基于邻接网络的频繁项目集发现算法①

陈富赞，李敏强
（天津大学管理学院，天津 300072）

摘要： 频繁项目集发现一直都是关联规则研究领域中最关键的问题.文章给出了一个新的频繁项目集发现算法，该算法的特别之处在于事先利用有向图进行的一次数据预处理，在预处理过程中将数据库预先存贮为每个结点都有一个域来记录其支持度的项目集邻接网络，从而把复杂的频繁项目集的发现问题转化为简单的图中搜索问题，这就大大提高了频繁项目集发现过程的效率.同时为了有效地解决预处理过程中的项目集支持度计算问题，采用了一种纵向的数据库表示格式.最后对所采用的算法给出实验结果.

关键词： 数据挖掘；关联规则；邻接网络；频繁项目集
中图分类号： TP311 **文献标识码：** A **文章编号：** 1007－9807(2006)03－0054－08

0 引 言

数据挖掘(data mining)，也称为数据库中的知识发现KDD(knowledge discovery in database)，是从大量原始数据中挖掘出隐含的、有用的、尚未发现的信息和知识.关联规则(association rules)是数据挖掘中的一个非常重要的研究内容.关联规则是表示数据库中一组对象之间某种关联关系的规则.关联规则挖掘的对象是交易(Transactional)数据库.例如，关联规则可以表示“购买了商品A和B的顾客中有80%的人又购买了商品C和D”.关联规则提供的信息可以用作商品目录设计、商场货架的布置、生产安排、具有针对性的市场营销等.

从大规模数据库中发现出所有频繁的关联规则的任务是一项非常困难并且具有挑战性的任务，而其中频繁项目集的发现更是一个瓶颈问题.围绕这个问题，Agrawal等在1994年提出Apriori算法[2]，基本的方法是重复扫描数据库，在第K次扫描产生长度为K的频繁项目集.Park等人提出的DHP算法，使用哈希(Hashing)技术有效地改进了备选集C_K的产生过程.Savasere等在1995年提出了一种把数据库分割处理的算法[4]，降低了挖掘过程中I/O操作的次数.以后更进一步地研究涉及分布和并行环境下挖掘关联规则，例如，Cheung等提出了一种关联规则的快速分布式挖掘算法(FDM).目前所研究出的算法从性质上来看大都是迭代性的，需要对数据库进行多次遍历，显然算法的开销会很大.一些采用了抽样技术的算法对数据的非均匀分布非常敏感，这也会对算法的性能产生非常巨大的影响.另外，大部分算法都采用了复杂的内部数据结构.

针对目前频繁项目集发现算法中存在的迭代次数多、数据结构复杂等问题，本文给出了一个新的算法.算法以图论为基础，将交易数据预先存贮在一个有向图——项目集邻接网络中，其中不仅可以通过有向边表示项目集间的次序关系，而且为了提高搜索效率还存贮了项目集结点的支持度.这就使得数据库中频繁项目集的发现问题转化为邻接网络中的搜索问题.以往对项目集支持度的计算采用的都是遍历整个数据库，并累计其支持交易数的方式，显然这种计算过程无论是I/O

① 收稿日期：2003－03－24；修订日期：2006－03－06.
基金项目：国家自然科学基金资助项目(70571057).
作者简介：陈富赞(1972—)，女，河北人，博士，副教授.

开销，还是计算开销都十分巨大，为了解决这一问题，本文采用了纵向数据库，将每个项目都与支持它的一个交易列表对应，这样在计算项目集的支持度时为不再需要对整个数据库进行遍历，通过项目对应的交易列表间的交集操作就可以计算出来．另外，针对问题搜索空间巨大，难以在主存中一次完成全部处理的特点，利用等价类的概念将问题空间进行了分解，使得分解后的每个子空间都可以在主存中独立处理．

1　关联规则挖掘任务定义

1.1　关联规则的定义[1]

设 I 为一个由 m 个项目组成的集合 $I=\{i_1,i_2,\cdots,i_m\}$，T 是针对 I 的交易，每一个交易包含若干个项目 $i_i,i_j,\cdots,i_k\in I$，即交易 T 为由 I 中项目组成的 I 的子集（$T\subseteq I$）．由 I 中项目组成的 I 的子集称为项目集（itemset），项目集中所包含的项目的个数称为它的长度或基数．长度为 k 的项目集称为 $k(k=|X|)$ 维项目集，记为 k-*itemset*．每一个项目集都有一个统计指标称为“支持度”（*support*）：数据库 D 中支持项目集 X 的交易所占的比例称为 X 在 D 中的支持度，记为 $supp(X)$ 或 $S(X)$．设 *minsup* 为给定的最小支持度，如果 $supp(X)\geqslant minsup$，则称项目集 X 是频繁的（frequent）．一个项目集的最小支持度是该项目集被认为是有意义时，支持它的交易占数据库 D 中所有交易总合的最小比例，通常是根据经验来确定的．

