

# 基于 FP-tree 的支持度计数优化策略

赵 阳,白 凡

(江南计算技术研究所,江苏 无锡 214083)

**摘 要:**关联规则挖掘过程中,频繁项集的挖掘是最关键的步骤。最大频繁项集是最常用的频繁项集简化表示。基于 FP-tree 的最大频繁项集挖掘算法多数都需要自底向上地搜索 FP-tree 来计算项集的支持度。而已有的支持度计算方法在计算当前项集的支持度时没有考虑已完成的支持度计算过程所获得的信息,因而造成了不必要的开销。针对该问题,提出了基于 FP-tree 的支持度计数优化策略(Support Count Optimization Method on FP-tree,SCOM),在付出很小的额外空间代价的条件下,充分利用已完成的支持度计数过程中获取的路径对项集的支持信息和项集之间的关系进行搜索剪枝,并设计实验将该策略应用到 DMFIA 算法上。实验结果表明,应用该策略的最大频繁项集挖掘算法 DMFIA 获得了较大的性能提升。SCOM 对基于 FP-tree 的支持度计数进行优化,因此能够应用到所有利用 FP-tree 进行支持度计数的算法之中。

**关键词:**关联规则挖掘;FP-tree;最大频繁项集;支持度计数;搜索剪枝

**中图分类号:**TP311

**文献标识码:**A

**文章编号:**1673-629X(2017)10-0030-04

doi:10.3969/j.issn.1673-629X.2017.10.007

## Support Count Optimization Method Based on FP-tree

ZHAO Yang,BAI Fan

(Jiangnan Institute of Computer Technology,Wuxi 214083,China)

**Abstract:**In the association rules mining,mining frequent itemsets is the most critical step. Maximum frequent itemsets is the most common simplified representation of frequent itemsets. Maximum frequent itemsets mining algorithms based on FP-tree are most needed to search the FP-tree bottom-up to count the support of the itemsets,but they have not considered the information obtained by completed support counting while counting the current itemset,resulting in unnecessary overhead. To solve it,Support Count Optimization Method on FP-tree,called SCOM for short,is proposed. With a small additional space cost,it can make full use of the information that whether a path supports a itemset and the relation between the itemsets to prune the search. Experimental results show that the maximum frequent itemsets mining algorithm applied obtains a performance boost with SCOM which optimizes the support count based on FP-tree,so it can be applied to all algorithms that use FP-tree to count support.

**Key words:**association rules mining;FP-tree;maximum frequent itemsets;support count;search prune

## 0 引 言

关联规则的概念是由 Agrawal 等提出的<sup>[1-2]</sup>,反映了大量数据中项目集之间有趣的关联或相关关系。频繁项集挖掘是关联规则挖掘中最关键的一步。实践中,由事务数据集产生的频繁项集的数量可能非常大,因此,从中识别出可以推导出其他所有频繁项集的、较小的、具有代表性的项集是有用的。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<sup>[3]</sup>。

频繁模式树(Frequent Patten tree,FP-tree)是由 Han 等提出的对于事务数据集的一种压缩存储方式<sup>[4]</sup>。基于 FP-tree 的算法可以避免 Apriori 系列算法需要多次读取事务数据集而造成的额外开销。目前,基于 FP-tree 的最大频繁项集挖掘算法主要有 FP-Max<sup>[5]</sup>,DMFIA<sup>[3]</sup>,IDMFIA<sup>[6]</sup>,FPMFI<sup>[7]</sup>,BDRFIA<sup>[8]</sup>,等等。FP-Max 与 FP-growth 类似,通过递归构建 FP-tree 的方式挖掘频繁项集,用超级检验的方法保证最

收稿日期:2016-11-14

修回日期:2017-03-16

网络出版时间:2017-07-11

基金项目:国家科技重点专项“核高基”(2015ZX01040-201)

