IJCTA, 9(26) 2016, pp. 153-160
© International Science Press

An Improved Transaction Weighted Itemlist for Mining High Utility Itemset

V. G. Vijilesh[1] and S. Hari Ganesh[2]

ABSTRACT

High utility itemset mining is an emerging trend in data analysis that exhibits the traders with the most profitable items for the promotion of the products. The traditional literature of high utility itemset mining algorithms rely upon candidate set generation which consumes more execution time and memory space. HHUI-Miner is a novel high utility itemset algorithm that mines the transactions set by constructing a k-utility-itemset. The method computes the transaction-weighted utility itemset by adding the product of quantity with utility and multiplies with the total number of occurrence of items in overall transactions. This would help in consuming the time and memory spaces and an additional construction of transaction utility table. The proposed method is compared with HUI-Miner and UP Growth algorithms to prove the novelty and superiority of the algorithm.

Keywords: utility, itemset, HUI-Miner, UP Growth, exhaustive search, transaction.

1. INTRODUCTION

The rapid growth of data let the database techniques to facilitate the storage and appliance of massive analysis on business corporations, governments, medical, education and scientific researches. Frequent itemset mining is a popular market basket analysis which aims at analyzing the regularities of the shopping habitual of customers in online shopping or departmental stores. To be more specific it finds the collections of products that are frequently bought together. The applications of frequent itemset mining include the arrangement of products in the racks or to be displayed in the catalogue pages, supporting product suggestions and detection of frauds and so on.

High utility itemset is the one of the applications of frequent itemset mining which refers to the discovery of itemsets that offers high utility like profit [5]. One of the important problems in frequent itemset mining is the high utility itemset mining problem as traditional frequent itemset mining algorithms do not have the ability to evaluate the utility information about itemsets. For instance, in a supermarket database each item has a unique price and profit and each item in a transaction is associated with the distinct quantity. The existing high utility itemset algorithms works on the basis of generating candidate for high utility itemsets and computes the exact utilities for the candidates to identify the high utility itemsets by scanning the database [6]. But the algorithms generate a very large number of candidate itemsets which in turn occupies excessive memory space and large amount of execution time.

Hence, the goal of this paper is to propose a novel utility-list structure to extract the utility information about an itemset through heuristic approach. Moreover, this paper also introduces a HHUI-Miner (Heuristic High Utility Itemset Miner to mine the high utility itemsets after the construction of initial utility-lists.

2. REVIEW OF LITERATURE

Several algorithms have been developed for high utilityitemset mining such as, IHUP [1], UP-Growth [2] and Two-Phase [3]. Two-Phase algorithm [3] is proposed by Liu et al. the algorithm contains two phases. In

[1] Senior Lecturer, International Business, Science and Technology University, ISBAT University, Kampala, Uganda.
[2] Assistant Professor, Department of Computer Science, H.H. The Rajah's College, Pudukottai.

phase I, Two-Phase algorithm makes use of Apriory based technique to enumerate HTWUIs. It creates next set of candidate itemsets from the previous set of candidate itemsets and prunes candidate itemsets by TWDC property. In each pass, HTWUIs and their estimated utility values are calculated by scanning the database. After this, the complete set of HTWUIs is collected. In phase II, the original database is scanned to discover the high utility itemsets and their utilities.

Although Two-Phase algorithm efficiently reduces the search space and discovers the complete set of high utility itemsets, it still creates too many candidates for HTWUIs and requires multiple database scans. To overcome this issues Ahmed et al. [1] have developed a tree-based algorithm, called IHUP. To retain the information of high utility itemsets and transactions the algorithm uses an IHUP-Tree. Every node in IHUP-Tree contains an item name, a support count, and a TWU value. The algorithm works in three steps, in first step, items in the transaction are rearranged in a fixed order such as lexicographic order. The IHUP-tree is then created using rearranged transactions. In the second step, HTWUIs are created from the IHUP-Tree. In third step, by scanning the original database, high utility itemsets and their utilities are recognized from the set of HTWUIs.

