法对项集格的搜索采用了一种深度优先/宽度优先相混合的策略，同时采用了子集修剪与动态记录子节点的方法。通过动态的记录，可以修剪位于每个节点尾部以外的非频繁项，减小了搜索空间。它主要存在的问题是修剪机制的效率尚待提高。深度优先算法中比较有代表性的算法有 MAFIA、SmartMiner 和 FpMax 等。MAFIA[6]算法沿用了 DepthProject 算法的事务数据库和项集的位图表示方法，除了传统的前瞻剪枝外，MAFIA 新采用一种称为父等价(parent equivalence)的方法来做进一步的剪枝。SmartMiner[7]使用一种称为尾信息的数据结构来指导挖掘过程，能充分利用已挖掘出来的最大频繁项集中的相关信息。虽然 SmartMiner 不需要超集检测，但是其尾信息的建立和访问开销也相当大。FpMax[8]算法基于 FP-Tree 来挖掘最大频繁项集，它使用 FP-Tree 来压缩

* 基金项目：国家自然科学基金(70571057,70771074)，新世纪优秀人才支持计划 (NCET-05-0253)。
通信作者：陈富赞，天津大学管理学院，副教授，e-mail：fzchen@tju.edu.cn。

表示事务数据库中与频繁项集相关的信息，并将最大频繁项集保存在一棵 MFI-tree（maximal frequent item sets tree）中，在一定程度上提高了最大频繁项集的存取速度。

本文设计了一个新的算法：针对实际应用中最大频繁项集通常都有长有短，单纯使用 Top-Down 和 Bottom-Up 搜索方法效率都不会很高的情况，提出一种更为适用的双向混合（Two-Way Hybrid）搜索策略，其基本思想是分别从两个方向进行搜索，以期在迭代早期发现最大频繁项集，还给出了能够在两个方向上实现信息共享的搜索空间分解及剪枝策略，可以快速地缩小搜索空间、从而加快最大频繁集的查找速度。

1 搜索空间描述及分解问题

定义 1 设项集 $X,Y\subseteq I$，如果 $X\cap Y=\phi$ 并且 $X\cup Y=T$，则 T 可以表示为 $T=X:Y$，Y 的幂集记为 $P(Y)=\{Z|Z\subseteq Y\}$。项集 T 上关于 X 和 Y 的搜索空间定义为：$[X:Y]=\{X\cup Z|Z\in P(Y)\}$，记为 S_Y^X 或 S。称 X 为 S 的前项、记为 $S.anc$，Y 为 S 的后项、记为 $S.chi$，$X\cup Y$ 为 S 的边界元素、记为 $S.bor$。

定义 2 设 $S,S_i(1\leqslant i\leqslant k)$ 为搜索空间，当且仅当 $S=S_1\cup S_2\cup\cdots\cup S_k$ 并且 $S_i\cap S_j=\phi$ 时，称 $(S_1,S_2,\cdots,S_k)$ 为 S 的一个划分。此时搜索空间 S 可以记为 $S=S_1+S_2+\cdots+S_k$。

图 1 中给出了当 $I=\{a,b,c,d,e\}$ 搜索空间的层次结构，在实际应用时可以根据需要选择一个适合的分解层次。这种搜索空间描述及划分机制不仅可以突破内存及存贮空间的限制，同时也为频繁项集挖掘过程中的剪枝提供了便利。

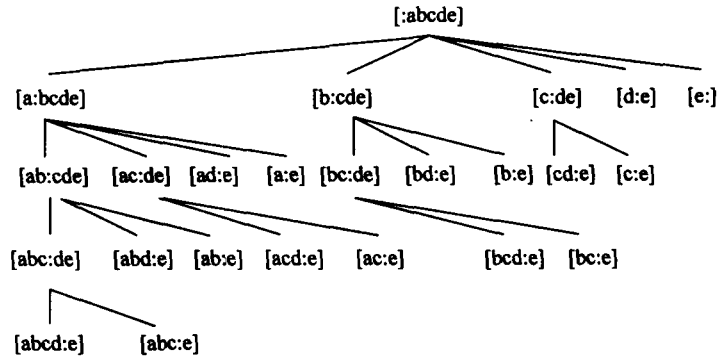

图 1 $I=\{a,b,c,d,e\}$ 上搜索空间的层次结构

2 最大频繁项集发现问题

2.1 最大频繁项集搜索及剪枝策略

在搜索空间中最大频繁项目的搜索策略通常有两种，即自顶向下（Top-Down）搜索和自底向上（Bottom-Up）搜索策略。Top-Down 搜索把某个 n 项集作为初始候选项集，通过每一次验证缩短候选项集的长度。当某个 k 项集被验证为非频繁的，那么它的所有 $(k+1)$ 项子集不用在下次验证中进行检验。最大频繁项集较长时该搜索策略比较适合。Bottom-Up 搜索分为两步：1）由频繁 k 项集通过连接、剪枝生成 $(k+1)$ 项候选项集；2）扫描数据库，计算 $(k+1)$ 项候选项集的支持度，生成频繁 $(k+1)$ 项

