© 2018 IJSRST | Volume 4 | Issue 5 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Science and Technology

Comparative Analysis of On-Shelf Utility Mining Algorithm

Dr. S. Vijayarani[1], Mrs. C. Sivamathi [2],Ms. V. Jeevika Tharini[3]
[1]**Assistant Professor, Department of Computer Science, Bharathiar University, Coimbatore,** Tamilnadu, India
[2]**Ph. D Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore,** Tamilnadu, India
[3]**PG student, Department of Computer Science, Bharathiar University, Coimbatore,** Tamilnadu, India

ABSTRACT

Data mining is a process of retrieving previously unknown and needed patterns from database. Utility mining is one of the important fields in data mining. Utility mining is a process of finding high utility itemsets from a database. An item is termed as high utility item if the item's utility is more than minimum threshold value. Utility of an item is based on user's interest or preference. Recently, temporal data mining has become a core data processing technique to deal with the changing data. On-shelf utility mining includes the on-shelf time period of item and gets the exact utility values of itemsets in temporal database. In traditional on-shelf utility mining, profits of all items in databases are considered as positive values. However, in real applications, some items may have negative profit. In this work both FOSHU (Faster On-Shelf High Utility) and TS-HOUN (Three-Scan Algorithm for Mining On-shelf High Utility Itemsets with Negative profit) algorithms are compared and their performances were measured.

Keywords : Utility mining, On-Shelf utility mining, temporal database, relative utility, periodical utility.

I. INTRODUCTION

Data mining is the process of extracting interesting information or patterns from large information repositories. It task includes finding association rules, classification rules, clustering rules. Among those data mining, association rule mining is the most popular task in data mining. It has two phases. In first phase, it discovers all the frequent itemsets based on a user-defined minimum support threshold value. In second phase, it generates the association rules from the discovered frequent itemsets based on the user-defined minimum confidence threshold value. In this, frequent itemsets considers only the frequency of an item in a database. The relative importance such as price, weight or profit of an item inside a transaction is not considered. However, in real world business, some items or itemsets with low support in the data set may bring high profits due to their high price or high frequency within transactions. Such useful, profitable itemsets are missed by frequent itemset mining [1].

In Weighted Frequent itemset mining, weights of each item such as unit profits of items in the databases are considered. If items appear infrequently, they might still be found if they have high weights. But in this framework, the quantities of items are ignored. Therefore it cannot satisfy the requirements of users who are interested in finding the itemsets with consideration of both quantity and profit [2]. Recently, Utility itemset mining [3] has been proposed to eliminate the limitation of frequent and weighted itemset mining. It considers utility of an item which is based on interesting measures like user's preference or frequent patterns of interest. Utility mining measures the importance of an item. Thus it is useful in real world market data analysis. Utility of an item in a database is the product of external and local transaction utility values. The local transaction utility is defined as the quantity of a item and the external utility is the profit of a item in utility mining. The utility of an itemset is calculated by the product of quantity and profit. If utility of an itemset is greater than the threshold (predefined (user defined)

minimum utility) or equal to threshold, then this itemset is considered as high utility itemset.

Temporal data mining [4, 5] has attracted a lot of attention due to its practicality in nature. Temporal data exist extensively in economics, finance, communication, and other areas like weather forecasting. Temporal transaction database is divided into several partitions according to the time periods in it. In specific time or season, some items or itemsets may have high frequency. Thus mining time-related database is interest to search and useful in real world. In real-world applications, products available in stores can be put on shelf and taken off multiple times when they are in need. In order to indentify such itemsets, on-shelf utility mining is proposed [6] in which an on-shelf period of products is calculated to get more accurate utility values of itemsets in temporal databases. In practical situations, utility of some items may have negative profit or it is difficult to calculate its profit like free products. But combination of such products with positive profit products may give profit to the business. In practical business applications, temporal databases are dynamic. They are continually appended or updated. Thus the discovered high utility itemsets need to be updated.