作者简介:赵 阳(1991-),男,硕士研究生,研究方向为数据挖掘、文本分析及可视化;导师:刘镇江,高级工程师,研究方向为数据挖掘、信息处理、数据可视化。

网络出版地址:cnki.net/kcms/detail/61.1450.TP.20170711.1457.090.html

终结果是最大频繁项集。DMFIA 是一种自顶向下的最大频繁项集挖掘算法,与 FP-Max 相比,其无需递归生成 FP-tree,对于最大频繁项集维度较大的数据效果较好;IDMFIA 是在 DMFIA 基础上提出的双向搜索算法,充分利用了低维项集信息进行剪枝,扩大了算法的适用数据范围;FPMFIA 使用基于投影的方法减少了超集检测的时间;BDRFI 则改进 FP-tree 为 DFP-tree (Digital Frequent Patten tree, 数字频繁模式树),运用多种预测剪枝策略,快速降低候选项集维度,减少了超级检测的时间。调研中还发现各文献对于“自顶向下”和“自底向上”的项集空间搜索的定义较为混乱<sup>[9-11]</sup>,因此文中的“自顶向下”的项集空间搜索特指从高维项集到低维项集的搜索。

上述基于 FP-tree 的最大频繁项集挖掘算法中多数都需要通过遍历 FP-tree 或其子树来计算项集的支持数,以避免多次扫描数据库或递归生成条件 FP-tree。但是在项集支持度计数的过程中,没有充分利用在计算之前的项集支持度时所获得的信息,因而造成了不必要的检索开销。针对该问题,提出了基于 FP-tree 的支持度计数优化策略 (Support Count Optimization Method on FP-tree, SCOM)。根据集合的性质,在搜索 FP-tree 计算项集的支持度时,记录路径对项集的支持信息,以达到减少搜索路径数的目的;并以 DMFIA 算法为例,应用该策略对算法进行改进,并设计实验进行验证。

## 1 相关知识

### 1.1 相关定义和性质

设  $I = \{i_1, i_2, \dots, i_m\}$  是由  $m$  个项组成的集合,  $D$  是由一组事务组成的事务数据集且  $D$  中事务都由  $I$  中的项组成。给定项集  $X \subseteq I$ ,  $X$  在  $D$  中的支持数是指  $D$  中包含  $X$  的事务数,  $X$  在  $D$  中的支持度是指  $X$  的支持数占  $D$  中所有事务数的百分比。

定义 1: 对于项集  $X \subseteq I$ , 若  $X$  在  $D$  中的支持度  $s_x$  大于等于给定的最小支持度  $s$ , 则称项集  $X$  为事务数据集  $D$  在最小支持度  $s$  下的频繁项集;反之,则称项集  $X$  为事务数据集  $D$  在最小支持度  $s$  下的非频繁项集。

定义 2: 对于项集  $X \subseteq I$ , 给定最小支持度  $s$ , 若  $X$  的所有超集都是非频繁项集,则称项集  $X$  为事务数据集  $D$  在最小支持度  $s$  下的最大频繁项集。

设  $R = \{n_1, n_2, \dots, n_m\}$  为由事务数据集  $D$  生成的 FP-tree 中的一条路径,所有项集和路径都是按照支持度降序排列的。

性质 1: 对于项集  $X$ , 若  $X \subseteq R$ , 则  $\forall Y \subseteq X, Y \subseteq R$  且  $\forall R' \supseteq R, X \subseteq R'$ ; 若  $X \not\subseteq R$ , 则  $\forall Y \supseteq X, Y \not\subseteq R$  且  $\forall R' \subseteq R, X \not\subseteq R'$ 。

证明: 根据集合的性质容易得证。

对于项集  $X, Y \subseteq X$  且  $Y$  的最后一项为  $e_i$ ,  $R'$  为  $R$  的以  $e_i$  结尾的前缀子路径。

性质 2: 若  $X \subseteq R$ , 则必有  $Y \subseteq R'$ ; 若  $Y \not\subseteq R'$ , 则必有  $X \not\subseteq R$ 。

证明: 若  $X \subseteq R$ , 由于所有项集和路径都是按照支持度降序排列的, 则对于  $X$  的以  $e_i$  为结尾的前缀子集  $X'$ , 必有  $X' \subseteq R'$ , 而  $Y \subseteq X'$ , 故  $Y \subseteq R'$ , 同理可证若  $Y \not\subseteq R'$ , 则必有  $X \not\subseteq R$ , 证毕。

### 1.2 FP-tree 与 DMFIA

FP-tree 是一种输入数据的压缩表示法,它通过逐个读入事务,并把每个事物映射到 FP-tree 中的一条路径来构造。由于不同的事物可能会有若干个相同的项,因此它们的路径可能部分重叠。路径相互重叠越多,使用 FP 树结构获得的压缩效果越好。如果 FP-tree 足够小,能够存放在内存之中,就可以直接从这个内存中的结构提取频繁项集,而不必重复扫描存放在硬盘上的数据。

