Mining Frequent Weighted Closed Itemsets

Bay Vo[1], Nhu-Y Tran[2], and Duong-Ha Ngo[2]

[1] Information Technology College, Ho Chi Minh City, Viet Nam
vdbay@itc.edu.vn
[2] Department of Information Technology, Ho Chi Minh City of Food Industry, Viet Nam
{ytn,hand}@cntp.edu.vn

Abstract. Mining frequent itemsets plays an important role in mining association rules. One of methods for mining frequent itemsets is mining frequent weighted itemsets (FWIs). However, the number of FWIs is often very large when the database is large. Besides, FWIs will generate a lot of rules and some of them are redundant. In this paper, a method for mining frequent weighted closed itemsets (FWCIs) in weighted items transaction databases is proposed. Some theorems are derived first, and based on them, an algorithm for mining FWCIs is proposed. Experimental results show that the number of FWCIs is always smaller than that of FWIs and the mining time is also better.

Keywords: data mining, frequent weighted support, frequent weighted itemsets, frequent weighted closed itemsets.

1 Introduction

Classical association rule mining does not take into consideration the relative benefit of items. However, in some applications, we are interested in relative benefit (weighted value) associated with each item. For example, suppose bread incurs profit 20 cents and a bottle of milk 40 cents. It is thus desirable to identify new methods for applying Association Rule Mining (ARM) techniques to this kind data so that such relative benefits are taken into account.

In 1998, Ramkumar et al. [12] proposed a model for describing the concept of Weighted Association Rules (WARs) and presented an Apriori-based algorithm for mining Frequent Weighted Itemsets (FWIs). Some further ideas concerning with the mining of WARs are also discussed in [2]. Since then many Weighted Association Rule Mining (WARM) techniques have been proposed [14, 17, 20]. Khan et al. [4] extended the concept of WARM and proposed a method for mining fuzzy WARs.

The advantage offered by adopting FWCI mining is that there are fewer FWCI's than frequent itemsets, and hence computational efficiencies may be introduced. However, for mining efficient association rules, we need mine FWCIs [1, 9-10, 16, 21-22]. This paper proposes a technique for mining FWCIs. Some theorems are also proposed. Based on them and WIT-tree [5-6, 17], we develop an algorithm for fast mining FWCIs.

N.T. Nguyen, T. Van Do, and H.A. Le Thi (Eds.): *ICCSAMA 2013*, SCI 479, pp. 379–390.
DOI: 10.1007/978-3-319-00293-4_29

The rest of this paper is organized as follows. Section 2 presents some related work concerning the mining of FWIs and WARs. Section 3 presents a proposed modification of WIT-tree [5-6, 17] for compressing the database into a tree structure. An algorithm for mining FWCIs using WIT-trees is discussed in section 4. Experimental results are presented in section 5. Conclusions and future work will present in section 6.

2 Related Work

2.1 Weighted Items Transaction Databases

A weighted items transaction database (*D*) is defined as follows: *D* comprises a set of transactions $T = \{t_1, t_2, ..., t_m\}$, a set of items $I = \{i_1, i_2, ..., i_n\}$ and a set of positive weights $W = \{w_1, w_2, ..., w_n\}$ corresponding to each item in *I*. For example, consider the data presented in Table 1 and Table 2. Table 1 presents a data set comprising six transactions $T = \{t_1, \ldots, t_6\}$, and five items $I = \{A, B, C, D, E\}$. The weights of these items are presented in Table 2, $W = \{0.6, 0.1, 0.3, 0.9, 0.2\}$.

Table 1. The transaction database

Transactions	Bought items
1	*A, B, D, E*
2	*B, C, E*
3	*A, B, D, E*
4	*A, B, C, E*
5	*A, B, C, D, E*
6	*B, C, D*

Table 2. Item weights

Items	Weight
A	0.6
B	0.1
C	0.3
D	0.9
E	0.2

a) Galois connection

Let $\delta \subseteq I \times T$ be a binary relation, where *I* is a set of items and *T* is a set of transactions contained in the database *D*. Let $X \subseteq I$ and $Y \subseteq T$. Let $\boldsymbol{P}(\boldsymbol{S})$ include all subsets of $\boldsymbol{S}$. Two mappings between $\boldsymbol{P}(\boldsymbol{I})$ and $\boldsymbol{P}(\boldsymbol{T})$ are called Galois connections as follows [22]:

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

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

The mapping $t(X)$ is the set of transactions in the database which contain *X*, and the mapping $i(Y)$ is an itemset that is contained in all the transactions *Y*.

