ICRIS 2019

International Conference on Robots & Intelligent System
15-16 June 2019 • Haikou, China

海南热带海洋学院 HAINAN TROPICAL OCEAN UNIVERSITY 1954

海南大学 HAINAN UNIVERSITY HN 1958

CENTRAL SOUTH UNIVERSITY 中南大学

IEEE Computer Society Order Number E6678
ISBN-13: 978-1-7281-2632-6
BMS Part # CFP19H71-CDR

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: <reprints@computer.org>

Editorial production by Juan E. Guerrero
Cover art production by Annie Jiu
Printed in the United States of America by Applied Digital Imaging

CPS
Conference Publishing Services

IEEE Computer Society
***Conference Publishing Services* (CPS)**
http://www.computer.org/cps

2019 International Conference on Robots & Intelligent System (ICRIS)

ICRIS 2019

Table of Contents

Session 1: Robotics and Intelligent Control

Session 8: Model, Optimization and Simulation

Mining Frequent Weighted Itemsets Using Extended N-list and Subsume

Zhai Yue, Sun Jianmei, Liu Ruijie
Department of Information Science, Dalian Institute of Science and Technology, Dalian, 116000
E-mail: 714021781@qq.com

Abstract—Discovering frequent weighted itemsets(FWIs) is an important research in practical applications of field of data mining. Recently the PrePost algorithm based on the idea of N-lists has been presented. In this paper, we propose an improved version method ENSFWI(extended N-list subsume-based algorithm for finding FWIs). The subsume concept and related theorems are proposed to calculate the weighted supports of itemsets fast and generate directly FWIs without extended N-list intersection, and then an algorithm is built based on these concept for efficiently mining FWIs. It is shown by experimental results that our approach not only results in shorter execution times, but also reduces the memory usage when run on very large and dense database.

Keywords-Data mining, Frequent weighted itemsets, Extended N-list, Subsume

I. INTRODUCTION

Frequent itemsets mining plays an important role in association rule mining. This model only considers whether an item is present in a transaction, but does not take into account the weight of an item within a transaction. Thus mining frequent weighted itemsets has been an increasing demand in data mining research. Many mining techniques have been proposed[1,2]. In recent studies of frequent itemsets mining propose N-list structure[3]. However that method doesn't fit to mining FWIs. Lee et al propose approach by using new types of prefix tree which mines FWIs more efficiently without storing any transaction ID[4]. However it is time-consuming as it needs to traverse tree when dealing with a very large and dense database. Bui et al proposed NFWI[5] algorithm for mining FWIs by using weighted N-list structure[5]. But it is more suitable for mining FWIs when mining in sparse databases.

In this paper, we propose extended N-list subsume-based algorithm for efficiently finding FWIs(ENSFWI). ENSFWI employs a novel extended N-list(EN-list)to represent weighted itemsets. EN-list stores all crucial information and calculate EN-list of k-itemset by intersecting EN-list of two (k-1)-itemsets. The use of the subsume concept to indentify FWIs without needing to compute the EN-list intersection. Then ENSFWI algorithm is shown base some theorems. The experimental results on a variety of databases show the effectiveness of proposed method, especially when run on very large and dense database.

The rest of the paper is organized as follows: Section 2 describes some primitive concepts. Section 3 presents the extended N-list structure and related properties. Section 4 refers to the subsume concept and develops ENSFWI algorithm. Then, Section 5 shows the results of experiments comparing the runtime and memory usage of ENSFWI with NFWI[5] in various databases. Finally we conclude and offer some future research in Section 6.

II. RELATED WORK

A. *Weighted Itemsets Transaction Databases*

Let T= {t_1, t_2…… t_m} be a set of transactions. In order to characterize the importance of the itemset, a set of items I={i_1,i_2,……i_n} and a set of weights W={ w_1,w_2,……w_n }, $0 \le w_n \le 1$, corresponding to each item in I. For example, Table 1A presents a weighted dataset DB comprising five transactions T={t_1,……t_5}, and five items I={A,C,D,T,W}. The weights of these items are presented in Table1B.