在 FP-tree 中,每个节点由 4 个域组成<sup>[12]</sup>: 节点名称 item-name、节点计数 item-count、节点链 item-link (用于指向树中具有相同 item-name 的下一个节点), 及父节点指针 item-parent。同时为方便树的遍历,FP-tree 还包含一个频繁项头表 Htable, 它由两个域组成: 项目名称 item-name 和指向 FP-tree 中具有相同 item-name 的首节点指针 head of node-link。

为了获得较好的压缩效果,通常需要将事务中的项按照支持度降序排列,尽管这样得到的 FP-tree 并不一定是最小的<sup>[13]</sup>。通过两次扫描数据集来构造 FP-tree:

(1) 第一次扫描数据集,确定每个项的支持度计数,丢弃非频繁项;按照支持度降序构建频繁项头表 Headers,创建 FP-tree 的根节点,标记为“null”;

(2) 第二次扫描数据集,对于每个事务 Trans, 根据项的支持度降序排列。令当前节点为根节点,顺序遍历 Trans 中的项,若当前节点包含与该项同名的子节点,则该子节点计数加 1,当前节点变为该子节点;否则,创建当前节点的新的子节点, item-name = 当前项的名称, item-count = 1, item-parent 指向当前节点, Headers 中同名链表的尾节点的 item-link 指向该子节点。直到遍历完数据集中所有的事务, FP-tree 构建完成。

文献[9]提出了 FP-tree 的一种改进—数字频繁模式树 (DFP-tree), 用频繁项降序排列后的序号代替名称,有利于提高超集检测的效率。

DMFIA 采用 FP-tree 的存储结构和自顶向下的搜索策略。它采用双重循环的方式挖掘最大频繁项集,

外层循环是在 MFCS 非空状态下进行,内层循环以自底向上的方式进行处理,若 MFCS 中项集的支持度大于等于最小支持度阈值,则将该项集加入到最大频繁项集(Maximum Frequent Sets,MFS)中;对非频繁项集,通过循环每次删除该项集中的一个项来产生新的候选项集,对于新的候选项集需要判断在 MFS 和 MFCS 中是否存在超集,若不存在则将其加入 MFCS 中,否则将其删除。

2 SCOM

2.1 主要思想

SCOM 策略的主要思想是:在搜索 FP-tree 计算候选项集的支持度时,记录各个相关路径的支持状态;当通过该项集生成新的候选项集时,根据性质 1、2 将旧的项集的路径支持信息传递给新的候选项集,这样在计算新的候选项集的支持度时就可以剪除不必要的路径与候选项集的匹配。SCOM 策略理论上适用于所有的 DMFIA 同类算法。

2.2 算法步骤

为了让 SCOM 策略能够较好地发挥优势,需要为 FP-tree 的每一个节点增加一个域 item\_num,用来记录该节点在 Headers 中同名链表的序号,也就是在计算以该项结尾的候选项集的支持度时检索的路径序号;为每一个候选项集  $X$  增加一个二进制向量标记(binary vector)用以表示路径对  $X$  的支持情况,记为  $X.bv$ 。 $X.bv(i)$  表示  $X.bv$  的第  $i$  位,  $X.bv(i)=0$  表示路径  $i$  不支持  $X$ ,  $X.bv(i)=1$  表示路径  $i$  支持  $X$ 。

DMFIA 同类算法计算最大频繁项集的过程大致可分为如下几个步骤:

- (1) 初始化候选项集集合;
- (2) 选取部分候选项集,计算支持度;
- (3) 将最大频繁项集加入到结果集;
- (4) 通过非最大频繁项集候选项集生成新的候选项集,通过“超级检测”等策略筛选后,代替步骤(2)中候选项集加入到候选项集集合;
- (5) 重复步骤(2)~(4)直到候选项集集合为空。

