IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-6, Issue-4 E-ISSN: 2347-2693

Discovering high average utility itemsets with multiple minimum supports

Neha Agrawal[1*],Amit Sariya[2]

[1*]Department of Computer Science, Alpine Institute of Technology,RGPV University, Ujjain, M.P,India
[2] Department of Computer Science, Alpine Institute of Technology,RGPV University, Ujjain, M.P,India

[*]*Corresponding Author: agrawal.na@gmail.com, Tel.: +91-9753362015*

Available online at: www.ijcseonline.org

Received: 14/Mar/2018, Revised: 20/Mar/2018, Accepted: 05/Apr/2018, Published: 30/Apr/2018

Abstract— High average-utility itemsets mining (HAUIM) is a key data mining task, which aims at discovering high average-utility itemsets (HAUIs) by taking itemset length into account in transactional databases. Most of these algorithms only consider a single minimum utility threshold for identifying the HAUIs. In this paper, we address this issue by introducing two phase algorithm with pruning strategy in which the task of mining HAUIs is done with multiple minimum average utility thresholds , where the user may assign a distinct minimum average-utility threshold to each item or itemset.

Keywords—Frequent itemsets ,minimum supports, utility mining,high utility mining

I. INTRODUCTION

The main purpose of knowledge discovery in database (KDD) is to discover implicit and useful information in a collection of data. Association-rule mining(ARM) or frequent itemset mining (FIM) plays an important topic in KDD, which has been extensively studied [1,2]. A major limitation of traditional ARM and FIM is that they focus on mining association rules or frequent itemsets in binary databases, and treat all items as having the same importance without considering factors. To address this limitation, the problem of high utility itemset mining (HUIM) [3,8,16,17] was introduced. An important limitation of traditional HUIM is that the utility of an itemset is generally smaller than the utility of its supersets. Hence, traditional HUIM tends to be biased toward finding itemsets of greater length (containing many items), as these latter are more likely to be high utility itemsets. The utility measure used in traditional HUIM thus does not provide a fair measurement of the utility of itemsets. To alleviate the influence of an itemset's length on its utility, and find more useful high utility itemset for recommendation, Hong et al. [5] proposed the average utility measure, and the problem of high average utility itemset mining (HAUIM). The average utility of an itemset is defined as the total utility of its items in transactions where the itemset appears, divided by the number of items in the itemset. Numerous algorithms have been designed to more efficiently mine high average-utility itemsets (HAUIs) [9,11,12,14] but most of them rely on a single minimum average-utility threshold to mine HAUIs. In real-life situations, each item or itemset may be more or less important to the user. It is thus unfair to measure the utility of all items in a database using the same minimum utility threshold.

To address this issue, this paper proposes a novel framework for high average-utility itemset mining with multiple minimum average-utility thresholds (HAUIM-MMS). Based on the proposed framework, a two-phase algorithm named HAUI-MMS is proposed to discover HAUIs. To improve the performance of the proposed algorithm,a efficient pruning strategy is designed to prune unpromising itemsets early, thus reducing the search space and speeding up the discovery of HAUIs. Extensive experiments were conducted on both real-life and synthetic datasets to show that the proposed algorithm can efficiently mine the complete and correct set of HAUIs in databases, while considering multiple minimum average-utility thresholds to assess the utility of itemsets.

II. RELATED WORK

In recent years, HUIM [3,8,16,17] has become a key research topic in the field of data mining. Chan et al. [3] presented a framework to mine the top-*k* closed utility patterns based on business objectives. Yao et al. [15,1] defined the problem of utility mining while considering both purchase quantities of items in transactions (internal utility) and their unit profits (external utility). Liu et al. [9] introduced the transaction-weighted utility (TWU) model and the transaction weighted downward closure (TWDC) property. Lin et al. [8] adopted the TWU model to design the high-utility pattern (HUP)-tree for mining HUIs using a condensed tree structure called HUP-tree. Liu and Qu [13] proposed the HUI-Miner algorithm to discover HUIs without generating candidates using a designed utility-list structure. Fournier-Viger et al. [4] then presented the FHM algorithm and the Estimated Utility Co-occurrence Structure (EUCS) to mine HUIs.Hong