Several researches about utility mining were proposed in the recent years, most of them emphasis on how to efficiently find out the high utility itemsets from the databases [7, 8,9].Many studies and researches [1, 4, 5, and 11] were proposed to dynamically mine using association rules. An example for dynamic itemset mining is to find the frequent patterns for On-shelf products. Though, a product may be put on shelf and taken off from shelf multiple times in a store. High On-Shelf utility itemset not only considers individual profit and quantity of each item in a transaction but also the on-shelf periods of all items in an itemset. For example, consider the following rule “In the winter, customers usually purchase overcoats and stockings together”. The itemset {overcoats, stockings} may be not frequent all through the entire database, but may be a high utility itemset in the winter. Hence 3-scanmining algorithm has been introduced, called TS-HOUN (Three-Scan Algorithm for Mining On-shelf High Utility Itemsets with Negative profit) Mine, for effectively andefficiently finding the itemsets from a database.

II. RELATED WORK

Li et al. [13] proposed two phase algorithms MHUI-BIT (Mining High-Utility Itemsets based on BITvector) and MHUI-TID (Mining High-Utility Itemsets based on TIDlist) for to find high utility itemsets from the data streams. They have used Bitvector and TIDlist (Transaction ID) to improve performance of utility mining. They have also developed MHUI-BIT-NIP (MHUI-BIT with Negative Item Profits) and MHUI-TID-NIP (MHUI-TID with Negative Item Profits) for discovering itemsets with negative profit over continuous data stream.

Lan and Tseng [10] proposed a new topic named on-shelf utility mining, which considered the on-shelf periods of each items with the quantities and unit profits of each items.

Lan, Hong et al. [11] proposed two-phased mining algorithm to discover high on-shelf utility itemsets in an efficient way. In the first phase, the possible candidates for on-shelf utility itemsets within each time period are extracted. In the second phase, generated candidates from the on-shelf utility itemsets will checked for their actual utility values.

Lan et al. [14] proposed an algorithm HOUN (High On-shelf Utility mining with Negative item values) to discover the on-shelf high utility items which have both positive and negative profit. This algorithm performs three database scan. In first database scan, preprocessing is done by transforming database into corresponding time period and discover high periodical utility upper bound 2-itemsets. In the second scan of the database, high periodical utility itemsets are discovered. In third scan, it discovers high on-shelf utility itemsets with negative values.

Lin et.al [15] proposed a new approach that integrates the previous two-phase procedure for utility mining and the FP-tree concept to utilize the downward-closure property and generate a compressed tree structure. Experimental results also show that the proposed approach has a better performance than Liu et al.’s two-phase algorithm in execution time. At last, the numbers of tree nodes generated from three different item ordering methods

are also compared; with results showing that the frequency ordering produces less tree nodes than the other two. Many algorithms were proposed to mine association rules, most of which were based on item frequency values. Considering a customer may buy many copies of an item and each item may have different profits, mining frequent patterns from a traditional database is not suitable for some real-world applications. The high utility pattern tree (HUP tree) is designed and the HUP-growth mining algorithm is proposed by Lin et, al., to derive high utility patterns effectively and efficiently.

Liang et al. [12] proposed THUI (Temporal High Utility Itemsets)-Mine approach. Utility of an itemset is considered as the value of this itemset, and utility mining aims at identifying the itemsets with high utilities. The temporal high utility itemsets are the itemsets whose support is larger than a pre-specified threshold in current time window of the data stream. Discovery of temporal high utility itemsets is an important process for mining interesting patterns like association rules from data streams. The THUI (Temporal High Utility Itemsets)-Mine is used for mining temporal high utility itemsets from data streams efficiently and effectively. This is the first work on mining temporal high utility itemsets from data streams. The novel contribution of THUI-Mine is that it can effectively identify the temporal high utility itemsets by generating fewer candidate itemsets such that the execution time can be reduced substantially in mining all high utility itemsets in data streams. In this way, the process of discovering all temporal high utility itemsets under all time windows of data streams can be achieved effectively with less memory space and execution time.

III. PRELIMINARY DEFINITIONS

The following are some of the preliminary definitions for discovering On-Shelf high utility Itemsets [16].

Definition 1: The internal utility of an item i_p is a transaction dependent numerical value. In general the quantity of an item in transaction is taken as an internal utility.

Definition 2: The external utility of an item $i_p$is transaction independent numerical value, defined by the user. It reflects importance of the item. Its common practice is to choose profit as external utility. External utilities are stored in a separate utility table.

Definition 3: Utility function f is the product of internal and external utility and it is considered as utility function.

Definition 4: The utility of an item i_p in transaction T is the calculated using utility function. Utility of an item in a particular transaction = Product of its internal utility in that transaction and its external utility.