Even though IHUP discovers HTWUIs without creating any candidates for HTWUIs and attains a better performance than Two-Phase, it still produces too many HTWUIs in phase I. To overcome this problem, Vincent S. Tseng, Cheng-Wei Wu, Bai-En Shie, and Philip S. Yu [2] developed the UP-Growth algorithm. A compact tree structure, called utility pattern tree (UP-Tree), for find out high utility itemsets and maintaining significant information related to utility patterns within databases are projected. High-utility itemsets can be created from UP-Tree efficiently with only two scans of original databases. Four new strategies are proposed namely DGU, DGN, DLU and DLN. First two strategies are applied on UP Tree to globally reduce unpromising items from obtained potential high utility itemsets [4]. The next two strategies namely DLU and DLN are applied by the UP-Growth on the UP-Tree for reducing the local unpromising items. The actual high utility itemsets are then defined from a set of potential high utility itemsets.

All these algorithms first create candidate itemset which needs more time and space. But the proposed HHUI algorithm do not generate candidate for high utility itemsets.

3. HHUI-MINER – PROPOSED METHODOLOGY

The architectural diagram for the proposed Heuristic High Utility Itemset Miner is presented in fig.1. which denotes the extraction of transaction set from databases, utility list construction and HHUI mining.

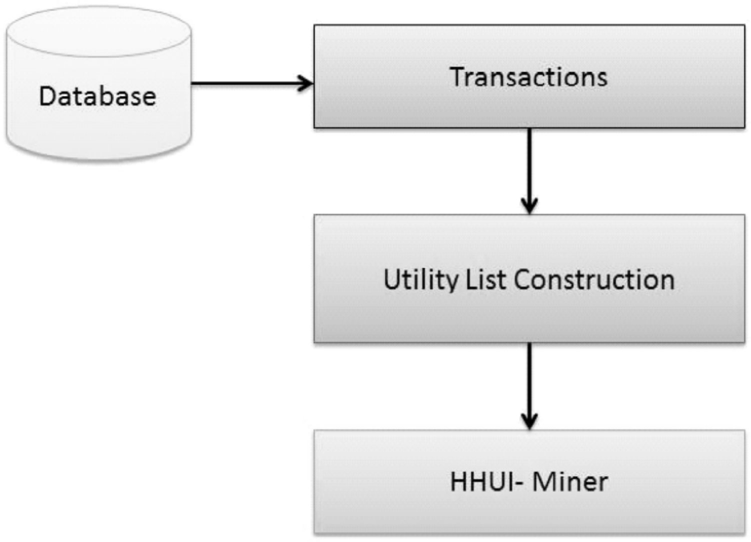


Figure 1: Architecture of HHUI-Miner

Definitions and Terminologies

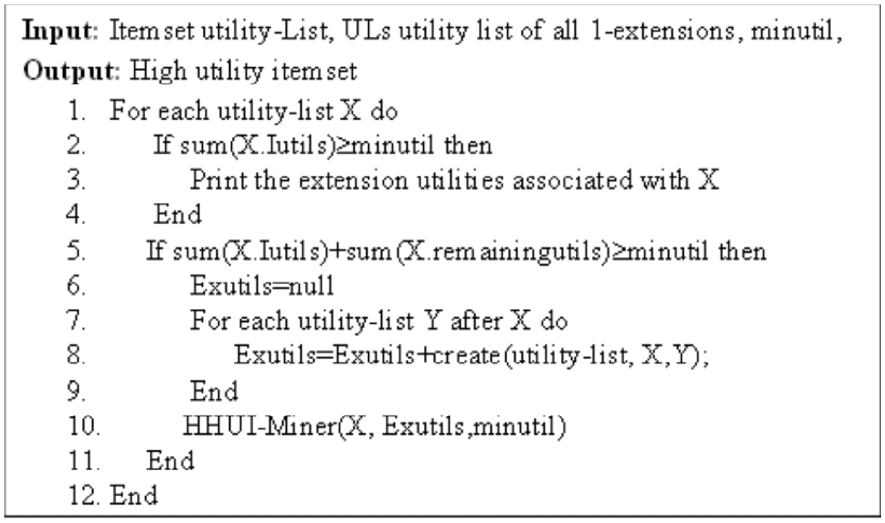
Let *I* be a set contains *‘n’* items such that*I= (i_1, i2, i3, i4….i_n)* and T be a transaction denoted by unique identifier *(t_{id})*. The construction of HHUI- miner is defined as [8]

