ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 7, July 2017

Mining of Top-K High Utility Itemsets from Transaction Database using a Single Phase Algorithm (TKSA) and an Algorithm ACE for Customer Evaluation

G.Gayathri[1], J.Nagamuneiah[2]

P.G. Student, Department of CSE, Chadalawada Ramanamma Engineering College, Tirupati, India[1]

Professor, Department of CSE, Chadalawada Ramanamma Engineering College, Tirupati, India[2]

ABSTRACT: Utility mining is a rising research domain in the data mining. High utility itemsets (HUIs) mining refers to discovering all itemsets having a utility meeting a user-specified minimum utility threshold min_util. Setting a appropriate value for min_util is a difficult problem for users. Also, finding an appropriate minimum utility threshold value is a tedious process for users. If min_util is set too low, too many HUIs will be generated, which may cause the mining process to be very inefficient. On the other hand, if min_util is set too high, it is likely that no HUIs will be found. To address this limitation an algorithm TKSA is proposed for mining top k-HUIs in a single phase without candidate generation. Also another algorithm ACE is proposed for mining HUIs which is completely based on Customer frequency.

KEYWORDS : Utility Mining, High Utility Itemset, Frequent Itemset Mining, Top –k pattern Mining.

I. INTRODUCTION

Data Mining is an inter-disciplinary subject applied by many businesses such as health care, retail, manufacturing etc. for retrieving valuable information from the large amount of data which are used to make better business decisions. Finding interesting patterns has been an important data mining task, and has a variety of applications, for example, genome analysis, condition monitoring, cross marketing, and inventory prediction, where interestingness measures play an important role. With frequent pattern mining, a pattern is regarded as interesting if its occurrence frequency exceeds a user specified threshold. For example, mining frequent patterns from a shopping transaction database refers to the discovery of sets of products that are frequently purchased together by customers.

The traditional Frequent Itemset Mining(FIM) may discover a large amount of frequent but low-value itemsets and lose the information on valuable itemsets having low selling frequencies. Hence, it cannot satisfy the requirement of users who desire to discover itemsets with high utilities such as high profits. To address these issues, utility mining emerges as an important topic in data mining. In utility mining, each item is associated with a utility (e.g. unit profit) and an occurrence count in each transaction (e.g. quantity). The utility of an itemset represents its importance, which can be measured in terms of weight, value, quantity or other information depending on the user specification. An itemset is called high utility itemset (HUI) if its utility is no less than a user-specified minimum utility threshold min_util.

Efficient mining of HUIs in databases is not an easy task because the downward closure property used in FIM does not hold for the utility of itemsets. In other words, pruning search space for HUI mining is difficult because a superset of a low utility itemset can be high utility. To tackle this problem, the concept of transaction-weighted utilization (TWU) model was introduced to facilitate the performance of the mining task. In this model, an itemset is called high transaction-weighted utilization itemset (HTWUI) if its TWU is no less than min_util, where the TWU of an itemset represents an upper bound on its utility.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 7, July 2017

Although many studies have been devoted to HUI mining, it is difficult for users to choose an appropriate minimum utility threshold in practice. Depending on the threshold, the output size can be too small or too large. Besides, the choice of the threshold greatly influences the performance of the algorithms. If the threshold is set too low, too many HUIs will be presented to the users and it is difficult for the users to comprehend the results. On the contrary, if the threshold is set too high, no HUI will be found. To find an appropriate value for the min_util threshold, users need to try different thresholds by guessing and re-executing the algorithms many times until being satisfied with the results. This process is both inconvenient and time-consuming.

Most of the prior utility mining algorithms with the itemset share framework does not address the above challenges and adopt a two-phase, candidate generation approach. They first find candidates of high utility patterns in the first phase, and then scan the raw data one more time to identify high utility patterns from the candidates in the second phase. The challenge is that the number of candidates can be huge, which is the scalability and efficiency bottleneck.

To effectively control the output size and discover the itemsets with the highest utilities without setting the thresholds, a promising solution is to redefine the task of mining HUIs as mining top-k high utility itemsets (top-k HUIs). The idea is to let the users specify k, i.e., the number of desired itemsets, instead of specifying the minimum utility threshold. Setting k is more intuitive than setting the threshold because k represents the number of itemsets that the users want to find whereas choosing the threshold depends primarily on database characteristics, which are often unknown to the users. Therefore a novel Algorithm named TKSA is proposed which identifies the complete set of top-k HUIs in single phase without the need to specify the min_util threshold. For TKSA, the novel strategy RUC (Raising the threshold by the Utilities of Candidates)for pruning the search space is integrated. In addition to TKSA, another algorithm named ACE which is completely based on Customer frequency is proposed for mining HUIs.