集。反复执行这两步直至找到 MFS。Apriori 算法是一种典型的自底向上模式,最大频繁项集较短时该搜索策略比较适合。

Two-Way Hybrid 搜索过程由 Bottom-Up 及 Top-Down 两部分组成,在迭代过程中 Bottom-Up 搜索从 1-维项集开始通过频繁项集 L_k 自连接以长度逐步增长方式生成候选频繁项集,并判断当前的频繁项集是否是最大频繁项集;与此同时 Top-Down 对搜索空间$[X: Y]$进行逐层分解剪枝,使得$|Y|$值不断减少,进而缩小搜索空间。

在搜索过程中对搜索空间的剪枝策略非常重要,有效的剪枝策略可以大幅度缩小搜索空间,从而提高搜索效率。Two-Way Hybrid 不仅利用 Apriori 性质进行了剪枝,并且在 Top-Down 过程中将搜索空间从 k 层到 $k+1$ 层分解时,也共享 Bottom-Up 过程产生的频繁项集及非频繁项集进行剪枝:如果项目集 R 是频繁的,则它的所有子集 $A\in[:R]$也一定是频繁的;如果项目集 R 是非频繁的,则它的所有超集 $B\in[R: V]$也一定是非频繁的。以下将就 Two-Way Hybrid 搜索中如何利用已知非频繁项集(或频繁项集)R 对搜索空间 $S=[X: Y]$进行剪枝的问题进行讨论。

定理 1 设 $X,Y,R\subseteq I, R=\{i_1,i_2,\cdots,i_k\}$(其中 $i_j\in I$ 且 $i_j\notin X,Y$),搜索空间 $S=[X: RY]$可以被划分为如下子空间:

$$[X: i_1 i_2\cdots i_k Y]=[X: Y]+\sum_{j=1}^{k}[Xi_j: i_{j+1}\cdots i_k Y]$$

证明 对任意 $w\in I$ 项集 wV 的幂集 $P(wV)=\{Z\mid Z\subseteq wV\}=\{Z\mid Z\subseteq V\}\cup\{w,wV\}$,根据搜索空间的定义可得$[X: wV]=[X: V]+[Xw: V]$。依次将 $i_j\in R$ 应用于上式中定理即可得证。

引理 1 设 $X,Y\subseteq I, u\in I$,如果项集 $R=Xu$ 是非频繁的,那么搜索空间 $S=[X: uY]$可以被 R 进行剪枝,并且剪枝后的搜索空间为$[X: Y]$。

定理 2 设 $X,Y\subseteq I$,如果存在 $u\in Y$,使得 $R=Xu$ 是一个非频繁项集,那么将搜索空间 $S=[X: Y]$分解为$[Xu: (Y-\{u\})]+[X: (Y-\{u\})]$最有效。

证明 搜索空间$[X: Y]$的大小取决 $|Y|$ 的值,设 $Y=\{i_1,i_2,\cdots,i_k\}$,那么根据定义 2 可知 $|S|=2^k$。S根据定理1可以分解成$\sum_{j=1}^{k}[Xi_j: i_{j+1}\cdots i_k]$,如果存在满足已知条件的$u\in Y$,即$R=Xu$是一个非频繁项集,设 $S_j=[Xi_j: i_{j+1}\cdots i_k]$,根据性质 2 可知 S_j 可以被完全剪枝。显然当 $j=1$(即 $i_1=u$) 时 $|S_j|$ 值最大。结合定理 1 该定理得证。

2.2 大频繁项集发现算法

输入一个项目集合 $I=\{i_1,i_2,\cdots,i_m\}$上的交易数据集 D 及其上的通过算法 1 中给出的 Two-Way Hybrid 搜索可以发现其中的最大频繁项集。Two-Way Hybrid 算法分别从 Bottom-Up 和 Top-Down 两个方向通过迭代循环向中间层逼近,在第 k 次迭代中 Bottom-Up 过程采用与 Apriori 算法相同的频繁项集连接方式(Apriori-gen[3])产生候选项集 C_k 并计算出其中的频繁 k 维项集 L_k 与$\sim L_k$;Top-Down 过程对当前的搜索空间 $S=[X: Y]$根据定理 2 用$\sim L_k$ 进行剪枝并分解。