Given X, X_1, $X_2 \in P(I)$ and Y, Y_1, $Y_2 \in P(T)$. The Galois connection satisfies the following properties [22]:

i) $X_1 \subset X_2 \Rightarrow t(X_1) \supseteq t(X_2)$

ii) $Y_1 \subset Y_2 \Rightarrow i(Y_1) \supseteq i(Y_2)$

iii) $X \subseteq i(t(X))$ and $Y \subseteq t(i(Y))$

b)Mining weighted association rules [17]

Definition 1. The transaction weight (tw) of a transaction t_k is defined as follow:

$$tw(t_k) = \frac{\sum_{i_j \in t_k} w_j}{|t_k|}$$

Definition 2. The weighted support of an itemset is defined as follow:

$$ws(X) = \frac{\sum_{t_k \in t(X)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)}$$

where T is the list of transactions in the database.

Example 1. Consider tables 1, 2, and definition 1, we can compute the $tw(t_1)$ value as follow:

$$tw(t_1) = \frac{0.6 + 0.1 + 0.9 + 0.2}{4} = 0.45$$

Table 3 shows all tw values of transactions in Table 1.

Table 3. Transaction weights for transactions in Table 1

Transactions	*tw*
1	*0.45*
2	*0.2*
3	*0.45*
4	*0.3*
5	*0.42*
6	*0.43*
Sum	*2.25*

From tables 1, 3, and definition 2, we can compute the $ws(BD)$ value as follow:

Because BD appears in transactions {1, 3, 5, 6}, $ws(BD)$ is computed:

$$ws(BD) = \frac{0.45 + 0.45 + 0.42 + 0.43}{2.25} \approx 0.78$$

2.2 Mining Frequent Closed Itemsets

An itemset X is called a frequent closed itemset if it is frequent, and it does not exist any frequent itemset Y such that $X \subset Y$ and $\sigma(X) = \sigma(Y)$. There are many methods proposed for mining frequent closed itemsets (FCIs) from data. They are divided into the following four categories [18]:

i) **Generate-and-test:** These methods are founded on the Apriori algorithm that uses a level-wise approach to discover FCIs. Some example algorithms include Close [10] and A-Close [9].

ii) **Divide-and-conquer:** These methods adopt a divide-and-conquer strategy and use compact data structures extended from a frequent-pattern (FP) tree to mine FCIs. Example algorithms include Closet [11], Closet+ [19] and FPClose [3].

iii) **Hybrid approaches:** These methods integrate both of the above two strategies to mine FCIs, which first transform the data into a vertical data format, and then develop properties and use a hash-table to prune non-closed itemsets. Example methods that use the hybrid approach include CHARM [23] and CloseMiner [13] also belong to them.

iv) **Hybrid approaches without duplication:** These methods differ from those using the hybrid approach in that they do not use the subsumption-checking technique, so identified FCIs need not be stored in the main memory. Methods within this category also do not use the hash-table technique as in the case of CHARM [23]. Example algorithms include DCI-Close 7], LCM [15] and PGMiner [8].

3 WIT-Tree Data Structure

In [5], authors proposed the WIT-tree (Weighted Itemset-Tidset tree) data structure, an expansion of the IT-tree proposed in [22], to mine high utility itemsets. To mine FWCIs, we modify the WIT-tree by changing *twu* to *ws* property. Using the WIT-tree, our proposed algorithm (see Section 4) only scans the data once because it is based on the intersection of Tidsets to compute the weighted support in next steps. Thus, it saves the time for the database scan and makes the algorithm to be done faster.

Each vertex in a WIT tree includes 3 fields:

i. X: an itemset.
ii. $t(X)$: the set of transaction contains X.
iii. ws: the weighted support of X.

The vertex is denoted $X \times_{ws(X)} t(X)$.

The value of $ws(X)$ is computed by summing all tw values of transactions, $t(X)$, which their *tids* belong to and then dividing this by the sum of all tw values. Thus, computing of $ws(X)$ is achieved using the Tidset. Arcs connect vertices at k^{th} level (called X) with vertices at the $(k+1)^{th}$ level (called Y).