The rest of the paper is organized as follows. Section II presents related works and problem definition. Section III presents the proposed system. Section IV presents the TKSA which is completely profit oriented and the ACE algorithm which is based on customer frequency and Section V presents Experimental Evaluation followed by conclusion and future work.

II. RELATED WORKS

In recent years, high utility itemset mining has received lots of attention and many efficient algorithms have been proposed. Some prior techniques introduced for mining High Utility Itemsets are briefly listed.

Philippe Fournier-Viger, Cheng-Wei Wu, Souleymane Zida, Vincent S.Tseng [3] proposed an algorithm named FHM which reduces the number of join operations by introducing a novel structure named EUCS(Estimated Utility Co-occurrence Structure). However FHM supports only static databases and it is time-consuming.

Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu, proposed an efficient algorithm for mining high utility itemsets from transactional databases. In this work, two algorithms, namely utility pattern growth (UP-Growth) and UP-Growth+, is proposed for mining high utility itemsets with a set of effective strategies for pruning candidate itemsets. The information of high utility itemsets is maintained in a tree-based data structure named utility pattern tree (UP-Tree) such that candidate itemsets can be generated efficiently with only two scans of database. Since this framework requires multiple phases to scan the database it is highly complex.

Sen Su, Shengzhi Xu, Xiang Cheng, Zhengyi Li, and Fangchun Yang proposed a private FP-growth algorithm, which is referred to as PFP-growth. The PFP-growth algorithm consists of a pre-processing phase and a mining phase. In the pre-processing phase, to improve the utility and privacy trade off, a novel smart splitting method is proposed to transform the database. For a given database, the pre-processing phase needs to be performed only once. In the mining phase, to offset the information loss caused by transaction splitting, we devise a run-time estimation method to estimate the actual support of itemsets in the original database. The limitation is that if itemsets of the same length are generated simultaneously, the number of FP-trees stored in memory will grow at an exponential rate.

The main characteristic of one-phase algorithms is that they discover high utility itemsets using only one phase and produce no candidates. Junqiang Liu, Ke Wang, Benjamin C.M. Fung proposed an algorithm, d^2HUP, namely Direct Discovery of High Utility Patterns, for mining high utility itemsets in one phase without candidate generation. It is an integration of the depth-first search of the reverse set enumeration tree, the pruning techniques that drastically reduces the number of patterns to be enumerated, and a novel data structure that enables efficient computation of utilities and upper bounds. HUI-Miner considers a database of vertical format and transforms it into utility-lists. The utility-list

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 7, July 2017

structure used in HUI-Miner allows directly computing the utility of generated itemsets in main memory without scanning the original database.
Although the above studies may perform well in some applications, they are not developed for top-k high utility itemset mining and they still suffer from the subtle problem of setting appropriate thresholds.

III. PROBLEM DEFINITION

Let be a finite set of distinct items $I^* = \{I_1, I_2, \ldots, I_N\}$, such that each item $I_i \in I^*$ is associated with a positive number called its external utility (e.g. unit profit). A transactional database $D = \{T_1, T_2, \ldots, T_M\}$ is a set of transactions, where each transaction $T_r \in D$ is a subset of I^* and has an unique identifier r, called its Tid. In a transaction T_r, each item $Ii \in I^*$ is associated with a positive number called its internal utility in Tr (e.g. purchase quantity). An itemset $X = \{ I_1, I_2, \ldots, I_k \}$ is a set of k distinct items, where $I_i \in I^*$ $(1 \leq i \leq k)$ and k is the length of X. A k-itemset is an itemset of length k.

Table 1. An example transaction database

TID	CID	Transaction
T_1	C_1	(p,1)(r,1)(s,1)
T_2	C_2	(p,2)(r,6)(t,2)(v,5)
T_3	C_3	(p,1)(q,2)(r,1)(s,6)(t,1)(u,5)
T_4	C_4	(q,4)(r,3)(s,3)(t,1)
T_5	C_1	(q,2)(r,2)(t,1)(v,2)

Table 2. External utility values

Item	p	q	r	s	t	u	v
Profit	5	2	1	2	3	1	1

Table 3. Transaction utility values(TU)

TID	T_1	T_2	T_3	T_4	T_5
TU	8	27	30	20	11

Table 4. TWU values of the items

Item	p	q	r	s	t	u	v
TWU	65	61	96	58	88	30	38

Let Table 1 be an example database containing five transactions. Each row in Table 1 represents a transaction, where each letter represents an item and has a purchase quantity (i.e., internal utility). CID represents customer IDs. The unit profit (i.e., external utility) of each item is shown in Table 2.

