Mining Generalized Closed Frequent Itemsets of Generalized Association Rules

Kritsada Sriphaew and Thanaruk Theeramunkong

Sirindhorn International Institute of Technology, Thammasat University
131 Moo 5, Tiwanont Rd., Bangkadi, Muang, Pathumthani, 12000, Thailand
{kong,ping}@siit.tu.ac.th

Abstract. In the area of knowledge discovery in databases, the generalized association rule mining is an extension from the traditional association rule mining by given a database and taxonomy over the items in database. More initiative and informative knowledge can be discovered. In this work, we propose a novel approach of generalized closed itemsets. A smaller set of generalized closed itemsets can be the representative of a larger set of generalized itemsets. We also present an algorithm, called *cSET*, to mine only a small set of generalized closed frequent itemsets following some constraints and conditional properties. By a number of experiments, the *cSET* algorithm outperforms the traditional approaches of mining generalized frequent itemsets by an order of magnitude when the database is dense, especially in real datasets, and the minimum support is low.

1 Introduction

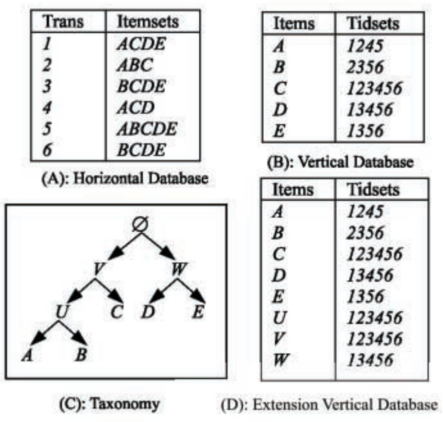
The task of association rule mining (ARM) is one important topic in the area of knowledge discovery in databases (KDD). ARM focuses on finding a set of all subsets of items (called itemsets) that frequently occur in database records or transactions, and then extracting the rules representing how a subset of items influences the presence of another subset [1]. However, the rules may not provide informative knowledge in the database. It may be limited with the granularity over the items. For this purpose, generalized association rule mining (GARM) was developed using the information of pre-defined taxonomy over the items. The taxonomy may classify products (or items) by brands, groups, categories, and so forth. Given a taxonomy where only leaf items present in the database, more initiative and informative rules (called generalized association rules) can be mined from the database. Each rule contains a set of items from any levels of the taxonomy.

In the past, most previous works focus on efficient finding all generalized frequent itemsets. As an early intensive work, Srikant et al. [2] proposed five algorithms that apply the horizontal database format and breath-first search strategy like Apriori algorithm. These algorithms waste a lot of time in multiple scanning a database. As a more recent algorithm, Prutax, was proposed in [3] by applying a vertical database format to reduce the time needed in database scanning. Nevertheless, a limitation of this work is the cost of checking whether their ancestor itemsets are frequent or not using a hash tree. There exists a slightly different task for dealing with multiple minimum supports as shown in [4, 5, 6]. A parallel algorithm was proposed in [7]. The recent applications of GARM were shown in [8, 9]. Our efficient approach to mine all generalized frequent itemsets is presented in [10]. Furthermore to improve time complexity of the mining process, the concepts of closed itemsets have been proposed in [11, 12, 13]. The main idea of these approaches focus to find only a small set of closed frequent itemsets, which is the representative of a large set of frequent itemsets. This technique helps us reduce the computational time. Thus, we intend to apply this traditional concept to deal with the generalized itemsets in GARM. In this work, we propose a novel concept of generalized closed itemsets, and present an efficient algorithm, named cSET, to mine only generalized closed frequent itemsets.