Table1: The Transaction Database DB and Item Weights

A		B		C	
TID	Sorted Items	Item	Weights	Item	ws
1	A,B,C,D	A	0.9	A	1
2	A,B	B	0.6	B	0.81
3	A,C	C	0.3	C	0.62
4	A,B,C,E	D	0.2	D	0.43
5	A,B,C,D,E	E	0.1	E	0.42
6	A,B,D,E				

B. *Mining Frequent Weighted Itemsets*

A frequent itemset can be formally defined as follows. Let DB be a transaction database and I be the set of items in DB.

Definition 1. The transaction weight(tw) of a transaction tk is defined as follow: tw(tk)= $\frac{\sum_{i_k \in t_k} w_i}{|t_k|}$.

Definition 2[6]. The weighted support of an itemset X 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.

For example, consider Tables 1 and Definition 1, the tw(t_1) value is compute as follow: tw(t_1)= $\frac{0.9+0.6+0.3+0.2}{4} = 0.5$.

Table 2 shows all tw values of transactions in Table1.

Table 2: Transaction Weights in Table 1

Transactions	1	2	3	4	5	6
tw	0.5	0.75	0.6	0.475	0.42	0.45
sumtw	3.195					

From Tables 1, 2, and Definition 2, ws(A) is computed:ws(A)= =1. Similarly, ws(AC)= =0.62.

Definition 3. An itemset X is called a frequent weighted itemset if ws(X) minsw, where minsw is a predefined minimum weighted support threshold.

Theorem 1.(downward-closure property) if X1 X2 then ws(X1) ≥ws(X2) [7].

978-1-7281-2632-6/19/$31.00 ©2019 IEEE
DOI 10.1109/ICRIS.2019.00132

C. N-list structure

The N-list structure was proposed by Deng to mine all frequent itemsets with the PrePost algorithm[3]. This method compresses a database into a tree and uses the N-list structure extracted from the tree to represent the database and mine FIs.

Definition 4 (The PPC-tree) A PPC-tree is a tree where each node holds five values: Ni.name, Ni.frequency, Ni.childnodes, Ni.pre and Ni.post which are: the frequent item identifier, the associated frequency count, the set of children node associated with this node, the order index of this node when traversing this tree in a pre-order manner and the order index of this node when traversing this tree in post-order manner.

Definition 5 (The PP-code) The PP-code, Ci, of each node Ni in a PPC-tree comprises a tuple of the form: Ci=(Ni.pre, Ni.post, Ni.frequency).

III. Extended N-list Structrue

A. WPPC-tree and WPP-code

Definition 6. (The WPPC-tree) R, is a tree where each node holds five values: Ni.name, Ni.weight, Ni.childnodes, Ni.pre and Ni.post which weight is the sum of tw values of the transactions passing through the node.

Definition7.(The WPP-code) The weighted PP-code comprise a tuple of form: Ci=(Ni.pre,Ni.post,Ni.weight).

For example, Consider the weighted database in Table1. First we obtain a sorted database in Table 1 A which are sorted in decreasing order of ws values of Table 1 C. The final results of the WPPC-tree is shown in Fig1.

B. Extended N-list

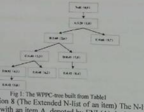

Fig 1: The WPPC-tree built from Table1

Definition 8 (The Extended N-list of an item) The N-list associated with an item A, denoted by ENL(A), is the set of WPP-codes associated with nodes in the PPC-tree whose name is equal to A. ENL(A)= where C_i is the WPP-code associated with N_i.

The EN-Lists of 1-itemsets are shown in fig2.

ENL(A) → {(1,8),3.2}
ENL(B) → {(2,6),2.6}
ENL(C) → {(3,3),1.4} – {(9,7),0.6}
ENL(D) → {(4,1),0.92} – {(7,5),0.45}
ENL(E) → {(5,0),0.42} – {(6,2),0.48} – {(8,4),0.42}

Fig 2: EN-Lists of 1-itemsets

Theorem 2. Let A be an item with its EN-list ENL(A). The weighted support for A ws(A) is calculated by:

$$ws(A)=\frac{\sum_{i=1}^{n} w_i}{\sum_{t_k \in T} tw(t_k)}$$

Definition 9. (The EN-list of k-itemset) Let XA and XB be two (k-1)-itemsets with the same prefix X. The EN-list associated with XAB is determined as follows:

(1) For each WPP-code C_i [ENL(XA) and C_j [ENL(XB), if C_i is an ancestor of C_j, the algorithm will add C_i:pre; C_i:post; C_j:weight to ENL(XAB).

