International Journal of Database Theory and Application
Vol. 2, No. 3, September 2009

Fast Algorithm for Mining Generalized Association Rules

Bay Vo[1] and Bac Le[2]

[1]*Faculty of Information Technology, Ho Chi Minh City University of Technology Ho Chi Minh, Vietnam*
[2]*Faculty of Information Technology University of Science, Ho Chi Minh, Vietnam*
vdbay@hcmhutech.edu.vn, lhbac@fit.hcmus.edu.vn

Abstract

In this paper, we present a new algorithm for mining generalized association rules. We develop the algorithm which scans database one time only and use Tidset to compute the support of generalized itemset faster. A tree structure called GIT-tree, an extension of IT-tree, is developed to store database for mining frequent itemsets from hierarchical database. Our algorithm is often faster than MMS_Cumulate, an algorithm mining frequent itemsets in hierarchical database with multiple minimum supports, in experimental databases.

Keywords: *generalized association rules, GIT-tree, hierarchical database, multiple minimum supports.*

1. Introduction

Mining association rules plays an important role in knowledge discovery and data mining (KDD) [1, 9]. Its purpose is mining the hidden knowledge in databases.

Mining association rules in hierarchical database has been proposed in [2, 3, 5, 8, 10, 11]. In [8], the "Mining Generalized Association Rules" problem is mining association rules among items in hierarchical tree that satisfy minSup and minConf. However, this study does not change the support in different hierarchical levels. The paper [3] proposed the uniform minimum support in each level, items in the same level get the same minimum support. The purpose is mining association rules among itemsets in the same level. Another association rule mining method with multiple minimum supports was proposed in [10]. This allows users identify the different minimum supports for different items, so we can mine both frequent and rare rules. However, this method does not traverse the whole hierarchism so that it is very difficult to find association rules among items in different levels.

The research in [3] also considered in mining association rules among items in different levels. However, the limitations of this method are still based on Apriori method and used a uniform minimum support for each level.

Paper [5] introduced MMS_Stratify and MMS_Cumulate algorithms for mining frequent itemsets from hierarchical database with multiple minimum supports and proposed the computing of the minimum support of items based on the lift measure. This makes us no need to identify the minimum supports of items.

The main contributions of paper are as follows:

- We develop GIT-tree data structure, an extension of IT-tree [6].

- We propose an algorithm for mining frequent itemsets in hierarchical databases based on GIT-tree: using formulas in [5], we develop an algorithm of mining frequent itemsets based on GIT-tree for reducing run-time.

Section 2 presents concepts and definitions. Section 3 proposes GIT-tree for mining frequent itemsets in hierarchical database. We present our algorithm in section 4. Experimental results are presented in section 5.

2. Concepts

Let $I = \{i_1, i_2,\ldots, i_m\}$ be set of items, $D = \{t_1, t_2,\ldots, t_n\}$ be set of transactions, each transaction $t_i = <tid, X>$ determines a unique tid and an itemset X ($X \subseteq I$). Suppose that hierarchical database is a graph in $I \cup J$, where $J = \{j_1, j_2,\ldots, j_p\}$ represents for a general set of items that derives from I. An arc in graph G is an is-a relation, means that if an arc starts from j to i, then j is parent of i.

Figure 1 illustrates hierarchical database which are built for I = {Laser printer, Ink-jet printer, Dot matrix printer, Desktop PC, Notebook, Scanner} and J = {Non-impact printer, Printer, PC}.