Definition 5: The utility of an itemset S in transaction T is defined as $u(S,T) = \sum u(ip,T)$, $\forall ip \in S, S \subseteq T$.

*Definition 6:*The periodical utility $pu(X,t_j)$ of an itemset X is the sum of the utility values of X in all transactions including X within the j^{th} period (t).

*Definition 7:*The periodical total transaction utility $pttu(t_j)$ is the sum of the transaction utilities of all transactions within the j^{th}time period t_j.

*Definition 8:*The periodical utility ratio $pur(X,t_j)$ of an itemset X is the periodical utility $pu(X,t_j)$ of X within the j^{th} period t_j over the periodical total utility $pttu(t_j)$ of the time period t_j. The periodical utility ratio $pur(X,t_j)$ of an itemset X is the periodical utility $pu(X,t_j)$ of X within the j^{th} period t_j over the periodical total utility $pttu(t_j)$ of the time period t_j.

*Definition 9:*The On-shelf utility ratio of an itemset X, our (X), is the sum of utilities of X within all on-shelf time periods of X over the sum of all the transaction utilities within the union of the on-shelf time periods of X.

TS-HOUN MINING ALGORITHM

TS-HOUN algorithm works as follows: The first step is to transform the occurring time of each transaction into the corresponding time period. Then it initializes a periodical total transaction utility (PTTU) table as a zero table, in which the row number is the time period number and each entry in the PTTU table is set as 0. In step 3, for each y^{th} transaction $Trans_{jy}$ in each period t_j, calculate the utility value and calculate transaction utility of items. Then calculate Transaction weighted utility of items to obtain the periodical total transaction utility $pttu_j$. Now it finds the 2-itemsets with high periodical utility upper-bound values in

each time period t_j. Next step it denotes the set of high-periodical-utility upper-bound 2- itemsets in each t_j as HPUUj2 and the union of allHPUUj2's for all time periods as HPUU2. Then, the itemsets with high periodical utility values in each timeperiod(t_j) is retrieved. Denote the set of high-periodical- utility itemsets in each t_j as HPU_j and the union of all HPU_j's for all time periods as HPU.For each itemset X in HPU, find its actual on-shelf utility. If the actual utility value of X in each of its On-shelf time periods is known, and $u(X)^{actual}$/pttu(X)=k, then X is a high-on-shelf-utility itemset; Then for each itemset X in HPU scan the database to find its actual periodical utility value. Calculate the actual on-shelf utility u(X)actual of each itemset X in HPU. If $U(X)^{actual}$/pttu(X)=k, then X is a High-On-Shelf Utilityitemset[16].

Algorithm: TS-HOUN algorithm
Input: Transaction database, MT – Minimum utility threshold.
Output: On-shelf high utility itemsets;
1. Pttu=0;
2. For each time period ti
2.1. For each transaction $Trans_j$ in t_i
2.1.1. $u(i_z, Trans_{jy})=s(i)*q(i\ Trans_{jy})$;
2.1.2. $tu(Trans_{jy}) =\sum_{i\in Trans_{jy}} u(iz, Transjy)$;
2.2. End for
3. End for
4. $pttu_j = \sum tu(Trans_{jy})$;
5. Generate candidates using itemset generation;
6. HPUi= candidate2_itemset.PTTU > MT // High Periodical Utility value.
7. HOUN=NULL; //On-shelf High Utility Itemsets with Negative profit
8. for each itemset X in HPU
8.1 $U(X)^{appearing} = \sum_{X\in HPU\ ^\ tj\in COSX} uj(X)$;
8.2 If $[U(X)^{actual}$= pttu(X)] >=k
8.2.1 HOUN = HOUN U X && HPU = HPU– X
9. End for;
10. For each itemset X in HPU
10.1.$U(X)^{act} = U(X)^{appear} + \sum_{X\in HUI\ ^\ XRHUIj\ ^\ tj\in COSX} U(X)_j^{scan}$
10.2. If $(U(X)^{actual}$/pttu(X]) >= MT
10.2.1 HOUN = HOUN U X&& HPU = HPU– X
11. End for;
12. Return HOUN;