算法 1 最大频繁项集发现算法

Algorithm Maximal Frequent Itemsets Search (Transaction Set: D, Item Set: I)

Output: MFI

Begin

$MFI_T=\Phi$; $MFI_B=\Phi$; $MFC=\Phi$; $k=1$;

$\Psi_1=\{[: I]\}$; $C_k=I$; $L_k=$ Partition $(C_k, minsup)$;

```
    ~L_k = Partition (C_k, minsup);
    While Ψ_k ≠ Φ do
      //Top-Down Search
      Ψ_{k+1} = Φ;
      Forall S ∈ Ψ_k do
        B = S.bor;
        If countsupport(B) < minsup then
          If {R | R ⊇ B and R ∈ MFI_T} ≠ Φ then
              Ω = Decompose(S, L_k, ~L_k);
              Ψ_{k+1} = Ψ_{k+1} ∪ Ω;
          Endif;
        Else
            MFI_T = MFI_T ∪ B;
        Endif;
      Endfor;
      //Bottom-Up Search
      C_{k+1} = Apriori-gen(L_k);
              //candidate (k+1)-itemsets
      L_{k+1} = Partition (C_{k+1}, minsup);
      ~L_{k+1} = Partition (C_{k+1}, minsup);
      Forall X ∈ L_k do
            If X ∩ {Y | Y ∈ ~L_{k+1}} ≠ φ then MFI_B = MFI_B ∪ X;
      Endfor;
      k++ ;
    Endwhile;
    Return MFI_T ∪ MFI_B;
End;
```

算法 2　将候选项集 C_k 根据最小支持度阈值 *minsup* 划分为频繁项集 L_k 及 $\sim L_k$

```
Procedure Partition (set of candidate itemsets: C_k, minimum support: minsup)
//classify L_k and ~L_k from set of candidate k-itemsets C_k
Output: L_k, ~L_k
Begin
    L_k = Φ; ~L_k = Φ;
    Forall X_j ∈ C_k do
        if CountSupport(X_j) ≥ minsup then
            L_k = L_k ∪ X_j;
        else
            ~L_k = ~L_k ∪ X_j;
        Endif;
```

Endfor;

End;

算法 3 用 $\sim L_k$ 对前项维度为 k-1 搜索空间 $S=[X; Y]$进行剪枝

Procedure Decompose (search space: $S=[X; Y]$, frequent k-itemsets: L_k, infrequent k-itemsets: $\sim L_k$)

Output: search space set: Ω

Begin

$\Omega=\Phi$;

$\sim P_k=\{Z|Z\supset X \text{ and } Z\in\sim L_k\}$; $P_k=\{U|U\supset X \text{ and } U\in L_k\}$;

$R=Y-\bigcup\{V|V\in\sim P_k\}$; $R=R\cap\{T|T\in P_k\}$;

$n=|R|$;

Forall($i=1$; $i=n-1$; i++) do

If $Xu_i\in P_k$ and $|Xu_iu_{i+1}\cdots u_k|>k$ then

$\Omega=\Omega\cup[Xu_i: u_{i+1}\cdots u_k]$;

Endfor;

Return Ω;

End;

3 算法分析及实验

为了测试上文所给出算法的性能,我们在 PC 机上进行了实验,实验环境为: Intel P4 2.93GHz CPU、1GHz 内存、Windows XP。实验数据用 IBM Almaden 研究中心提供的程序生成(http://www.almaden.ibm.com/software/projects/hdb/Resources/datasets/syndata.html#assocSynData),设定项目个数为 1000,测试数据集实例以 Tx.Iy.DmK 表示。

实验 1 数据集采用 T10.I6.D10K,在不同的最小支持度取值下分别用 bottom-up、top-down 及 two-way hybrid 搜索策略发现最大频繁项目集。实验结果如图 2 及图 3 所示,从图 2 中可以发现当最小支持度阈值大于 2%时,bottom-up 搜索策略的性能稍好于其他两种策略,原因在于随着最小支持度的增大,在搜索过程中产生的频繁项目集会大幅度减少,bottom-up 搜索策略可以表现出较好的性能;而小于 2%时 two-way hybrid 搜索策略的性能较好,主要原因是它可以充分利用双向产生的信息,尽快产生较长及较短的最大频繁项目集,并且可以通过有效的剪枝策略缩小搜索空间。当最大频繁项目集普遍较短时 bottom-up 搜索策略比较适合,反之 top-down 比较适合,而 two-way hybrid 搜索策略会在最大频繁项目集长度分布不均匀时表现出良好的性能。

