计算机科学2004Vol.31№.3

频繁项集挖掘算法

颜跃进　李舟军　陈火旺
（国防科技大学计算机学院　长沙410073）

摘　要　数据挖掘在最近几年里已被数据库界所广泛研究，而搜索频繁项集是诸如关联规则挖掘、序列模式挖掘等数据挖掘问题中的关键步骤。本文描述了频繁项集挖掘问题的特点，并根据搜索策略对已有各种频繁项集挖掘算法进行了分析和比较。

关键词　频繁项集，Apriori性质，支持度，宽度优先，深度优先

Frequent Item Sets Mining Algorithms

YAN Yue-Jin　LI Zhou-Jun　CHEN Huo-Wang
(School of Computer Science, National University of Defense Technology, Changsha410073)

Abstract　Nowadays dat mining is becoming one of most popular problems in the field of database research. Mining frequent item sets in a key step in many data mining problems, such as association rule mining, sequential pattern mining, and so on. In this paper, we introduce the characteristic of frequent item sets mining problems, and then we analyze and compare today's common approaches according to search strategy.

Keywords　Frequent item sets, Apriori property, Support, Breath-first search, Depth-first search

1　引言

随着数据库技术的迅速发展以及数据库管理系统的广泛应用，人们积累的数据越来越多。激增的数据背后隐藏着许多重要的信息，人们希望能够对其进行更高层次的分析，以便更好地利用这些数据。目前的数据库系统可以高效地实现数据的录入、存储、查询、统计等功能，但无法发现数据中存在的关系和规则，无法根据现有的数据预测未来的发展趋势。缺乏挖掘数据背后隐藏的知识的手段，导致了"数据爆炸但知识贫乏"的现象。

计算机技术的另一领域——人工智能自1956年诞生之后取得了重大进展。经历了博弈研究、自然语言理解、知识工程等阶段，目前的研究热点是机器学习。机器学习是用计算机模拟人类学习的一门科学，比较成熟的算法有神经网络、遗传算法等。

用数据库管理系统来存储数据，用机器学习的方法来分析数据，挖掘大量数据背后的知识，这两者的结合促成了数据挖掘(Data Mining)的产生。实际上，数据库中的知识发现是一门交叉性学科，涉及到机器学习、模式识别、统计学、智能数据库、知识获取、数据可视化、高性能计算、专家系统等多个领域。从数据库中发现出来的知识可以用在信息管理、过程控制、科学研究、决策支持等许多方面[1]。

在实际应用中，往往根据模式的实际作用将数据挖掘需要挖掘的模式细分为以下六种：分类模式、回归模式、时间序列模式、聚类模式、关联模式和序列模式。

关联模式反映一个事件和其他事件之间依赖或关联的知识。如果两项或多项属性之间存在关联，那么其中一项的属性值就可以依据其他属性值进行预测。最为著名的关联规则挖掘方法是R. Agrawal提出的Apriori算法。关联规则的挖掘可分为两步。第一步是迭代识别所有的频繁项目集，要求频繁项目集的支持率不低于用户设定的最低值；第二步是从频繁项目集中构造可信度不低于用户设定的最低值的规则。识别或挖掘所有频繁项目集是关联规则挖掘算法的核心，也是计算量最大的部分。

序列模式与关联模式相仿，而把数据之间的关联性与时间联系起来。为了挖掘序列模式，不仅需要知道事件是否发生，而且需要确定事件发生的时间。我们一般分如下五个步骤来找出所有的序列模式：排序、频繁项集搜索、转换、序列和选最大序列。其中频繁项集搜索是非常关键的一步。

本文主要介绍频繁项集挖掘算法的研究特点和现状。第2部分是对频繁项集挖掘问题的描述，包括问题的定义、性质和搜索策略；第3部分根据频繁集挖掘的搜索策略—深度优先和宽度优先来对当前的频繁项集挖掘算法进行分类介绍，并分析了各个算法自己特有的优化和存在的不足；最后对频繁项集挖掘算法的未来研究方向进行了展望。

