A New Method for Mining High Average Utility Itemsets

Tien Lu, Bay Vo, Hien Nguyen, Tzung-Pei Hong

A New Method for Mining High Average Utility Itemsets

Tien Lu[1], Bay Vo[2,3], Hien T. Nguyen[3], Tzung-Pei Hong[4]

1. *University of Sciences, Ho Chi Minh, Vietnam*
2. *Divison of Data Science, Ton Duc Thang University, Ho Chi Minh, Vietnam*
3. *Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh, Vietnam*
4. *Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan, ROC*

Email: lucaotien@gmail.com,vodinhbay@tdt.edu.vn, hien@tdt.edu.vn, tphong@nuk.edu.tw

Abstract. Data mining is one of exciting fields in recent years. Its purpose is to discover useful information and knowledge from large databases for business decisions and other areas. One engineering topic of data mining is utility mining which discovers high-utility itemsets. An itemset in traditional utility mining considers individual profits and quantities of items in transactions regardless of its length. The average-utility measure is then proposed. This measure is the total utility of an itemset divided by the number of items. Several mining algorithms were also proposed for mining high average-utility itemsets (HAUIs) from a transactional database. However, the number of generated candidates is very large since an itemset is not a HAUI, but itemsets generated from it and others can be HAUIs. Some effective approaches have been proposed to prune candidates and save time. This paper proposes a new method to mine HAUI from transaction databases. The advantage of this method is to reduce candidates efficiently by using HAUI-Tree. A new itemset structure is also developed to improve the speed of calculating the values of itemsets and optimize the memory usage.

Keywords: utility mining, average utility, itemset mining

1 Introduction

Pattern mining plays an important role in data mining. Frequent pattern mining is a task of searching for the associations and correlations among items in large transactional or relational datasets [2]. However, it considers only the occurrence of items, while the other factors such as price, profit are neglected. The significance of items is the same. The actual significance of an itemset thus cannot be easily recognized by using traditional approaches. Only frequency is thus not sufficient to identify highly profitable items. Chan proposed utility mining to solve this problem [10] or [7], his approach used a tree structure to mine high utility itemsets (HUIs) of which utility values are larger than or equal a predefined threshold. Many approaches have been proposed for mining HUIs such as [9], [4]. However, utility mining is based on only the utility of itemset and does not consider their length while the longer

length itemset is, the higher utility its value has. The average utility (au) measure was proposed by Hong in [3] where the author considered the length of itemset. However, due to the lack of "downward closure property", the cost of candidate generation for mining high utility itemsets is intolerable in terms of time and memory space. An itemset which is not a HAUI can combine with one or more other items to become HAUI. Therefore, the number of generated candidates is very large, which leads to a challenge for researchers. The solution of this problem is to reduce the number of candidates and the time for generating them. Using the average upper-bound value is one of solutions [3]. Some authors proposed structures to improve the generation steps. However, the speed was decreased trivially. Although the closure downward cannot be applied to HAUIs (high-average utility itemsets), HAUUBIs (high-average utility upper bound itemsets) have this feature. Thus, we utilize it to generate HAUIs effectively. We also proposed the HAUI-tree and a new structure for itemset to speed up calculation.

The rest of this paper is organized as follows. Section 2 presents some related researches in mining high utility itemsets and high average-utility itemsets. Section 3 presents the proposed algorithm. An illustrative example is described in section 4. Finally, section 5 includes experimental results and conclusions are in section 6.

2 Related Work

2.1 High Utility Itemset Mining

High Utility Itemsets (HUI) mining is different from Frequent Itemsets (FI) mining. FI mining regards to the appearance of itemsets in transactions, and does not consider other attributes of itemsets such as price and quantity. In HUI mining, the utility of item in a transaction is defined as multiplication of its quantity and its profit. The utility of itemset in a transaction is the sum of the utility of its items in that transaction. If the utility of itemset is greater than or equal to a predefined threshold, it is called a high utility itemset (HUI).

