第 38 卷 第 4 期
Vol.38 No.4
计 算 机 工 程
Computer Engineering
2012 年 2 月
February 2012
·软件技术与数据库· 文章编号：1000—3428(2012)04—0073—03 文献标识码：A 中图分类号：TP311.12

项约束先过滤的最大频繁项集挖掘算法

姚全珠，李如琼，王美君
(西安理工大学计算机科学与工程学院，西安 710048)

摘 要：在稠密型数据库中，现有最大频繁项集挖掘算法效率低、耗时长，挖掘结果模糊，不利于用户使用。为此，提出一种项约束先过滤的最大频繁项集挖掘算法——VCM。利用项包含约束过滤数据库，使用垂直数据表示数据集，采用深度优先的挖掘策略对数据库进行最大频繁相集的挖掘。实验结果表明，该算法快速有效，尤其在挖掘具有长模式的稠密数据库时优势明显。

关键词：关联规则；最大频繁项集；项约束；垂直数据格式；深度优先；稠密数据库

Mining Algorithm of Maximal Frequent Itemset with Item Constraint Filtering First

YAO Quan-zhu, LI Ru-qiong, WANG Mei-jun
(School of Computer Science & Engineering, Xi'an University of Technology, Xi'an 710048, China)

【Abstract】 In the dense database, mining maximal frequent itemsets takes too much time, and the results are too large to satisfy the users. This paper proposes a maximal frequent itemsets mining algorithm, called VCM. It filters the database with the constraints, uses the vertical data representation of data sets and adopts depth-first strategy for mining maximum frequent itemsets. Compared with other algorithms, experimental results show that the VCM algorithm is faster and more effective, and the advantage is remarkable when the databases are dense and with long patterns.

【Key words】 association rule; maximal frequent itemset; item constraint; vertical data format; depth-first; dense database
DOI: 10.3969/j.issn.1000-3428.2012.04.024

1 概述

文献[1]提出关联规则挖掘问题，指出频繁项集的挖掘是关联规则最为关键的步骤，决定了整个挖掘的性能。因此，包含了全部频繁项集信息，但规模却小得多的最大频繁项集的挖掘倍受青睐。

目前，最大频繁项集的挖掘算法有 GenMax[2]、MAFIA[3]、TT-Apriori[4]和 IAFP-max[5]等。其中，GenMax 算法的剪枝策略不如 MAFIA 算法；但 MAFIA 算法则对内存要求较高；TT-Apriori 算法在数据库规模较大时，事务树的优势并不明显；IAFP-max 算法基于 FP-tree[6]和阵列技术，但还存在着难以基于内存构造的瓶颈，且计算量也不容小视。

另外，关联规则挖掘过程只使用了支持度和置信度 2 个参数，使挖掘的结果太过庞大，且没有针对性。

为此，本文提出了基于垂直数据格式[7]的最大频繁项集算法——VCM，先用布尔表达式表达的项包含约束过滤数据库，再将数据垂直表示，然后采取深度优先策略，挖掘出带有长模式的稠密数据集中含有简洁性项包含约束的最大频繁项集。

2 问题的形式化描述

设 I={I1, I2, …, Im}是项的集合。设事务数据库 D 中每个事务 T 都是项的集合，每个事务有唯一标识符 TID，并且使得 T⊆I。其中，D 由事务标识符 TID 和项集 Items 组成。项 Item 的支持度 sup 就是 D 中事务包含 Item 的百分比。而包含项集的事务数称为此项集的绝对支持度计数，记为 sup_count。如果一个项集的支持度不小于用户指定的最小支持度(min_sup)，那么此项集就是一个频繁项集。

若项集 X 是频繁的，则一定有支持度计数不小于最小支持度计数，即 sup_count(X)≥min_count(X)。

定义 1(最大频繁项集) 如果一个项集 X 是频繁的，而且不存在超项集 Y 使得 X⊂Y，Y 在数据集 D 上是频繁的，那么就称项集 X 是数据集 D 中的最大频繁项集，记为 MFIS。

频繁项集具有一个重要性质：当一个项集是频繁的，则它的每个子集也是频繁的。

由此性质可以明显得出以下推理：

推理 1 任何频繁项集 X 都是某个最大频繁项集 Y 的子集。

推理 2 最大频繁项集 Y 的任何子集 X 都是频繁的。

本文中的约束采用布尔表达式表示，即设约束条件 B 为项集 I 上的布尔表达式，将 B 转换成析取范式(DNF)的表达形式，即 B=B1∨B2∨…∨Bn，其中，每个 Bi 形如 b1∧b2∧…∧bm，bj∈I。