2　问题描述

2.1　问题定义

设$I=\{x_1,\cdots,x_n\}$为一组不同的字母集合，其中的元素称为项(item)或者商品，I也是所有商品的集合。如果对一个集合X有：$X\subseteq I$且$k=|X|$，则X称为k项集，或者简单地称为一个项集。记D为事务(transaction)T的集合，这里事务T是项的集合，并且$T\subseteq I$。对应每一个事务有唯一的标识，如事务号，记作TID。设X是I中的一个项集，如果$X\subseteq T$，则称事务T包含X。如果X的支持度(X的出现频率)大于某个给定的最小阈值，我们称X是频繁的。频繁项集挖掘就是要在事务数据库里找出所有频繁项集和它们的实际支持度。

频繁项集挖掘的主要挑战在于理论上需要考虑的项集数

颜跃进　博士生，研究方向：数据挖掘与数据仓库；**李舟军**　教授，博士生导师，研究方向：数据仓库与数据挖掘，计算机科学理论；**陈火旺**　工程院院士，研究方向：软件理论与软件工程。

目巨大。实际上，频繁项集的数量随着|I|成指数增长。我们不愿意、在实际中也不可能挖掘如此之多的频繁项集，这些频繁项集通常用一个量化的最小阈值来进行限制，这个量化的阈值称为支持度，可记为：minSupp。在 D 中，X 的支持度定义为支持 X 的事务 T 的个数与数据库总事务个数的比 $Supp(X)=|\{T\in D|X\subseteq T\}|/|D|$。$X$ 是频繁的是指 X 的支持度大于 minSupp。在寻找频繁项集过程中，会产生一些可能是频繁项集的中间项集，我们称之为候选项集或者候选。这些候选在计算支持度后，如果满足最小支持度，即支持度大于 minSupp，则成为频繁项集；否则，不是频繁项集，需要在搜索过程中舍弃。

2.2 频繁项集自身的约束性质

在频繁项集的搜索过程中，文[2]提出的一种叫 Apriori 的重要性质用于压缩搜索空间。

Apriori 性质：一个频繁项集的所有非空子项集也是频繁。Apriori 性质基于以下观察：根据定义，若项集 X 不频繁，即不满足最小支持度，则添加某个项 A 到 X 中，新项集($X\cup A$)不可能比 X 更频繁出现，也就是说，$X\cup A$ 也不频繁。

文[5]中采用一种分段计算项集支持度的思想，进一步利用了频繁项集的潜在约束性质，大大减小了有可能成为频繁项集的项集数量，从而进一步缩小了搜索空间。分段约束性质详见文[5]。

在频繁项集的搜索过程中，除了利用频繁项集自身的性质来压缩搜索空间，减少候选产生的策略外，遍历搜索空间的方法，如何计算候选支持度以及采用的事务和候选的组织数据结构都是非常重要的[5]。

2.3 遍历搜索空间

由于我们需要找到满足最低支持度的所有项集，在实际的应用中，如果对项集 I 的所有子项集进行遍历搜索，将因为搜索空间爆炸而导致失败。事实上我们还要考虑，项数的线性增长也会导致项集的指数增长。我们将整个搜索空间组织成图1的树，频繁项集在图中树的上部，而非频繁项集在下部。虽然我们没有显式地将每个项的支持值在图中表示出来，我们不妨假设黑粗线将频繁项集和非频繁项集分割开来。根据频繁项集支持度的 Apriori 性质，对任何事务数据库和任意给定的最小阈值，这条黑线总是存在的。