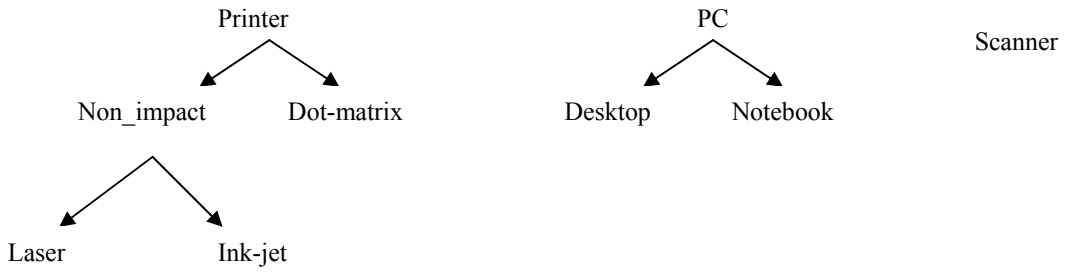

Figure 1. Example of hierarchical tree [5]

2.1. Definition 1: Given transaction t = <tid, X>, itemset Y belongs to t if every items of Y belong to X, or parent node of an item belongs to X. An itemset X has the support s, called sup(X), in D if s% of transactions in D contains X.

2.2. Definition 2: Given set of transactions D and a hierarchical tree G, a general association rule is the form:

$$X \rightarrow Y - X$$

where $X, Y \subset I \cup J, \varnothing \neq X \subset Y$, there is no item in Y which is the parent node of any item in X. The support of rule is sup(Y). The confidence of rule, $conf(X \rightarrow Y - X) = sup(Y)/sup(X)$.

2.3. Definition 3: Let ms(x) be the minimum support of an item $x \in I \cup J$. An itemset $X = \{x_1, x_2,\ldots, x_k\}$, where $x_i \in I \cup J$ and $1 \leq i \leq k$, is frequent if the support of X is greater or equal the smallest value of items in X.

$$\sup(X) \geq \min_{x_i \in X} ms(x_i) \quad (1)$$

2.4. Definition 4: An association rule X $\rightarrow$ Y – X is strong if its frequent satisfies $\sup(X \rightarrow Y - X) \geq \min_{y_i \in Y} ms(y_i)$ and conf(X $\rightarrow$ Y - X) $\geq$ minConf.

The greatest difficulty of general association rule mining is the identification of minimum support for each item. To solve this problem, the paper [2] proposed a formula to compute the minimum support:

$$ms(x) = \begin{cases} \alpha \times \sup(x) & if \quad \alpha \times \sup(x) > \min Sup \\ \min Sup & if \quad else \end{cases} \tag{2}$$

The parameter α ($0 \leq \alpha \leq 1$) is used to adjust the minimum support of single item x, which relates to its reality frequent in databases. If $\alpha = 0$, it becomes the homogeneous form. As the formula, user still identifies the minSup and parameter α.

The formula above make user difficult in identifying the parameter α. Thus, authors in [5] proposed the formula to identify the minimum support:

$$ms(x_i) = \begin{cases} \sup(x_i) \times \max\{\min Conf, \sup(x_{i+1})\} if\ 1 \leq i \leq n-1 \\ \sup(x_i) \qquad if \quad i = n \end{cases} \tag{3}$$

Example: consider the database

Table 1. Transaction database [5]

tid	Items
1	Notebook, Laser printer
2	Scanner, Dot-matrix printer
3	Dot-matrix printer, Ink-jet printer
4	Notebook, Dot-matrix printer, Laser printer
5	Scanner
6	Desktop Computer

From Table 1, and equation 3, we have the result:

Table 2. Items are sorted according to supports and *minConf* = 50%

Item	Support (%)	ms (%)
Desktop	16.7	8.3
Ink-jet	16.7	8.3
Laser	33.3	16.7
Notebook	33.3	16.7
Scanner	33.3	16.7
Dot-matrix	50.0	25.0
Non-impact	50.0	25.0
PC	50.0	33.3
Printer	66.7	66.7

2.5. Definition 5: The minimum support of an itemset:

$$ms(X) = \min_{x \in X} ms(x) \tag{4}$$

Remark 1: From equation 4, we see that the order of items in X = {x_1, x_2, …, x_m} is increasing follow ms, then ms(X) = ms(x_1). Besides, if we have ms(X) and ms(Y), then ms(X∪Y) = min{ms(X), ms(Y)}. Based on this remark, we need not use equation 4 (use the loop) to compute ms(X∪Y).

