Journal of mathematics and computer science 8 (2014), 282-296

Contents list available at JMCS
Journal of Mathematics and Computer Science
Journal Homepage: www.tjmcs.com

A Method for Reducing Repetitive items on Weighted Data using the WIT-WFI Algorithm

Akbar Rashidi[1] , Arash Ghorbannia Delavar[2]

Department of Computer,Payame Noor University,
PO BOX 19395-3697, Tehran, IRAN

rashidi.igs@gmail.com [1], a_ghorbannia@pnu.ac.ir[2]

Article history:
Received August 2013
Accepted September 2013
Available online September 2013

Abstract

Trying and mining frequent item sets plays an important role in the mining of association rules .in a dataset that stored with items and transactions an items can used for various significance .association rules is a important and considerable ways in data mining without presidency . one of discussion that today investigate is mining and finding frequent weighted item set and reduce run time of algorithm and reduce production frequent item sets is one of problem for research .at this paper we purpose present some method and ways for reduce run time of algorithm and reduce production frequent item set .all methods and ways applying on WIT algorithm and WIT-Tree structure .in first section we express and description classic association rule method (Apriori) and then WIT and then WIT-Diff algorithm and finally explain my proposed ways and experimental results.

Keywords: Data mining – Frequent items – Weighted item sets – WIT- Tree – Association rules.

1. Introduction

Association Rules Mining (ARM) is an important part in the domain of knowledge discovery in data (KDD) [1,2]. Association rule mining is used for finding and mining frequent patterns and relationship between transactions in a database or dataset. Association rules used of presidency learning principle, that purpose the this principle is obvious and we understand that research what knowledge unlike without presidency way than result and purpose the mining not clear. Given a set of items $I = \{i1, i2,. . . , in\}$, a transaction is defined as a subset of I. The input to an ARM algorithm is a dataset D comprising a set of transactions. Given an item set $X \subseteq I$, the support of X in D, denoted as $\sigma(X)$, is the number of transactions in D which contain X [18]. An item set is described as being frequent if its support is larger

than or equal to a user supplied minimum support threshold (min Sup). A ‘‘classical’’ Association Rule (AR) is an expression of the form {X→Y (sup, conf)}, where X, Y ⊆ I and X∩Y = Ø. The support of this rule is sup = σ(XY) and the confidence is conf= $\frac{\sigma(XY)}{\sigma(X)}$. Given a specific min Sup and a minimum confidence threshold (min Conf), we want to mine all association rules whose support and confidence exceeds min Sup and min Conf respectively [3, 6].
However, Classical association rule have some problem that very great run time and many scan of database for finding item sets .if we add computation time of items with weight as time of algorithm raising .purpose the this paper is expansion WIT algorithm for mining frequent items in view run time and reduce product item set. The rest of this paper is organized as follows. Section 2 presents some related work about the mining of frequent weighted items and weighted association rules and some terms and equations. Section 3 we explain WIT-Tree structure and in Section 4 explain WIT algorithm .in Section 5 explain and descript WIT-DIFF algorithm .in Section 6 explain and descript my proposal methods. Some experimental results are present in Section 7 .and my conclusion in Section 8.

2. Related works

This section presents some related works. The section begins with a formal definition of weighted transaction databases. A weighted transaction database (D) is defined as follows: D comprises a set of transactions $T=\{t_1,t_2,,,t_n\}$, 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 Tables 1 and 2. Table 1 presents a data set comprising six transactions $T=\{t_1,t_2\ldots t_6\}$ 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} [4,5].

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: Items weight

Items weight.

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

The equations that we used in this paper consist:

- Calculation weight of transaction (tw[1]).

[1] Transaction weight

- Calculation weight of items or weighted support (ws).

For acquire weight of a transaction we must sum of all items weight in transaction and then calculate average of items with divide sum of items in count of items in each transactions. See definition 2.1, we can compute the transaction weight [7, 8].

$$Tw_{(t_k)} = \frac{\sum i_j \epsilon t_k W_j}{|t_k|} \qquad 2.1$$

t_k: Transaction k.

i_j: j th items in transaction.

$|t_k|$: Size of transaction k, count of items .

w_j: Weight of j th item.

For compute weighted support must compute sum of transaction weight (table 3) and divide to sum of transaction. See definition 2.2, we can compute the transaction weight.

$$WS(X) = \frac{\sum_{t_k \epsilon t(x)} tw(t_k)}{\sum_{t_k \epsilon T} tw(t_k)} \qquad 2.2$$

X: The item.

$tw(t_k)$: Weight of transaction t_k .

t(X): Transaction consist item (X).

T: total of dataset.