设给定交易数据库 D 和约束条件 B，挖掘项包含约束条件 B 最大频繁项集问题的本质就是挖掘所有满足以下 2 个条件的项集：此项集是最大频繁项，且满足项包含约束条件 B。

定义 2(稠密数据集) 那些项数和事务长度相差不大的数据集被称为稠密数据集。源自现实生活中的真实稠密数据集有 chess、connect、mushroom 等，它们都包含大量长模式，最长的频繁项集能包含 30 个~40 个项，挖掘难度较大。

3 VCM 算法描述

VCM 算法大致可分为 4 步，即先对数据集按照约束分别进行过滤，将其表示为垂直数据格式，对垂直格式数据进

作者简介：姚全珠(1960－)，男，教授、博士，主研方向：数据库技术，自然语言处理，数据挖掘；李如琼、王美君，硕士研究生
收稿日期：2011-08-01 **E-mail：**alpsbear@163.com

行挖掘，最后将其与约束条件合并，全部放入 MF 中。

3.1 数据库过滤——Filter

设 D 是原始的数据集，数据集 D′为 D 中符合约束条件的所有事务的集合。对于 D 中所有满足约束条件 B 的事务 t，将其写入 D′中，从而组成一个新的数据集 D′。D′和 D 满足以下定理：

定理 在 D 中符合约束条件 B 的频繁项集 X，在 D′中也频繁，且 X 在 D 和 D′中的支持数相同。此定理在文献[8]有详细证明。

推理 3 如果项 X 在 D′中是非频繁的项集，则 X 在 D 中也不可能是满足约束条件 B 的频繁项集。

根据以上定理，在有关约束性频繁项集挖掘之前，都可以先将数据集过滤，缩小其规模，而且能保证挖掘结果的正确性。

Filter 算法描述如下：

Step1 约束过滤数据库 D

```
(1)  scan the database D once
(2)  while(t ∈ D){
(3)      if (t satisfied Bi)
(4)      write t to D′;}
```

3.2 I_Ts 转换

使用 Apriori 算法和 FP-Growth 算法挖掘频繁模式时，都是挖掘形如{TID：itemset}的水平数据格式的事务集，但在海量的稠密数据集中，这样的数据格式会使得扫描时间与计算时间大大增加。而垂直数据格式——{item：TID_set}，在确定项集支持度的时候，不需要频繁的扫描数据库，从而可节省大量时间。

I_Ts 转换函数 I_Ts( )如下：

Step2 对于每个 Di′，将其表示为垂直数据格式 VDi′

```
(1)  scan the database VDi′ once
(2)  for each frequent_1 itemset i∈ Di′{
(3)      write i to VDi′;
(4)      write the corresponding TID to VDi′;}
(5)  count the TID_Len;
(6)  Ascending sequence the VDi′;
```

3.3 挖掘函数 mining()

推理 4 如果 X 是最大频繁项集，Y 是频繁项集，且 X⊇Y，设所有包含 X 的事务集合为 TID_X，所有包含 Y 的事务集合为 TID_Y，则 TID_X⊆TID_Y，且|TID_X|≤|TID_Y|。

证明：因为 X 是 Y 的超集，根据推理 1 可得：TID_X∩TID_Y=TID_X。即 TID_X⊆TID_Y，且|TID_X|≤|TID_Y|。

从推理 4 可容易得出推理 5，即：

推理 5 如果 X 是最大频繁项集，Y 是频繁项集，且 TID_X=TID_Y，则 X⊇Y。

根据推理 4 和推理 5 能明显看出，若包含频繁项集 Y 的所有频繁的超集{X1, X2,…, Xn}都已找到，那么就不需要再考虑 Y，从而剪枝，然后分别寻找{X1, X2,…, Xn}的超集，进一步剪枝、循环，直到找到所有包含 Y 的最大频繁项集为止。

对垂直数据集 VD′进行挖掘函数 mining( )如下：

Step3 挖掘过程

```
(1)  TID_Set={∅};
(2)  for each Item in Itemset{
(3)      FI_Temp = Item_next = Item;
(4)      FI = FI_Temp × Item_next;
(5)      if ( FI ∈ MFC ){
(6)          FI_Temp = FI;
(7)          Item_next + = 1;}
(8)      else TID_C = FI_temp ∩ Item_next;
(9)      if( TID_C ≥ min_count){
(10)         join TID_C into TID_Set;
(11)         join FI into MFC;
(12)         FI_temp = FI;
(13)         Item_next + =1;}
(14)     else //由推理 4、推理 5 剪枝{
(15)         FI = FI_temp;//回溯
(16)         if(Item_next is the last one){
(17)             join FI_temp into MF;
(18)             FI = FI_temp = {∅}; }}
(19)   TID_Set = 0 ;}
```