et al. [5] first proposed the average utility measure, and the stated the problem of HAUIM. The average-utility of an itemset is the sum of the utilities of its items, in transaction where it appears, divided by its length (number of items).The average-utility model provides an alternative measure to assess the utility of itemsets. Because the average utility measure considers the length of itemsets, it is more suitable and applicable in real-life situations, than the traditional measure.Lin et al. [9] then developed a high average-utility pattern (HAUP)-tree structure to mine HAUIs more efficiently. Lan et al. [11] developed a projection based average-utility (PBAU) mining algorithm to mine HAUIs. Lan et al. [12] then also extended the PBAU algorithm and designed a PAI approach using an improved strategy for mining the HAUIs. Lu et al. [14] then developed a HAUI-tree structure to mine HAUIs without candidate generation.
In the past, the MSApriori [7] was first proposed to mine FIs under multiple minimum support thresholds. The CFPgrowth algorithm [6] was then designed to build the MIS-tree and perform a recursive depth-first search to output the FIs. The MHU-Growth algorithm [15] extends CFP-Growth, to mine high utility frequent itemsets with multiple minimum support thresholds. Lin et al. [9] then developed the HUIM-MMU model for discovering HUIs with multiple minimum utility thresholds.

III. METHODOLOGY

Let $I = \{i1, i2, \ldots, ir\}$ be a finite set of r distinct items occurring in a database D, and $D = \{T1, T2, \ldots, Tn\}$ be a set of transactions, where for each transaction $Tq \in D$, Tq is a subset of I and has a unique identifier q, called its *TID* (transaction identifier). For each item ij and transaction Tq, a positive number $q(ij, Tq)$ represents the purchase quantity of ij in transaction Tq. Moreover, a profit table $ptable = \{p(i1), p(i2), \ldots, p(ir)\}$ is defined, where $p(im)$ is a positive integer representing the unit profit of item im ($1 \leq m \leq r$). A set of k distinct items $X = \{i1, i2, \ldots, ik\}$ such that $X \subseteq I$ is said to be a k-itemset, where k is the length or level of the itemset. An itemset X is said to be contained in a transaction Tq if $X \subseteq Tq$. An example quantitative database is shown in Table 1. It consists of five transactions and six items, represented using letters from (a) to (f). The profit of each item in Table 2

Table 1. A quantitative database.

TID	Items
1	*b*:7, *c*:2, *d*:3, *e*:1
2	*b*:4, *c*:3, *d*:3
3	*a*:2, *d*:1
4	*a*:1, *c*:6 , *f* :4
5	*b*:2, *c*:3, *d*:1, *f* :2

Table 2. A profit table.

Item	Profit
a	5
b	2
c	1
d	2
e	4
f	1

Definition 1. The minimum average-utility threshold to be used for an item ij in a database D is a positive integer denoted as $mau(ij)$). A *MMAU-table* is used to store the minimum average-utility thresholds of all items in D, and is defined as:

$$MMAU - table = \{mau(i1), mau(i2), \ldots, mau(ir)\}$$

In the following, it will be assumed that the minimum average-utility thresholds of all items for the running example are defined as: $\{mau(a):8,\ mau(b):8, mau(c):13,\ mau(d):14,\ mau(e):20,\ mau(f):9\}$

Definition 2. For a k-itemset X, the minimum average-utility threshold of X is denoted as $mau(X)$, and is defined as:

$$mau(X) = \frac{\sum_{ij \in X} mau(ij)}{|X|} = \frac{\sum_{ij \in X} mau(ij)}{k}$$

Definition 3. The average-utility of an item ij in a transaction Tq is denoted as $au(ij, Tq)$, and is defined as:

$$au(ij, Tq) = \frac{q(ij, Tq) \times p(ij)}{1}$$

where $q(ij, Tq)$ is the purchase quantity of item ij in Tq, and $p(ij)$ is the unit profit of item ij .

Definition 4. The average-utility of a k-itemset X in a transaction Tq is denoted as $au(X, Tq)$, and defined as:

$$au(X, Tq) = \frac{\sum_{ij \in X \wedge X \subseteq Tq} q(ij, Tq) \times p(ij)}{|X|} = \frac{\sum_{ij \in X \wedge X \subseteq Tq} q(ij, Tq) \times p(ij)}{k}$$

where k is the number of items in X.

Definition 5. The average-utility of an itemset X in a database D is denoted as $au(X)$, and is defined as:

$$au(X) = \sum_{X \subseteq Tq \wedge Tq \in D} au(X, Tq)$$

Definition 6. The average-utility upper bound of an itemset X is defined as the sum of the maximum utilities of transactions where X appears:

$$auub(X) = \sum_{X \subseteq Tq \wedge Tq \in D} mu(Tq)$$

where $mu(Tq)$ is the maximum utility of transaction Tq, defined as $mu(Tq) = max(q(ij, Tq) \times p(ij)) \forall ij \in I$.