Liu proposed the two-phase algorithm for HUI mining. In the first phase, database is scanned and generated candidates are pruned by using the upper bound threshold. In next phase, database is scanned again, the actual utility value of remain candidates are calculated and the algorithm then gets HUI from candidates. The problem of this method is to reduce candidates and time for rescanning database.

Ahmed et al. pruned candidates by using HUC-Tree [1]. High utility itemsets are mined after scanning database two times. In 2011, Hong et al. proposed a new structure called the HUP-Tree [8] for mining high utility itemsets. First, the algorithm processes correlative utility values and creates 1-itemsets as candidates. Second, database is rescanned and the HUP-tree is generated. High utility itemsets can be obtained from HUP-tree.

2.2 High Average-Utility Itemset Mining

In high utility itemset mining, the utility of an itemset is the sum of the utilities of its items in all transactions including the itemset. The longer the length of itemset is, the larger its utility increases. Thus, using the same threshold for all itemsets is not fair. The average utility (au) was proposed in [3] where the measure also considered the number of all items in an itemset in addition to profits and quantities of items used in the original utility measure. The average utility of an itemset was defined as the summation of all utility values of its items in transactions divided by the number of its items. The downward-closure property used in utility mining is not directly applied to the average-utility mining. An itemset which is not a high-average utility itemset (HAUI) can be combined with one or more items to become a HAUI. This property leads to the larger number of generated candidates and slower.
To reduce candidates, most authors used the average utility upper-bound value (ub). In [3], Hong et al. used average utility upper-bound to prune generated candidates and create the set of itemsets having upper-bound value are greater than or equal threshold. The set of (r+1)-itemset is generated by combining r-itemsets and 1-itemsets which have high average utility upper-bound. However, some (r+1)-itemset was generated in many times. Furthermore, calculating values was not fast. So, in [6], Hong et al. improved the speed of calculation by using a structure which is the same as Index Table. Then, an algorithm was proposed in [5]. This algorithm, calculates values of itemsets and filters them were processed two times. However, this method requires much effort and time to computing values of itemsets.

3 Proposed Algorithm

In the high average utility itemsets mining algorithm proposed in [3], Hong et al. used the average utility upper-bound (ub) value to reduce the number of candidates by using items which have suitable ub. The ub values of these items are greater than or equal the threshold. A structure then used to speed up calculating was the Index Table proposed in [6]. This structure stored the position of the last item of current itemset in transactions to retrieve its values easily. In 2012, Hong et al. continued improving Hong's method using ub. They used ub to prune two times. Firstly, they removed unsuitable item and then calculated again without unsuitable item. However, this method took much time. Especially, downward closure properties can be applied for high average utility upper-bound itemsets. An itemset is the high average utility upper bound itemset, all its subitemsets is also the high average utility upper bound itemsets. This new method generated candidates faster and compared their average utility values with threshold to get HAUIs. This approach is similar to WIT-tree in [11] which is used for mining frequent itemsets.

3.1 Notations

Notation 1. $I=\{i_1,i_2, \ldots, i_n\}$ is a set of items, which may appear in transactions.

Notation 2. An itemset X is a subset of items, $X \subset I$. If $|X|$=r, the itemset X is called a r-itemset. For example, an itemset {AB} which contains 2 items is called a 2-itemset.

Notation 3. T={t_1, t_2, …,t_m} is a set of m transactions in database.

Notation 4. P={p_1, p_2, …, p_n} is a set of profits for items which may appear in transactions.

Notation 5. λ is predefined threshold.

Notation 6. ub_X is an average utility upper-bound value of an itemset X.

Notation 7. au_X is an average utility value of an itemset X.

Notation 8. T_X is a set of transactions which contain itemset X.

Notation 9. Pos_X is a set of correlative positions of the last item of itemset X in transactions which include it (T_x).

Notation 10. q_{ij} is the quantity of item i_i in transaction t_j.