现有算法的基本原理是使用这条粗线来有效地裁减搜索空间。一旦发现这条线，我们就可以把注意力集中在计算这条线上部的项集的支持度上，而忽略下部的项集。

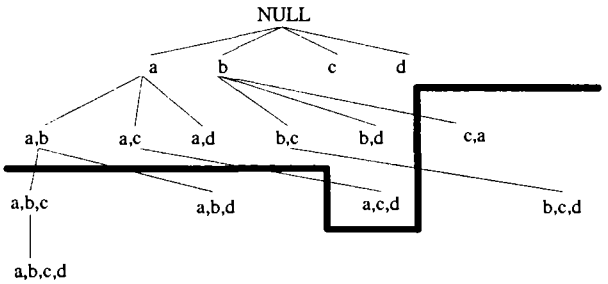

图1 $I=\{a,b,c,d\}$时的树

这个规律允许我们有效地限制需要考察的项集的数量。在搜索过程中，我们只需要多计算那些分割频繁项集和非频繁项集分界线上的项集的支持度的值。当然，实际的搜索策略在于我们自己的选择。当今比较流行的搜索策略有宽度优先(BFS)和深度优先(DFS)两种[6]。在宽度优先中，任何 k 项集支持度的计算在所有 k－1项集之后进行。相反，深度优先沿着树的子孙往下搜索。

3 频繁项集挖掘算法研究现状

正如前面所说，我们可以在逻辑上将频繁项集组织成一棵树，搜索频繁项集的过程可以认为是遍历频繁项集树的过程。这样的话，按频繁项集树的搜索次序，所有的频繁项集生成算法可以分为宽度优先和深度优先两种。我们将现有主要频繁项集挖掘算法依照遍历搜索空间的策略进行分类，并分析和比较了各个算法自己特有的优化和存在的不足。最后，对其他类型的频繁项集挖掘算法进行了简要介绍。

3.1 宽度优先

这类算法最流行的是 Apriori 算法[2,6]，频繁项集支持度的 Apriori 性质也是在这个算法里开始被介绍的。Apriori 算法中使用这个性质来进行剪枝，即将那些具有非频繁子项集的候选项集在计算支持度之前裁减掉。由于宽度优先允许一个候选的所有子项集的支持度事先计算，所有这种方法是切实可行的。Apriori 算法在一遍对数据库的扫描里面计算所有 k 项集的支持度。文[6]中针对这个问题提出一个 hash 树结构。在这个结构里，每个候选项集的一项对应 hash 树里的一个节点，具有相同前缀的候选可以共享节点。对每个事务，我们在该树中使用 hash 函数进行遍历，当我们达到 hash 树的叶节点时，就可以找到一组包含在该事务中的具有相同前缀的候选。

AprioriTID 算法[6]是基本 Apriori 算法的扩展。与使用原始数据库中的数据不同，AprioriTID 算法内部用每个事务当前所包含的候选来表示该事务。AorioriHybrid[6]结合了两种算法。SETM[7]是类似 Apriori(或者 AprioriTID)算法的进一步扩展，它的目标是直接用 SQL 语句实现。

DIC[8]算法是 Apriori 算法的更新的一个变种。它软化了计算和产生候选的界限。只要候选满足最小的阈值。即使还没有全部计算这个候选项集在每个事务中是否出现，DIC 中就开始由它产生下一步的候选。正因为如此，它采用了一个前缀树。与 hash 树不同的是，前缀树中每个节点，包括叶节点和内部节点，都代表某个候选频繁项集各个频繁子项集。