2 Problem Definitions

A generalized association rule can be formally stated as follows. Let $I = \{A, B, C, D, E, U, V, W\}$ be a set of distinct items, $T = \{1, 2, 3, 4, 5, 6\}$ be a set of transaction identifiers (*tids*). The database can be viewed into two formats, i.e. horizontal format as shown in Fig. 1A, and vertical format as shown in Fig. 1B. Fig. 1C shows the taxonomy, a directed acyclic graph on items. An edge in a taxonomy represents *is-a* relationship. V is called an *ancestor* item of U, C, A and B. A is called a *descendant* item of U and V. Note that only leaf items of a taxonomy are presented in the original database. Intuitively, the database can be extended to contain the ancestor items by adding the record of ancestor items of which tidsets are given by the union of their children as shown in Fig. 1D.

A set $I_G \subseteq I$ is called a *generalized itemset* (GI) when I_G is a set of items where no any items in the set is an ancestor item of the others. The *support* of I_G, denoted by $\sigma(I_G)$, is defined as a percentage of the number of transactions in which I_G occurs as a subset to the total number of transactions. Only the GI that has its support greater than or equal to a user-specified *minimum support* (*minsup*) is called a *generalized frequent itemset* (GFI). A rule is an implication of the form R: $I_1 \mapsto I_2$, where I_1, $I_2 \subseteq I$, $I_1 \cap I_2 = \emptyset$, $I_1 \cup I_2$ is GFI, and no item in I_2 is an ancestor of any items in I_1. The confidence of a rule, defined as $\sigma(I_1 \cup I_2)/\sigma(I_1)$, is the conditional probability that a transaction contains I_2, given that it contains I_1. The rule is called a *generalized association rule* (GAR) if its confidence is greater than or equal to a user-specified *minimum confidence* (*minconf*). The task of GARM can be divided into two steps, i.e. 1) finding all GFIs and 2) generating the GARs. The second step is straightforward while the first step takes intensive computational time. We try to improve the first step by exploiting the concept of closed itemsets to GARM, and find only a small set of generalized closed itemsets to reduce the computational time.



Trans	Itemsets
1	ACDE
2	ABC
3	BCDE
4	ACD
5	ABCDE
6	BCDE

(A): Horizontal Database

Items	Tidsets
A	1245
B	2356
C	123456
D	13456
E	1356

(B): Vertical Database

$$\begin{tikzcd}
 & & & \varnothing \arrow[dl] \arrow[dr] & \\
 & & V \arrow[dl] \arrow[dr] & & W \arrow[dl] \arrow[dr] \\
 & U \arrow[dl] \arrow[dr] & & C \quad D & & E \\
A & & B & & &
\end{tikzcd}$$

(C): Taxonomy

Items	Tidsets
A	1245
B	2356
C	123456
D	13456
E	1356
U	123456
V	123456
W	13456

(D): Extension Vertical Database

Fig. 1. Databases and Taxonomy

3 Generalized Closed Itemset (GCI)

In this section, the concept of GCI is defined by adapting the traditional concept of closed itemsets in ARM [11, 12, 13]. We show that a small set of generalized closed frequent itemsets is sufficient to be the representative of a large set of GFIs.

3.1 Generalized Closed Itemset Concept

Definition 1. *(Galois connection): Let the binary relation $\delta \subseteq I \times T$ be the extension database. For an arbitrary $x \in I$ and $y \in T$, $x\delta y$ can be denoted when x is related to it y in database. Let it $X \subseteq I$, and $Y \subseteq T$. Then the mapping functions,*

$t : I \mapsto T,\ t(X) = \{y \in T \mid \forall x \in X,\ x\delta y\}$

$i : T \mapsto I,\ i(Y) = \{x \in I \mid \forall y \in Y,\ x\delta y\}$

define a Galois connection between the power set of I ($\mathcal{P}(I)$) and the power set of T ($\mathcal{P}(T)$). The following properties hold for all $X, X_1, X_2 \subseteq I$ and $Y, Y_1, Y_2 \subseteq T$:

1. $X_1 \subseteq X_2 \Longrightarrow t(X_1) \supseteq t(X_2)$
2. $Y_1 \subseteq Y_2 \Longrightarrow i(Y_1) \supseteq i(Y_2)$
3. $X \subseteq i(t(X))$ *and* $Y \subseteq t(i(Y))$