Notation 11. The utility of item i_i in transaction t_j: $u_{ij}=q_{ij}$ x p_i.

Notation 12. $maxU_{t_j} = \max(u_{1j}, u_{2j}, u_{3j}, \dots, u_{nj})$ is maximum utility of items in transaction t_j. So, $u_{ij} \leq maxU_{t_j}$

Notation 13. The utility of itemset X in transaction t_j: $u_{Xj}=\sum_{i \in X} u_{ij} \times q_{ij}(t_j \in T_X)$.

Notation 14. The average utility of itemset X: $au_X=\frac{\sum_{t_j \in T_X} u_{Xj}}{|X|}$

Notation 15. The average utility upper-bound of itemset X: $ub_X=\sum_{t_j \in T_X} maxU_{t_j}$

Notation 16. HAUUBI is a set of high average utility upper-bound itemsets.

Notation 17. HAUI is a set of high average utility itemsets.

3.2 The downward closure properties of HAUUBI

Theorem 1: If itemset X is a high average utility upper bound itemset, then all its sub-itemsets are also the high average utility upper bound itemsets.

Proof:

Let r-itemset X with $ub_X=\sum_{t_j \in T_X} maxU_{t_j}$ and (r+1)-itemset X' with X is sub-itemset of X' (X⊂X'). Thus, $|T_{X'}| \leq |T_X|$ and $T_{X'} \subseteq T_X$. Furthermore, $ub_{X'}=\sum_{t_j \in T_{X'}} maxU_{t_j} \leq ub_X=\sum_{t_k \in T_X} maxU_{t_k}$. If X' ∈ HAUUBI or $ub_{x'} \geq \lambda$. Thus, $ub_X \geq \lambda$ or X ∈ HAUUBI.

3.3 A new structure of itemset

To speed up calculating, we use some properties of an itemset as follow:

- List of items belong to itemset X.
- au_X: average utility of itemset X.
- ub_X: average utility upper bound of itemset X.
- T_X: a set of transactions including itemset X.

- Pos_X: a set of position of last item of itemset X in correlative transactions. $(|T_X|=|Pos_X|)$

3.4 Algorithm for mining HAUI

Input: A set of transactions T, a set of profits of items P, predefined threshold λ (All items in transactions are ordered to generate candidates and calculate faster).

Output: High average utility itemsets.

Processes:

```
Prepare ()
{
   STEP 1: Initialize MaxU is a set of maxU of t_j (t_j ∈ T)
   STEP 2: Initialize L_1 is a set of 1-itemset generated from
items. Each itemset X includes au_X, ub_X, T_X, Pos_X
   STEP 3: For each transaction t_i in T, do the following
substeps:
o      Calculate maxU_{t_i}
o      Calculate au_{Xi} (X ∈ t_i, X ∈ L_1)
o      Add t_i to T_X, add the position of itemset X in t_i to
       Pos_X
o      Update au_x, ub_x of itemset X in L_1

   STEP 4: For each 1-itemset X in L_1 to choose itemsets, add
to HAUUBI_1 (1-itemsets have ub, which sastify predefined
threshold) and HAU_1(1-itemsets have au, which sastify predefined
threshold)
   STEP 5: createHAUI(HAUUBI_1, n_HAUUBI1)
}
```

The `createHAUI(L_r, n_{L_r})` (L_r: a set of itemsets, n_{Lr}: the quantity of itemsets in L_r)

```
createHAUI(L_r , n)
{
    for i=1 to n-1 do
    {
        Initialize L_{r+1};
        for j=i+1 to n do
        {
            Itemset X = merge(L_r[i], L_r[j]);
            if(ub_X > λ)
            {
                Add X to L_{r+1};
                if(au_X> λ)   Add X to HAUI;
            }
        }
        createHAUI(L_{r+1} , n_{L_{r+1}})
    }
}
```