基于 Apriori 算法框架的所有算法不可避免地有两个致命的弱点：一是需要产生大量候选项集，而真正有价值的不多；二是需要多次扫描事务数据库，效率低下。

Partition 算法[9]是一个象 Apriori 算法一样的宽度优先算法，但是它用集合交来决定候选项集的支持度。一个 *tid* 是唯一的事务标识。对一个项来说，*tidlist* 是与该项有关的一组事务标识的集合。当然，对项集也存在 *tidlist*，一个项集 X 的 *tidlist* 用 $X.tidlist$ 表示，如果候选 $C=X\cap Y$，那么 $C.tidlist=X.tidlist\cap Y.tidlist$。这里为了有效地交运算，*tidlist* 集合用字母升序排列。在前面所说的 Apriori 算法中，所有 k－1项候选在 k 项候选前面产生。Partition 算法中想用 k－1项集的 *tidlist* 产生 k 项集的 *tidlist*。很显然，这些中间结果的大小很容易就超过了一般机器的内存限制。为此，Partition 将事务数据库划分成一个个独立处理的块，块大小的选择依赖于这些中间 *tidlist* 结果正好能载入内存。在每个事务数据库块中决定了频繁项集以后，Partition 算法需要一个附加的扫描来决定该频繁项集是不是全局频繁的。不过，当子数据库数目增大时，Partition 算法生成的无效总候选频繁项集数目快速增长，导致效率降低，因此 Partition 算法在较大数据库上的性能还不如 Apriori 算法。

TreeProjection[10]是最新的宽度优先算法。TreeProjection中使用字母树中的节点来表示频繁项集，并使用了事

务投影技术和矩阵计算支持度的技术来减少事务集的个数。该算法只考察了事务子集，通过自上而下的搜索策略，可使频繁模式都保存在这些事务子集里面。这种方法极大地提高了计算包含频繁项集的事务数量的计算性能。TreeProjection 主要基于纯宽度优先策略。它具有与 Apriori 算法同样的缺点，即：由投影导致的模式匹配的高代价，巨大的频繁项集树和太多次数的数据库扫描。

3.2 深度优先

文[16]中介绍的 Eclat 算法使用了深度优先搜索策略，它也是利用 tidlist 集合相交来计算候选项集的支持度。在深度优先搜索过程中，该算法在从根节点到当前考察的候选项集始终保持一份 *tidlist*。这样，就不再需要 Partition 算法中使用的划分方法。

值得一提的是，Eclat 中使用了一种叫做"快速交"的优化方法。当我们将两个 *tidList* 集合相交的时候，我们仅仅对那些支持度大小可能满足最小支持度的结果 *tidList* 集合进行相交。换句话说，一旦我们发现某个交运算将不可能满足最小支持度值，我们就要舍弃这次不必要的交运算。

最近出现了 FP-growth 算法[12]，FP-growth 算法提出一种叫做 FP 树的高度压缩的事务数据表示结构。在开始的一个预处理过程中，FP 算法扫描一次数据库生成 FP 树。产生 FP 树就是通过计算出现次数和深度优先搜索完成的。在第二步，FP-growth 算法使用前面生成的 FP 树来计算所有频繁项集的支持度的值。然而，条件 FP-Tree 树的数目与频繁项集数目是一个数量级的。算法对于稀疏数据库和极大数据库的扩展性也不太好。

文[13]中提出一种 H-Struct 结构，将稀疏数据库保存在内存中，使用基于 H-Struct 结构的算法 H-Mine，H-Mine 算法调用了 FP-growth 算法来挖掘密集数据库。在事务数据库非常庞大的时候，H-Mine 算法采用了基于划分的方法。由于各划分的局部频繁模式数量还是有可能非常大，因此 H-mine 算法在数据库非常大的时候仍然会有问题。

DepthProject[14]和 MAFIA 算法[15]是两个采用深度优先策略来挖掘最大频繁项集的新算法。DepthProject 采用一种可选择投影和水平位串来表示投影事务子集。而 MAFIA 使用垂直位图结合位图压缩机制来表示投影事务子集。然而，在事务中的项集均长比项总数小很多的情况下，它们的效率比基于数组的表示方法还要低效，而这种情况在大而稀疏的数据库是很常见的。另一方面，在挖掘密集型数据库时，如果一个节点表示了相对大数量的项集时，基于树的表示方法的压缩比率显得尤为重要。因此对于密集数据库，基于树的表示方法的效率应该优于，至少是等于位串和位图的表示方法。此外，伪投影方法要比有 DepthProject 的选择投影和 MAFIA 中的压缩策略有效得多。