Table 3: Transaction weight for transaction in table 1

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

Example for calculation transaction weight and weighted Support. For transaction weight used of table 1 , table 2 and definition 2.1, we can compute the transaction weight:

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

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

From Tables 1 and 3, and Definition 2.2, we can compute the ws(BD) value as follows: Because BD appears in transactions {1, 3, 5, 6}, ws(BD) is computed:

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

From equation 2.1 and 2.2 using in all algorithm and my proposal methods .in all methods a input value (threshold) used for filter algorithm input, we used this value for comparison and evaluation run time of algorithm and methods. The mining of FWI requires identification all item sets whose weighted support satisfies a user specified minimum weighted support threshold (minws[2]).

$$FWI=\{X\subseteq I|WS(X)\geq minWS\} \qquad 2.3$$

3. WIT-Tree structure

The structure that we used to explain my methods is WIT structure .for description this structure we first explain some terms [9].

X: set of items.

t(X): Set of transaction contain item(X).

ws: Value of weighted support for item(X).

for show a node of my tree used <X,T(x),ws> style. In this style x is my item and T(x) is transactions id which item X member of them. And ws is value of weighted support. For more description see Figure 1.

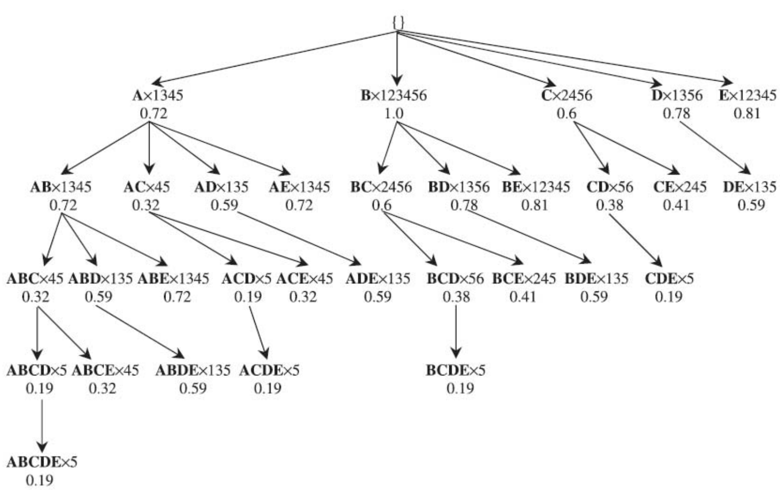


Figure 1: WIT-Tree structure example

In fig 1 level ∅ or root display with two accolade { } .if items member of a transaction consist same prefix they are called equivalence class items and display them with [ ] ,for example if item X is prefix item in some items we illustrate it with [X].in root level my prefix item is ∅ and equivalence class is

[2] Minimum weighted support

[∅].in this figure we see for all levels and items we compute transaction weight and weighted support. For compute number of all items in dataset used of 2^k-1 equation that variable k is all items in dataset and too for number of items in each level we used of combination equation $\binom{k}{L}$ .variable k is number of items and variable L is level numeral [2,19].
In this structure for create next level must used of two top level items or parent item. For example level 1 on fig 1 consist {A,B,C,D,E} items and a part of level 2 consist items with equivalence class [A] ,{AB,AC,AD,AE} .this process resume for all items. Inspection of Fig. 1 suggests that all item sets satisfy the downward closure property. So, we can prune an equivalence class in the WIT-tree if its ws value does not satisfy the minws. For example, suppose that minwus = 0.4, because ws(ABC) = 0.32 < minws we can prune the equivalence class with the prefix ABC, i.e., all child nodes of ABC can be pruned. In nest sections explain WIT and WIT-Diff methods.

4. WIT algorithm

In classic association rule (Apriori) for mining frequent weighted items we must for all items compute transaction weight and ws and too scan dataset for transaction id and location of each item in dataset. But in WIT we proceed to reduce repetitive calculation and whereupon reducing run time of my algorithm and methods. For this purpose we explain some theorems. we propose algorithms for mining FWI from weighted transaction databases [11,12]. First an algorithm for directly mining FWI from WIT-trees is presented. It uses a minws threshold and the downward closure property to prune nodes that are not frequent. Some theorems are then derived and based on these theorems, an improved algorithm is proposed. Finally, the algorithm is further developed, by adopting a Diffset strategy to allow for fast computating the weighted support of item sets in a memory efficient way. If we have two item set X ,Y that transaction id of item set X equal item set Y, otherwise t(X)=t(y) .in result can deduction for two item set X and Y , value of Weighted support is equal. Or otherwise WS(X)=WS(Y).