- Internal Utility *iu (i, T)* refers to the quantity associated with an item i in the transaction table.
- External Utility*eu (i)* refers to the profit of item i in the utility table.
- Utility*u (i, T)* is the product of *iu (i, T)* in transaction table *eu (i)* in utility table.
- The utility of itemset X in transaction T *u(X,T)* is the sum of utilities of all items in X in T
- The utility of itemset X *u(X)* is the sum aggregate of the utilities of X in all the transaction in the database
- The utility of transaction T *tu(T)* is the sum of the utilities of all the items in T
- The transaction –weighted utility of itemset X *twu(X)* in database is the product of quantity with utility of each item xof all the transactions containing X in database.
- The transactions are revised by deleting all the items whose weighted utilities are less than *‘minutil’* and sorting the remaining items in ascending order
- The set of all items after an itemset X in Transaction T is denoted as *T/X*
- The remaining utility of itemset X in transaction T is the accumulation of the utilities of all item in T/X which is denoted as *ru(X,T)*
- The construction of utility-list of k-itemset is the intersection of utility list of concatenating items
- Finally, the Set-Enumeration tree is constructed by representing an itemset by a node which is an itemset of an ancestor node.

Properties of High utility itemset

1. If the weighted utility of itemset X is less than the user given ‘minutil’, then the particular item is considered as a non-profitable or low utility item [7].
2. If an extension of X called X? is a combination of X with one or more items after X.
3. If the accumulation of all itemsetutility (*Iutil*) and remaining utility (*rutil*) in the utiliy –list is less than a user given ‘*minutil*’ then, the remaining utility X? is not high utility.

The proposed algorithm for HHUI miner is depicted in Fig. 2.



```
Input: Itemset utility-List, ULs utility list of all 1-extensions, minutil,
Output: High utility itemset
   1.  For each utility-list X do
   2.      If sum(X.Iutils)≥minutil then
   3.          Print the extension utilities associated with X
   4.      End
   5.      If sum(X.Iutils)+sum(X.remainingutils)≥minutil then
   6.          Exutils=null
   7.          For each utility-list Y after X do
   8.              Exutils=Exutils+create(utility-list, X,Y);
   9.          End
   10.         HHUI-Miner(X, Exutils,minutil)
   11.     End
   12. End
```

Figure 2: HHUI –Miner Algorithm

4. ILLUSTRATION

The step by step analysis of the proposed methodology is described in this section. The following tables 1 and 2 denote the transaction and utility tables respectively.

Table 1
Transaction Table

Transaction Id (Tid)	*Transaction (Count)*
Tid_1	{u(1),v(2),w(1),z(1)}
Tid_2	{t(4),u(1),v(3),w(1),x(1)}
Tid_3	{t(4),v(2),w(1)}
Tid_4	{v(2),x(1),y(1)}
Tid_5	{t(5),u(2),w(1),x(2)}
Tid_6	{t(3),u(4),v(1),y(2)}
Tid_7	{w(1),z(5)}

Table 2
Utility Table

Item	*Utility*
t	1
u	2
v	1
w	5
x	4
y	3
z	1

The novelty of the proposed HHUI-Miner is the computation of transaction-weighted utilities of itemset which is the sum of product of quantity with the profit. The sum is then multiplied with the occurrence of the item on the whole transactions. Table 3 represents the transaction-weighted utility of items. where*q* is the quantity, *u* is the utility, *s* is sum of quantity and utility of each item,*o* is the number of occurences and *twu*is the product of *s* and *o*.

Table 3
Construction of Transaction –weighted utility of items

Item	*T1 (q,u)*	*T2 (q,u)*	*T3 (q,u)*	*T4 (q,u)*	*T5 (q,u)*	*T6 (q,u)*	*T7 (q,u)*	*s*	*o*	*Twus,o*
t	-	4,1=4	4,1=4	-	5,1=5	3,1=3	-	16	4	64
u	1,2=2	1,2=2	-	-	2,2=4	4,2=8	-	16	4	64
v	2,1=2	3,1=3	2,1=2	2,1=2	-	1,1=1	-	10	5	50
w	1,5=5	1,5=5	1,5=5	-	1,5=5	-	1,5=5	25	5	125
x	-	1,4=4	-	1,4=4	2,4=8	-	-	16	3	48
y	-	-	-	1,3=3	-	2,3=6	-	9	2	18
z	1,1=1	-	-	—	-	-	5,1=5	6	2	12