3.3 其它工作[16]

如果一个频繁项集不是其它任何频繁项集的真子项集，那么称此频繁项集为最大频繁项集。除了前面提到的挖掘最大频繁项集的算法外，文[17,18]中都对基于最大频繁项集的频繁项集挖掘算法进行了研究。

频繁封闭项集是一组事务都包含的项的最大项集。一个数据库的频繁封闭项集集合与其频繁项集集合的信息量相等，但是前者比后者小得多，因此挖掘频繁封闭项集能够减少数据库扫描遍数和 CPU 开销，从而提升频繁项集挖掘的效率，并大幅减少输出冗余信息[19]。

用户往往只对满足一定约束条件的频繁项集感兴趣，因此如果频繁项集挖掘算法只挖掘满足用户指定约束条件的频繁项集，那么将大量节省用户处理不感兴趣的频繁项集的时间，同时也可提高算法执行速度[20]。

当项或者商品具有概念层次关系时，如果用户对包含有不同概念层次项的规则感兴趣，称这种规则为广义关联规则或多层关联规则挖掘。广义关联规则挖掘算法提高性能的关键也在于广义频繁项集挖掘的效率[21]。

Apriori 算法适用的前提之一是对于所有频繁项集使用相同的最小支持度。如果在数据库中存在出现频率比较低的项(称为稀疏项)，同时用户对这些稀疏项又比较感兴趣，那么就需要使用可变的最小支持度来挖掘包含稀疏项的频繁项集，同时不发现太多的用户不感兴趣的频繁项集[22,23]。

结束语 宽度优先算法的缺点就是对含有长模式的密集数据库效率比较低下。深度优先算法的缺点就是不能完全挖掘深度优先的潜力，不能很好地扩展到大型稀疏数据库的挖掘中去。两者的简单结合也肯定不能适应各种事务数据库特点，产生万能的算法。但是，将两者有机地结合起来应该能够适合更多情况下的事务数据库。

事务子集是事务数据库部分数据在内存中的投影。它由最初从整个原始事务数据库扫描发展表示成数组、树、垂直位图以及水平位字串，这些表示方法是有效投影和有效计算支持度的关键。任何一种表示形式都不可能适合所有的情况。事实上，最大的效率和扩展性是在不同的投影和支持度计算环境里来平衡各种不同表示形式。研究事务子集在各个环境下的表示形式将对频繁项集算法发展有着积极意义。

遍历搜索空间过程中，频繁项集由于自身的特点有一定的约束性质，例如：Apriori 性质和频繁项集的分段约束性质。有效地利用这些性质能大大减少候选项集数量，从而减少计算支持度的工作量，因此也大大压缩了搜索空间。但是，前面提到的两种约束性质都只是项集约束的充分条件，进一步深入研究频繁项集的约束性质能使候选项集更加逼近频繁项集，从而达到提高频繁项集产生效率的目的。

参考文献

1 http://www.dmgroup.org.cn/zs20.htm

2 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: Proc. 1993ACM-SIGMOD Int. Conf. Management of Data, Washington, D.C., May 1993. 207～216

3 Srikant R, Agrawal R. Mining sequential patterns: generalizations and performance improvements. In: 5th Intl. Conf. Extending Database technology, Mar. 1996

4 Srikant R, Agrawal R. Mining sequential patterns: generalizations and performance improvements. In: Proc. 5th EDBT, 1996. 3～17

5 Brin S, Motwani R, Silverstein C. Beyond market basket: Generalizing association rules to correlations. In: Proc. 1997ACM-SIGMOD Int. Conf. Management of Data, Tucson, Arizona, May 1997. 265～276

6 Silverstein C, Brin S, Motwani R, Ullman J. Scalable techniques for mining causal structures. In: Proc. 1997 Int. Conf. Very Large Data Bases, New York, NY, Aug. 1998. 594～605