关联规则表示为 $X\Rightarrow Y$，其中 $X,Y\subset I$，并且 $X\cap Y=\varnothing$．X 称做规则的前提或前项，Y 为结果或后项．每个规则也有两个度量标准：支持度（*support*）及可信度（*confidence*）．规则支持度的定义为

$$support(X\Rightarrow Y)=support(X\cup Y)$$

规则可信度的定义为

$$Confidence(X\Rightarrow Y)=\frac{support(X\cup Y)}{support(X)}$$

1.2　关联规则挖掘问题的定义

关联规则挖掘问题就是要找出这样的一些规则，它们的支持度或可信度分别大于用户指定的最小支持度值 *minsup* 和最小可信度值 *minconf*．因此，该问题可以分解成如下两个子问题：

1）产生所有支持度大于等于指定最小支持度值 *minsup* 的项目集，这些项目集称为频繁项目集（frequent itemsets），而其它的项目集则称为非频繁项目集（infrequent itemsets）．

2）对每个频繁项目集，产生可信度大于等于最小可信度 *minconf* 的规则，如下：对于一个频繁项目集 L 及任意 $S\subset L$，如果 $support(L)/support(S)\geqslant minconf$，那么规则 $S\Rightarrow L-S$ 就是一个有用规则．

关联规则挖掘问题的主要特征是数据量巨大，因此算法的效率是关键．目前研究的重点在第1步，即找出频繁项目集，相对而言，第2步更直接、更容易．

为了简化关联规则发现问题的讨论，设实例交易数据库 D 如表1所示，数据库中的项目集 $I=\{A,B,C,D,E\}$．设最小支持度 $minsup=0.3$，由表1可以得到最大频繁项目集 *ABCE*，*ACD*，*BCD*，*CDE*．

表1　交易数据库 D

Table 1 Transaction database D

TID	1	2	3	4	5	6	7	8	9	10
项目	*ABCE*	*CDE*	*ABCE*	*ACDE*	*ABCDE*	*BCD*	*DE*	*BCE*	*ABCD*	*BE*

2　项目集邻接网络

从图论中的有向图的定义可知，一个集合 $P=\{p_1,p_2,\cdots,p_m\}$ 上的关系 R 可以用一个有向图表示．一个有向图 $G=\langle V,E\rangle$ 可表示一个 X 上的关系 R．集合 P 中的元素可用图中的结点表示，即 V 表示 P；关系 R 的有序偶 (p_i,p_j) 可用图中从结点 p_i 到 p_j 的有向边表示，即 E 表示 R．这样数据库 D 中项目集 I 上的关系就可以用一个有向图来表示．

2.1　项目集邻接网络的定义

设 R 是集合 P 上的关系，如果 R 是自反的、非对称的、传递的，则称 R 是集合 R 上的偏序关系．偏序关系一般用 $\leqslant$ 表示，具有偏序关系的集合称为有序集合．如果关系 R 是非自反的、传递的，则称 R 是集合 P 上的拟序关系，拟序关系一般用 $<$ 表示．显然数据库 D 中项目集 I 所组成的幂集 $P(I)$ 上的关系“$\subset$”是拟序的．

定义1　设 P 是一个有序集合，$Y\subseteq P$，如果存在一个 $x\in P$，则

1) 如果在 P 中不存在一个元素 x' 有 $x \neq x'$ 并且 $x \leqslant x'$，则称 x 是 P 的极大元素或顶元素，记为 $\overline{P}$；

2) 如果在 X 中不存在一个元素 x' 有 $x \neq x'$ 并且 $x' \leqslant x$，则称 x 是 P 的极小元素或底元素，记为 $\underline{P}$.

定义 2 设 P 是一个有序集合，$x, y \in P$，且 x 是 P 的底元素，如果不存在一个 $x' \in P$，使得 $x < x' < y$，则称 y 是 P 的一个原子. P 中由原子组成的集合记为 $A(P)$.

定义 3 设 X 及 Y 分别为数据库 D 中的项目集 I 上的两个不同的项目集，如果在项目集 X 中加入一个项目可以得到项目集 Y，则称这两个项目集是相邻的. 此时称 X 是 Y 的父亲，而 Y 则称为 X 的孩子.