3.4 合并约束

将找出的最大频繁项集与约束条件合并。

Step 4 后续处理

MF={Bi × Item|Item ∈ MFi, i=1,2,…,n}//与约束合并

通过一个实例来具体说明 VCM 算法，如表 1 所示。

表 1 测试数据库 D

TID	Itemsets	TID	Itemsets
1	a b c d e	6	a d e f h
2	a b c d e f	7	a b c d e h
3	b c d e f h	8	c d e
4	c d e f	9	a b c d f
5	a b e h	10	c d e f h

表 1 是一个测试数据库 D，设 min_sup=20%，约束条件 B=c，具体执行如下：

Step1 假设约束条件为 c。最小支持度为 20%时，经约束条件过滤后的 D_1'包含的事务集为{T_1, T_2, T_3, T_4, T_7, T_8, T_9, T_{10}}。

Step2 将 D_1'转换成垂直数据格式 VD_1'，并依照 tid_len 升序排列，如表 2 所示。

表 2 升序排列后的 VD_1'

Itemset	TID	tid_len
h	3 7 10	3
a	1 2 7 9	4
b	1 2 3 7 9	5
f	2 3 4 9 10	5
e	1 2 3 4 7 8 10	7
d	1 2 3 4 7 8 9 10	8

Step3 挖掘过程如下：

说明：为简化叙述，下文步骤均代表 3.3 节 Step3 中所描述的相应步骤。

第 1 个项 h。根据步骤(5)、步骤(10)、步骤(16)、步骤(19)等求交、剪枝、回溯，可得出 MFC={hb, hbe, hbed}；MF1={hbed}；TID_Set 和 FI_temp 清零。

第 2 个项 a。根据步骤(5)可知，a 未在 MFC 中出现过，则开始逐步向下求交；通过计算得到 MFC={b, be, bed, a}；MF1={hbed, abed}；TID_Set 和 FI_temp 清零。

第 3 个项 b。根据步骤(5)可知，b 在 MFC 中出现过；则继续比对，项 bf 未在 MFC 中出现过，则继续向下求交计算，通过计算得 MFC={b, be, bed, a, f, fe, fed}；MF1={hbed, abed, bfed}；TID_Set 和 FI_temp 清零。

第 4 个项 f。根据步骤(5)可知，f 在 MFC 中出现过；继续比对，项 fe 也在 MFC 中出现过；再继续比对，项 fed 也

在 MFC 中出现过；根据步骤(19)，最终得到 MF1={hbed, abed, bfed}。

Step4 将 MF1 与约束条件结合，即 MF={chbed, cabed, cbfed}。

4　实验结果与分析

本文实验在 1.25 GB 内存、Inter(R) Pentium(R) 4-2.40 GHz 的 CPU、Windows XP 的平台上，用 VC++6.0 实现了本文算法及其对比算法。本文实验使用的是 chess 数据集。此数据集源自国际象棋最终游戏 KRKPA7 的真实稠密数据集，有 76 个项，3 196 个事务，事务平均长度为 37，且支持度越低的时候它越稠密，包含的长模式越多。本文的对比算法为第 1 节中介绍的 GenMax 算法和 MAFIA 算法。

下文具体分析这 3 种算法性能：

考虑若支持度发生变换时，这 3 种算法的执行时间的差异。随机选择约束条件 B=(22)∨(2)，2 种算法执行时间如图 1 所示。

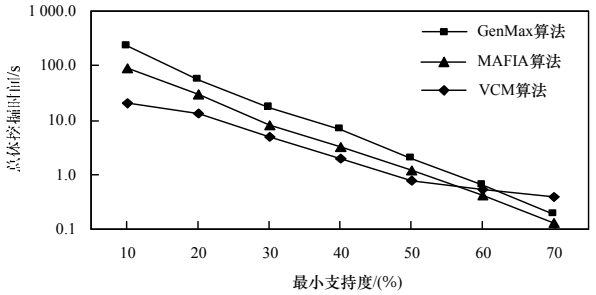

图 1　随最小支持度变化的挖掘耗时

本文算法先用约束条件对原数据集过滤，将有用数据库事务数缩减为 1 086 个。从图 1 中的数据变化趋势也可明显看出，在较小的支持度下，本文算法的计算时间代价比 GenMax 算法和 MAFIA 算法小得多。当最小支持度逐渐增大以后，较大的最小支持度过滤数据库，使数据规模锐减，此时约束条件先过滤策略将不再占优势。尤其 min_sup=50%以后，大部分的挖掘时间都是用来扫描数据库的，计算时间相差微小，此时 VCM 算法的 2 次扫描会使总体耗时加剧。但 min_sup=40%以前，最小支持度在缩减数据规模方面没有起到决定性的作用，在此时总体挖掘时间的比较中，可明显看出本文算法的高效性。