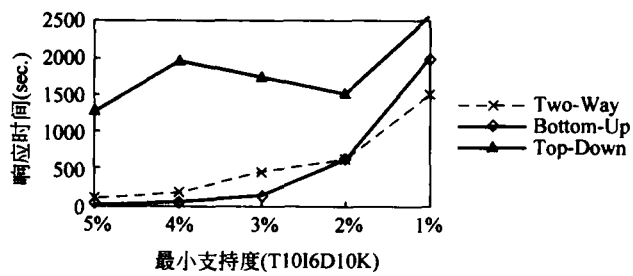

图 2 最小支持度对算法的影响

实验 2 数据集采用 T15. I8. D10K，在不同最小支持度阈值下测试 two-way hybrid 搜索策略的性能，实验结果如图 4 所示，实验表明随着最小支持度阈值的降低，会产生更多、更长的最大频繁项目集，算法需要检验更多的项目集，因此会需要更多的计算开销。

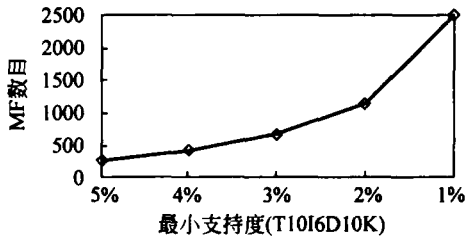

图 3 不同支持度下发现 MFI 的数目

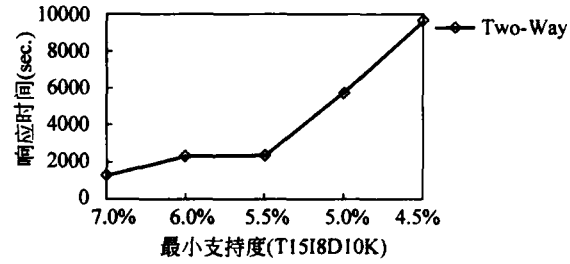

图 4 不同支持度下算法的性能

4 结束语

本文给出了一些能够有效发现最大频繁项集的新算法。该算法以一种层次结构的搜索空间组织形式为基础，结合 Top-Down 和 Bottom-Up 搜索方法提出一种适应性更广的双向混合(Two-Way Hybrid)搜索方法。在算法采用的层次结构组织形式搜索空间中每层都可以分解为若干个相互独立的子空间，每个子空间都可以单独处理，为算法的并行化提供了基础。最大频繁项集并行化搜索方法也是我们下一步准备研究的问题。

参考文献

[1] Agrawal R, ImielinSki T, Swami A. Mining association rules between sets of items in large database[C]. Proceedings of the ACMSIG2 MOD International Conference on Management of Data, Washington, DC, 1993, 2: 207-216.

[2] Srikant A. R. Fast algorithms for mining association rules[C]. Proceedings of the 20th International Conference Very Large Data Bases(VLDB'94), Santiago, Chile, 1994, 487-499.

[3] Bayardo R. Efficiently mining long patterns from databases[C]. Proceedings of the ACM SIGMOD, Int'l Conference On Management of Data. New York: ACM Press, 1998, 85-93.

[4] YanYJ, Li ZJ, Chen HW. Efficiently Mining of Maximal Frequent Item Sets Based on FP-Tree[J]. Journal of Software, 2005, 16(2): 215-222.

[5] AgarwalRC, Aggarwal CC, Prasad VVV. Depth First generation of long patterns[C]. Proceedings of the 6th ACM SIGKDD Int' l Conference On Knowledge Discovery and Data Mining. 2000, 108-118.

[6] Doug Burdick, Manuel Calimlim, J. E. Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases[C]. Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, April 2001.

[7] Zou Q, Chu W W, SmartMiner: A Depth First Algorithm Guided by Tail Information for Mining Maximal Frequent Itemsets[C]. Proceedings of the Second IEEE International Conference on Data Mining (ICDM'02), 2002, 570-577.

[8] Grahne G, Zhu Jianfei. High performance mining of maximal frequent itemsets[C]. Proceedings of the 6th SIAM International Workshop on High Performance Data Mining. San Francisco, 2003, 311-337.

An Efficiently Algorithm Based on Two-Way Hybrid Search for Finding Maximal Frequent Itemsets

CHEN Fuzan & LI Minqiang
(School of Management, Tianjin University, Tianjin 300072)

Abstract A new algorithm based on the two-way hybrid search and structure of multi-level search space organization for mining maximal frequent itemsets is proposed. This method can discover more maximal frequent itemsets as soon as possible. Some efficient decomposition and pruning strategies are implied in this method, which can reduce the original search space rapidly in the iterations. And experimental and analytical results are presented in the end.

Key words Data mining, Association rules, Maximal Frequent Itemsets, Two-Way Hybrid Search