Definition 2. *(Generalized closure): Let $X \subseteq I$, and $Y \subseteq T$, the composition of two mappings gc_{it}:$\mathcal{P}(I) \mapsto \mathcal{P}(I)$ and gc_{ti}: $\mathcal{P}(T) \mapsto \mathcal{P}(T)$ are generalized closure operator on itemset and tidset respectively. The mapping of $gc_{it}(X) = i \circ t(X) = i(t(X))$ while $gc_{ti}(Y) = t \circ i(Y) = t(i(Y))$.*

Definition 3. *(Generalized closed itemset and tidset): X is called a generalized closed itemset (GCI) when $X = gc_{it}(X)$, and Y is called a generalized closed tidset (GCT) when $Y = gc_{ti}(Y)$.*

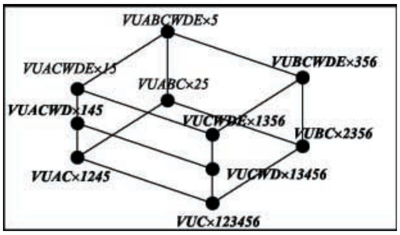


Fig. 2. Galois Lattice of Concepts

For X ⊆ I and Y ⊆ T, the generalized closure operators gc_{it} and gc_{ti} satisfy the following properties (Galois property):

1. $Y \subseteq gc_{ti}(Y)$.
2. $X \subseteq gc_{it}(X)$.
3. $gc_{it}(gc_{it}(X)) = gc_{it}(X)$, and $gc_{ti}(gc_{ti}(Y)) = gc_{ti}(Y)$.

For any GCI X, there exists a corresponding GCT Y, with the property that $Y = t(X)$ and $X = i(Y)$. Such a GCI and GCT pair $X \times Y$ is called a *concept*. All possible concepts can be formed a Galois lattice of concepts as shown in Fig. 2.

3.2 Generalized Closed Frequent Itemsets (GCFIs)

The support of a concept $X \times Y$ is the size of GCT (i.e. $|Y|$). A GCI is frequent when its support is greater than or equal to minsup.

Lemma 1. *For any generalized itemset X, its support is equal to the support of its generalized closure ($\sigma(X) = \sigma(gc_{it}(X))$).*

Proof. Given X, its support $\sigma(X) = |t(X)|/|T|$, and the support of its generalized closure $\sigma(gc_{it}(X)) = |t(gc_{it}(X))|/|T|$. To prove the lemma, we have to show that $t(X) = t(gc_{it}(X))$. Since gc_{ti} is the generalized closure operator, it satisfies the first property that $t(X) \subseteq gc_{ti}(t(X)) = t(i(t(X))) = t(gc_{it}(X))$. Thus $t(X) \subseteq t(gc_{it}(X))$. The $gc_{it}(X)$ provides the GI that is the maximal superset of X and has the same support as X. Then $X \subseteq gc_{it}(X)$, and $t(X) \supseteq t(gc_{it}(X))$ due to the Galois property [11]. We can conclude that $t(X) = t(gc_{it}(X))$.

Implicitly, the lemma states that all GFIs can be uniquely determined by the GCFIs since the support of any GIs will be equal to its generalized closure. In the worst case, the number of GCFIs is equal to the number of GFIs, but typically it is much smaller. From the previous example, there are 10 GCIs, which are the representatives of a large amount of all GIs as shown in Fig. 2. With minsup=50%, only 7 concepts (in bold font) are GCFIs.

4 Algorithm

4.1 cSET Algorithm

In our previous work [10], all GFIs can be enumerated by applying two constraints, i.e. subset-superset and parent-child, on GIs for pruning. We propose an algorithm called *cSET* algorithm, which specifies the order of set enumeration by using these two constraints and the generalized closures to generate only GCIs. Two constraints stated that only descendant and superset itemsets of GFIs should be considered in the enumeration process. For generating only GCFIs, the following conditional properties must be checked when generating the child itemsets by joining $X_1 \times t(X_1)$ with $X_2 \times t(X_2)$.