由以上定义可知一个项目集可能具有多个父亲及孩子. 对项目集 X 中的每个元素 i_k 来说 $X - \{i_k\}$ 都是 X 的父亲，因此项目集 X 所有父亲的数目都正好等于项目集合 X 的基数.

定义 4 数据库 D 中项目集 I 所组成的幂集 $P(I)$ 上的邻接网络 L 的结构如下：对项目集 $I = \{i_1, i_2, \cdots, i_m\}$ 来说，它是一个以项目集 $X \subseteq I$ 为结点的有向无环图，结点 $v(X)$ 内的项目都按一定次序排列，并且每个结点都有一个属性来记录它的支持度值，这一属性以 $S(X)$ 来表示；设 $v(X)$ 及 $v(Y)$ 分别为与项目集 X 及 Y 对应的两个结点，当且仅当 X 是 Y 的父亲时，在 $v(X)$ 及 $v(Y)$ 之间存在一条有向边. $v(X)$ 及 $v(Y)$ 之间的边以 $E(X, Y)$ 表示. 结点 $v(X)$ 称为边 $E(X, Y)$ 的尾，而 $v(Y)$ 则称为边 $E(X, Y)$ 的头.

项目集邻接网络 $L = P(I)$ 的顶元素为 I，底元素为 $\{\}$，原子集合为 I.

显然，项目集邻接网络中如果结点 $v(X)$ 与 $v(Z)$ 之间存在一条由 $v(X)$ 指向 $v(Z)$ 的通路，那么对项目集 X 与 Z 而言存在关系 $X \subset Z$. 此时称 X 是 Z 的一个祖先，而 Z 则称为 X 的一个后代.

图 1 中给出了实例数据库 D 中项目集 I 的所对应的项目集邻接网络 L. 另外图 1 中还给出了最大频繁项目集. 对于任意频繁项目集 X 及 Y 来说，$X \cap Y$ 也是频繁，但 $X \cup Y$ 不一定也是频繁的.

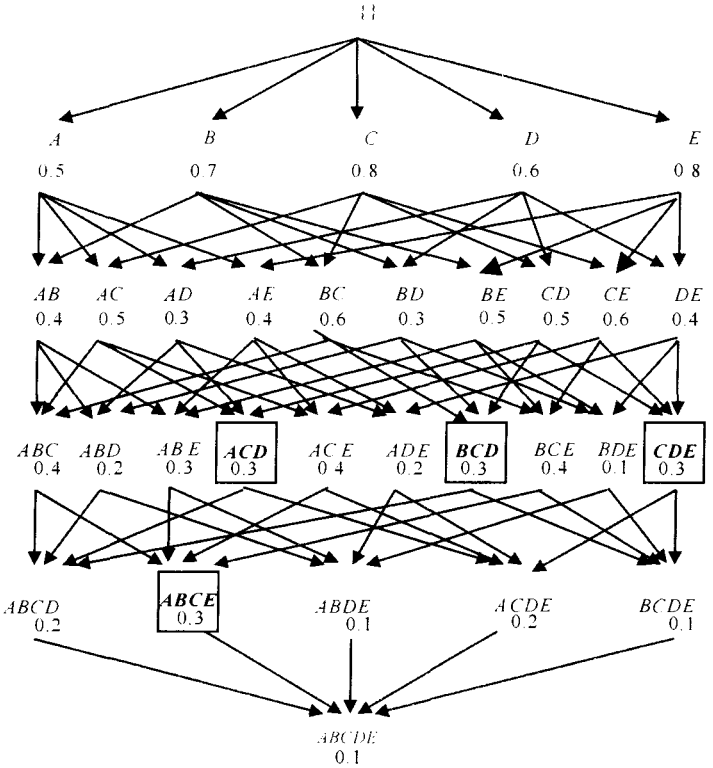


图 1 项目集邻接网络 $L = P(I)$

Fig.1 Adjacent itemsets lattice $L = P(I)$

性质 1 项目集邻接网络 L 中所有结点 $X \in L$ 都可以表示为 $X = \bigcup \{Y \in A(L) \mid Y \leqslant X\}$.

定义5　设 L 是一个项目集邻接网络，$S \subset L$，对所有结点 $X, Y \in S$，如果 $X \cup Y \in S$，并且 $X \cap Y \in S$，则称 S 是 L 的子网络.

在项目集邻接网络，每个结点都有一个属性对应其支持度值.由项目集邻接网络 L 可以得出以下结论：

定理1　在 L 中如果结点 $v(Y)$ 是结点 $v(X)$ 的一个后代，则必有 $S(Y) \leqslant S(X)$.