Definition 1 (Absolute utility of an item). The absolute utility of an item $I_j \varepsilon\ I^*$ in a transaction T_r is denoted as $EU(I_j,T_r)$ and defined as the product of internal and external utility.

Definition 2 (Absolute utility of an itemset in a transaction). The absolute utility of an itemset X in a transaction T_r is defined as $EU(X, T_r)=\Sigma_{Ij \varepsilon X} (I_j,T_r)$

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 7, July 2017

Definition 3 (Transaction utility and total utility). The transaction utility (TU) of a transaction Tr is defined as TU (T_r) = EU(T_r,T_r). The total utility of a database D is denoted as $TotalU_{DB}$ and defined as $\Sigma_{Tr \in D}$ TU(T_r). Table 3 shows transaction utility values calculated for each transaction.

Definition 4 (Utility of an itemset in a database). The (relative) utility of X is defined as U(X)=EU(X)/ $TotalU_{DB}$.

Definition 5 (High utility itemset). An itemset X is called high utility itemset (HUI) iff U(X) is no less than a user-specified minimum utility threshold min_util (0% ≤ min_util < 100%). Otherwise, X is a low utility itemset. Also an itemset X is high utility iff EU(X) ≥ abs_min_util, where abs_min_util=min_util × TotalUDB.

Definition 6 (High utility itemset mining). Let δ (0% < δ < 100%) be the minimum utility threshold, the complete set of HUIs in D is denoted as f_{HUI}(D, δ). The goal of HUI mining is to discover f_{HUI} (D, δ).

Because the utility of an itemset may be equal to, higher or lower than that of its supersets and subsets, we cannot directly use the anti-monotone property (also known as downward closure property) to prune the search space. To facilitate the mining task, Liu et al. introduced the concept of transaction-weighted downward closure property.

Definition 7: (Transaction-weighted utilization). The transaction-weighted utilization of an itemset X is the sum of the transaction utilities of all the transactions containing X. Table 4. shows TWU values for each item in the database.

Definition 8: (High TWU itemset). An itemset X is a high TWU itemset iff TWU(X) ≥abs_min_util.

Property 1:(TWDC property). The TWDC (Transaction Weighted Utilization Downward Closure) property states that for any itemset X if its TWU(X) is less than min_util, all supersets of X are low utility.

Property 2: Let KH be the complete set of top-k HUIs in D. KH may contain less than k itemsets when |f_{HUI}(D, 0) | < k. Besides, KH may contain more than k HUIs when some itemsets have the same utility.

IV. PROPOSED SYSTEM

The algorithm TKSA (Top-K High utility Itemset Mining in single phase) can discover top-k HUIs in only one phase. It utilizes the basic search procedure of HUI-Miner and its utility-list structure. Whenever an itemset is generated by TKO, its utility is calculated by its utility-list without scanning the original database. The proposed algorithm consists of two parts: (1) construction of initial utility-lists and (2) generation of top-k HUIs using the utility-lists.

Construction of Utility-List Structure:

The utility-lists of items are called initial utility-lists, which can be constructed by scanning the database twice. In the first database scan, the TWU and utility values of items are calculated. During the second database scan, items in each transaction are sorted in order of TWU values and the utility-list of each item is constructed.

Table 5. Transactions rearranged according to TWU values

TID	CID	Transaction	Transaction Utility (TU)
T_1	C_1	(s,1)(p,1)(r,1)	8
T_2	C_2	(v,5)(p,2)(t,2)(r,6)	27
T_3	C_3	(u,5)(s,6)(q,2)(p,1)(t,1)(r,1)	30
T_4	C_4	(s,3)(q,4)(t,1)(r,3)	20
T_5	C_1	(v,2)(q,2)(t,1)(r,2)	11

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 7, July 2017

Fig 1. Initial utility-lists

v		
2	5	22
5	2	9

s		
1	2	6
3	12	13
4	6	14

p		
1	5	1
2	10	12
3	5	4

u		
3	5	25

q		
3	4	9
4	8	6
5	4	5

t		
2	6	6
3	3	1
4	3	3
5	3	2

r		
1	1	0
2	6	0
3	1	0
4	3	0
5	2	0

Table 5 shows the example database where items in each transaction are arranged in ascending order of TWU values. Fig.1 shows the utility-lists of items for the database in Table 5. Each element in the utility-list of itemset X contains three fields: <tid, iutil, and rutil>. Field tid indicates a transaction T containing X. Field iutil is the utility of X in T, i.e., u(X, T). Field rutil is the remaining utility of X in T, i.e., ru(X, T) which is defined as follows.