$$\text{If} \quad t(X)=t(Y) \quad \text{Then} \quad WS(X)=WS(Y) \qquad 4.1$$

If item set X member of collection Y and too number of transaction common together equal, in result weighted support value of two item set X and Y is unify.

$$\text{if} \quad X\subseteq Y \quad \text{and} \quad |t(X)|=|t(Y)| \qquad 4.2$$
$$\text{then } WS(X)=WS(Y)$$

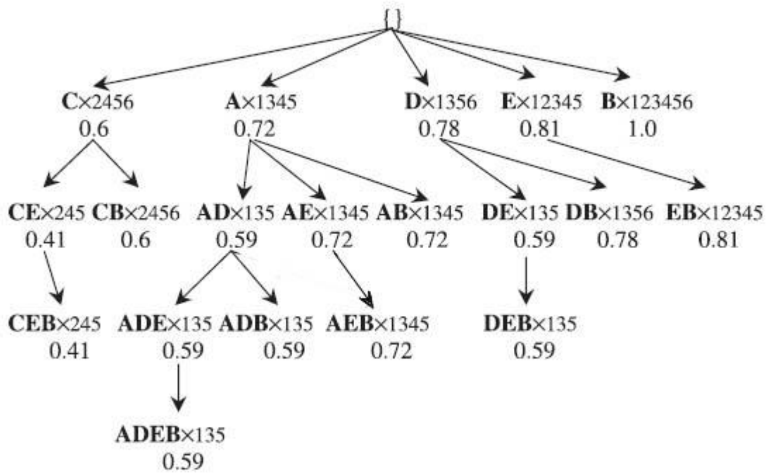


Figure 2: Example of WIT

For more description can see figures 2 and 3.

In Fig 2, first selection all items that satisfy min ws[3] and then sort them with increasing by their weighted support and set them in frequent weighted items (FWI) list. In line 4 of WIT algorithm Fig 3 call extend function for produce next item set from combination top level items. Too equation 4.1 and 4.2 used in 9 , 10 , 11 lines. With inspection In fig 2 for item <A, 1345, 0.72> and that combination with item set <B,123456,1> and item set <AB,1345,?> value of weighted support of item set AB with definition 4.1 and 4.2 equal with item A weighted support and not require for computing ws for item set AB .this process continue for next level and item sets to pending remain one item set in level.

Input: Database D and minimum weighted support threshold minws.
Output: FWI contains all frequent weighted Item sets that satisfy minws from D.

Method:
WIT ()

1. $L_{r=}$All items that their ws satisfy minws.
2. $sort$ Nodes in L_r increasing by their ws.
3. $FWI = \emptyset$.
4. $call$ Function FWI-Extend with the parameter is L_r.

FWI-Extend (L_r)

5. Consider each node l_i in L_1 DO.
6. Add($l_i.itemset.l_i.ws$) to FWI.
7. Create a new set L_i by join l_i with all l_j following it in $L_1$by:
8. Set X=$l_i.itemset \cup l_j.itemset$ and $Y = t(l_i) \cap t(l_j)$
9. If $|t(l_i)|$=$|Y|$ then ws(X)=ws(l_j)
10. Else if $|t(l_j)|$=$|Y|$ then ws(X)=ws(l_j)
11. If $Y = \emptyset$ $then$ $ws(X) = ws(l_i)$
12. Else $ws(X) = COMPUTE - WS(Y)$
13. if $ws(X)$ satisfies minws then

[3] Weighted Support

14. Add new Node $< X, Y, ws(X) >$ into L_i
15. $if\ number\ of\ nodes\ in\ L_i \geq 2\ then$
16. Call recursive the function FWI-Extend with the parameter is L_i

Figure 3: WIT algorithm

5. WIT-Diff algorithm

Proposed the Diffset[4] strategy for fast computing the support of item sets and saving memory to store Tidsets[5]. We recognize that it can be used for fast computing the ws values of item sets (1). Diffset computes the difference set between two Tidsets in the same equivalence class. In a dense database, the size of Diffset is smaller than the Tidset. Thus, using Diffset will consume less storage and allow for the fast computing of weighted support values. In this algorithm difference between PX and PY illustrated d(PXY) that X and Y are my items and P is prefix that illustrated equivalence class of items [13,15].(Fig 4)

$$d(pxy) = \frac{t(px)}{t(py)} \qquad 5.1$$

If have values of d(PX) and d(PY) and I will compute d(PXY) ,can used bellow equation:

$$\mathrm{d(pxy)} = \frac{\mathrm{d(py)}}{\mathrm{d(px)}} \qquad 5.2$$

Beneficial of equation 5.1 and 5.2 can reach an equation for calculate weighted support in this method:

$$\mathrm{WS(pxy)} = \mathrm{ws(px)} - \frac{\sum_{t \in d(pxy)} tw(t)}{\sum_{t \in T} tw(t)} \qquad 5.3$$

If value of d(PXY)=∅ then value of ws(PXY)=ws(PX) , this denote if d(PXY)=∅ then value of ws is equal parent ws value.