According to property 1, if the *twu(X)* is less than *minutil*, the all supersets of *X* are not high utility. Supposing if the *minutil* =30, then *twu({y,z})={18,12}<30*. Hence, all supersets of *{y,z}* are not high utilities. The revised transactions of all items whose *twu* are greater than the *minutil* are ordered in ascending sequence for creating the initial utility-lists. The sorted items according to *twu*are *x?v?u?t?w*. Table 4. presents the revised transactions

Table 4
Revised Transactions Table

Tid	*Item*	*Util.*	*Item*	*Util.*	*Item*	*Util.*	*Item*	*Util.*	*Item*	*Util.*	*TU*
Tid_1	v	2	u	2	w	5			9		
Tid_2	x	4	v	3	u	2	t	4	w	5	18
Tid_3	v	2	t	4	w	5					11
Tid_4	x	4	v	2							6
Tid_5	x	8	u	4	t	5	w	5			22
Tid_6	v	1	u	8	t	3					12
Tid_7	w	5									5

The remaining utility of itemset X in transaction T is computed by adding the utilities of all the remaining items after X as shown in fig. 3.

{x}

2	4	14
4	4	2
5	8	14

{v}

1	2	7
2	3	11
3	2	9
4	2	0
6	1	11

{u}

1	2	5
2	2	9
5	4	10
6	8	3

{t}

2	4	5
3	4	5
5	5	5
6	3	0

{w}

1	5	0
2	5	0
3	5	0
5	5	0
7	5	0

Figure 3: Initial Utility Lists

The creation of 2 item-sets utility –lists is shown fig. 4.

{x}

2	4	14
4	4	2
5	8	14

{v}

1	2	7
2	3	11
3	2	9
4	2	0
6	1	11

{xv}

2	7	11
4	6	0

Figure 4: Utility-list of 2-itemsets

The creation of k-itemsets utility-list for the itemset{eba} is shown fig.5.

{xu}

2	6	9
5	12	10

{xt}

2	8	5
5	13	5

{xut}

2	10	5
5	17	5

Figure 5: Utility-list of k-itemsets

Once when the construction of k-itemsets utility-list is over a set-Enumeration tree is constructed in which an itemset is represented by a node called an extension of itemset which is representation of an ancestor node of the node. The set-Enumeration tree of the revised transaction table is depicted in fig. 6.

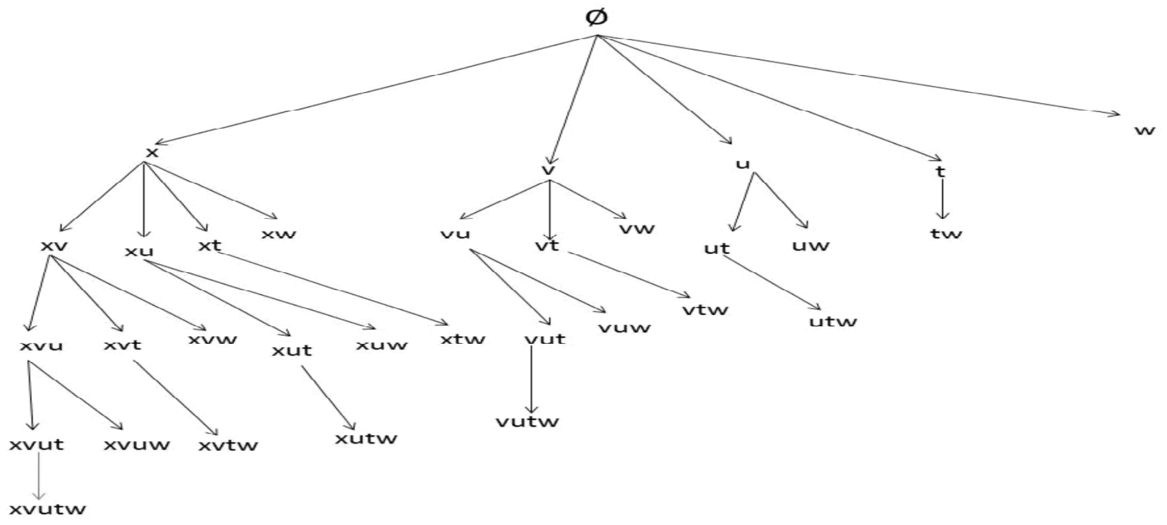


Figure 6: Set-Enumeration Tree

The tree is pruned using exhaustive searching strategy which is time consumingthan any other searching algorithms. The high utility itemset is computed by adding the sum of all the itemsetutils called iutils and remainingutils called rutils. If the sum of all iutils and rutils of the util-list is less than a given 'minutil', the extension of X? of X is not high utility.