证明　根据项目集相邻的概念，后代项目集 Y 是通过对祖先项目集 X 添加项目得到的，也即有 $Y \supset X$. 再由支持度的定义可证 $S(Y) \leqslant S(X)$.

定理2　项目集邻接网络中边的数目与所有结点项目集基数的合计数相等.

证明　项目集邻接网络中边的数目可以通过累计每个初始项目集存在的父亲的数目获得.一个项目集的所有父亲的数目与该项目集中项目的数目相等.由此定理得证.

2.2　项目集支持度的计算

目前大部分关联规则发现算法所采用的数据库表示形式都是横向的，如表1所示，数据库由一个交易的列表组成，其中每个交易中除了该交易中的项目列表以外还有一个交易标识.本文采用了一种纵向数据库格式，如表2所示，对每个项目保存一个交易列表.这就使得项目集支持度的计算通过简单的 tid-list 之间的逻辑交操作就可实现.

表2　交易数据库 D(纵向表示形式)

Table 2 Transaction database D(vertical format)

项目	A	B	C	D	E
tid-list	13459	1356891 0	12345689	245679	1234578 10

定义5　设 D 是一个纵向数据库，$L = P(I)$ 是数据库中项目集 I 对应的邻接网络.L 中每个原子 A 对应的、由支持它的交易的标识号组成 tid-list 的记为 $L(A)$，$L(A)$ 中元素的数目称为它的基数，记为 $|L(A)|$.

性质2　数据库 D 中的交易数目为 N，项目集 I 对应的项目集邻接网络 $L = P(I)$，对于任意 $X \in L$，设 $J = \{Y \in A(L) \mid Y \leqslant X\}$. 则有 $X = \bigcup_{Y \in J} Y$ 及 $S(X) = \left|\bigcap_{Y \in J} L(Y)\right| / N$.

性质2表明任何一个项目集都可以通过网络中一些原子的并操作得到，并且该项目集的支持度可以通过这些原子的 tid-list 的交操作来得到.在项目集的集合上一般化性质2可得：

性质3　数据库 D 中的交易数目为 N，项目集 I 对应的项目集邻接网络 $L = P(I)$，$J \subset L$，对于任意 $X \in L$，如果 $X = \bigcup_{Y \in J} Y$，则有 $S(X) = \left|\bigcap_{Y \in J} L(Y)\right| / N$.

性质3表明如果一个项目集是通过 J 中项目集的并操作得到的，那么它的支持度可以通过 J 中项目集对应的 tid-list 的交操作来得到.可见任何一个 k 维项目集的支持度都可以通过它的两个任意 $(k-1)$ 维子集所对应的 tid-list 之间简单的并操作来确定.因此，对于当前处理层上的一个项目集来说，取上一层中按一定顺序排列的前两个子集，就可以计算出该项目集的支持度.

定理3　设 X 与 Y 是两个项目集，如果 $X \subseteq Y$，则 $L(X) \supseteq L(Y)$.

证明　由支持度的定义可证.

定理3表明如果 X 是 Y 的一个子集，则 Y 的 tid-list 的基数(即 Y 的支持度)一定小于等于 X 的 tid-list 的基数.由以上推论可以得出一个重要结论：网络中的中间 tid-list 的基数会随着网络中层数的增加而减小.这一结果将大大加速交集操作及支持度的计算.

2.3　项目集邻接网络的生成

给定一个纵向数据库 D，N 为数据库 D 中交易数目，$I = \{i_1, i_2, \cdots, i_m\}$ 为数据库 D 中的项目集合.下面的算法可以生成出项目集的邻接网络 L.

算法1　邻接网络的生成算法

```
Algorithm ConstructLattice(TransactionDatabase: D)
Begin
  L = ∅;
  For all transaction t ∈ D do
    for each itemset X = {i_1, ..., i_r} in t do
      S(X) = |∩ L(i_j)| / N
      add the vertex v(X) to L with label S(X)
      add the edge E(X - {i_k}, X) to L for
            each k ∈ {1, ..., r}
    end for;
end;
```

2.4　项目集邻接网络的分解

在实际应用中系统往往没有足够大的主存空间来处理整个项目集邻接网络，因此就需要将过

大的搜索空间分割成较小的、可以独立在主存中处理的子空间,即项目集邻接网络的分解.本文在下面的部分就讨论这一问题.

定义 6 设集合 X 上的关系 R,如果它是自反的、对称的、传递的,则称 R 为等价关系,记为 $\equiv$.