Input: Database D and minimum weighted support threshold minws.
Output: FWI contains all frequent weighted Itemsets that satisfy minws from D.

Method:
WIT-Diff()
1. $L_{r=}$All items that their ws satisfy minws.
2. $sort$ Nodes in L_r increasing by their tid.
3. $FWI = \emptyset$.
4. $call$ Function FWI-Extend-Diff with the parameter is L_r.

FWI-Extend-Diff (L_r)

5. Consider each node l_i in L_1 DO.
6. Add($l_i.itemset.l_i.ws$) to FWI.
7. Create a new set L_i by join l_i with all l_j following it in $L_1$by:
8. Set X=$l_i.itemset \cup l_j.itemset$

[4] Difference set
[5] Transaction id number sets

9. If L_r is the first Level Then $Y = \frac{t(l_i)}{t(l_j)}$
10. Else $Y = \frac{d(l_i)}{d(l_j)}$
11. If $Y = \emptyset\ then \quad ws(X) = ws(l_i)$
12. Else $ws(X) = COMPUTE - WS - DIFF(Y)$
13. $if\ ws(X)$ satisfies minws then
14. Add new Node $< X, Y, ws(X) >$ into L_i
15. $if\ number\ of\ nodes\ in\ L_i \geq 2\ then$
16. Call recursive the function FWI-Extend-Diff with the parameter is L_i

Figure 4: WIT-Diff algorithm

For description example of this algorithm used of table 1, table 2 and fig 5.for compute difference between item B and D we must compute d(BD):

$$d(BD)=\frac{t(B)}{t(D)} \qquad 5.4$$

That t(B)=123456 and t(d)=1356 then difference of two items is 24 , while that result of unlike combination of item B and item D is null.

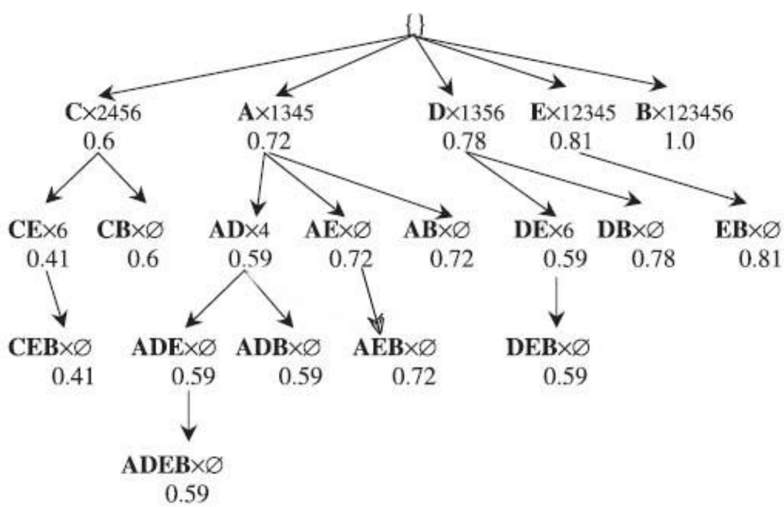


Figure 5: WIT-Diff example

Using the example data presented in Tables 1 and 3, and the algorithm in Fig. 5, we illustrate the WIT-DIFF algorithm with minws = 0.4 as follows. Level 1 of the WIT-tree contains single items, their tids[6], and their ws. They are sorted in increasing order by their |tids|. The purpose of this work is to compute Diffset faster [16,17].

A join D:

$$d(AD)=\frac{t(A)}{t(D)}=\frac{1345}{1356}=4 \Rightarrow ws(AD)$$

[6] Transaction identity number

$$=ws(A)-\frac{\sum_{t\epsilon d(AD)} tw(t)}{\sum_{t\in T} tw(t)}=0.72-\frac{0.3}{2.25}=0.59$$

A join B:

$$d(AB)=\frac{t(A)}{t(B)}=\frac{1345}{123456}=\emptyset$$

$$\Rightarrow \quad ws(AB)=ws(A)=0.72$$

6. The proposed methods:

In explained ways and methods we used of datasets that cleaned and removed the duplicate items and preprocessing with other tools such as Microsoft excel ,Clementine ,Weka or with other tools. We modified WIT-Diff algorithm at first remove duplicate items in each transactions and reduce transactions weight computations for similar transactions.