Definition 9: The remaining utility of itemset X in transaction T, denoted as ru(X, T), is the sum of the utilities of all the items after X in T, denoted as ru(X, T) = $\Sigma_{i \in (T/X)}$ u(i, T). [T/X denotes items after X in T]

The utility-list of 2-itemset {xy} can be constructed by the intersection of the utility list of {x} and that of {y}. The algorithm identifies common transactions by comparing the tids in the two utility-lists. Suppose the lengths of the utility-lists are m and n respectively, and then (m + n) comparisons are enough for identifying common transactions, because all tids in a utility-list are ordered.

To construct the utility-list of k-itemset $\{i_1 \cdots i_{(k-1)} i_k\}$ ($k \geq 3$), we can directly intersect the utility-list of $\{i_1 \cdots i_{(k-2)} i_{(k-1)}\}$ and that of $\{i_1 \cdot \cdot i_{(k-2)} i_k\}$ as we do to construct the utility-list of a 2-itemset. Generally, to calculate the utility of $\{i_1 \cdot \cdot \cdot i_{(k-2)} i_{(k-1)} i_k\}$ in T, the following formula holds: $u(\{i_1 \cdot \cdot \cdot i_{(k-2)} i_{(k-1)} i_k\}, T) = u(\{i_1 \cdots i_{(k-2)} i_{(k-1)}\}, T) + u(\{i_1 \cdots i_{(k-2)} i_k\}, T) - u(\{i_1 \cdots i_{(k-2)}\}, T)$.

Property 3: Given the utility-list of itemset X, if the sum of all the iutils and rutils in the utility-list is less than a given minutil, any extension of X is not high utility.

V. TKSA ALGORITHM

The TKSA algorithm takes as input the parameter k and a transactional database D transformed into vertical format such as initial utility-lists. The minimum utility threshold min_util is initially set to 0 and a min-heap structure TopK-CL-List is initialized for maintaining the current top-k HUIs during the search. In this algorithm items are sorted in transaction-weighted-utility-ascending order and processed in the same order.

Mining of top k-HUI patterns by using RUC strategy

Mining of top k-HUI patterns introduces a RUC (Raising the threshold by the Utilities of Candidates) strategy. This strategy can be incorporated with any one-phase mining algorithm where itemsets are found with their utilities. It adopts the Top K-List structure to maintain top-k HUIs, where itemsets are sorted by descending order of utility. Initially, Top K-List is empty. When an itemset X is found by the search procedure and its utility is no less than min_util, X is added to Top K-List. If there are more than k itemsets already in Top K-List, min_util can be safely raised to the utility of the k-th itemset in Top K-List. After that, itemsets having a utility lower than the raised min_util are removed from Top K-List.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 7, July 2017

PROCEDURE: TKSA

Input: (i) u(P) utility-list for some prefix P
(ii) ItemsP: set of itemsets containing the prefix P
(iii) ULSP: set of utility-lists containing the prefix P
(iv) min_util: minimum utility threshold
(v) TopK-CL-List: list for storing candidate itemsets
Result: Top-K-CL List gives the top K HUIs

```
1.  For each utility-list X in ItemsP do
2.    If (SUM(X.iutils) ≥ min_util )then
        // Raise the min_util by the RUC strategy
3.     min_util = RUC( X, TopK-CL-List)
4.    End
5.    If ((SUM(X.iutils) + SUM(X.rutils)) ≥ min_util)
6.      ItemsP= NULL ; ULSP= NULL
7.       For each utility list Y after X do
8.          Construct the utility list for the itemset XY
9.          Append the constructed utility list to be the new ULSP
10.         Append the itemset XY to ItemsP
11.      End
12.      //Call Recursive procedure
13.     TKSA(X, ItemsP, ULSX, min_util, TopK-CL-List)
14.   End
15.  End
```