3. GIT-tree

Based on IT-tree [6], we add one more field ms(X) to identify minimum support of itemset X.

3.1. Vertex: Includes 3 fields

X: an itemset.

Tidset: the set of transaction contains X.

ms: The minimum support of X.

A vertex is denoted: $X \times_{ms(X)} Tidset(X)$.

3.2. Arc: Connecting the vertex at k^{th} level (called X) with the vertex at $(k+1)^{th}$ (called Y) in which $X \equiv_{\theta_k} Y$ ($X \equiv_{\theta_k} Y$ if X and Y have the same k-prefix – see [6] for more details) and Y does not contain the item that is the parent of any item in X.

Consider the database in Table 1. We have results:

Table 3. Map items to IDs and ms of each item

ID	Item	Support(%)	ms(%)
A	Desktop	16.7	8.3
B	Ink-jet	16.7	8.3
C	Laser	33.3	16.7
D	Notebook	33.3	16.7
E	Scanner	33.3	16.7
F	Dot-matrix	50.0	25.0
G	Non-impact	50.0	25.0
H	PC	50.0	33.3
I	Printer	66.7	66.7

Table 4. Database in Table 1 after mapping

tid	Items
1	D, C
2	E, F
3	F, B
4	D, F, C
5	E
6	A

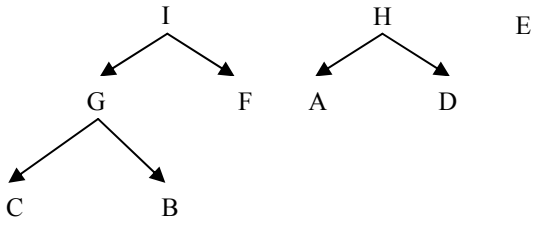

Figure 2. Hierarchical tree after mapping

Table 5. Database after add parent items

Tid	Items
1	D, C, H, G, I
2	E, F, I
3	F, B, I, G
4	D, F, C, H, I, G
5	E
6	A, H

Table 6. Convert into vertical transaction database

item	Tids
A	6
B	3
C	1, 4
D	1, 4
E	2, 5
F	2, 3, 4
G	1, 3, 4
H	1, 4, 6
I	1, 2, 3, 4

With minConf = 50%, we have results in Table 7.

Table 7. The support with ms+ of items (where ms+ = $\lceil ms*|D|/100 \rceil$)

ID	Support	ms+
A	1	1
B	1	1
C	2	1
D	2	1
E	2	1
F	3	2
G	3	2
H	3	2
I	4	4

4. Algorithm for mining frequent itemsets in hierarchical database based on GIT-tree

In [5], authors proposed algorithms MMS_Stratify and MMS_Cumulate based on Apriori algorithm that lead to do many database scans, and the candidate number will be very large. In this section, we apply GIT-tree to mine frequent itemsets in hierarchical database to reduce the mining time.

4.1. Algorithm

Input: Hierarchical database D and *minConf*

Output: Generalized frequent itemsets in D

Create IMS /* minimum supports table*/

Create IA /* The item and its parent node in hierarchical tree G*/

SMS = Sort(IMS) /* sort IMS table in increasing order of ms(x) */

F = F-gen (SMS, D, IA)

L_r = First level of GIT-tree contains nodes $i \times \underset{ms(i)}{Tidset(i)}$ with $i \in I \cup J$

ENUMERATE_GENERALIZED_FIs(L_r)

ENUMERATE_GENERALIZED_FIs (L_r)

for all node $X \times \underset{ms(X)}{Tidset(X)} \in L_r$ do

$L_c = \varnothing$

for all $Y \times \underset{ms(Y)}{Tidset(Y)} \in L_r$ with Y after X do

X' = X $\cup$ Y

if $\forall x \in X', \neg\exists y \in X'$: parent(x)=y then

T = Tidset(X) $\cap$ Tidset(Y)

ms(X') = min(ms(X), ms(Y))

if |T| $\geq$ ms(X') then

$$L_c = L_c \cup \left\{ \begin{matrix} X' \times T \\ ms(X') \end{matrix} \right\}$$