Definition 7. An itemset X is a high average-utility upper-bound itemset (HAUUBI) if its *auub* is no less than its minimum average-utility threshold.
The set of all HAUUBIs is thus defined as:
$HAUUBI \leftarrow \{X | auub(X) \geq mau(X)\}$

Definition 8 (Least Minimum Average-Utility, LMAU). The least minimum average-utility (LMAU) in the *MMAU-table* is defined as:
$LMAU = min\{mau(i1), mau(i2), \ldots, mau(ir)\}$
where r is the total number of items.
The *LMAU* in the running example is calculated as $LMAU = min\{mau(a), mau(b), mau(f), mau(c), mau(d), mau(e)\} = min\{8, 8, 9, 13, 14, 20\}(= 8)$.

Theorem 1 (Transaction- Maximum-Utility Downward Closure Property,TMUDC Property). *Without loss of generality, assume that items in itemsets are sorted in ascending order of mau values. Let X^k be a k-itemset ($k \geq 2$) , and X^{k-1} be a subset of Xk of length $k-1$. If Xk is a HAUUBI, then X^{k-1} is also a HAUUBI.*

Theorem 2 (HAUIs $\subseteq$ HAUUBIs). *Let be an itemset X^{k-1} of length $k-1$ and X^k be one of its supersets. If X^{k-1} has an auub value lower than the LMAU,X^{k-1} is not a HAUUBI nor a HAUI, as well as all its supersets. Hence, X^{k-1} and its supersets can be discarded.*

Problem Statement: The purpose of HAUIM-MMS is to efficiently discover the set of all high average-utility itemsets, where an itemset X is said to be a HAUI if its average utility is no less than its minimum average-utility threshold $mau(X)$ as:

$HAUI \leftarrow \{X | au(X) \geq mau(X)\}$

IV. THE PROPOSED ALGORITHM

The designed baseline HAUI-MMS algorithm consists of two phases. In the first phase, the designed HAUI-MMS algorithm performs a breadth-first search to mine the HAUUBIs. In the second phase, an additional database scan is performed to identify the actual high average-utility itemsets from the set of HAUUBIs

Algorithm 1. HAUI-MMS
Input: *D*, a quantitative transactional databases; *ptable*, a profit table; *MMAU-table*, the user predefined multiple minimum average utility threshold table.
Output: The set of complete high average-utility itemsets (HAUIs).

1 find the *LMAU* in the *MMAU-table*;
2 scan *D* to find *auub*(*ij*);
3 **for** *each item ij* **do**
  if *auub*(*ij*) $\geq$ *LMAU* **then**
    *HAUUBI*1 $\leftarrow$ *HAUUBI*1 $\cup$ *ij* ;
4 sort items in *HAUUBI*1 in ascending order of their *mau* values;
5 set $k \leftarrow 2$;
6 **while** $HAUUBI^{k-1} \neq null$ **do**
  $Ck = generate\ candidate(HAUUBI^{k-1})$;
  for *each k-itemset X* $\in$ *Ck* **do**
  scan *D* to calculate *auub*(*X*);
  if *auub*(*X*) $\geq$ *mau*(*X*) **then**
    $HAUUBI^k \leftarrow HAUUBI^k \cup X$;
7 set $k \leftarrow k + 1$;
8 $HAUUBIs \leftarrow \cup\ HAUUBI^k$;
9 Set m$\leftarrow$*1;*
10 **for** *each itemset X and k in HAUUBIs* **do**
  for each m < k **do**
  X^a = FindSubset (X^k, 0, m);
  X^b = FindSubset (X^k,m, k);
  If($X^a \cup X^b = X$ & $X^a \cap X^b = \varnothing$)
  If ($X^a \not\subset$ HAUI & $X^b \not\subset$ HAUI)
  set $k \leftarrow k + 1$;
  Go to 10 **else**
  scan *D* to calculate *au*(*X*);
  if *au*(*X*) $\geq$ *mau*(*X*) **then**
  HAUIs $\leftarrow$ *HAUIs* $\cup$ *X*;
  return *HAUIs*;

V. EXPERIMENTS

Table 3 Experimental databases

Database	#Trans	#Items
FoodMart	21,556	1,559
chess	3,196	75

#|*D*| Transaction count
#|*I*| Number of distinct items

The method proposed in [7] for assigning the multiple thresholds to items was adapted to automatically set the *mau* value of each item in the proposed HAUIM-MMS framework. The following equation is thus used to set the *mau* value of each item *ij* :
$mau(ij) = max\{\beta \times p(ij), GLMAU\}$

where β is a constant used to set the *mau* values of an item as a function of its unit profit. To ensure randomness and diversity in the experiments, β was respectively set in different interval for varied datasets. The constant *GLMAU* is user-specified and represents the global least average-utility value. Lastly, $p(ij)$ represents the external utility (unit profit) of the item *ij*. If β is set to zero, a single minimum average-utility threshold *GLMAU* is used for all items. In that case, the task of HAUI-MMS would become the same as traditional HAUIM.