6.1 . WIT-Odd or Even method:

In this way at first scanning dataset and selection all items that not repetitive and unique then classify all items in two group, odd items and even items .in continue of process the algorithm done individually for even items and odd items .and at the end we have two run times, one for even items and other run time for odd items .and have compounds of only odd and even items .in result we reducing run time and produce frequent item sets. (Fig 6)

Input: Database D and minimum weighted support threshold minws. Output: FWI contains all frequent weighted Itemsets that satisfy minws from D and Even and odd weight.
Method: WIT-Diff-even and odd() $IF \quad w_i \quad MOD \quad 2 \neq 0 \, then$ // odd items. Else if $w_i \quad MOD \quad 2 = 0 \, then$ //even items. 1. $L_{r=}$ All items that their ws satisfy minws. 2. $sort$ Nodes in L_r increasing by their ws. 3. $FWI = \emptyset$. 4. $call$ Function FWI-Extend-Diff –even and odd with the parameter is L_r. FWI-Extend-Diff-even and odd(L_r) 5. Consider each node l_i in L_1 DO. 6. Add ($l_i.itemset.l_i.ws$) to FWI. 7. Create a new set L_i by join l_i with all l_j following it in $L_1$by: 8. Set X=$l_i.itemset \cup l_j.itemset$ 9. If L_r is the first Level Then $Y = \frac{t(l_i)}{t(l_j)}$

10. Else $Y = \frac{d(l_i)}{d(l_j)}$
11. If $Y = \emptyset$ $then$ $ws(X) = ws(l_i)$
12. Else $ws(X) = COMPUTE - WS - DIFF - even\ and\ odd(Y)$
13. $if\ ws(X)$ satisfies minws then
14. Add new Node $< X, Y, ws(X) >$ into L_i
15. $if\ number\ of\ nodes\ in\ L_i \geq 2\ then$
16. Call recursive the function FWI-Extend-Diff –even and odd with the parameter is L_i

Figure 6: WIT-Diff-even and odd algorithm

6.2. WIT-Max of Even or odd method:

The previous method ,we use of two variables for keeping count of even and odd items then at the end counting down count of two variables and then each of have maximum count and value ,my algorithm run with it. In this way purpose is reduce more calculations. (Fig 7)

Input: Database D and minimum weighted support threshold minws.
Output: FWI contains all frequent weighted Item sets that satisfy minws from D With Max.

Method:
WIT-Diff-MAX()
1. $For\ All\ items\ of\ Dataset\ Do$
2. $IF\ item_{Weight} = ODD\ \ DO$
3. $\max_{odd} + +$
4. $Else\ if\ item_{weight} = EVEN\ Do$
5. $\max_{even} + +$
6. $IF\ \ max_{odd} > max_{even}\ \ then$
7. FWI-Extend-Diff-ODD()
8. $Else\ if\ max_{even} > max_{odd}\ then$
9. FWI-Extend-Diff-EVEN()

Figure 7: WIT-Diff-max of even or odd algorithm

6.3. WIT-Percent method:

In third method , In first counting each of items in total dataset and then compute percent of each item in all items. And then run method with specified percent of items. (Fig 8)

Input: Database D and minimum weighted support threshold minws.
Output: FWI contains all frequent weighted Item sets that satisfy minws from D.

Method:
WIT-Diff-Percent ( )
Array 1[ ][ ]= count of all items.
Array 2[ ][ ]=percent of each items in dataset.
Computing Percent of each item in dataset.
Input user threshold for percent of items.
1. $L_{r=}$All items that their ws satisfy minws.
2. $sort$ Nodes in L_r increasing by their ws.

3. $FWI = \emptyset$.
4. $call$ Function FWI-Extend-Diff-Percent with the parameter is L_r.

FWI-Extend-Diff-Percent (L_r)

5. Consider each node l_i in L_1 DO.
6. Add ($l_i.itemset.l_i.ws$) to FWI.
7. Create a new set L_i by join l_i with all l_j following it in $L_1$by:
8. Set X=$l_i.itemset \cup l_j.itemset$
9. If L_r is the first Level Then $Y = \frac{t(l_i)}{t(l_j)}$
10. Else $Y = \frac{d(l_i)}{d(l_j)}$
11. If $Y = \emptyset\ then \quad ws(X) = ws(l_i)$
12. Else $ws(X) = COMPUTE - WS - DIFF - Percent(Y)$
13. $if\ ws(X)$ satisfies minws then
14. Add new Node $< X, Y, ws(X) >$ into L_i
15. $if\ number\ of\ nodes\ in\ L_i \geq 2\ then$
16. Call recursive the function FWI-Extend-Diff-Percent with the parameter is L_i