(2) Traversing ENL(XAB) to combine the WPP-codes which has the same pre and post values.

Theorem 3. Given two different nodes N_1 and N_2 of a WPPC-tree, N_1 is an ancestor of N_2 if and only if N_1.pre$\leq N_2$.pre and N_1.pos$\geq N_2$.pos.

According to definition 9 and theorem 3 we calculated ENL(CE) as Fig3.

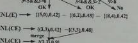

Fig 3: Calculation of ENL(EC)

C. EN-list Intersection

According to Definition 9, we can use the EN-lists of two (k−1)-itemsets to calculate the EN-list of one corresponding k-itemset. The EN-lists intersection algorithm is shown as follows.

```
Procedure ENL_intersection(ENL(I1),ENL(I2))
1. let ENL(I3) ← Ø, i=j=0
2. while i<|ENL(I1)| and j<|ENL(I2)| do
3.    if ENL(I1).pre< ENL(I2).pre then
4.    if ENL(I1).post>ENL(I2).post then
5.      if(|ENL(I3)|>0 and ENL(I3).pre= ENL(I1).pre)
6.        then increase ENL(I3).weight+=ENL(I2).weight
7. else add (ENL(I1).pre, ENL(I1).post, ENL(I2).weight)
to ENL(I3) and j++
8.    else sF=sF-ENL(I2).weight and i++
9. else sF=SF-ENL(I1).frequency and j++
10.   if sF<minws then return null
11.   return ENL(I3)
```

Theorem 6. Given an itemset I and an item X, X $\notin$ I, if ws(I)=ws({I} ∪ {X}),then for every itmset S that satisfier(S | I= Ø and X$\notin$ S) we have ws(S∪I) =ws(S∪I∪X).

This theorem will be used to reduce time complexity of the EN-lists intersection algorithm. For example, we have: ENL(E)->{(5,0),0.42}-{(6,2),0.48}-{(8,4),0.42}; ENL(AE)->{(1,8),0.42}-{(1,8),0.48}-{(1,8),0.42}; ENL(BE)->{(2,6),0.42}-{(2,6),0.48}-{(2,6),0.42};ENL(CE)->{(3,3),0.42}-{(3,3),0.48}. Since ws(E)= ws(AE)= ws(BE)= (0.42+0.48+0.42)/3.195 ≈ 0.41. According to Theorem6, we have ws(ABE)= ws(E) without the need to calculate ws(ABE). Similarly, ws(CE)= ws(ACE)= ws(BCE)= (0.42+0.48)/3.195.

IV. SUBSUME CONCEPT AND ENSFWI ALGORITHM

A. *The subsume concepts*

Definition 10[8]. The subsume index of a frequent item A, denoted by subsume(A) is defined as follows: subsume(X_1)=$\{X_2 \in I_1 | g(X_1) \subseteq g(X_2)\}$,g(X)=$\{TID | T \in DB$ and $X \subseteq T\}$.

For example, let X_1=$\{D\}$ and X_2=$\{B\}$, we have $g(X_1)$ = $\{1,5,6\}$ and $g(X_2)$ = $\{1,2,4,5,6\}$. Because $g(X_1) \subseteq g(X_2)$, thus $X_2 \in$ subsume(X_1).

Theorem 7. Let the subsume index of an item X be$\{X_1, X_2, \ldots, X_m\}$. The support of each of the 2^m-1 nonempty subsets of $\{X_1, X_2, \ldots, X_m\}$ combined with X is equal to the weighted support of X.

For example, subsume(D)=$\{A,B\}$, ws(D)=ws(AD)= ws(BD)=ws(ABD).

The subsume index associated with each frequent 1-itemset based on EN-list

Theorem 8[9]. Let A be a FWI, then subsume(A) =$\{B \in I1 | \forall C_i \in$ EN-list(A), $\exists C_j \in$ EN-list(B) such that C_j is an ancestor of $C_i\}$

Proof: all WPP codes in ENL(A) have a WPP-code ancestor in ENL(B), this means that all transactions that contain A also contain B. then $g(A) \subseteq (g(B)$, which implies that $B \in$ subsume(A).

For example, ENL(B)=$\{(2,6),2.6\}$.According to theorem 4,ENL(B) are descendants of NL(A)=$\{(1,8),3.2\}$.Therefore, $A \in$ subsume(B).