Definition 3. [23] – The equivalence class

Let I be a set of items and $X \subseteq I$, a function $\boldsymbol{p(X,k) = X[1:k]}$ as the k length prefix of X and a prefix-based equivalence relation θ_K on itemsets as follows: $\forall X,Y \subseteq I, X \equiv_{\theta_k} Y \Leftrightarrow p(X,k) = p(Y,k)$.

The set of all itemsets which having the same prefix X is called an equivalence class, and denoted as the equivalence class with prefix X is $[X]$.

Example 2: Consider tables 1 and 3 above, the associated WIT-tree for mining frequent weighted itemsets is as follows: The root node of the WIT-tree contains all 1-itemset nodes. All nodes at level 1 belong to the same equivalence class with prefix $\{\}$ (or $[\varnothing]$). Each node at level 1 will become a new equivalence class using its item as the prefix. With each node in the same prefix, it will join with all nodes following it to create a new equivalence class. The process will be done recursively to create new equivalence classes in the higher levels. For example, nodes $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{E\}$ belong to the equivalence class $[\varnothing]$. Consider node $\{A\}$, this node will join with all nodes following it ($\{B\}$, $\{C\}$, $\{D\}$, $\{E\}$) to create a new equivalence class $[A] = \{\{AB\}, \{AC\}, \{AD\}, \{AE\}\}$. $[AB]$ will become a new equivalence class by also joining with all nodes following it ($\{AC\}$, $\{AD\}$, $\{AE\}$); and so on.

We can see [17] for more details about WIT-tree applying mining FWIs.

4 Mining Frequent Weighted Closed Itemsets

Definition 4: Let $X \subseteq I$ be a frequent weighted itemset, X is called a frequent weighted closed itemset if and only if it does not exist the frequent weighted itemset Y such that $X \subset Y$ and $ws(X) = ws(Y)$.

From the definition 4, there are a lot of FWIs that are not closed. For example, A, AB, AE are not closed because ABE has the same ws values with them. The purpose of the section is mining **FWCIs** from weighted items transaction databases fast.

Theorem 1. Given two itemsets X, Y where $X \subseteq Y$, $ws(X) = ws(Y) \Leftrightarrow t(X) = t(Y)$.

Proof:

✦ if $ws(X) = ws(Y)$ then $t(X) = t(Y)$

We have $ws(X) = \frac{\sum_{t_k \in t(X)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)}$, $ws(Y) = \frac{\sum_{t_k \in t(Y)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)}$ and $ws(X) = ws(Y)$

$$\Rightarrow \frac{\sum_{t_k \in t(X)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)} = \frac{\sum_{t_k \in t(Y)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)}$$

$$\Rightarrow \sum_{t_k \in t(X)} tw(t_k) = \sum_{t_k \in t(Y)} tw(t_k) \quad (1)$$

According to property i) of Galois connection, we also have $X \subseteq Y \Rightarrow t(X) \supseteq t(Y)$

It impies that $\sum_{t_k \in t(X)} tw(t_k) = \sum_{t_k \in t(Y)} tw(t_k) + \sum_{t_k \in t(X) \setminus t(Y)} tw(t_k)$ (2)

From (1) and (2) we have $\sum_{t_k \in t(Y)} tw(t_k) + \sum_{t_k \in t(X) \setminus t(Y)} tw(t_k) = \sum_{t_k \in t(Y)} tw(t_k)$

$\Rightarrow \sum_{t_k \in t(X)-t(Y)} tw(t_k) = 0$

$\Rightarrow t(X) \setminus t(Y) = \emptyset$ (Because $tw(t_k) > 0$).

$\Rightarrow t(X) \subseteq t(Y)$ *or* $t(X) = t(Y)$.

✦ *if* $t(X) = t(Y)$ then $ws(X) = ws(Y)$

According to Theorem 4.1 [17].

By theorem 1, we can use tidset to check the itemset is closed or not.

Theorem 2. Let $X \times_{ws(X)} t(X)$ and $Y \times_{ws(Y)} t(Y)$ are two nodes in the equivalence class [*P*], we have:

i) If $t(X) = t(Y)$ then *X*, *Y* are not closed.

ii) If $t(X) \subset t(Y)$ then *X* is not closed.

iii) If $t(X) \supset t(Y)$ then *Y* is not closed.