Figure 8: WIT-Diff-Percent of items algorithm

We can use data mining tools (such as Clementine software) for acquire count of items and percent of items in total of datasets. (See Fig 9 and Fig 10)

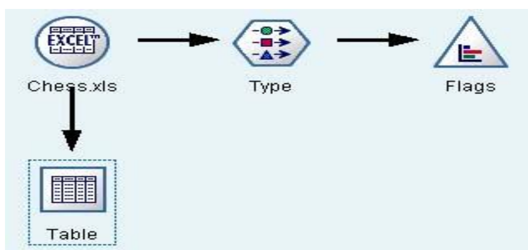
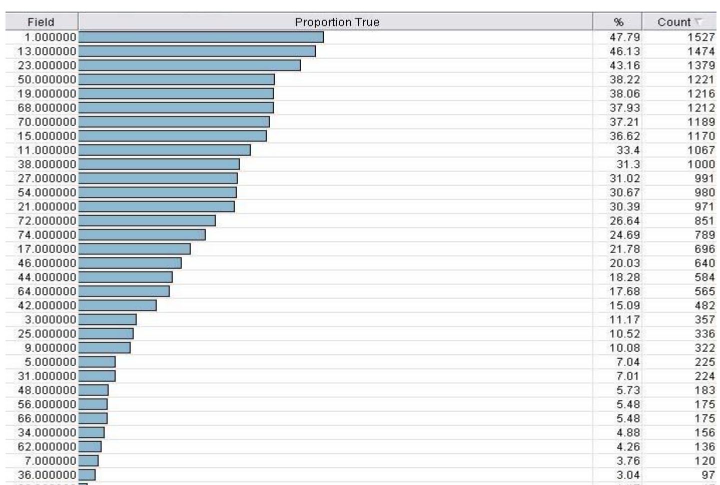


Figure 9: Clementine workspace



Field	Proportion True	%	Count
1.000000		47.79	1527
13.000000		46.13	1474
23.000000		43.16	1379
50.000000		38.22	1221
19.000000		38.06	1216
68.000000		37.93	1212
70.000000		37.21	1189
15.000000		36.62	1170
11.000000		33.4	1067
38.000000		31.3	1000
27.000000		31.02	991
54.000000		30.67	980
21.000000		30.39	971
72.000000		26.64	851
74.000000		24.69	789
17.000000		21.78	696
46.000000		20.03	640
44.000000		18.28	584
64.000000		17.68	565
42.000000		15.09	482
3.000000		11.17	357
25.000000		10.52	336
9.000000		10.08	322
5.000000		7.04	225
31.000000		7.01	224
48.000000		5.73	183
56.000000		5.48	175
66.000000		5.48	175
34.000000		4.88	156
62.000000		4.26	136
7.000000		3.76	120
36.000000		3.04	97
60.000000		1.47	47

Figure 10: Out of Clementine workspace

6.4. WIT-Scope of weight method:

In fourth method, use of domain for weight of items .as respects we assign for each item a value for weight with a random function .we can filter items with their weight and then running algorithm for selected items. In all method we purpose reducing the input of algorithm and as a result reduce run time and produce item sets. (Fig 11)

Input: Database D and minimum weighted support threshold minws.
Output: FWI contains all frequent weighted Item sets that satisfy minws from D.

Method:
WIT-Diff-Scope ()
Input two Scope for weight Domains
From $scope_1$ *to* $scope_2$ *Do*

1. $L_{r=}$All items that their ws satisfy minws.
2. $sort$ Nodes in L_r increasing by their ws.
3. $FWI = \emptyset$.
4. $call$ Function FWI-Extend-Diff-Scope with the parameter is L_r.

FWI-Extend-Diff-Scope (L_r)

5. Consider each node l_i in L_1 DO.
6. Add $(l_i.itemset.l_i.ws)$ to FWI.
7. Create a new set L_i by join l_i with all l_j following it in $L_1$by:
8. Set X=$l_i.itemset \cup l_j.itemset$
9. If L_r is the first Level Then $Y = \frac{t(l_i)}{t(l_j)}$
10. Else $Y = \frac{d(l_i)}{d(l_j)}$
11. If $Y = \emptyset$ *then* $ws(X) = ws(l_i)$
12. Else $ws(X) = COMPUTE - WS - DIFF - SCOPE(Y)$
13. *if* $ws(X)$ satisfies minws then
14. *Add* new Node $< X, Y, ws(X) >$ into L_i
15. *if number of nodes in* $L_i \geq 2$ *then*
16. Call recursive the function FWI-Extend-Diff-Scope with the parameter is L_i

Figure 11: WIT-Diff-Domain of weight algorithm

7. Experimental results

All experimental described below were performed on a Intel(R) Core ™ i5 2.2 GHz .4GB RAM memory, Windows 7, using visual studio C# 2010 .the experimental datasets used for the experimentation were downloaded from http://finin.cs.helsinki.fi/data [14].we add a value for weight each of items with random function (values in the range of (1 to 10) for each datasets). In table 4 see more information of experimental datasets.