7 Agrawal R, Srikant R. Mining sequential patterns. In: Proc. 1995 Int. Conf. Data Engineering, Taipei, Taiwan, March 1995. 3～14

8 李雄飞，苑森淼，董立岩，全勃. 多段支持度数据挖掘算法研究. 计算机学报，2001，24(6)：661～665

9 Agrawal R, Mannila H, Srikant R, Toivonen H, Verkamo A. Fast Discovery of Assocation Rules. 1996. 307～328

10 Houtsma M, Swami A. Set-oriented mining for Association Rules in Relational Databases. In: Proc. of the 11th Intl. Conf. on Data Engineering. March 1995. 24～32

11 Brin S, Motwani R, ullman J, Tsur S. dynamic Itemset counting and implication rules for market basket data. SIGMOD97, 1997. 255～264

(下转第124页)

测量、生产能力、市场销售预测量)继续寻找相应的模型和数据,直到完成任务的分解。

3)决策资源代理调用网格的资源管理服务进行决策资源的查询,将各子任务分配到合适的软硬件资源上,规划并控制各子任务的执行,保证子任务的顺利完成。

4)集成子任务的执行结果,将最终的产品年生产计划建议,通过决策者终端应用返回给决策者,为决策者的决策提供支持。

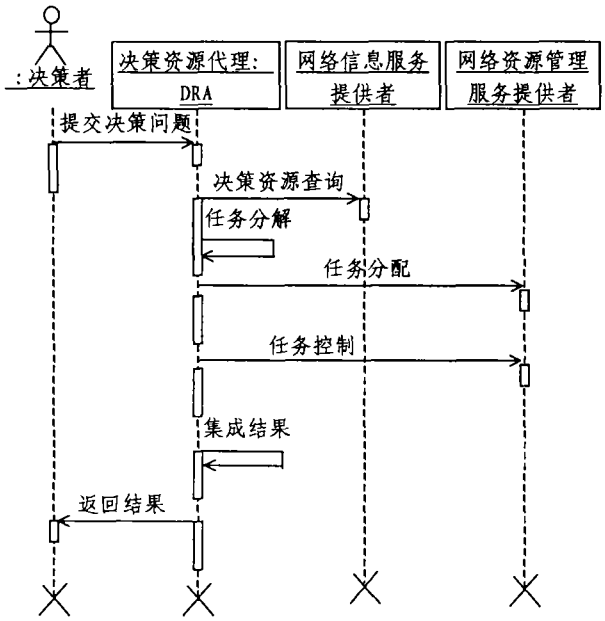

图3 基于网格的开放式决策支持系统运行顺序图

从上面的流程可以看出,基于网格的开放式决策支持系统能够充分利用网格环境提供的信息服务和资源管理功能,动态有效地组织和集成现有的信息资源,为决策者提供更好的决策支持服务。

总结与展望 本文在对目前DSS研究中存在的问题和网格技术进行分析的基础上,给出了一个基于网格的开放式决策支持系统模型。本文认为网格平台能够从根本上克服以往DSS开发平台的许多缺点,能够解决目前DSS研究中存在的一些问题,因而基于网格的决策支持系统具有许多优秀的特性。决策支持系统具有信息资源需求量大、需要大量协同工作的特点,非常适合应用网格技术。网格技术和决策支持系统的结合能充分发挥网格技术的优势,提高决策支持的能力,推动两者的发展。

目前基于网格的决策支持系统的研究还不多见,但随着网格技术的逐渐成熟和网格应用的逐渐增多,基于网格的决策支持系统研究将成为今后决策支持系统研究的一个重要方向。网格技术处于不断的发展和变化之中,未来网格的商业运作模式对基于网格的DSS的功能、结构和实现方式会产生很大的影响。