定义 7 设 S 是一个集合,$A_1, A_2, \cdots, A_m$ 是它的子集,如果它们满足下列条件:

1) 所有 A_i 间均是分离的,即对所有 $i, j = 1, 2, \cdots, m$,如果 $i \neq j$,则 $A_i \cap A_j = \varnothing$.

2) $A_1 \cup A_2 \cup \cdots \cup A_m = S$

则集合 $A = \{A_1, A_2, \cdots, A_m\}$ 称为 S 的一个划分,而 $A_1, A_2, \cdots, A_m$ 称为这个划分的块.

定义 8 设 R 是集合 X 上的等价关系,对任意一个 $x \in X$,可以构造一个 X 的子集 $[x]_R$,称为 x 对于 R 的等价类:$[x]_R = \{y \mid y \in X \text{ 且 } x \equiv y\}$.

定理 4 集合 X 上的等价关系 R 所构成的类产生一个集合 X 的划分.

定理 5 任一集合 X 上的一个划分 C 可产生一个等价关系.

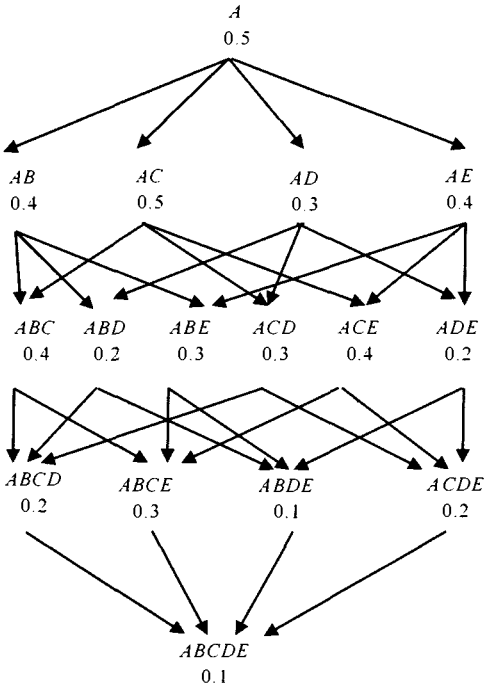
项目集邻接网络 $L = P(I)$ 中的结点为项目集 I 的幂集中的元素,由定理 6 可知项目集邻接网络中存在等价关系.

定义 9 f 为项目集 X 上的一个函数关系,它的功能是取内部项目按一定次序排列的项目集 X 上的前 k 个项目,即函数 $f(X, k) = X[1 : k]$.设 $\equiv_k$ 为项目集邻接网络 L 上的满足函数关系 f 的等价关系,即对任意 $X, Y \in L$,使得 $X \equiv_k Y \Leftrightarrow (X, k) = f(Y, k)$.

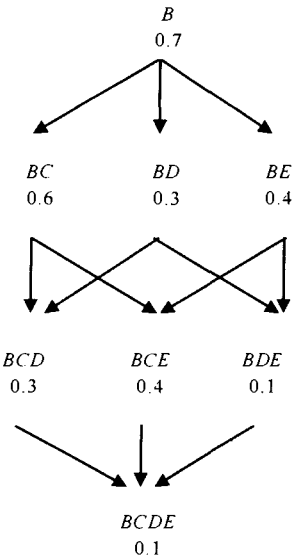
定理 6 在项目集邻接网络 L 中,每个由等价关系 $\equiv_k$ 产生的等价类 $[X]_k$ 都是 L 的一个子网络.

证明 设项目集 $A, B \in [X]_k$,根据定义 12 项目集 A 及 B 都具有相同的项目子集 X,由 $A \cup B \supseteq X$ 可知 $A \cup B \in [X]_k$;并且由 $A \cap B \supseteq X$ 可知 $A \cap B \in [X]_k$,根据定义 7 中对子网络定义定理可证.

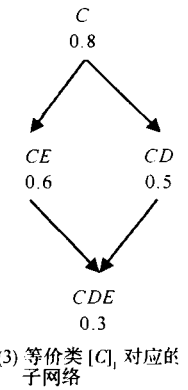
对图 1 中的项目集网络 $L = P(I)$ 应用 $\equiv_1$ 可以得到的 5 个各自独立的子网络,图 2 中给出 $[A]_1$、$[B]_1$ 及 $[C]_1$ 对应的子网络,$[D]_1$ 中包括 D、DE 两个项目集,而 $[E]_1$ 中只有一个项目集 E.