SCOM 策略主要体现在支持度计算(步骤(2))和新的候选项集生成(步骤(4))之中。支持度计算过程如下:

```
ProcedureComputeCount(FP-tree, Headers, m)
/* Headers 为频繁项头表, m 为待计算的最后一项为 i 的候选项集 */
begin
    搜索项目头表 Htable 的项目名称域 item-name, 假设 Htable
    [ q1 ]. item-name = i ;
    根据 Htable[ q1 ]. head 找到 FP-tree 中节点名称为 i 的节点
```

```
n1, n2, ..., nh ;
    根据 n1, n2, ..., nh 及其前缀节点的父节点指针域, 找到包含
    i 的所有路径 P1, P2, ..., Ph ;
    for( j = 1; j ≤ h; j++) do begin
        if m . bv 的第 j 位为 1 then // 项集空间搜索方向为“自顶向下”时
            m 支持数增加 ndj. node-count, continue; // 性质 1
        /*
        if m . bv 的第 j 位为 0 then // 项集空间搜索方向为“自底向上”时
            continue; // 性质 2
        */
        if 路径 Pj 包含 m, then //
            m 的支持数增加 ndj. node-count;
            m . bv 的第 j 位置 1
        end
    end
    对每个非最大频繁项集候选项集 m, 设 M' 是由 m
    生成的新的候选项集, 则应用 SCOM 策略的算法如下:
    begin
        for all m' ∈ M' do begin
            if m' 与 m 的最后一项相同, then
                m' . bv = m . bv;
            else
                ComputeBV(FP-tree, m, m'); // 由性质 2 根据路径的包含
                关系计算路径支持信息
            end
        end
        ProcedureComputeBV(FP-tree, Headers, m, m')
        /* 由性质 2 根据已计算过路径支持信息的 m 计算新的候
        选项集 m' 的路径支持信息, I(0) 表示全 0 的二进制向量, | m .
        bv | 表示 m . bv 的长度 */
        /* 针对“自顶向下”的项集空间搜索 */
        begin
            m' . bv = I(0);
            for i = 0 to | m . bv | do begin
                if m . bv(i) = 0 then continue;
                由 Headers 找到 item_name = m 末项名称的第 i 个节点 Ni ;
                if Ni 存在 item_name = m' 末项名称的祖先节点 Nj, Nj . item_
                num = j then
                    m' . bv(j) = 1;
            end
        /* 针对“自底向上”的项集空间搜索 */
        begin
            m' . bv = I(1);
            for i = 0 to | m . bv | do begin
                if m . bv(i) = 1 then continue;
                由 Headers 找到 item_name = m 末项名称的第 i 个节点 Ni ;
                if Ni 存在 item_name = m' 末项名称的子孙节点 Nj, Nj . item_
                num = j then
```

```
m'.bv(j)=0
end
```

2.3 算法实例

举例说明该策略的优化过程。图 1 为一棵数字频繁模式树。

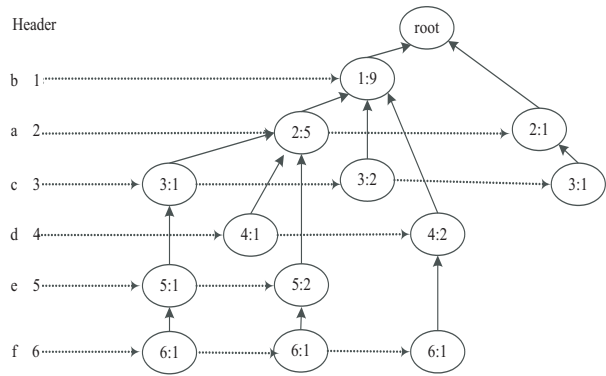


图 1 数字频繁模式树

设  $X = \{2, 3, 5\}$  为已经计算过支持度的候选项集, 则  $X.bv = (1, 0)$ ;  $Y_1 = \{2, 5\}$ ,  $Y_2 = \{2, 3\}$  为“自顶向下”的项集空间搜索算法生成的新候选项集,  $Y_3 = \{2, 3, 5, 6\}$  为根据自底向上搜索算法生成的新候选项集, 根据 SCOM 策略能够快速得出  $Y_1.bv = X.bv = (1, 0)$ , 由 ComputeBV 得  $Y_2.bv = (1, 0)$ ,  $Y_3.bv = (1, 0, 1)$ , 则在计算  $Y_1$ 、 $Y_2$  的支持度时, 只需搜索 item\_name = 6 的第二条路径, 在计算  $Y_3$  的支持度时, 只需搜索第一条路径与第三条路径。

3 实验结果

为了验证 SCOM 策略的实际优化效果, 在 8 G RAM, Intel Core i5-2430 M CPU 2.40 GHz, Windows7 操作系统上用 Java 实现了 DMFIA 原算法和应用了 SCOM 策略的 DMFIA 算法 SCOM-DMFIA。实验采用的测试数据集为 mushroom, 包含有 8 124 条记录, 记录平均长度为 23, 共有 115 个蘑菇属性。图 2 为在不同最小支持度下(分为 5%, 10%, 15%, 20%, 25% 五档)两种算法的执行时间对比结果。