Table 4 The execution time and memory usage of two algorithms along with fixed value of GLMAU and varied β

Database	Value of B	Execution time (in sec)		Memory Usage (in MB)	
		Before Pruning	After Pruning	Before Pruning	After Pruning
foodmart (GLMAU =1k)	[1-10]	700	10	90	100
	[10-20]	200	0.5	90	92
	[20-30]	80	0.5	90	87
	[30-40]	70	0.5	90	82
	[40-50]	40	0.5	90	78
Chess(GLMAU=168k)	[1-200]	120	60	120	110
	[200-400]	120	58	120	102
	[400-600]	120	55	120	95
	[600-800]	85	35	120	90
	[200-1000]	65	20	120	84

VI. CONCLUSION

In this paper, the high average-utility itemset mining with multiple minimum average-utility thresholds (HAUIM-MMS) framework was designed to mine high average-utility itemsets (HAUIs) with multiple minimum average-utility thresholds. The baseline HAUI-MMS algorithm is a two phases algorithm, which relies on several designed theorems to find the HAUIs. An extensive experimental study was conducted on both synthetic and real datasets to evaluate the performance of the algorithms in terms of runtime, number of candidates, and memory usage. Results show that the designed algorithms can efficiently discover the HAUIs and can effectively reduce the search time after applying pruning strategy in second phase

REFERENCES

[1] Agarwal, R., Imielinski, T., Swami, A.: Database mining: a performance perspective.IEEE Trans. Knowl. Data Eng. **5**(6), 914–925 (1993)

[2] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in largedatabases. In: International Conference on Very Large Data Bases, pp. 487–499 (1994)

[3] Chan, R., Yang, Q., Shen, Y.D.: Mining high utility itemsets. In: IEEE International Conference on Data Mining, pp. 19–26 (2003)

[4] Fournier-Viger, P., Wu, C.-W., Zida, S., Tseng, V.S.: FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning. In: Andreasen, T.,Christiansen, H., Cubero, J.-C., Ra´s, Z.W. (eds.) ISMIS 2014. LNCS, vol. 8502,pp. 83–92. Springer, Heidelberg (2014)

[5] Hong, T.P., Lee, C.H., Wang, S.L.: Effective utility mining with the measure of average utility. Expert Syst. Appl. **38**(7), 8259–8265 (2011)

[6] Kiran, R.U., Reddy, P.K.: Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. In: ACM International Conference on Extending Database Technology, pp. 11–20 (2011)

[7] Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports.In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 337–341 (1999)

[8] Liu, Y., Liao, W., Choudhary, A.K.: A two-phase algorithm for fast discovery of high utility itemsets. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 689–695. Springer, Heidelberg (2005)

[9] Lin, C.-W., Hong, T.-P., Lu, W.-H.: Efficiently mining high average utility itemsets with a tree structure. In: Nguyen, N.T., Le, M.T., ´Swi _ atek, J. (eds.) ACIIDS 2010.LNCS, vol. 5990, pp. 131–139. Springer, Heidelberg (2010)

[10] Lin, J.C.W., Gan, W., Fournier-Viger, P., Hong, T.P.: Mining high-utility itemsets with multiple minimum utility thresholds. In: International C* Conference on Computer Science and Software Engineering, pp. 9–17 (2015)

[11] Lan, G.C., Hong, T.P., Tseng, V.S.: A projection-based approach for discovering high average-utility itemsets. J. Inf. Sci. Eng. **28**(1), 193–209 (2012)

[12] Lan, G.C., Hong, T.P., Tseng, V.S.: Efficiently mining high average-utility itemsets with an improved upper-bound strategy. Int. J. Inf. Technol. Decis. Making **11**(5),1009–1030 (2012)

[13] Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In:ACM International Conference on Information and Knowledge 12)

[14] Lu, A new method for mining high Sn´a˘sel, V. (eds.) CISIM 2014. Heidelberg (2014)

[15] Ryan high utility itemsets with multiple Anal. 1047 (2014)

[16] Yao tional approach to mining itemset ternational Conference on Data Mi

[17] Yao, H utilities from transaction databases. Data Knowl. Eng. 59(3), 603–626 (2006)