Proof:

i) We have $t(X \cup Y) = t(X) \cap t(Y) = t(X) = t(Y)$ (because $t(X) = t(Y)$) $\Rightarrow$ according to theorem 1, we have $ws(X) = ws(Y) = ws(X \cup Y) \Rightarrow X$ and *Y* are not closed.

ii) We have $t(X \cup Y) = t(X) \cap t(Y) = t(X)$ (because $t(X) \subset t(Y)$) $\Rightarrow$ according to theorem 1, we have $ws(X) = ws(X \cup Y) \Rightarrow X$ is not closed.

iii) We have $t(X \cup Y) = t(X) \cap t(Y) = t(Y)$ (because $t(X) \supset t(Y)$) $\Rightarrow$ according to theorem 1, we have $ws(Y) = ws(X \cup Y) \Rightarrow Y$ is not closed.

When we sort nodes in equivalence class P by increasing order according to cardinality of tidset, condition iii) of theorem 2 will not occur, so that we only consider conditions i) and ii).

In the process of mining **FWCIs**, considering nodes in the same equivalence class will consume a lot of time. Thus, we need to group nodes that satisfy condition i) at level 1 of WIT-tree together. In the process of creating a new equivalence class, we will group nodes that satisfy condition i) also. This reduces the cardinality of the equivalence class, so that the mining time decreases significantly. This approach differs from Zaki's approach [23] in that it decreases significantly the number of nodes, and it need not to remove the nodes that satisfy condition i) in an equivalence class.

4.1 The Algorithm

Input: A database *D* and *minws*

Output: **FWCIs** contains all frequent weighted closed itemsets that satisfy *minws* from *D*.

Method:

WIT-FWCIs()

1. $[\varnothing] = \{i \in I: ws(i) \geq minws\}$

2. **FWCIs** = ∅
3. **SORT**([∅])
4. **GROUP**([∅])
5. **FWCIs-EXTEND** ([∅])

FWCIs-EXTEND([*P*])
6. for all $l_i \in [P]$ do
7. $[P_i] = \emptyset$
8. for all $l_j \in [P]$, with j > i do
9. *if* $t(l_i) \subset t(l_j)$ then
10. $l_i = l_i \cup l_j$
11. *else*
12. $X = l_i \cup l_j$
13. $Y = t(l_i) \cap t(l_j)$
14. $ws(X)$ = **COMPUTE-WS** (Y)
15. if $ws(X) \geq minws$ then
16. if $X \times Y$ is not subsumed then
17. Add { $\underset{ws(X)}{X \times Y}$ } to $[P_i]$ //sort in increasing by $|Y|$
18. Add (l_i, $ws(l_i)$) to **FWCIs** if it is closed
19. **FWCIs-EXTEND** ($[P_i]$)

Fig. 1. WIT-FWCIs algorithm for mining frequent weighted closed itemsets

The algorithm (in Fugure 1) commences with an empty equivalence class which contains simple items with their *ws* values satisfying *minws* (line 1). The algorithm then sorts nodes in equivalence class [∅] by increasing order according to cardinality of tidset (line 3). After that, it groups all nodes which have the same *tids* into the unique node (line 4), and calls procedure **FWCIs-EXTEND** with parameter [∅] (line 5). Procedure **FWCIs-EXTEND** uses equivalence class [*P*] as an input value, it considers each node in the equivalence class [*P*] with equivalence classes following it (lines 6 and 8). With each pair l_i and l_j, the algorithm considers condition ii) of theorem 2, if it satisfies (line 9) then the algorithm replaces equivalence class $[l_i]$ by $[l_i \cup l_j]$ (line 10), otherwise the algorithm creates a new node and adds it into equivalence class $[P_i]$ (initially it is assigned by empty value, line 7). When tidset of X (i.e., Y) is identified, we need to check whether it is subsumed by any node or not (line 16), if not, then it is added into [*P*]. Adding a node $\underset{ws(X)}{X \times Y}$ into $[P_i]$ is performed similarly as level 1 (i.e., consider it with nodes in $[P_i]$, if exists the node that has the same tidset, then they are grouped together, line 17). After consider l_i with all nodes following it, the algorithm will add l_i and its *ws* into **FWCIs** (line 18). Finally, the algorithm is called recursively to generate equivalence classes after $[l_i]$ (line 19).