Fig 2: Pseudo code of TKSA algorithm

RUC STRATEGY

Input: X, Current min-heap TopK-CL-List
Output: Rearranged min-heap tree

```
1. Check if  HUI List is full or not
2.   If NO, then add X with Utility List
3.   If YES,
4.       Compare utility value with the root of the min-
             heap tree
5.       If utility value greater than the root value,
6.          Remove lowest utility item value from the root
7.          Add the new value and rearrange the min-heap tree
8.          Set threshold = new root value
9.      Add X with Utility
10.  Return
```

Fig 3. Implementation of RUC Strategy

Fig. 2 shows the pseudo code of the proposed algorithm. When the initial utility-lists are constructed from a database, they are sorted and processed in transaction-weightedutility-ascending order. Therefore, all the utility-lists in u(P) are ordered as the initial utility-lists are ordered. To explore the search space, the algorithm intersects X and each utility-list Y after X in u(P)s. It continues to mine itemsets that are concatenations to the itemset X if the sum of iutils and rutils of X is not less the minimum threshold value. Suppose X is the utility-list of itemset P_x and Y that of itemset P_y, and then construct procedure is implemented to construct the utility-list of itemset P_{xy}. Finally, the set of utility-lists

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 7, July 2017

of all the 1-extensions of itemset Px is recursively processed. Given a database and a minutil, after the initial utility-lists are constructed, the algorithm can mine all high utility itemsets. Fig 3 shows the implementation of RUC strategy in the min-heap tree for pruning search space.
During the search, TKO updates the list of current top-k HUIs in Top K-List and gradually raises the min_util threshold by the information of Top K-List. When the algorithm terminates, the Top K-List captures the complete set of top-k HUIs in the database.

ACE ALGORITHM

The present algorithm aims to study the customer pattern ie., the customers who have purchased the high utility itemsets and how often. The frequency of customer visits are also studied. Finally the customers are listed on the decreasing order of purchase quantity of high utility itemsets(HUIs).

Architecture:

The Item Entry Table include item name, item number and selling price. The sales Entry Table includes Transaction number TID, items sold, CID (customer ID), quantity and profit associated are entered. Multiple transactions are possible for the same customer. Customer accounts are created using CID, phone number and such details.

PROCEDURE : **ACE**
Input : (i) Transactional database D
(ii) min–heap tree TopK-CL-List and minimum utility threshold min_util calculated earlier

Result : List of customer IDs along with frequency count in CUST-LIST

1. Read a tuple from the transaction database
2. Check whether the transaction has HUIs contained within it
3. If true
4. Append the CID in CUST-LIST
5. If the CID is already in the list increase its frequency count.
6. Read the next tuple and continue from step 2 until all transactions are read completely.
7. End

VI. EXPERIMENTAL EVALUATION

Four different sizes of datasets were used in the experiments. Databases 1, 2, 3 4 and 5 were generated by Synthetic Data Generation Code. The databases do not provide item utility (external utility) and item count for each transaction (internal utility). Here internal utilities for items are generated randomly ranging from 1 to 10 and unit profit of items are generated randomly from 1 to 5 by transaction utility values Generation code. Table 6 shows the statistical information about these databases, including the size on disk, the number of transactions, the number of distinct items.

Table 6: Statistical information about the database

Database	Size(KB)	No. of transactions	No. of items
1	100	2000	48
2	200	4100	48
3	300	8000	50
4	400	15000	50
5	500	30000	60

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

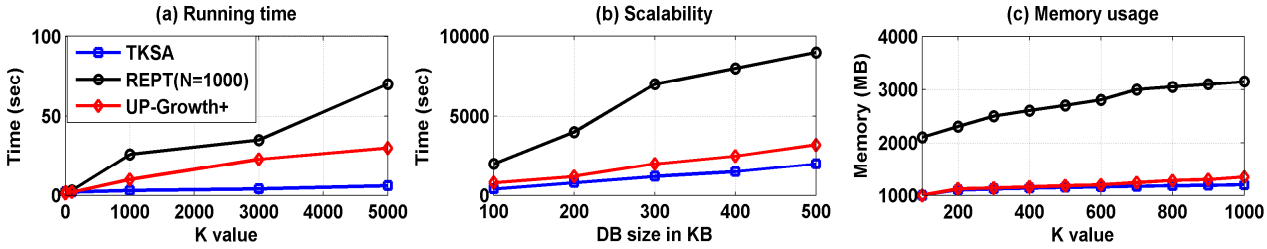
(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 7, July 2017

We evaluate the performance of the proposed algorithm TKSA against REPT[16] and UP-Growth+. The results show that the proposed algorithm performs better than the existing algorithms.