B. *ENSFWI Algorithm*

Bui and Huong presented an algorithm for fast mining the frequent itemsets from NFWI[5]. We simple modify this method to apply it in Extended N-list and subsume FWI (ENSFWI).The proposed ENSFWI algorithm for generating FWIs from Extended N-list is shown as follows.

Input: An weighted dataset DB and threshold minws
Output: FWIs
function Generate_FWIs_From_ENSFWI()
1. Build_ WPPC-tree() to generate I_1 and T
2. Call Generate_ENLists(T,I_1)
3. Find_Subsume(I_1)
4. Let FWIs $\leftarrow$ I_1 and subsume $\leftarrow$ {}
5. Find_FWIs(I_1,subsumes)
6. return FWIs
function Generate_ENLists(T,I_1)
1. let C $\leftarrow$ {T.pre,T.post,T.weight}
2. add C to ENL(T.name)
3. increase ENL(T.name).weight by C.weight
4. for each child in T.children do
5. Generate_ENList(child)
function Find_FWIs(I_s,S)
1. for i $\leftarrow$ I_s.size-1 to 0 do
2. let $FWIs_{next} \leftarrow \emptyset$
3. if | I_s[i].subsume|>0 then let S be the set of subset generated from all elements of I_s[i].subsume
4. for each s in S do
5. add (s,I_s[i]. weight) to FWIs // using theorem 7
6. else if |is[i]|=1 then S $\leftarrow$ {}
7. indexS=|I_s[i].subsumes|-1
8. for j $\leftarrow$ i-1 to 0 do
9. if indexS $\geq$ 0 and Is[i].Subsumes[indexS]=j then
10. indexS=indexS-1 and continue
11. let e_{first} be the first item of I_s[j]
12. FWIs $\leftarrow$ $\{e_{first}\}$+I_s[i] ; (ENL(FI) FI.weight) $\leftarrow$ENL_intersection (ENL(i),ENL(j))
13. if ENL(FI)=null then continue
14. add FWI to FWIs
15. insert FI at first in FWIsnext
16. for each subsume in S do
17. let f=FWI+subsume ;f.weight=FWI.weight
18. add f to FWIs ;Find_FWIs(FWIsnext,S)

We will illustrate the operation of the ENFWI algorithm with the example in Table 1. After calling the function of Build_WPPC_Tree, the obtained result is the WPPC-tree shown in Fig 1 with minws = 0.4. The EN-lists of the 1-FWIs are shown in Fig 2. ws(I_1)=$\{1,0.81, 0.62,0.42,0.41\} \geq$ minws. Thus the 1-FWIs are added into the set of FWIs=$\{A,B,C,D,E\}$.The subume index associated with the FWIs is shown in Table 3.

Table 3: The subsume index associated with FWIs

FWIs	A	B	C	D	E
Subsume index		A	A	A,B	A,B

The algorithm traverses Find_FWIs method to create FWIs. For detail, find the 2^m-1 subsets from the m find the m frequent weighted 1-itemsets in subsume($\{e\}$) and combine them with $\{e\}$ to generate the 2^m-1 frequent weighted itemsets S. In this case, subsume($\{B\}$)= $\{A\}$, therefore S = $\{AB\}$; Similarly, we don't combined with A because A is subsume of C to create candidate 2-itemsets $\{AC\}$, candidate 2-itemsets $\{AD, BD,AE,BE\}$ does not compute. Because ws(CD),ws(CE)and ws(DE)< minws, it updates FWIs= $\{A,B,C,D,E,AB,AC,BC,AD,BD,AE,BE\}$. Next the algorithm combines the elements in $FWIs_{next}$ with the elements in S to create further frequent weighted itemsets without calculating their support. In this case, $\{ABD\}$and $\{ABE\}$ are created; and use the elements in $FWIs_{next}$ to combine together to create the candidate 3-itemsets. because ws(ABC)=1.4/3.195< minws, ABC is not a FWI. In this case, this algorithm will stop here because $FWIs_{next}$ has only one element.