Two nodes in the same equivalence class do not satisfy condition i) of theorem 2 because the algorithm groups these nodes into one node whenever they are added into [*P*]. Similarly, condition iii) does not occur because the nodes in the equivalence class [*P*] are sorted according to increasing order of cardinality of tidset.

4.2 Example

Using the example data presented in Tables 1 and 3, we illustrate the **WIT-FWCIs** algorithm with *minws* = 0.4 as follows. First of all, [∅] = {*A*, *B*, *C*, *D*, *E*}. After sorting

and grouping, we have the result as [∅] = {*C*, *D*, *A*, *E*, *B*}. Then, the algorithm calls the function **FWCIs-EXTEND** with input nodes {*C*, *D*, *A*, *E*, *B*}.

With the equivalence class [*C*]:

Consider *C* with *D*: we have a new itemset *CD*×56 with *ws*(*CD*) = 0.38 < *minws*.

Consider *C* with *A*: we have a new itemset *CA*×45 with *ws*(*CA*) = 0.32 < *minws*.

Consider *C* with *E*: we have a new itemset *CE*×245 with *ws*(*CE*) = 0.41 ⇒ [*C*] = {*CE*}.

Consider *C* with *B*: we have *t*(*C*) ⊂ *t*(*B*) (satisfy the condition ii) of theorem 2) ⇒ Replace [*C*] by [*CB*]. It means that all equivalence classes following [*C*] are replaced *C* into *CB*. Therefore, [*CE*] is replaced into [*CBE*].

After making the equivalence class [*C*] (become [*CB*] now), *CB* is added to **FWCIs** ⇒ **FWCIs** = {*CB*}. The algorithm will be called recursively to create all equivalence classes following it.

Consider the equivalence class [*CBE*] ∈ [*CB*]: Add *CBE* to **FWCIs** ⇒ **FWCIs** = {*CB*, *CBE*}.

With the equivalence class [*D*]:

Consider *D* with *A*: we have a new itemset *DA*×135 with *ws*(*DA*) = 0.59 ⇒ [*D*] = {*DA*}.

Consider *D* with *E*: we have a new itemset *DE*×135 ⇒ Group *DA* with *DE* into *DAE* ⇒ [*D*] = {*DAE*}.

Consider *D* with *B*: we have *t*(*D*) ⊂ *t*(*B*) (satisfy the condition ii) of theorem 2) ⇒ Replace [*D*] by [*DB*]. It means that all equivalence classes following [*D*] are replaced *D* into *DB*. Therefore, [*DAE*] is replaced into [*DBAE*].

After making the equivalence class [*D*] (become [*DB*] now), *DB* is added to **FWCIs** ⇒ **FWCIs** = {*CB*, *CBE, DB*}. The algorithm will be called recursively to create all equivalence classes following it.

Consider the equivalence class [*DBAE*] ∈ [*DB*]: Add *DBAE* to **FWCIs** ⇒ **FWCIs** = {*CB*, *CBE, DB, DBAE*}.

Similar to equivalence classes [*A*], [*E*], [*B*]. We have all **FWCIs** = {*CB*, *CBE, DB, DBAE, AEB, EB, B*}.

Results show that the number of FWCIs is smaller than FWIs (7 compare to 19), and the number of search levels in a tree by WIT-FWCIs is also less than that of WIT-FWIs (2 compared with 4). Thus, we can say that FWCIs mining is more efficient than FWIs.

5 Experimental Results

All experiments described below were performed on a Centrino core 2 duo (2×2.53 GHz), 4GBs RAM memory, Windows 7, using C# 2008. The experimental data sets used for the experimentation were downloaded from http://fimi.cs.helsinki.fi/data/. Some statistical information regarding these data sets is given in Table 4.

Table 4. Experimental databases

Databases (DB)	#Trans	#Items	Remark
Mushroom	8124	120	Modified
Connect	67557	130	Modified

Each database is modified by creating a table to store weighted values of items (value in the range of 1 to 10).

5.1 Number of Itemsets

Results from Table 5 show that the number of FWCIs is always smaller than FWIs. For example, consider Mushroom database with *minws* = 20%, the number of FWIs is 53,513 and the number of FWCIs is 1199, the ratio is 1199/53513 × 100% ≈ 2.24%.