In table 4 view databases name and number of items and transactions, and in table 5 view result of run time of algorithms and methods. View number of FWI(Frequent weighted items) based MinWs threshold.

Table 4: Information of datasets

Database (DB)	# Transactions	#Items	Modified
Chess	3196	75	Insert duplicate items on each transaction
Mushroom	8124	120	Insert duplicate items on each transaction

Table 5: Number of FWI from databases

Database	MinWs	#FWI
Chess	80	8063
Chess	70	16039
Chess	60	23208
Chess	50	29431
Mushroom	50	436
Mushroom	40	3038
Mushroom	30	5347
Mushroom	20	11634

Table 6: Number of FWI with methods

Database	MinWs	Even	Odd	Percent	Max	Domain of Weights
Chess	80	150	996	$70\% \rightarrow 5900$	996	$(20:90) \rightarrow 315$
Chess	70	450	9376	$60\% \rightarrow 12912$	9376	$(20:90) \rightarrow 2955$
Chess	60	3730	13815	$50\% \rightarrow 18126$	13815	$(20:90) \rightarrow 1043$
Chess	50	5216	16057	$40\% \rightarrow 28422$	16057	$(20:90) \rightarrow 13817$
Mushroom	50	18	17	$80\% \rightarrow 31$	18	$(20:90) \rightarrow 25$
Mushroom	40	112	54	$70\% \rightarrow 31$	112	$(20:90) \rightarrow 42$
Mushroom	30	133	71	$40\% \rightarrow 5331$	133	$(20:90) \rightarrow 91$
Mushroom	20	689	134	$30\% \rightarrow 9452$	134	$(20:90) \rightarrow 336$