3.5 Purposes of using T_X and Pos_X

Let r-itemset $\{XI_i\}$ and $\{XI_j\}$ be two itemsets which have the same prefix X. We merge them to create (r+1)-itemset $\{XI_iI_j\}$ and calculate values of itemset $\{XI_iI_j\}$ by following formulas:

$$T_{\{XI_iI_j\}} = T_{\{XI_i\}} \cap T_{\{XI_j\}}$$

$$ub_{\{XI_iI_j\}} = ub_{\{XI_i\}} - \sum_{t_i \in T_{\{XI_i\}} - T_{\{XI_j\}}} \max U(t_i)$$

$$u_{\{XI_iI_j\}} = au_{\{XI_i\}} * r - \sum_{t_i \in T_{\{XI_i\}} - T_{\{XI_j\}}} u_{\{XI_i\}i} + \sum_{t_k \in T_{XI_j} \cap T_{XI_i}} u_{\{I_j\}t_k}$$

$$au_{\{XI_iI_j\}} = u_{\{XI_iI_j\}}/(r+1)$$

4 Illustration

A set of transactions T and a set of profits of items P are shown in Table 1 and Table 2 below. The predefined threshold λ=25. This example is the same to example in [6].

Firstly, we calculate au, ub of 1-itemsets generated from input items. For example, the average-utility of 1-itemset {A} in transactions that belong to $T_{\{A\}}$ is calculated by (1+2+3+2)*3/1=24 in which the profits of item A is 3 and the quantities of 1-itemset A are 1, 2, 3, 2 in correlative transactions. $T_{\{A\}}$={1,5,8,9}, $Pos_{\{A\}}$={1,1,1,1}. The results are shown in Table 3, L_1 includes 1-itemsets X with properties au, ub, T_X, Pos_X and the mining process is based on L_1.

Table 1. Transactions in database

TID	A	B	C	D	E	F
1	1	0	2	1	1	1
2	0	1	25	0	0	0
3	0	0	0	0	2	1
4	0	1	12	0	0	0
5	2	0	8	0	2	0
6	0	0	4	1	0	1
7	0	0	2	1	0	0
8	3	2	0	0	2	3
9	2	0	0	1	0	0
10	0	0	4	0	2	0

Table 2. Profits of Items

Item	Profit
A	3
B	10
C	1
D	6
E	5
F	2

Table 3. Results after calculating values

TID	$\mathbf{u_{Ak}}$	$\mathbf{u_{Bk}}$	$\mathbf{u_{Ck}}$	$\mathbf{u_{Dk}}$	$\mathbf{u_{Ek}}$	$\mathbf{u_{Fk}}$	$\mathbf{mu_k}$
1	3	0	2	6	5	2	6
2	0	10	25	0	0	0	25
3	0	0	0	0	10	2	10
4	0	10	12	0	0	0	12
5	6	0	8	0	10	0	10
6	0	0	4	6	0	2	6
7	0	0	2	6	0	0	6
8	9	20	0	0	10	6	20
9	6	0	0	6	0	0	6
10	0	0	4	0	10	0	10
ub	42	57	75	**24**	56	42	
au	**24**	40	57	**24**	45	**12**	

Table 4. List of 1-itemsets with properties in L_1

1-Itemset	$\mathbf{au_X}$	$\mathbf{ub_X}$	$\mathbf{T_X}$	$\mathbf{Pos_X}$
A	24	42	{1,5,8,9}	{1,1,1,1}
B	40	57	{2,4,8}	{1,1,2}
C	57	75	{1,2,4,5,6,7,10}	{2,2,2,2,1,1,1}
D	24	24	{1,6,7,9}	{3,2,2,2}
E	45	12	{1,3,5,8,10)	{4,1,3,3,2}
F	56	42	{1,3,6,8}	{5,2,3,4}