Table 5. Compare the number of FWCIs with the number of FWIs

Databases	*minws*(%)	#FWIs	#FWCIs
Connect	97	512	297
	96	1147	513
	95	2395	861
	94	4483	1284
Mushroom	35	1159	252
	30	2713	423
	25	5643	696
	20	53513	1199

5.2 The Mining Time

Experimental results from Figures 2 and 3 show the efficiency of FWCIs mining. The time of FWCIs mining is faster than FWIs mining in these two databases. Especially, when *minws* is low, mining FWCIs is more efficient than mining FWIs. For example, consider Mushroom database, when *minws* = 35%, the time for mining FWIs is 0.284(s) while the time for mining FWCIs is 0.15(s). When we decrease *minws* to 20%, the time for mining FWIs is 5.88(s) while the time for minng FWCIs is 0.541(s).

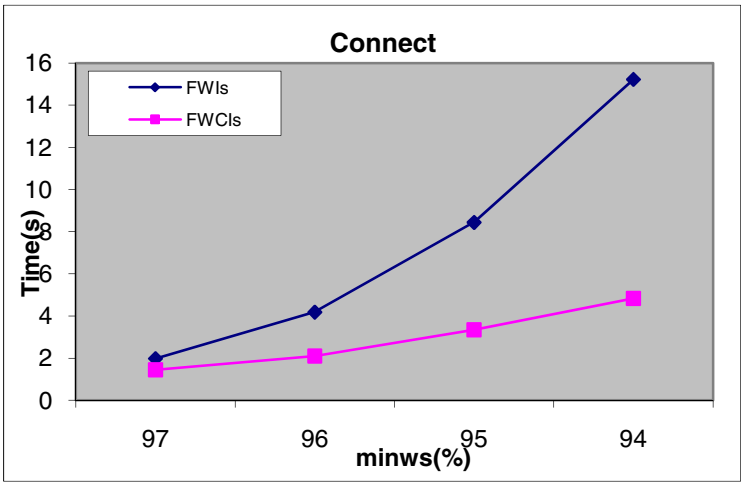

Fig. 2. Compare the run time of FWIs and FWCIs in Connect database

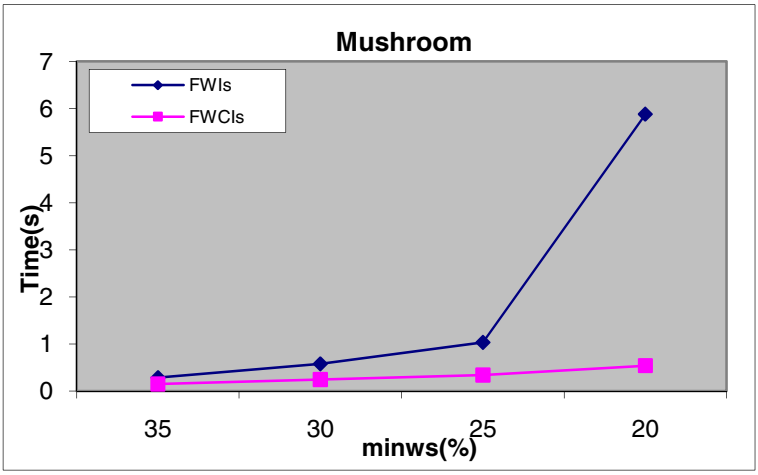

Fig. 3. Compare the run time of FWIs and FWCIs in Mushroom database

6 Conclusions and Future Work

This paper has proposed the method for mining frequent weighted itemsets and frequent weighted closed itemsets from weighted items transaction databases, and the efficient algorithm is also developed. As above mentioned, the number of FWCIs is always smaller than that of FWIs and the mining time is also better. By WIT-tree, the proposed algorithm only scans the database one and uses tidset to fast determine the weighted support of itemsets.

In the future, we will study how to mine efficient association rules from frequent weighted closed itemsets.

Acknowledgment. This work was supported by Vietnam's National Foundation for Science and Technology Development (NAFOSTED) under Grant Number 102.01-2012.47.