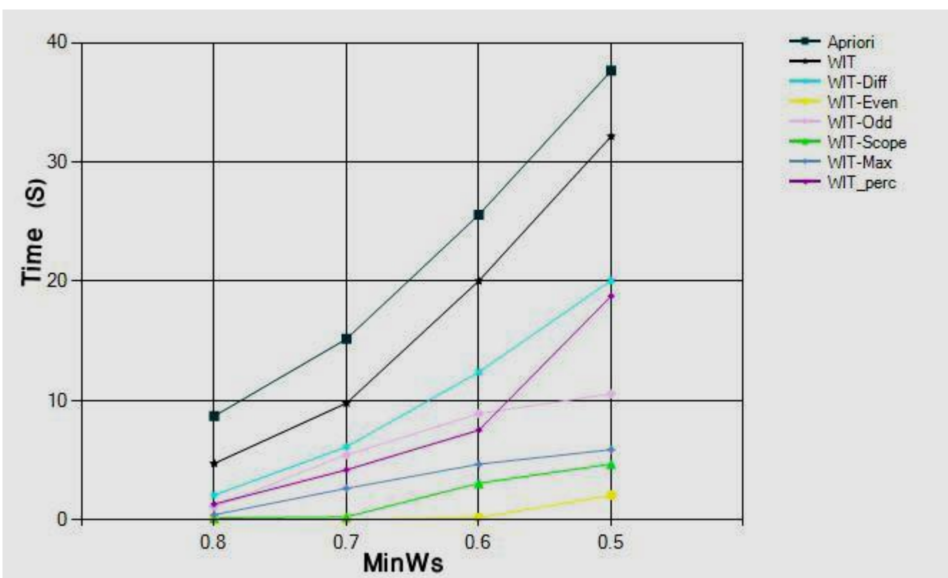


Figure 12: Run time for the eight methods in Chess dataset

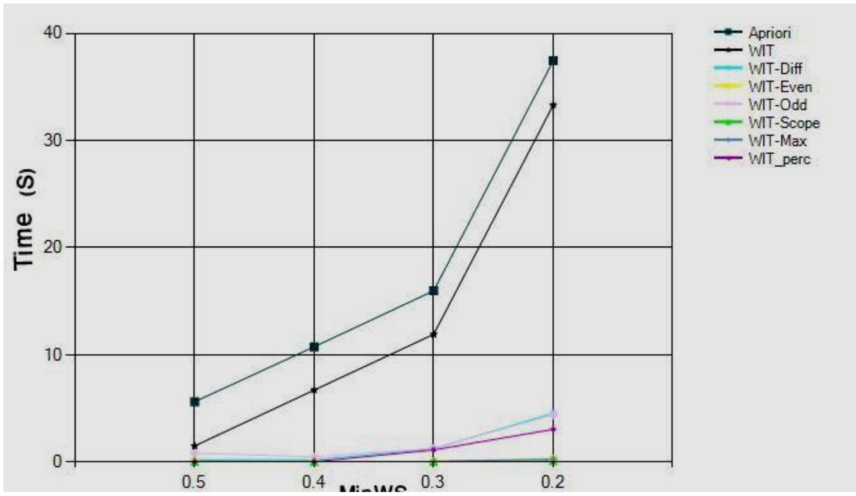


Figure 13: Run of the eight methods in Mushroom dataset

8. Conclusion

This paper has presented some method for mining frequent weighted item sets from weighted item transaction databases with reduce run time and reduce produce frequent item sets. And several efficient algorithms proposed. We use of WIT-Tree structure and apply my method to WIT-Diff algorithm that have less than run time from other algorithm. In this paper, we have concentrated only on the on the mining of FWIs using the proposed WIT-Tree data structure. And in my proposed methods at the first remove duplicated items in a transaction because, they not efficacy in computations and not compute similar transactions weight since used weight transaction of previous similar transaction.

9. References

[1]. *A new method for mining Frequent Weighted* . **Vo, B,Frans Coenen & Bac Le.** 2013, VLDB94, pp. 489-499.

[2]. *Fast algorithms for mining association rules.* **Agrawal, R., & Srikant, R.** 1994, VLDB'94, pp. 487–499.

[3]. *Mining association rules.* **Cai, C. H., Fu, A. W., Cheng, C. H., & Kwong, W. W.** 1998, international database engineering and, pp. 68–77.

[4]. *An effective mining approach for up-todate.* **Hong, T. P., Wu, Y. Y., & Wang, S. L.** 2009, Expert Systems with Applications, pp. 9747–9752.

[5]. *An efficient and effective association-rule.* **Hong, T. P., & Wang, C. J.** 2010, Expert Systems with, pp. 618–626.

[6]. *A weighted utility framework.* **Khan, M. S., Muyeba, M., & Coenen, F.** 2008, second uksim european symposium on, pp. 87–92.

[7]. *A novel algorithm for mining high utility*. **Le, B., Nguyen, H., Cao, T. A., & Vo, B.** 2009, Asian conference on intelligent information and database, pp. 13–16.

[8]. *An efficient strategy for mining high utility*. **Le, B., Nguyen, H., & Vo, B.** 2011, International Journal of Intelligent Information and Database Systems, pp. 164–176.

[9]. *http://fimi.cs.helsinki.fi/data/*. **Dataset.**

[10]. *A dynamic bit-vector approach.* **Vo, B., Hong, T. P., & Le, B.** 2012, Expert Systems with Applications, pp. 7196–7206.

[11]. *Interestingness measures for association rules.* **Vo, B., & Le, B.** 2011, Expert Systems with Applications, pp. 11630–11640.

[12]. *Mining minimal non-redundant association rules.* **Vo, B., & Le, B.** 2011, Journal of Intelligent Systems Technology and Applications, pp. 92–106.

[13]. *Mining association rules with weighted items.* **Cai, C. H., Fu, A. W., Cheng, C. H.** 1998, international database engineering and, pp. 68–77.

[14]. *A new methid for mining frequent weighted itemsets based on wit-trees.* **Vo, B,Frans Coenen & Bac Le.** 2013.

[15]. *Mining non-redundant association rules.* **Zaki.M.J.** 2004.

[16]. *Weighted association rule mining using weighted support and significance framework.* **Tao.F.Murtagh.** 2003.

[17]. *Weighted association rule mining via a graph based connectivity model.* **Russel Pears, Yun Sing Koh,Gillian Dobbie,Wai Yeap.** s.l. : Information Sciences, 2012.

[18]. *Surveying Robot Routing Algorithms with Data Mining Approach.* **Rouhollah Maghsoudi, Somayye Hoseini.** No.2, s.l. : The Journal of Mathematics and Computer Science, 2011, Vol. Vol .2.

[19]. *Rule Extraction for Blood Donators with Fuzzy Sequential Pattern Mining.* **Fatemeh Zabihi, Mojtaba Ramezan, Mir Mohsen Pedram, Azizollah Memariani.** 1, s.l. : The Journal of Mathematics and Computer Science, 2011, Vol. 2.