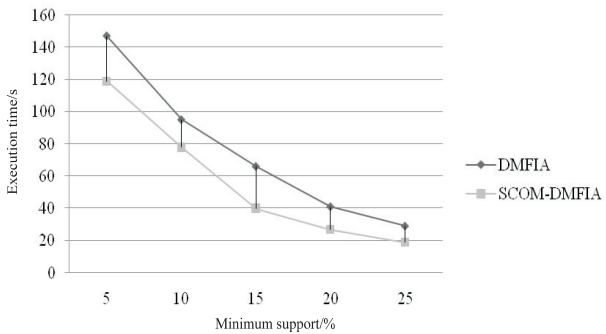


图 2 mushroom 数据集上两种算法的执行时间对比

从图 2 可以看出, 应用了 SCOM 策略的 DMFIA 算法的整体运行时间明显少于原算法。为了进一步说明

SCOM 策略通过优化支持度计数进而提高最大频繁项集挖掘算法整体性能的过程, 实验中还监测了两种算法用于支持度计数的时间开销, 如图 3 所示。

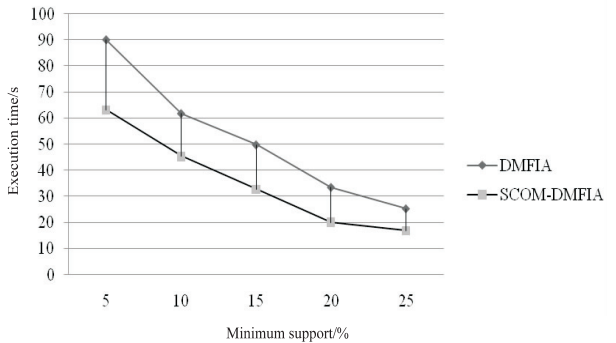


图 3 mushroom 数据集上两种算法用于支持度计数的时间对比

从图 3 中可以看出, SCOM 明显降低了 DMFIA 算法用于支持度计数的时间。

以上实验结果说明, SCOM 策略的确能在基于 FP-tree 的支持度计数过程中减少搜索路径, 降低时间开销, 进而提高整个最大频繁项集挖掘算法的效率。

4 结束语

文中提出的基于 FP-tree 的支持度计数优化策略—SCOM, 通过记录支持度计算过程中路径对项集的支持情况, 依据相关性质在支持度计算中搜索 FP-tree 时避免不必要的搜索路径, 以达到搜索优化的目的。实验结果表明, 该策略有效提高了最大频繁项集挖掘算法 DMFIA 的运行效率, 并且可以推广应用到 DMFIA 的同类算法中。下一步的工作中将进一步探究 SCOM 在其他种类频繁项集挖掘算法中的适用性。

参考文献:

[1] Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large database [C]//Proceedings of 1993 ACM SIGMOD conference on management of data. New York: ACM, 1993: 207-216.

[2] Han J, Kamber M. Data mining: concepts and techniques [M]. Beijing: High Education Press, 2001.

[3] 宋余庆, 朱玉全, 孙志挥, 等. 基于 FP-tree 的最大频繁项目集挖掘及更新算法 [J]. 软件学报, 2003, 14 (9): 1586-1592.

[4] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation [C]//Proceedings of the 2000 ACM-SIGMOD international conference on management of data. New York: ACM, 2000: 1-12.

[5] Grahne G, Zhu J. High performance mining of maximal frequent itemset [EB/OL]. [2014-07-06]. <http://www.docin.com/p-773109811.html>.



表 1 多路径拥塞控制算法的优缺点

多路径拥塞控制算法	优点	缺点
非耦合的简单多路径拥塞控制算法	多路径间单独使用拥塞控制,互不干扰	无法满足 TCP 友好性,同时不能均衡拥塞
基于路径加权的拥塞控制算法	满足 TCP 友好性,利用权重因子合理规划网络资源	多路径与 TCP 链路竞争链路资源时没有优势
耦合多路径算法	满足 TCP 友好性,同时高负载链路能转移到低负载链路	多路径之间数据的频繁转移造成链路抖动,网络不稳定
CMT/RP 拥塞控制算法	满足 TCP 友好性,同时实现负载均衡,提高网络鲁棒性	只适用于 CMT-SCTP
Linked Increases 算法	通过抢占因子控制拥塞窗口的增减已达到平衡拥塞	只适用于 MPTCP