In Fig 4, The final results include FWIs which are shown below:FWIs=$\{A,B,C,D,E,AB,AC,BC,AD,BD,AE,BE,ABD, ABE\}$. The ENFWI algorithm does not compute and store the EN-list of the nodes $\{AB, AC, AD, BD, AE,BE, ABD,ABE\}$which are shown by dotted lines. Therefore, using the subsume index concept it not only reduces the runtime but also reduce the memory usage.

Fig 4. All frequent weighted itemsets on table1(minws = 0.4)

V. EXPERIMENTAL RESULTS

The experimental databases are downloaded from http://fimi.cs.helsinki.fi/data/ to use for experiments. The The experiments are conducted using Chess and Mushroom dense datasets. Using random number [0,1] as the weight of item to simulate the importance of itemsets.

A. The runtime of the ENSFWI Algorithm

We compare the proposed algorithm ENSFWI with NFWI[5] in terms of runtime. The experimental results are shown in Fig 5-6.From these figures it can be shown that ENSFWI ran fairly fast on dense database especially with low thresholds, such as Mushroom and Chess. This is explained as follows. Generating the subsume index involves extended cost. However, the subsume index associated with each of the weighted frequent items in a sparse datasets usually have few elements. Therefore, using the subsume concept don't work well in this case. Therefore, The ENSFWI is capable of good data compression by building a WPPC-tree.we conclude ENSFWI outperforms NFWI in terms of the runtime on dense database.

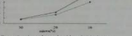

Fig 5. Execution time of two algorithms in Mushroom

Fig 6. Execution time of two algorithms in Chess

B. The Memory usage

Fig 7. The memory usage using two algorithms in Mushroom

The experimental results with respect to the memory usage experiments are shown in Fig7-8. the operation of ENSFWI is better than NFWI in the case of dense datasets. The ENSFWI algorithm does not compute and store EN-list of nodes which generated using the subsume index. Therefore, using the subsume index concept is effective with respect to dense datasets.

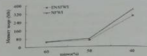

Fig 8. The memory usage using two algorithms in Chess

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an effective approach ENSFWI for mining frequent weighted items using the EN-list and subsume concepts. Firstly, we utilize the extended N-list structure and develop some theorems to improve intersection between two EN-lists. Then subsume index of frequent weighted items based on EN-list and its theorems calculate the weighted supports of the weighted itemsets and determine some nodes without using EN-lists intersections. The experimental results on a variety of databases show that ENSFWI is more efficient with respect to dense datasets. Besides. ENSFWI algorithm use the least memory on dense database.In the future, we will focus on applications of the EN-list structure to various data mining problems, such as mining frequent weighted closed/maximal itemsets, mining frequent weighted itemsets in incremental databases.

Acknowdgements

This work is supported by Social Science Foundation of Liaoning Province (Grant No L18AGL004).

REFERENCES

[1] VO, Bay,et al. A new method for mining Frequent Weighted Itemsets based on WIT-trees. Expert Systems with Applications, 2013, 40.4: 1256-1264.

[2] A.M.J.Md.et al, Weighted Support Association Rule Mining using Closed Itemset Lattices in Parallel, IJCSNS International Journal of Computer Science and Network Security, Vol.9,247-253,2009.

[3] Deng, Z, Wang, Z, & Jiang, J. A new algorithm for fast mining frequent itemsets using N-lists. Science China Information Sciences, 55 (9) (2012). 2008–2030.

[4] LEE, et al. Mining frequent weighted itemsets without storing transaction ids and generating candidates. International Journal of Uncertainty, 2017, 25.01: 111-144.

[5] BUI, et al. A weighted N-list-based method for mining frequent weighted itemsets. Expert Systems with Applications, 2018, 96: 388-405.

[6] Bay Vo,et al. Mining Frequent Weighted Closed Itemsets, Advanced Computational Methods for Knowledge Engineering Studies in Computational Intelligence Vol.479, 379-390,2013.

[7] Lu, Songfeng, Heping Hu, and Fan Li. "Mining weighted association rules." Intelligent data analysis 5.3 (2001): 211-225.

[8] VO, Bay, et al. Mining frequent itemsets using the N-list and subsume concepts. International Journal of Machine Learning and Cybernetics, 2016, 7.2: 253-265.

[9] YUE, Zhai; LIJUAN, Wang; NING, Wang. Efficient weighted association rule mining using lattice. In: Control and decision conference (2014 CCDC), the 26th Chinese. IEEE, 2014. p. 4913-4917.