(1) 等价类 $[A]_1$ 对应的子网络



(2) 等价类 $[B]_1$ 对应的子网络



(3) 等价类 $[C]_1$ 对应的子网络

图 2 由等价关系 $\equiv_1$ 产生的 $L = P(I)$ 的子网络

Fig.2 Sub-lattices of $L = P(I)$ induced by $\equiv_1$

如果对项目集邻接网络 L 利用 $\equiv_1$ 进行一层 分解后得到的子网络还是太大,则可以对子网络

用 $\equiv_2$ 进行递归的等价类分解，直到所有的子网络都能够在主存中处理.

3　频繁项目集的搜索

3.1　邻接网络中的搜索算法

为了在项目集邻接网络或子网络 L 中检索出能满足最小支持度 s 约束条件的所有项目集，需要在网络 L 中解决以下搜索问题.

问题 1　对给定的项目集 X，检索出满足下列条件的所有项目集 Y：在网络 L 中 $v(X)$ 与 $v(Y)$ 之间存在一条通路，并且 $S(Y) \geqslant s$.

在项目集邻接网络中，从网络的底元素出发可以通过一条路途到达的结点数目虽然很多，但是能够满足最小支持度 s 约束条件的结点数目却是有限的.利用网络结构可以限定需要检验的结点数目.

设在项目集邻接网络及子网络的每一层中，项目集按支持度的大小进行逆序排列.下面将主要讨论在各个子网络中列举出频繁项目集 Breadth-First 搜索策略.Breadth-First 搜索过程首先从子网络的底元素开始，从上往下逐层搜索.即首先处理完一层中的所有项目集后，然后才开始下一层中项目集的处理过程，由于搜索过程是广度优先的，因此该技术可以保证列举出所有频繁项目集.图 3 中给出了子网络 $[A]_1$ 中各结点的搜索顺序.

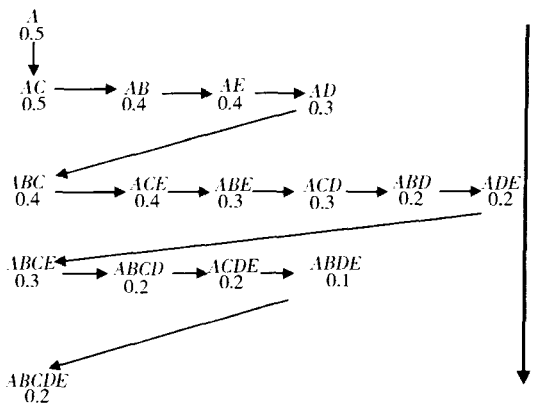


图 3　Breadth-First 搜索过程

Fig.3 Breadth-First search in sub-lattice

算法 2 给出了 Breadth-First 搜索过程.算法的输入是底元素为 B，顶元素为 T 的子网络 S，输出的是子网络 S 中只由频繁项目集结点组成的部分 FS. 搜索过程从子网络中底元素 B 及 $List = \{v(B)\}$ 开始，检查 $v(B)$ 的每个未访问过的孩子结点 $v(Y)$，如果支持度 $S(Y) \geqslant minsup$，则将该结点加入到 $List$ 中，将结点及边加入到 FS 中，然后将 $v(B)$ 从 $List$ 中删除.重复该过程直至 $List$ 为空.对 $List$ 中给定的一个结点，它所有未到达过的支持度大于等于 $minsup$ 的孩子结点都会被搜索到.算法终止时，FS 中包含了所有满足 $S(Y) \geqslant minsup$ 条件的项目集及反映邻接关系的边，并且 $Y \supseteq B$.

算法 2　Breadth-First 搜索过程

```
Algorithm Breadth-First(S)
Begin
  FS = {(v(B),S(B))}; List = {v(B)};
  While List ≠ Ø do
  Select v(X) from List;
  For each unvisited child v(Y) of v(X) do
    if S(Y) ≥ minsup do
      List = List ∪ v(Y);
      add (v(Y),S(Y)) to FS
      for each i_j ∈ Y add E(Y - {i_j},Y) to FS
    end for;
  end do;
  delete v(X) from List
end;
```

由于非频繁项目集的超集也都是非频繁的，因此在搜索过程中可以利用这一性质进行剪枝以提高算法的效率.

3.2　算法分析

与以往的需要对数据库进行多次遍历的算法相比，本文所采用的方法只需要在数据预处理过程中进行两次数据库遍历：第 1 次将数据库表示形式由横向向纵向转换，第 2 次构造项目集邻接网络.在邻接网络生成过程中，生成一个新的项目集结点时，只对结点任意两个子集所对应的 tid-list 进行操作，就可以确定出新结点的支持度.