Pseudo code for TS-HOUN

FOSHU MINING ALGORITHM

Algorithm: FOSHU algorithm
Input: D: a transaction database, minutil: auser-specified threshold
Output: the set of high on-shelf utility itemsets
1. Scan D to calculate TWU({i}), TWU({i},h) and pto(h) for each period h and each item i, as well as the set of all time periods PE;
2. Let I^*= {i}∃h ∈PE ^ TWU({i},h)/pto(h)>=minutil};
3. Let > be the global TWU ascending order on I^*;
4. Scan D to build the utility-list of each item i∈I^*;
5. Search (Ø, I^*, minutil);

Pseudo code for FOSHU

In Search procedure (the step 5 in FOSHU algorithm), it takes an itemset P as input, extensions of P having the form Pz meaning that Pz was previously obtained by appending an item z to P, and minutil. The search procedure operates as follows. For each extension Px of P, the search procedure first scans the utility list of Px to calculatesumIUtil(Px, h) for each period h where Px appears. At the same time, the total utility of time periods where Px appears (to(Px)) is calculated, as well as the total utility of Px (which is equal to sumIUtil(Px)). Then, the relative utility of Px is calculated as ru(Px) = sumIUtil(Px)/to(Px). If ru(Px) is no less than minutil, Px is a high on-shelf utility itemset and it is output. Then, if there exist a time period h such that the sum of sumIUtil(Px; h)+sumRUtil(Px; h))=to(Px) is no less than minutil, it means that extensions of Px should be explored (if∃h ∈ pi(Pxy)). This is performed by merging Px with all extensions Py of P such that y >x to form extensions of the form Pxy containing |Px|+1 items. The utility-list of Pxy is then con- structed as in FHM by calling the Construct procedure to join the utility-lists of P, Px and Py. This latter procedure is the same as in FHM and is thus not described in more details. Then, a check is performed to determine if Pxy and its extensions may be high on-shelf utility itemsets by

using the TWU measure, based on Property 4. This is done by scanning the utility list of Pxy to calculate TWU(Pxy; h) for each time period h where Pxy appears. If oTWU(Pxy, h)/p to (h) >=minutil for at least one period h, then Pxy will be added to Extensions ofPx, the set of extensions of Px which will be considered for further extensions with a recursive call to the Search procedure. Note that the check at line (if∃h ∈ pi(Pxy)) may seem redundant since the TWU is a less tight upper bound than the sum of iutil and rutil values used in the check at if∃h ∈ pi(Pxy) such that (sumIUtil(Px; h)+ sumRUtil(Px, h))/to(Px) <= minutil (that would be performed in a recursive call to Search with Pxy). However, there is a good reason for performing the check. It is that pruning an itemset Pxy before the recursive call to Search will avoid comparing Pxy with potentially many other extensions of Px in the recursive call. Since the Search procedure starts from single items, it recursively explore the search space of itemsets by appending single items, it can be easily seen and that procedure is correct and complete to discover all high utility on-shelf itemsets.

IV. EXPERIMENTAL EVALUATION

To evaluate the performance of TS-HOUN and FOSHU algorithms, an experiment is conducted. Both the algorithms was implemented in Java and executed in a machine with 3.20 GHz CPU. Comparison is performed for the algorithm FOSHU with the state of the art algorithm TS-HOUN for high on-shelf utility itemset mining with negative unit profits. All the memory measurements and Time measurements were done using the Java API. Experiments were carried on with five real-life datasets having varied characteristic. Candidates count and execution time for each algorithm are compared. Also the numbers of items with various minimum threshold values are compared.

TABLE I

COMPARISON OF TS-HOUN AND FOSHU USING FIVE DATASETS

Data set	Exe. Time of TSHOUN	Exe. Time FOSHU	Mem space TSHOUN	Mem space FOSHU
Accidents	63222	2058	31.66	19.17
Kosarak	81279	2496	16.2	20.24
Mushroom	67067	2176	16.57	20.1
Pumsb	61940	2227	25.45	29.12
Retail	62245	2235	30.62	25.99