ENUMERATE_GENERALIZED_FIs(L_c)

Algorithm 1. Mining frequent itemsets in hierarchical database using GIT-tree

First, we create the minimum support table of items based on equation 3, then create parent-child relation table in hierarchical tree G. Finally, algorithm sorts the IMS in increasing order of minimum support (aim to create set F that includes single items satisfying minimum support threshold of the smallest minimum support item).

Function ENUMERATE_GENERALIZED_FIs(L_r) creates GIT-tree to mine generalized frequent itemsets. It creates a new equivalence class L_c by considering every Y after X to form itemset X' and Tidset of X' (T). If items in X have not parent-child relation each other, we consider whether its support satisfies the minimum support or not (ms(X') is computed by remark 1). If it satisfies ms(X'), then we add this node to L_c.

4.2. Illustration

Consider the database in Table 6, we have results as follows:

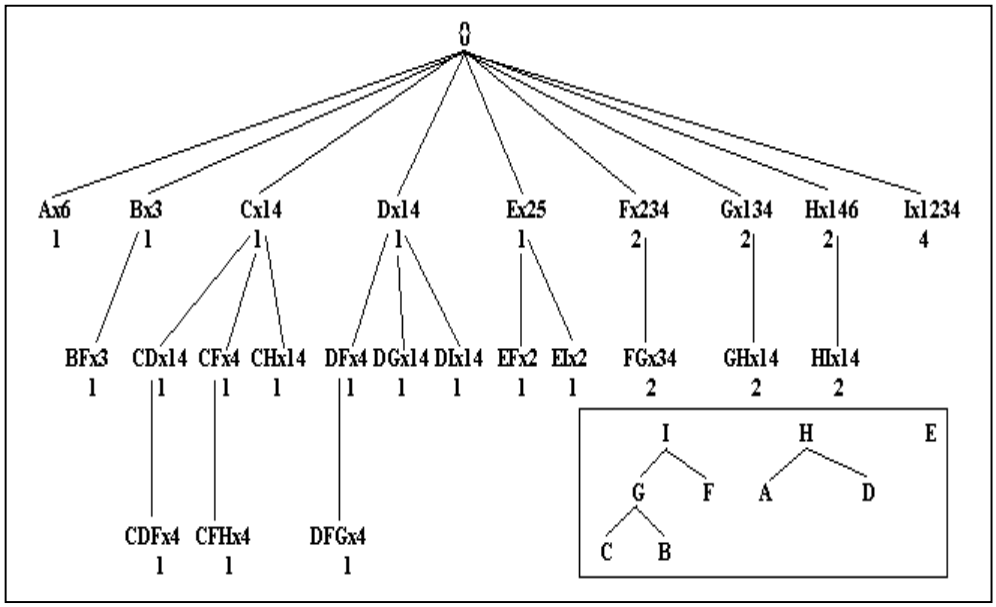

Figure 3. GIT-tree for mining generalized frequent itemsets

Each node in GIT-tree includes 3 elements: itemset, Tidset and ms. Example: DG belongs to transactions (tids) 1, 4 and ms(DG) = 1. When we combine nodes X, Y at higher level to be node at lower level, we compute the intersection between two tidsets and find min{ms(X), ms(Y)}.
Example:

Consider the combination between D and G to be DG, we compute Tidset(DG) = Tidset(D) ∩ Tidset(G) = 14, and then compute ms(DG) = min{ms(D), ms(G)} = min{1, 2} = 1. Because |Tidset(DG)| ≥ ms(DG) so we add DG to next level.

Note: Although GI has support is 3 (Tidset(GI) = Tidset(G) ∩ Tidset(I) = 134 ∩ 1234 = 134) ≥ ms(GI) = min(ms(G), ms(I)) = 2, but I is a generalized item of G, GI does not combine together.