Table 5. $HAUUB_1$

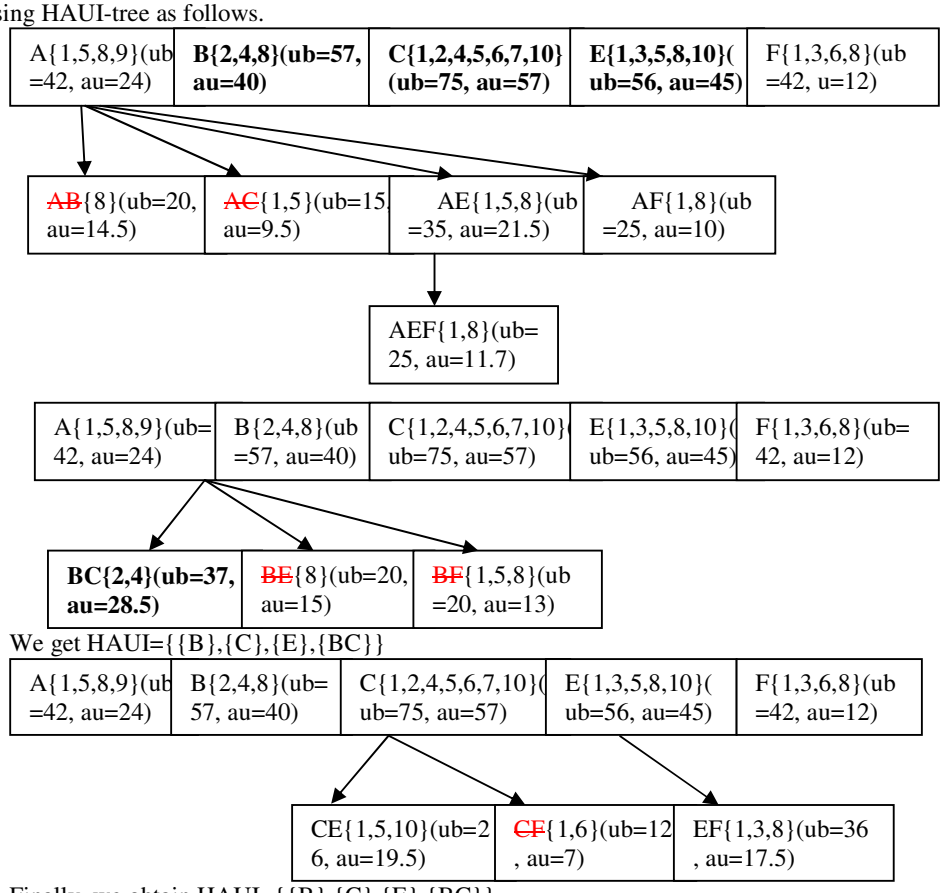
1-Itemset	$\mathbf{au_X}$	$\mathbf{ub_X}$	$\mathbf{T_X}$	$\mathbf{Pos_X}$
A	24	42	{1,5,8,9}	{1,1,1,1}
B	40	57	{2,4,8}	{1,1,2}
C	57	75	{1,2,4,5,6,7,10}	{2,2,2,2,1,1,1}
E	45	56	{1,3,5,8,10)	{4,1,3,3,2}
F	56	42	{1,3,6,8}	{5,2,3,4}

Table 6. $HAUI_1$

1-Itemset	au_X	ub_X	T_X	Pos_X
B	40	57	{2,4,8}	{1,1,2}
C	57	75	{1,2,4,5,6,7,10}	{2,2,2,2,1,1,1}
E	45	12	{1,3,5,8,10)	{4,1,3,3,2}

We obtain HAUI={{C},{B},{E}}

We illustrate the processing of the procedure createHAUI(L_r,n_{Lr}) procedure by using HAUI-tree as follows.



We get HAUI={{B},{C},{E},{BC}}

Finally, we obtain HAUI={{B},{C},{E},{BC}}.

5 Experiments

All algorithms were coded by C# 2010. The configuration of computer used for experimental evaluation is Intel core 2 dual with 2.20 GHz CPU, 2GB RAM and Windows 7 OS. Experimental databases have features as follow:

Table 7. Experimental databases

Database	#Trans	#Items
BMS-POS	515597	1657
Chess	3196	76

Table 8. The execution time of two algorithms along with different thresholds.