References

[1] Bastide, Y., Pasquier, N., Taouil, R., Stumme, G., Lakhal, L.: Mining minimal non-redundant association rules using frequent closed itemsets. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 972–986. Springer, Heidelberg (2000)

[2] Cai, C.H., Fu, A.W., Cheng, C.H., Kwong, W.W.: Mining Association Rules with Weighted Items. In: Proceedings of International Database Engineering and Applications Symposium (IDEAS 1998), pp. 68–77 (1998)

[3] Grahne, G., Zhu, J.: Fast Algorithms for Frequent Itemset Mining Using FP-Trees. IEEE Transaction on Knowledge and Data Engineering 17(10), 1347–1362 (2005)

[4] Khan, M.S., Muyeba, M., Coenen, F.: A Weighted Utility Framework for Mining Association Rules. In: Proceedings of Second UKSIM European Symposium on Computer Modeling and Simulation Second UKSIM European Symposium on Computer Modeling and Simulation, pp. 87–92 (2008)

[5] Le, B., Nguyen, H., Cao, T.A., Vo, B.: A novel algorithm for mining high utility itemsets. In: The First Asian Conference on Intelligent Information and Database Systems, pp. 13–16 (2009) (published by IEEE)

[6] Le, B., Nguyen, H., Vo, B.: An Efficient Strategy for Mining High Utility Itemsets. International Journal of Intelligent Information and Database Systems 5(2), 164–176 (2011)

[7] Lucchese, B., Orlando, S., Perego, R.: Fast and Memory Efficient Mining of Frequent Closed Itemsets. IEEE Transaction on Knowledge and Data Engineering 18(1), 21–36 (2006)

[8] Moonestinghe, H.D.K., Fodeh, S., Tan, P.N.: Frequent Closed Itemsets Mining using Prefix Graphs with an Efficient Flow-based Pruning Strategy. In: Proceedings of 6th ICDM, Hong Kong, pp. 426–435 (2006)

[9] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering Frequent Closed Itemsets for Association Rules. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 398–416. Springer, Heidelberg (1999)

[10] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of Association Rules using Closed Itemset Lattices. Information Systems 24(1), 25–46 (1999)

[11] Pei, J., Han, J., Mao, R.: CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. In: Proc. of the 5th ACM-SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Dallas, Texas, USA, pp. 11–20 (2000)

[12] Ramkumar, G.D., Ranka, S., Tsur, S.: Weighted Association Rules: Model and Algorithm. In: SIGKDD 1998, pp. 661–666 (1998)

[13] Singh, N.G., Singh, S.R., Mahanta, A.K.: CloseMiner: Discovering Frequent Closed Itemsets using Frequent Closed Tidsets. In: Proc. of the 5th ICDM, Washington DC, USA, pp. 633–636 (2005)

[14] Tao, F., Murtagh, F., Farid, M.: Weighted Association Rule Mining using Weighted Support and Significance Framework. In: SIGKDD 2003, pp. 661–666 (2003)

[15] Uno, T., Asai, T., Uchida, Y., Arimura, H.: An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases. In: Suzuki, E., Arikawa, S. (eds.) DS 2004. LNCS (LNAI), vol. 3245, pp. 16–31. Springer, Heidelberg (2004)

[16] Vo, B., Le, B.: Mining Minimal Non-redundant Association Rules using Frequent Itemsets Lattice. Journal of Intelligent Systems Technology and Applications 10(1), 92–106 (2011)

[17] Vo, B., Coenen, F., Le, B.: A new method for mining Frequent Weighted Itemsets based on WIT-trees. Expert Systems with Applications 40(4), 1256–1264 (2013)

[18] Vo, B., Hong, T.P., Le, B.: Mining most generalization association rules based on frequent closed itemsets. Int. J. of Innovative Computing Information and Control 8(10B), 7117–7132 (2012)

[19] Wang, J., Han, J., Pei, J.: CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 236–245 (2003)

[20] Wang, W., Yang, J., Yu, P.S.: Efficient Mining of Weighted Association Rules. In: SIGKDD 2003, pp. 270–274 (2003)

[21] Zaki, M.J.: Generating Non-Redundant Association Rules. In: Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, Massachusetts, USA, pp. 34–43 (2000)

[22] Zaki, M.J.: Mining Non-Redundant Association Rules. Data Mining and Knowledge Discovery 9(3), 223–248 (2004)

[23] Zaki, M.J., Hsiao, C.J.: Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure. IEEE Transactions on Knowledge and Data Engineering 17(4), 462–478 (2005)