5. EXPERIMENTATION

Performance of proposed algorithm is evaluated with 2.20 GHz Core2 Duo Processor with 2GB memory. The operating system is Windows 8. The algorithms are implemented in Java language. The data setis obtained from FIMI Repository. The description of the dataset is given in Table 5.

Table 5
Dataset Description

Dataset Name	*Size*	*Transactions*	*Items*	*Average Length*
Chess	642Kb	3196	75	37

The algorithm is compared with two algorithms HUI-Miner and UP-Growth based on two parameters such execution time and consumption of memory. The results yielded by the proposed algorithms are presented here.

Execution Time

When the execution time of the chess dataset exceeded 10000 seconds the mining task is terminated, and the results have proven that performance of HHUI-Miner is superior in terms of execution time as the process directly starts from transaction –weighted utility items. The results become very significant when the minutil decreases. Fig. 7. depicts the execution time of the HHUI-Miner

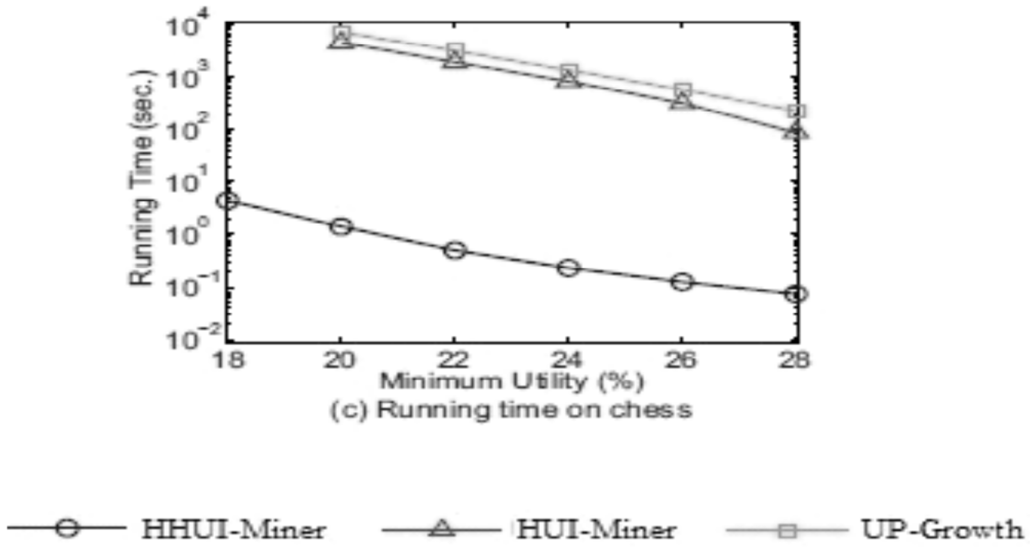


Figure 7: Execution Time- HHUI -Miner

Memory Consumption

HHUI-Miner consumes less memory space than HUI-Miner as the memory space needed to store the sum of the transaction utility is consumed as it has been removed from the process. Moreover, all the computations are performed using the transaction-weighted utility items itself. The memory consumption for UP Growth algorithm is bit higher than the HUI-Miner algorithm. Fig. 8. shows the memory consumption of HHUI-Miner.