1. If $t(X_1) = t(X_2)$, then (1) replace X_1 and children under X_1 with $X_1 \cup X_2$, (2) generate taxonomy-based child itemsets of $X_1 \cup X_2$, and (3) remove X_2 (if any).
2. If $t(X_1) \subset t(X_2)$, then (1) replace X_1 with $X_1 \cup X_2$ and (2) generate taxonomy-based child itemsets of $X_1 \cup X_2$.
3. If $t(X_1) \supset t(X_2)$, then (1) generate join-based child itemset of X_1 with $X_1 \cup X_2$, (2) add hash table with $X_1 \cup X_2$, and (3) remove X_2 (if any).
4. If $t(X_1) \neq t(X_2)$ and $t(X_1 \cup X_2)$ is not contain in hash, then generate join-based child itemset of X_1 with $X_1 \cup X_2$.

Using the given example in Fig. 1 with minsup=50%, the *cSET* algorithm starts with an empty set. Then, we add all frequent items in the second level of the taxonomy, that are item V and W, and form the second level of the tree shown in Fig. 3. Each itemset has to generate two kinds of child itemsets, i.e. *taxonomy-based* and *join-based* itemsets, respectively. We first generate taxonomy-based itemset by joining last items in itemsets by its child according to taxonomy. One taxonomy-based itemset of V is VU. The first property holds for VU, which results in replacing V with VU and then generating VUA and VUB. The second taxonomy-based itemset is joined with the current itemset (VU), which produces VUC. Again, the first property holds for VUC, which results in replacing VU and the children in tree under VU with VUC. Next, the join-based child itemset of V, VW, is generated. The third property holds for VW, which results in removing W and then generating VW under V. In the same approach, the process recursively occurs until no new GCFIs are generated. Finally, a complete itemset tree is constructed without excessive checking cost as shown in Fig. 3. All remaining itemsets in Fig. 3, except ones of crossed itemsets, are GCFIs.

4.2 Pseudo-code Description

The formal pseudo-code of *cSET*, extended from *SET* in [10], is shown below. The main procedure is `cSET-MAIN` and a function, called `cSET-EXTEND`, creates a subtree followed by a proposed set enumeration. `cSET-EXTEND` is executed recursively to create all itemsets under the root itemsets. The `NewChild` function

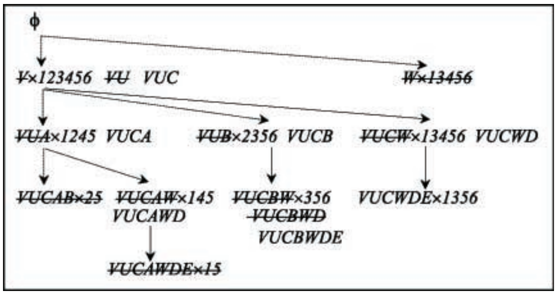


Fig. 3. Finding GCFIs using *cSET* with minsup=50%

creates a child itemset. For instance, `NewChild`(V, U) creates a child itemset VU of a parent itemset V, and adds the new child in a hash table. The `GenTaxChild` function returns the taxonomy-based child itemsets of that GI. Line 8-11 generates the join-based child itemsets. The function, called `cSET-PROPERTY`, checks for four conditional properties of GCIs and makes the operations with the generated itemset. Following the *cSET* algorithm, we will get a tree of all GCFIs.