4 结束语

针对新型互联网多路径传输协议受限于单路径传输的传统拥塞控制算法,提出了多路径拥塞控制机制的设计原则以及适用于多路径传输协议的公平性准则。同时引入了两种单纯采用传统 SCTP 拥塞控制思想的多路径拥塞控制算法和基于资源池思想的三种多路径拥塞控制算法。通过对多路径传输协议拥塞控制机制的研究,可以发现:虽然提出了 TCP-Friendliness 要求,但确实需要一种能够合理描述多路径资源分配公平性的度量准则,能够保证在不同网络场景多路径传输的普适性;多路径拥塞实现了平衡拥塞能力后,在资源最大化利用上仍有很大提升,因此在实际网络中,如何充分考虑各个路径的拥塞、丢包情况,来保证网络资源的最优利用值得探究;多路径传输应考虑路径的最优传输,如何从传输性能差异较大的多路径中选择最优的传输路径也是值得考虑的问题。

下一步将针对这几种不同的多路径拥塞控制算法,开展在实际网络(新型互联网)的开发测试,并针对实际网络,定量分析拥塞控制算法的设计和拥塞控

制的效果,找出最适用于新型互联网传输协议的拥塞控制算法。

参考文献:

[1] 张宏科,苏伟.新网络体系基础研究——体化网络与普适服务[J].电子学报,2007,35(4):593-598.

[2] 董平,秦雅娟,张宏科.支持普适服务的一体化网络研究[J].电子学报,2007,35(4):599-606.

[3] 赵珊珊.新型互联网传输控制协议的研究与设计[D].北京:北京交通大学,2016.

[4] 代志刚.新一代流控制传输协议 SCTP[J].电信快报,2001(3):32-34.

[5] Raiciu C,Wischik D,Handley M. Practical congestion control multipath transport protocols[S]. [s.l.]:[s.n.],2009.

[6] Iyengar J R,Amer P D,Stewart R. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths[J]. IEEE/ACM Transactions on Networking,2006,14(5):951-964.

[7] Wischik D,Handley M,Braun M B. The resource pooling principle[J]. ACM SIGCOMM Computer Communication Review,2008,38(5):47-52.

[8] 林开司,余东.计算机网络拥塞控制综述[J].科技资讯,2008(3):106-107.

[9] 刘宇苹.基于拥塞控制算法的研究[J].武汉船舶职业技术学院学报,2009,8(3):37-39.

[10] 韩鹏.SCTP 拥塞控制机制的研究与改进[D].南京:南京邮电大学,2011.

[11] Floyd S,Fall K. Promoting the use of end-to-end congestion control in the Internet[J]. IEEE/ACM Transactions on Networking,1999,7(4):458-472.

[12] Chiu D M,Jain R. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks[J]. Computer Networks and ISDN Systems,1989,17(1):1-14.

[13] Dreibholz T,Becke M,Adhari H,et al. On the impact of congestion control for concurrent multipath transfer on the transport layer[C]//Proceedings of the 2011 11th international conference on telecommunications. [s.l.]:IEEE,2011:397-404.

(上接第 33 页)

[6] 吉根林,杨明,宋余庆,等.最大频繁项目集的快速更新[J].计算机学报,2005,28(1):128-135.

[7] 颜跃进,李舟军,陈火旺.基于 FP-Tree 有效挖掘最大频繁项集[J].软件学报,2005,16(2):215-222.

[8] 钱雪忠,惠亮.关联规则中基于降维的最大频繁模式挖掘算法[J].计算机应用,2011,31(5):1339-1343.

[9] 颜跃进,李舟军,陈火旺.一种挖掘最大频繁项集的深度优先算法[J].计算机研究与发展,2005,42(3):462-467.

[10] 陈晨,鞠时光.基于改进 FP-tree 的最大频繁项集挖掘算法[J].计算机工程与设计,2008,29(24):6236-6239.

[11] 王黎明,赵辉.基于 FP 树的全局最大频繁项集挖掘算法[J].计算机研究与发展,2007,44(3):445-451.

[12] 付冬梅,王志强.基于 FP-tree 和约束概念格的关联规则挖掘算法及应用研究[J].计算机应用研究,2014,31(4):1013-1015.

[13] Tan Pangning. 数据挖掘导论:英文[M].北京:人民邮电出版社,2006.