Fig 4. Experimental results for different algorithms



The running time of the proposed algorithm is better than the other algorithms for almost all database sizes. The memory consumption of all the algorithms is proportional to the number of candidate itemsets they generate. The proposed algorithm comsumes less memory because the number of candidate itemsets that are generated and stored in the memory are less. Also the proposed algorithm scales better for different database sizes.

VII. CONCLUSION AND FUTURE WORK

This paper presents Mining Top K-HUI patterns by using two algorithms namely TKSA which is a single phase algorithm and ACE which is based on customer frequency. The TKSA utilizes the basic search procedure of HUI-Miner and its utility-list structure. Utility is calculated by its utility-list without scanning the original database. In ACE, the purchasing power of the customers are studied using the top-k HUIs calculated earlier.

As a future work the two Algorithms, TKSA and ACE can be combined for pruning the search space. Top K-HUI mining task can be extended to discover different types of top-k high utility patterns such as top-k high utility episodes, top-k high utility web access patterns and top-k mobile high utility sequential patterns. The work can be extended to handle data streams and negative utility values also.

REFERENCES

[1] R. Agrawal et al., “Mining association rules between sets of items in large databases”, in proceedings of the ACM SIGMOD International Conference on Management of data, page no: 207-216.
[2] R. Agrawal and R. Srikant, ―Fast Algorithms for Mining Association Rules, in Proc. of Int'l Conf. on Very Large Data Bases, pp. 487-499, 1994.
[3] J. Han, J. Pei, Y. Yin, “Mining frequent patterns without candidate generation”, in Proceedings of the ACMSIGMOD Int'l Conf. on Management of Data, pp. 1-12.
[4] R. Chan, Q. Yang, and Y. Shen, “Mining high utility Itemsets,” The Third IEEE International Conference on Data Mining, pp. 19-26, 2003.
[5] Y. Liu, W. Liao and A. Choudhary, “A fast high utility itemsets mining algorithm”, in Proceedings of the Utility-Based Data Mining Workshop.
[6] Philippe Fournier-Viger, Cheng-Wei Wu, Souleymane Zida, Vincent S.Tseng, 2014, ‘FHM: Faster High-Utility Itemset Mining using Estimated Utility Co-occurrence Pruning’, Proc.21st international Symposium on Methodologies for intelligent Systems(ISMIS 2014), Springer, LNAI, pp.83-92.
[7] Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu, Fellow, IEEE,’ Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases’.
[8] G. Liu, H. Lu, J. X. Yu, W. Wang, and X. Xiao, “ An efficient implementation of pattern growth approach”, in proceedings of IEEE Int’l Conf. Data Mining Workshop Frequent Itemset Mining Implementations, 2003.
[9] Junqiang Liu, Ke Wang, Benjamin C.M. Fung, 2016, ‘Mining High Utility Patterns in One Phase without Generating Candidates’ IEEE Transactions on Knowledge and Data Engineering.
[10] S. Krishnamoorthy, “Pruning strategies for mining high utility itemsets,” Expert Syst. Appl., vol. 42, no. 5, pp. 2371–2381, 2015.
[11] P. Fournier-Viger, C. Wu, V. S. Tseng, ―Mining Top-K Association Rules, in Proc. of Int’l Conf. on Canadian conference on Advances in Artificial Intelligence, pp. 61–73, 2012.
[12] P. Fournier-Viger, V. S Tseng, ―Mining Top-K Sequential Rules, in Proc. of Int’l Conf. on Advanced Data Mining and Applications, pp. 180-194, 2011. [9]
[13] Mengchi Liu and JunfengQu, “Mining high utility itemsets without candidate generation”, in Proceedings of the 21st ACM international conference on Information and knowledge management, page no: 55–64.
[14] M. Liu and J. Qu, “Mining high utility itemsets without candidate generation,” in Proc. ACM Int. Conf. Inf. Knowl. Manag., 2012, pp. 55–64.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirset.com

Vol. 6, Issue 7, July 2017

[15] T. Quang, S. Oyanagi, and K. Yamazaki, "ExMiner: An efficient algorithm for mining top-k frequent patterns," in Proc. Int. Conf. Adv. Data Mining Appl., 2006, pp. 436 – 447.
[16]H. Ryang and U. Yun, "Top-k high utility pattern mining with effective threshold raising Strategies," Knowl.-Based Syst., vol. 76, pp. 109–126, 2015.
[17] Frequent Itemset Mining Implementations Repository [Online]. Available: http://fimi.cs.helsinki.fi/