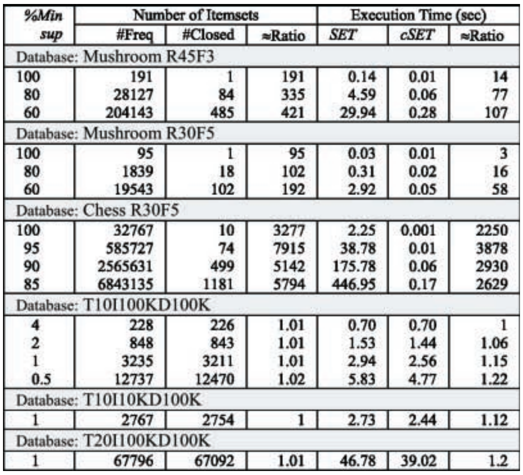
```
cSET-MAIN (Database,Taxonomy,minsup):
 1. Root = Null Tree                  //Root node of set enumeration
 2. NewChild(Root, GFIs from second level of taxonomy)
 3. cSET-EXTEND(Root)

cSET-EXTEND(Father)
 4. For each $F_i$ in  Father.Child
 5.      C = GenTaxChild($F_i$)         //Gen taxonomy-based child itemset
 6.      If  supp(C) $\geq$ minsup then
 7.          cSET-PROPERTY(Nodes,C)
 8.      For j = i+1 to |Father.Child|   //Gen join-based child itemset
 9.      C = $F_i \cup F_j$
10.      If  supp(C) $\geq$ minsup then
11.          cSET-PROPERTY(Nodes,C)
12.      If $F_i$.Child $\neq$ NULL then cSET-EXTEND($F_i$)

cSET-PROPERTY(Node,C)
13. if t($F_i$) = t($F_j$) and Child($F_i$)= $\emptyset$ then          //Prop.1
14.      Remove($F_j$);  Replace($F_i$) with C
15. else if t($F_i$) $\subset$ t($F_j$) and Child($F_i$)= $\emptyset$ then     //Prop.2
16.      Replace($F_i$) with C
17. else if t($F_i$) $\supset$ t($F_j$) then                       //Prop.3
18.      Remove($F_j$); if !Hash(t(C)) then NewChild($F_i$,C)
19. else if !Hash(t(C)) then NewChild($F_i$,C)          //Prop.4
```

5 Experimental Results

Since the novel concept of GCIs has never appeared in any researches, there are no existing algorithms for finding GCFIs. In our experiment, the *cSET* algorithm is evaluated by comparing with the current efficient algorithms for mining GFIs, i.e. SET algorithm [10]. All algorithms are coded in C language and the experiment was done on a 1.7GHz PentiumIV with 640Mb of main memory running Windows2000. The syntactic and real datasets are used in our experiment. The syntactic datasets are automatically generated by a generator tool from IBM Almaden with some slightly modified default values. Two real datasets from the UC Irvine Machine Learning Database Repository, i.e. mushroom and chess, are used with our own generated taxonomies. The original items contain in the leaf-level of taxonomy.

Table. 1 shows the comparison of using *SET* and *cSET* to enumerate all GFIs and GCFIs, respectively. In real datasets, the number of GCFIs is much smaller than that of GFIs. With the same datasets, the ratio of the number of GFIs to that of GCFIs typically increases when we lower minsup. The higher the ratio is, the more time reduction is gained. The ratio can grow up to around 7,915 times, which results in reduction of running time around 3,878 times. Note that in syntactic datasets, the number of GFIs is slightly different from the number of GCFIs. This indicates that the real datasets are dense but the syntactic datasets are sparse. This result makes us possible to reduce more computational time by using *cSET* in real situations.