参 考 文 献

1 Morton M S S. Management decision systems. Computer bassed support fordecision making, Division of Research, Harvard University, Cambridge, Massachusetts, 1971
2 徐志伟. 网格的称谓. 计算机世界报, vol. 43, 2002-10-31
3 Dong C S J, Loo G S. Flexible Web-Based Decision Support System Generator (FWDSSG) Utilising Software Agents, DEXA Workshop 2001. 892～897
4 Bhargava H, Krishnan R, Mueller R, Decision Support on Demand: On Emerging Electronic Markets for Decision Technologies. Decision Support System, 1997, 19(3): 193～214
5 赵新昱,陈文伟,陈卫东,等. DSS中广义模型服务器规范化研究与实现. 小型微型计算机系统,2000,21(6):595～599
6 Chervenak A, Kesselman C. Class material of "Introduction to Grid Computing". USC Information Sciences Institute, 2000
7 Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, San Francisco, 1999
8 网格计算网罗一切. 中国计算机报, vol. 26, 2002-04-15
9 Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal Supercomputer Applications, 2001, 15(3)

(上接第114页)

12 Savasere A, Omiecinski E, Navathe S. An efficient algorithm for mining association rules in large databases. In: Proc. 1995 Int. Conf. Very Large Data Bases, Zurich, Switzerland, Sept. 1995. 432～443
13 Agarwal R, Aggarwal R, Prasad V V V. A tree projection algorithm for generation of frequent itemsets. J. Parallel and Distributed Computing, 2000
14 Liu J Q, Pan Y H, Wang K, Han J W. Mining Frequent Item Sets by Opportunistic Projection, KDD'02, Edmonton, Canada, July 2002
15 Han J, Pei J, Yin Y. Mining Frequent Patterns without Candidate Generation. In: Proc. 2000ACM-SIGMOD Int. Conf. on Management of Data(SIGMOD'00), Dallas, TX, May 2000
16 Pei J, Han J, Lu H, et al. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases, In: Proc. 2001 Int. Conf. on Data Mining(ICDM'01), San Jose, CA, Nov. 2001
17 Agarwal R, Aggarwal C, Prasad V V V. Depth first generation of long patterns. In: Proc. of SIGKDD Conf. 2000
18 Burdick D, Calimlim M, Gehrke J. MAFIA: A maximal frequent itemset algorithm for transactional databases. In: proc. of the 17th Intl. Conf. on Data Engineering, Heidelberg, Germany, April 2001
19 http://bbs.nuj.edu.cn/dataming/2350.htm
20 Zaki M J, Parthasarathy S, Ogihara M, Li W. New Algorithms for Fast Discovery of Association Rules. In: Proc. of the 3rd Int'l Conf. on Knowledge discovery in Databases (KDD'97). Aug. 1997. 283～286
21 Lin D, Kedem Z M. Pincer-Search: A New Approach for Discovering the Maximum Frequent Set. In: Proc. of Sixth European Conf. on Extending Database Technology
22 Pei J, Han J, Mao R. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. In: Proc. 2000 ACM-SIGMOD Int. Workshop on Data Mining and Knowledge Discovery (DMKD'00), Dallas, TX, May 2000
23 Klemettinen M, Mannila H, Ronkainen P, Toivonen H, Verkamo A I. Finding interesting rules from large sets of discovered association rules. In: Proc. 3rd Int. Conf. Information and Knowledge Management, Gaithersburg, Marylad, Nov. 1994. 401～408
24 Han J, Fu Y. Discovery of Multiple-Level Association Rules from Large Databases. In: Proc. of 1995 Int'l Conf. on Very Large Data Bases(VLDB'95), Zurich, Switzerland. Sept. 1995. 420～431
25 Liu B, Hsu W, Ma Y. Mining association rules with multiple minimum supports. KDD99
26 Wang K, He Y, Han J. Mining Frequent Itemsets Using Support Constraints. In: Proc. 2000 Int. Conf. on on Very Large Data Bases (VLDB'00), Cairo, Egypt, Sept. 2000
27 Hipp J, Guntzer U, Nakaeizadeh G. Algorithms for Association Rule Mining—A General Survey and Comparison. In: Proc. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, 2000