4.3. Diffset for fast computing the support

In [7], authors used Diffset for fast computing the support of itemset and saving memory that stores Tidset. We also use it for mining generalized frequent itemsets to reduce run-time and memory.

Table 8. Average size of Tidset and Diffset in database from UCI [7]

Database	*MinSup* (%)	Avg. of Diffset size	Avg. of Tidset size	Scale Tidset/Diffset
chess	0.5	26	1820	70
connect	90	143	62204	434.99
mushroom	5	60	622	10.37
pumsb_star	35	301	18977	63.04
pumsb	90	330	45036	136.47
T10I4D100K	0.1	31	230	7.42
T40I10D100K	0.5	96	755	7.86

5. Experimental results

Experimental results are performed in databases from Microsoft Foodmart2000 of Microsoft SQL2000. We call the algorithm based on GIT-tree is MMS_GIT-Tree. Database Foodmart is results from sales_fact_1997 and synthetized database is gotten from sales_fact_1997, sales_fact_1998 and sales_fact_dec_1998 in Foodmart2000. Hierarchical items include 3 levels: 1560 items in level 1 (products), 110 general items in level 2 (product subset) and 47 general items in top level (product types). Both of them have 5581 transactions.

For accurate purpose, we run 5 times with each minConf, and the result is averaged of 5 times.

Table 9. Comparing run-time in database Sales_fact_1997 with minConf thresholds

minConf (%)	Time (s)	
	MMS_Cumulate	MMS_GIT-Tree
90	25.14	24.68
85	25.38	24.82
80	26.12	24.96
75	27.86	25.05
70	30.92	25.21
65	39.25	25.39
60	57.69	26.85
55	104.85	27.09
50	257.36	27.36

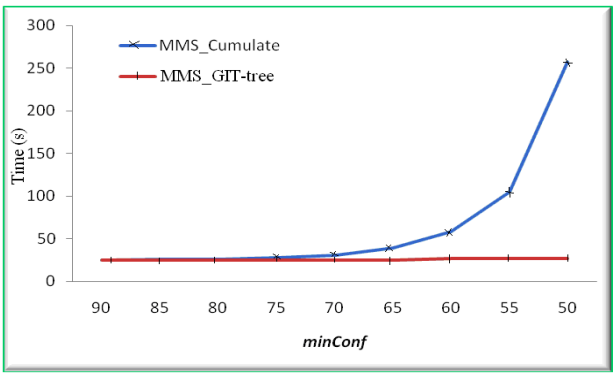

Figure 4. Comparing the run-time between MMS_Cumulate and MMS_GIT-Tree in Sales_fact_1997

Table 10. Comparing the run-time in database sales_fact_synthetized

minConf (%)	Time (s)	
	MMS_Cumulate	MMS_GIT-Tree
90	104.36	101.88
85	115.83	102.29
80	156.22	103.02
75	252.91	103.34
70	580.10	108.21

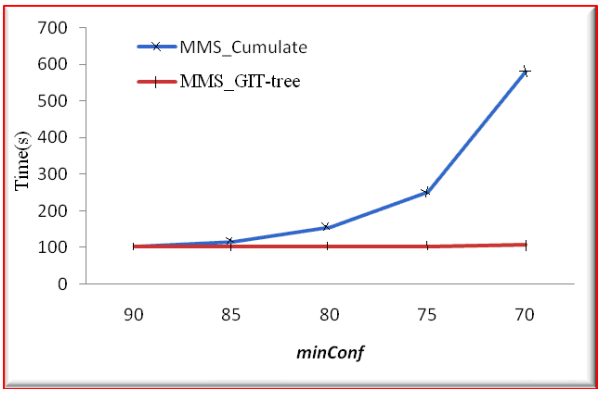

Figure 5. Comparing run-time between MMS_Cumulate and MMS_GIT-Tree in Sales_fact_synthetized

From Figures 4 and 5, we see that the run-time based on GIT-tree is more effective than that of MMS_Cumulate, especially when minConf is low. It happens because the smaller the minConf is, the larger the number of candidates MMS_Cumulate are. This costs a lot of time to mine candidates and compute the support of candidates. In contrast, GIT-tree based on the intersection of Tidset to compute the support of itemsets fast, so the mining time is faster.