项目集具有这样一个性质：如果一个项目集存在一个非频繁子集，那么该项目集就是一个非频繁项目集.这一性质表明，在频繁项目集搜索过程中，只需要对底元素频繁的子网络进行搜索就可以确定出频繁项目集，而在底元素不是频繁项目集的子网络中，根本不可能存在频繁项目集.这一性质也可以大大减少对数据的读取.

从时间摊销的角度来看，频繁项目集发现过程通常采用的方式是根据给定的最小支持度来临时从数据库中搜索出频繁项目集；本文所采用的

方式是在预处理过程中计算出项目集的支持度，在频繁项目集搜索过程中不需要对整个数据库进行读取，而只是对相应子网络进行操作.这样预处理的时间可以分摊在多次任务中，相应地任务执行次数越多，本算法的优势也就越显著.

4 实验计算与结果分析

为了测试上文所给出算法的性能，在一个高性能 PC 机上实现了算法. 合成数据是采用 IBM Almaden 研究中心 Rekesh Agrawal 教授领导的 KDD 研究小组提供的源程序生成的，见 Http://www.almaden.ibm.com/cs/quest/syndata # AssocSynData. 设 D 为数据库中交易记录的数目，T 为交易数据记录的平均长度，I 为最大潜在频繁项目集的平均长度.一个测试数据库实例记为 Tx.Iy.DmK.

本文所给出的频繁项目集发现问题解决方案由三部分组成，首先将交易数据库 D 由横向格式转换为纵向格式，然后再生成与数据库 D 对应的项目集邻接网络，最后再用不同的搜索过程从网络中搜索出频繁项目集.因此，整个实验也由这三个部分对应的过程组成.

图 4 中给出了数据库 D 由横向到纵向的转换时间与交易量的关系.转换过程如下：读取一个交易，对其中的每个项目将交易号加入到该项目所对应的交易列表中. 实验数据库限定在 $T = 5$、$I = 3$ 条件下，每次增加 1K 的交易量，由图 4 中可以看出，转换所需的时间基本上与交易量成正比关系.同时在实验中还发现，在数据库转换过程中所需要的时间同时也受数据库中所包含的交易项目数的一定影响.

图 5 中给出了项目集邻接网络生成时间与交易长度的关系.实验数据限定在 $D = 1K$、$I = 3$ 条件下，交易平均包含的项目个数每次增加 1，由图 5 中可以看出，生成项目集邻接网络所需要的时间基本上与交易平均长度成指数增长关系.在项目集邻接网络的生成过程中，对生成所需时间及网络规模影响最大的因素是平均交易长度，另外，在平均交易长度不大的情况下，如果个别交易包含的项目数非常大，那么无论是从生成时间还是从网络规模上也都会产生很大影响.另外生成时间及网络规模也受交易量、交易项目数等因素的影响.

图 6 中给出了项目集邻接网络中用 Breadth-First 搜索策略对频繁项目集进行搜索所需要的时间与输出量的关系.实验数据为 T5.I3.D1K，在它对应的项目集邻接网络中共存在 105 964 个项目集，由图 6 中可以看出，搜索频繁项目集所需的时间基本上与输出量成正比关系.原因在于，首先项目集邻接网络中每层中的结点都按支持度从大到小进行了排序；其次，在搜索过程中涉及的只是简单的对结点的访问操作，而不涉及诸如项目集支持度计算之类的复杂操作.因此搜索所需要的时间将主要受输出量的影响，受输入数据的影响不是很大.

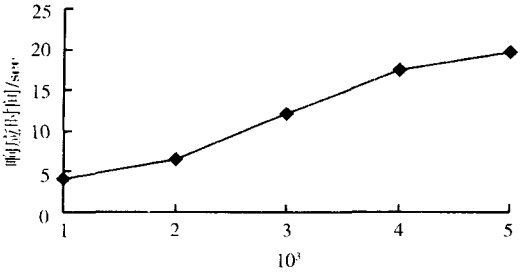


图 4 数据库格式转换时间与交易量的关系

Fig.4 Response time variation with number of transactions

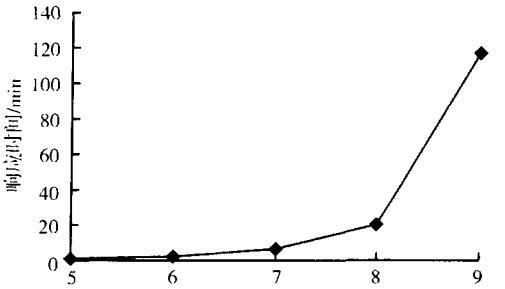


图 5 网络生成时间与交易长度的关系

Fig.5 Response time variation with average transaction size during lattice constructing