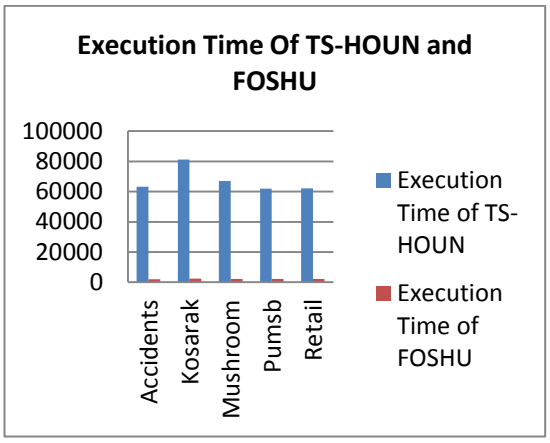

Fig 1. Comparison of execution time of TS-HOUN and FOSHU algorithm

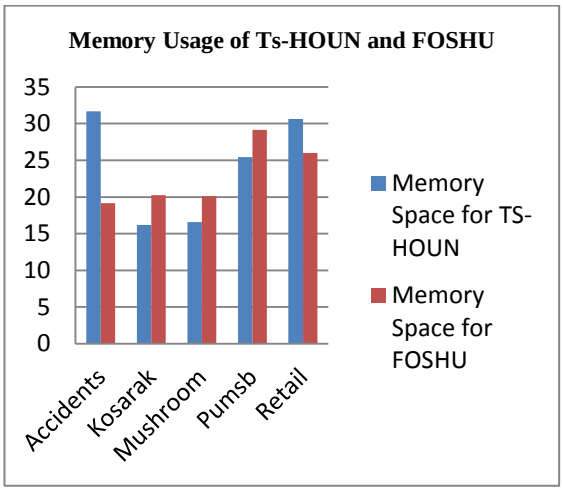

Fig 2. Comparison of memory usage of TS-HOUN and FOSHU algorithm

TABLE II

COMPARISON OF TS-HOUN AND FOSHU USING CHESS DATASET

Threshold	Execution Time of FOSHU	Execution Time TS-HOUN	Memory space FOSHU	Memory space TS-HOUN
0.8	656	7001	15.47	10.92
0.7	918	13594	15.81	22.61
0.6	1063	26517	18.47	29.8
0.5	1782	52487	27.15	35.61
0.4	5988	101271	24.89	46.08
0.3	17751	364535	28.8	58.46