6. Conclusion and future works

Identifying the minimum support (minSup) for mining association rules in database and hierarchical database has many difficulties. Thus, authors in [5] proposed the method to direct compute the minimum support of items based on their support. However, the Apriori-based algorithm consumed a lot of time. This paper introduces an efficient algorithm to mine frequent itemsets in hierarchical database with multiple minimum supports that based on GIT-tree. Experimental results show the effect of proposed algorithm.

In future, we will develop the efficient algorithm for fast mining generalized association rules which are generated among generalized frequent itemsets. Besides, the evaluation of rules through measures is considered to find the best rules for users. Next, efficient algorithm for mining frequent closed itemsets from hierarchical database will be discussed.

References

[1] A. Savasere, E. Omiecinski, S. Navathe, "An efficient algorithm for mining association rules in large databases", in: Proc. 21st Int. Conf. on Very Large Data Bases, Zurich, Switzerland, 1995, pp. 432–444.
[2] B. Liu, W. Hsu, Y. Ma, "Mining association rules with multiple minimum supports", in: Proc. 1999 Int. Conf. on Knowledge Discovery and Data Mining, San Deige, CA, 1999, pp. 337–341.
[3] J. Han, Y. Fu, "Discovery of multiple-level association rules from large databases", in: Proc. 21st Int. Conf. on Very Large Data Bases, Zurich, Switzerland, pp. 420–431, 1995.
[4] J. Han, M. Kamber, "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers, pp. 250 – 254, 2006.
[5] M.C. Tseng , W.Y. Lin, "Efficient mining of generalized association rules with non-uniform minimum support", Data & Knowledge Engineering 62, ScienceDirect, pp. 41–64, 2007.
[6] M. J. Zaki, C.J. Hsiao, "Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure", IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No 4, April 2005, pp. 462-478, 2005.
[7] M.J. Zaki and K. Gouda, "Fast Vertical Mining Using Diffsets", in: Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, Aug. 2003, pp. 326–335.
[8] R. Agrawal, R. Srikant, "Fast algorithms for mining generalized association rules", in: Proceedings of the 20th International Conference on Very Large Databases (VLDB94), Santiago, Chile, pp. 487–499, 1994.
[9] R. Agrawal, T. Imielinski, A. Swami, "Mining association rules between sets of items in large databases", in: Proc. 1993 ACMSIGMOD Int. Conf. on Management of Data, Washington, DC, pp. 207–216, 1993.
[10] R. Srikant, R. Agrawal, "Mining generalized association rules", Future Generation Computer Systems 13 (2–3), pp. 161–180, 1997.
[11] S. Brin, R. Motwani, C. Silverstein, "Beyond market baskets: generalizing association rules to correlations", in: Proc. 1997 ACMSIGMOD Int. Conf. on Management of Data, pp. 207–216, 1997.

Authors

Bay Vo is currently Ph.D student at Computer Science department, University of Science, Ho Chi Minh City, Vietnam. He received his Bachelor of Science (2002) and Master of Science (2005) degrees from University of Science, Ho Chi Minh City, Vietnam. His research interests include association rules, classification, data mining in multidimensional database, distributed database and privacy preserving in data mining.

Bac Le received the BSc degree, in 1984, the MSc degree, in 1990, and the PhD degree in Computer Science, in 1999. He is an Associate Professor, Vice Dean of Faculty of Information Technology, Head of Department of Computer Science, University of Science, Ho Chi Minh City. His research interests are in Artificial Intelligent, Soft Computing, and Knowledge Discovery and Data Mining.