Database	Threshold (%)	#HAUI	Execution time (minutes)	
			HAUI mining by UB and Index Table [6]	Proposed algorithm
BMS-POS	0.6	143	>180(unfinished)	1.11
	0.8	56	157	0.66
	1.0	34	62	0.55
	2.0	6	31	0.1
Chess	4.0	38	18.39	7.69
	5.0	1	2.45	1.22
	6.0	0	1.33	0.19

We compare our algorithm with the algorithm using ub and Index Table in [6]. Our algorithm is always faster than the algorithm using ub and Index Table. With BMS-POS database, when the threshold is very small, the algorithm in [6] run very slow. For example, with the threshold is 0.8, it takes 157 minutes while our algorithm takes 0.66 minutes. When we decrease the threshold to 0.6, the runtime is more than 180 minutes.

With HAUI-Tree, the generation of candidates is faster than other methods, because HAUI-Tree creates candidate for the next generations from the previous one. Besides, the other algorithms create candidates from scan each transaction. The quantity of transactions can be very large. So they is slower than HAUI-Tree. Furthermore, HAUI-Tree uses some properties of itemset to calculate utility values for the next generation quickly.

6 Conclusions and Future Work

This paper has presented a new method for mining high average utility itemsets from databases by using the HAUI-Tree algorithm and a new structure for itemset. The proposed algorithm prunes and generates candidates fast and scans the database only once. Furthermore, the new structure for itemset improved the calculation and execution time. The proposed algorithm can be applied to large database (which include many items and transactions). Besides we are also interested in developing application for high utility average itemsets. In future, we will research efficient methods to find useful rules from high utility average itemsets.

References

1. Ahmed, C. F., Tanbeer, S. K., Jeong, B.S., Lee, Y.K., *HUC-Prune: An efficient candidate pruning technique to mine high utility patterns*. Appl Intell 34, 181–198, 2009.
2. Han, J., Kamber, M., *Data mining: Concepts and Techniques*. 2th edition, Morgan Kaufmann, p743, 2006.
3. Hong, T.P., Lee, C.H., Wang, S.L., *Effective utility mining with the measure of average utility*. Expert Systems with Applications 38, 8259–8265, 2011.
4. Lan, G.C., Hong, T.P., Tseng, V.S., *Mining High Transaction-Weighted Utility Itemsets*. Second International Conference on Computer Engineering and Applications, 2010.
5. Lan, G.C., Hong, T.P., Tseng, V.S., *Efficiently mining high average-utility itemsets with an improved upper-bound strategy*. International Journal of Information Technology & Decision Making, 1009-1030, 2012.
6. Lan, G.C., Hong, T.P., Tseng, V.S., *A Projection-Based Approach for Discovering High Average-Utility Itemsets*. Journal of Information Science and Engineering 28, 193-209, 2012.
7. Lan, G.C., Hong, T.P., Lu, W.H., *An effective tree structure for mining high utility itemset.* Expert Systems with Applications 38, 7419–7424, 2010.
8. Le, B., Nguyen, H., Cao, T.A., Vo, B. , *A Novel Algorithm for Mining High Utility Itemsets.* First Asian Conference on Intelligent Information and Database Systems, 2009.
9. Liu, Y., Li, J., Liao, W.K., Shi, Y., Choudhary, A., *High Utility Itemsets Mining*. International Journal of Information Technology & Decision Making, Vol. 9, No. 6, 905–934, 2010.
10. Liu, Y., Liao, W.K., Choudhary, A., *A two-phase algorithm for fast discovery of high utility itemsets*, Lecture Notes in Computer Science, 3518, pp.689-695.
11. Vo, B., Coenen, F., Le, B., *A new method for mining Frequent Weighted Itemsets based on WIT-trees*. Expert Systems with Applications 40, 1256–1264, 2013.