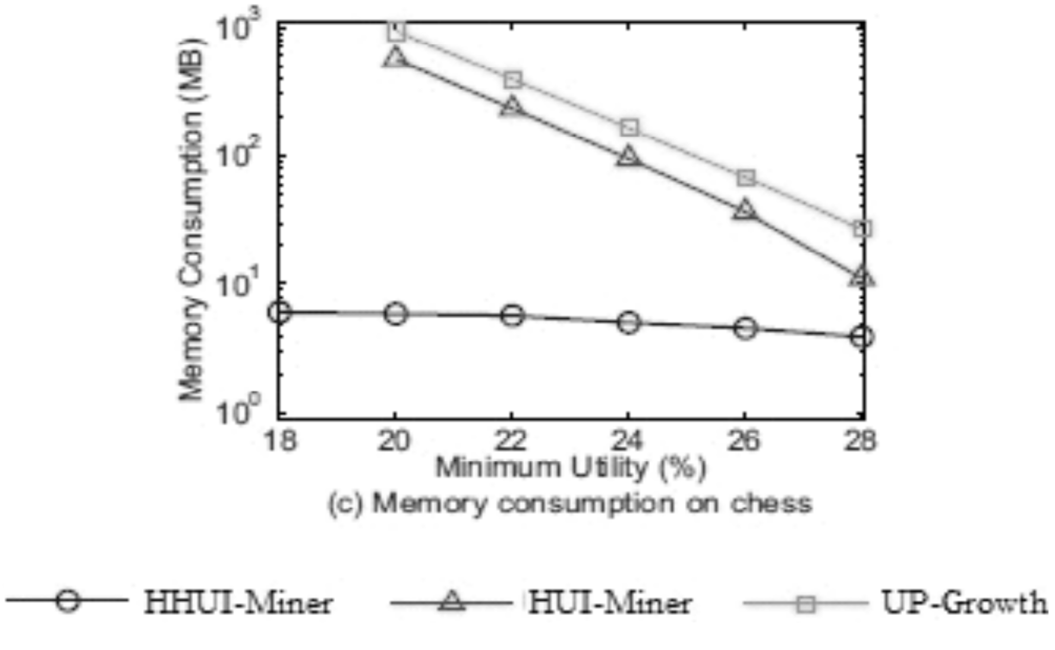


Figure 8: Memory Consumption- HHUI-Miner

6. CONCLUSION

This paper proposes a novel transaction –weighted utility list called HHUI-Miner for mining the high utility itemset. The utility list provided by the HHUI-Miner not only provides the information about the itemsets but also explicates the steps of pruning HHUI-Miner. One of the biggest advantages of HHUI-Miner is its ability to identify the high utility itemset without generating a candidate set which is time consuming. Moreover, the performance of HHUI-Miner is significant with respect to time and space as the transaction-weighted utility evades the computation and storage of transaction utility table. One of the limiting factors of high utility itemset algorithms is the user intervention for setting the minutil, which can be automated in future.

REFERENCES

[1] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee. "Efficient tree structures for high utility pattern mining in incremental databases."*In IEEE Transactions on Knowledge and Data Engineering*, 1708-1721, 2009.

[2] Vincent S. Tseng, Bai-En Shie, Cheng Wei Wu, and Philip S. Yu, Fellow, "Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases" *IEEE Transactions on Knowledge and Data Engineering*, 1772-1786, 2013.

[3] Liu, Ying, Wei-keng Liao, and AlokChoudhary. "A fast high utility itemsets mining algorithm." In *Proceedings of the 1st international workshop on Utility-based data mining*, pp. 90-99, 2005.

[4] Pillai, Jyothi, and O. P. Vyas. "Overview of itemset utility mining and its applications." *International Journal of Computer Applications* 5, 9-13, 2010.

[5] Ravi, M.A.P.M.D., Subathra, M.R. and Coimbatore, C.C., "HUI Miner Algorithms for Transactional Databases.",*International Journal On Engineering Technology and Sciences*, 5-13, 2015.

[6] Liu, Mengchi, and Junfeng Qu. "Mining high utility itemsets without candidate generation." *In Proceedings of the 21st ACM international conference on Information and knowledge management*, 55-64, 2012.

[7] Guo, Shi-Ming, and Hong Gao. "HUITWU: An Efficient Algorithm for High-Utility Itemset Mining in Transaction Databases." *Journal of Computer Science and Technology*, 776-786, 2016.

[8] Fournier-Viger, Philippe, Cheng-Wei Wu, SouleymaneZida, and Vincent S. Tseng. "FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning." *In International Symposium on Methodologies for Intelligent Systems*, 83-92. 2014.