Table 1. Number of itemsets and Execution Time (GFIs vs. GCFIs)



%Min sup	Number of Itemsets			Execution Time (sec)		
	#Freq	#Closed	≈Ratio	*SET*	*cSET*	≈Ratio
Database: Mushroom R45F3						
100	191	1	191	0.14	0.01	14
80	28127	84	335	4.59	0.06	77
60	204143	485	421	29.94	0.28	107
Database: Mushroom R30F5						
100	95	1	95	0.03	0.01	3
80	1839	18	102	0.31	0.02	16
60	19543	102	192	2.92	0.05	58
Database: Chess R30F5						
100	32767	10	3277	2.25	0.001	2250
95	585727	74	7915	38.78	0.01	3878
90	2565631	499	5142	175.78	0.06	2930
85	6843135	1181	5794	446.95	0.17	2629
Database: T10I100KD100K						
4	228	226	1.01	0.70	0.70	1
2	848	843	1.01	1.53	1.44	1.06
1	3235	3211	1.01	2.94	2.56	1.15
0.5	12737	12470	1.02	5.83	4.77	1.22
Database: T10I10KD100K						
1	2767	2754	1	2.73	2.44	1.12
Database: T20I100KD100K						
1	67796	67092	1.01	46.78	39.02	1.2

6 Conclusion and Further Research

A large number of generalized frequent itemsets may cause of high computational time. Instead of mining all generalized frequent itemsets, we can mine only a small set of generalized closed frequent itemsets and then result in reducing computational time. We proposed an algorithm, named *cSET*, by applying some constraints and conditional properties to efficiently enumerate only generalized closed frequent itemsets. The advantage of *cSET* becomes more dominant when minimum support is low and/or the dataset is dense. This approach makes us possible to mine the data in real situations. In further research, we intend to propose a method to extract only a set of important rules from these generalized closed frequent itemsets.

Acknowledgement

This paper has been supported by Thailand Research Fund (TRF), and NECTEC under project number NT-B-06-4C-13-508.

References

[1] Agrawal, R., Imielinski, T., Swami, A. N.: Mining association rules between sets of items in large databases. In Buneman, P., Jajodia, S., eds.: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D. C. (1993) 207–216 476

[2] Srikant, R., Agrawal, R.: Mining generalized association rules. Future Generation Computer Systems **13** (1997) 161–180 476

[3] Hipp, J., Myka, A., Wirth, R., Güntzer, U.: A new algorithm for faster mining of generalized association rules. In: Proceedings of the 2nd European Conference on Principles of Data Mining and Knowledge Discovery (PKDD '98), Nantes, France (1998) 74–82 476

[4] Chung, F., Lui, C.: A post-analysis framework for mining generalized association rules with multiple minimum supports (2000) Workshop Notes of KDD'2000 Workshop on Post-Processing in Machine Learing and Data Mining 477

[5] Han, J., Fu, Y.: Mining multiple-level association rules in large databases. Knowledge and Data Engineering **11** (1999) 798–804 477

[6] Lui, C. L., Chung, F. L.: Discovery of generalized association rules with multiple minimum supports. In: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD '2000), Lyon, France (2000) 510–515 477

[7] Shintani, T., Kitsuregawa, M.: Parallel mining algorithms for generalized association rules with classification hierarchy. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data. (1998) 25–36 477

[8] Michail, A.: Data mining library reuse patterns using generalized association rules. In: International Conference on Software Engineering. (2000) 167–176 477

[9] Hwang, S. Y., Lim, E. P.: A data mining approach to new library book recommendations. In: Lecture Notes in Computer Science ICADL 2002, Singapore (2002) 229–240 477

[10] Sriphaew, K., Theeramunkong, T.: A new method for fiding generalized frequent itemsets in generalized association rule mining. In Corradi, A., Daneshmand, M., eds.: Proc. of the Seventh International Symposium on Computers and Communications, Taormina-Giardini Naxos, Italy (2002) 1040–1045 477, 480, 482

[11] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. Lecture Notes in Computer Science **1540** (1999) 398–416 477, 478, 479

[12] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. Information Systems **24** (1999) 25–46 477, 478

[13] Zaki, M. J., Hsiao, C. J.: CHARM: An efficient algorithm for closed itemset mining. In Grossman, R., Han, J., Kumar, V., Mannila, H., Motwani, R., eds.: Proceedings of the Second SIAM International Conference on Data Mining, Arlington VA (2002) 477, 478