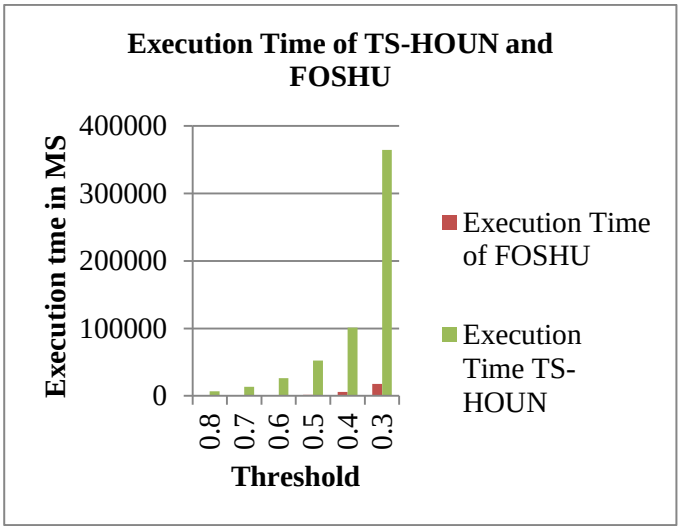

Fig 3. Comparison of execution time of TS-HOUN and FOSHU algorithm

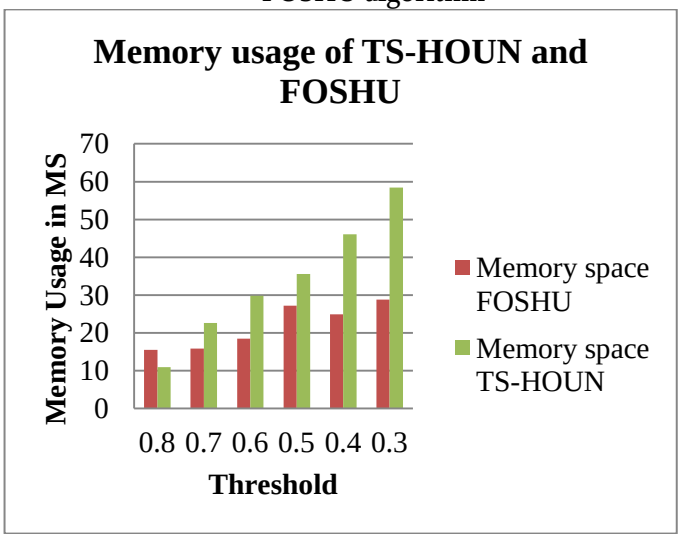

Fig 4. Comparison of memory usage of TS-HOUN and FOSHU algorithm

V. CONCLUSION

In recent years, more number of researches has been done on utility mining. In frequent itemset mining, the mining process does not concentrate on profit, number of items purchased or cost of an itemset. In real life applications there may be some items, which may not be frequent but may have high profit. Utility Mining finds such high utility itemsets from transaction database and in On-Shelf utility mining it considers the on shelf time periods also. It is very beneficial in several real-life applications. In this work a brief overview of On-Shelf utility mining algorithms for mining high utility itemset with negative profit was presented and those algorithms are compared with various performance factors like Execution time, memory used and number of candidates generated. It was observed that FOSHU is faster than TS-HOUN algorithm.

VI. REFERENCES

1. R Agrawal, R. Srikant, et al., ?Fast algorithms for mining association rules," in Proc. 20th int. conf. very large data bases, VLDB, vol. 1215, pp. 487-499, 1994.
2. F Tao, F. Murtagh, and M. Farid, ?Weighted association rule mining using weighted support and significance framework," in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 661-666, ACM, 2003.
3. R Chan, Q. Yang, and Y.-D. Shen, ?Mining high utility itemsets," in Data Mining, 2003. ICDM 2003. Third IEEE International Conference on, pp. 19-26, IEEE, 2003.
4. J M. Ale and G. H. Rossi, ?An approach to discovering temporal association rules," in Proceedings of the 2000 ACM symposium on Applied computing-Volume 1, pp. 294-300, ACM, 2000.

5. Chang, C. Y., Chen, M. S., & Lee, C. H. (2002). Mining general temporal association rules for items with different exhibition periods. In The third IEEE international conference on data mining (pp. 59–66).
6. T-P. H. Gu-Cheng Lan and V. S. Tseng, ?A three-scan mining algorithm for high on-shelf utility itemsets," Expert Systems with Applications.
7. CJ. Chu, Vincent S. Tseng, and T. Liang, Mining temporal rare utility itemsets in large databases using relative utility thresholds, International Journal of Innovative Computing, Information and Control, vol.4, issue.8, 2008.
8. Y Liu, W. Liao, and A. Choudhary, A fast high utility itemsets mining algorithm, The Utility-Based Data Mining Workshop, pp.90-99, 2005.
9. H Yao, and H.J. Hamilton, Mining itemset utilities from transaction databases, Data & Knowledge Engineering, vol.59, no.3, pp.603-626, 2006.
10. C.H. Lee, C.R. Lin, and M.S. Chen, “On mining general temporal association rules in a publication database,” The 2001 IEEE International Conference on Data Mining, pp.337-344, 2001.
11. H. T. P. T. V. S. Lan, G. C., ?Discovery of high utility itemsets from on-shelf time periods of products," Expert Systems with Applications, vol. 38, no. 5, pp. 5851-5857, 2011.
12. Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Choi, H. J. (2012). Interactive mining of high utility patterns over data streams. Expert Systems with Applications, 39(15), 11979-11991.
13. H. H. Y. L. S. Y. Li, H. F., ?Fast and memory efficient mining of high utility itemsets from data streams: with and without negative item profits," Expert Systems with Applications, vol. 28, no. 3, pp. 495-522, 2011.
14. G.-C. Lan, T.-P. Hong, J.-P. Huang, and V. S. Tseng, ?On-shelf utility mining with negative item values," Expert Systems with Applications, vol. 41, no. 7, pp. 3450-3459, 2014.
15. Shie, B. E., Philip, S. Y., & Tseng, V. S. (2012). Efficient algorithms for mining maximal high utility itemsets from data streams with different models. Expert Systems with Applications, 39(17), 12947-12960.
16. Lan, G. C., Hong, T. P., Huang, J. P.& Tseng, V. S. (2014). On-shelf utility mining with negative item values. Exper Sysytems with Applications, 41(7),3450-3459.
17. P. Fournier-Viger, C.-W. Wu, S. Zida and V.S. Tseng.Faster High-Utility Itemset Mining using EstimatedUtility Co-occurrence Pruning. In Proc. 21st Intern.Symp. Methodologies Intell. Systems, Springer,pp. 83{92, 2014}.