另外，由于本文算法在挖掘前先按约束条件的析取范式分别过滤数据集，针对每个数据集分别挖掘，即将满足约束的事务组成一个数据分片，然后只将此数据分片调入内存进行挖掘，既使得计算复杂度大大降低，又节约了 I/O 时间开销。而其他算法虽然不需要额外的辅助存储空间，但随着约束条件的增长，其计算规模却呈指数增长，需花费更多的计算时间。图 2 是在 min_sup=30%、随机约束条件 B=2∨10∨11∨22∨28∨39(约束条件从 1 项扩展到 6 项的过程)时，3 种算法的执行时间对比，可以明显地看出，本文算法有较高的挖掘效率。

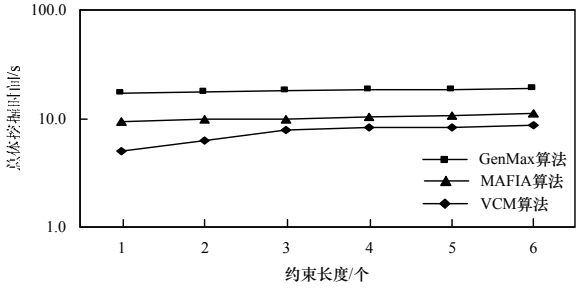

图 2　随约束条件长度增加的挖掘耗时

5　结束语

本文提出一种 VCM 算法，使用垂直数据表示方法，减少了访问数据库的次数，另外本文所提出的约束先过滤方法和有效剪枝方法，缩减了数据规模，提高了挖掘的速度。实验结果表明，本文算法 VCM 更适合稠密型数据集中挖掘最大频繁项集。

参考文献

[1] Agrawal R, Imielinski T, Swami A. Mining Association Rules Between Sets of Items in Large Databases[C]//Proc. of ACM SIGMOD International Conference on Management of Date. Washington D. C., USA: [s. n.], 1993: 207-216.

[2] Gouda K, Zaki M J. GenMax: An Efficient Algorithm for Mining Maximal Frequent Item-sets[J]. Data Mining and Knowledge Discovery, 2005, 11(3): 1-20.

[3] Burdick D, Calimlim M, Flannick J, et al. MAFIA: A Maximal Frequent Itemset Algorithm[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(11): 1490-1504.

[4] 张忠平，郑为夷. 基于事务树的最大频繁项集挖掘算法[J]. 计算机工程, 2009, 35(15): 97-99.

[5] Wang Huajin, Hu Chun'an. Mining Maximal Patterns Based on Improved FP-tree and Array Technique[C]//Proc. of the 3rd International Symposium on Intelligent Information Technology and Security Informatics. Jinggangshan, China: [s. n.], 2010: 567-571.

[6] Han Jiawei, Pei Jian, Yin Yiwen, et al. Mining Frequent Patterns Without Candidate Generation[C]//Proc. of ACM SIGMOD International Conference on Mangement of Data. Dallas, USA: [s. n.], 2000: 1-12.

[7] Zaki M J. Scalable Algorithms for Association Mining[J]. IEEE Transactions Knowledge and Data Engineering, 2000, 12(3): 372-390.

[8] 崔立新，范森淼，赵春喜. 约束性相联规则发现方法及算法[J]. 计算机学报, 2000, 23(2): 216-219.

编辑　任吉慧

(上接第 72 页)

[3] 崔　涛. 使用 ipmitool 实现 Linux 系统下对服务器的 ipmi 管理[EB/OL]. (2004-07-01). http://www.ibm.com/developerworks/cn/linux/lipmi/index.html.

[4] David M, Stephane E. IA64 Linux Kernel Design and Implementation[M]. Boston, USA: Addison-Wesley Professional Press, 2004.

[5] Rochkind M J. Advanced Unix Programming[M]. 2nd ed. Boston, USA: Addison-Wesley Professional Press, 2004.

编辑　陈　文