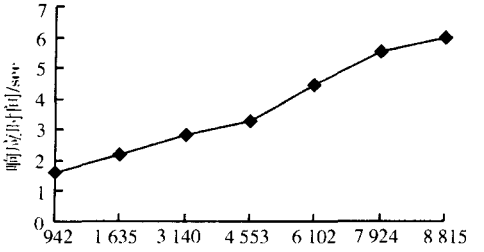


图 6 搜索时间与输出量的关系

Fig.6 Response time variation with number of frequent itemsets searched

5 结束语

本文提出了一些能够有效地发现出频繁项目集的新算法.同时还给出了能够将搜索空间划分成较小的、相互独立的子空间的等价类方法.从而使得每个子问题都可以主存中利用搜索过程来处理.并且对数据库进行预处理后,在整个处理过程中算法完全不需要对整个数据库进行读取,而只对取预处理后的数据中的一部分即可.

由于本算法有效运行的基础是纵向格式数据库,同时邻接网络的构造与存贮也需要更大的空间,因此可以说本算法在时间上的优势是由更大的存贮空间换来的,这可能会在一定范围内限定的该算法的应用,但是随着海量存贮设备的发展及高速处理器的出现,该算法的应用范围会不断扩大.

参 考 文 献:

[1]Agrawal R, Swani A. Mining Association Rules Between Sets of Items in Very Large Databases[C]. Proceedings of the ACM SIGMOD Conference on Management of Data, Washington, DC, ACM Press, 1993. 207—216.

[2]Agrawal R, Srikant R. Fast Algorithm for Mining Association Rules in Large Databases[C]. Proceedings of the 20th International Conference on Very Large Databases, 1994. 478—499.

[3]Aggarwal C, Yu P S. Online Generation of Association Rules[R]. IBM Research Report, RC-20899, 1997.

[4]Zaki M J. Scalable Data Mining for Rules[D]. New York: University of Rochester, http://www.cs.rpi.edu/~zaki/.1998.

[5]Han J, Pei J, Yin Y, Mao R. Mining Frequent Patterns without Candidate Generation[C]. Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00), Dallas, TX, May 2000. 1—12.

[6]Aggarwal C, Yu P S. A New Framework for Itemset Generation[R]. IBM Research Report, RC-21064.

[7]Han J, Kamber M. Data Mining: Concepts and Techniques[M]. San Mateo: Morgan Kaufmanns Publishers, 2000.

[8]Goulbourne G, *et al*. Algorithms for computing association rules using a partial-support tree[J]. Knowledge-based Systems, 2000, 13: 141—149.

[9]Guangling Lee, *et al*. Efficient graph-based algorithms for discovering and maintaining association rules in large database[J]. Knowledge and Information Systems, 2001, 3: 338—355.

[10]Loo K K, *et al*. A lattice-based approach for I/O efficient association rule mining[J]. Information Systems, 2002, 27: 41—74.

[11]Lin Weiyang. Efficient adaptive-support association rule mining for recommender systems[J]. Data Mining and Knowledge Discovery, 2002, 6: 83—105.

[12]Masahiro Terabe. Attribute generation based on association rules[J]. Knowledge and Information Systems, 2002, 4(3): 329—349.

[13]Lin Yi. Support vector machines and the bayes rule in classsification[J]. Data Mining and Knowledge Discovery, 2002, 6: 259—275.

[14]Brock B, Howard Hamilton J. Extracting share frequent itemsets with infrequent subsets[J]. Data Mining and Knowledge Discovery, 2003, 7(2): 153—185.

Algorithm based on adjacent lattice for finding frequent itemsets

CHEN Fu-zan, LI Min-qiang

School of Management, Tianjin University, Tianjin 300072, China

Abstract: It is well known that the task of finding frequent itemsets in large database is the bottleneck problem in the research of association rules mining. A new algorithm for mining frequent itemsets is proposed in this paper. Based on the graph theory, the algorithm converts the origin transaction database to an itemsets adjacent lattice in the preprocessing, where each itemset vertex has a label to save its support. The algorithm changes the complicated task of mining frequent itessets in the database to a simpler one of searching vertex in the lattice, which can speed up greatly the mining process. Furthermore, to compute the support of each itemset, the algorithm uses a vertical tid-list database format, where each itemset is associated with a list of transactions in which it occurs. At the end, we carried out the algorithm, and analyzed the result of the experiment.

Key words: data mining; association rules; adjacent-lattice; frequent itemset