International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August, 2019

Business Intelligence for Evaluating the Intangible Benefits of On-Shelf High Utility Itemset from the Temporal Transaction Database

S. Vijayarani, V. Jeevika Tharini, C. Sivamathi

Abstract: ***Utility Mining is the progression of identifying High Utility Itemsets (HUI's) from enormous transaction data. Utility mining plays a decisive role in the inspection of the data or giving actionable information to help managers, sales executives, and other commercial end-users to generate versed business decisions. In the hypermarkets, the showcase period of every item in display will vary such as new products, seasonal products, and so on. Itemsets with time period are not retrieved by existing utility mining algorithms. Hence, On-Shelf Utility Mining algorithms were proposed to discover HUI's and a general on-shelf period of all items in temporal databases is considered. Research work aims to propose an algorithm called LOSUM (List On-Shelf Utility Mining) to retrieve on-shelf HUI's from a temporal transaction database by reducing the data stores scan. The algorithm is enhanced by implementing a list structure to store utility information of every itemset. The candidate itemsets are generated from the list itself. This reduces the supplementary scan of a database. The LOSUM is compared with FOSHU using Chess, Accident, Kosarak, and Mushroom datasets. The experimental results illustrate that the LOSUM is efficient than the existing algorithm FOSHU (Fast On-Shelf High Utility itemset mining) algorithm.***

Keywords: ***Utility Mining, High Utility Itemset, List Structure, Periodic utility, On-shelf utility value.***

I. INTRODUCTION

Data mining (DM) is the progression of mining needed patterns or interesting information from repositories and its task includes finding classification, association and clustering rules. Among these DM methods, an Association Rule Mining (ARM) is the most popular task. ARM has two segments, the first segment finds out all the repeated itemset based on a user-defined minimum threshold value. In the subsequent phase, the ARM is generated from the identified repeated itemset. Frequent Itemset Mining (FIM) considers only the incidence of an item in a database. The significance likely price, weight or profit of an item inside a transaction is not considered. Some items or itemset in a dataset may have low support value that may bring higher earnings due to their high price or high profit of the frequency of an item within transactions. In this case, such valuable and profitable itemset are failed to spot by FIM [1]. In Weighted Frequent Itemset Mining (WFIM), weights (each item has individual weight) of the item is considered for mining needed information.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
Jaya Koshta*, Department of Electronics and Comm. Engg., MANIT, Bhopal, India.
Kavita Khare, Department of Electronics and Comm. Engg., MANIT, Bhopal, India.

If an item with the highest weight but its occurrence is infrequent, in this situation also, this item is still found in the transaction. This kind of framework ignores the number of incidences of items. Users may also look for itemsets having consistent profit as well as desired quantity, which cannot be satisfied by WFIM [2]. Utility Mining [3] is developed to overcome the limitations of FIM/WFIM and considers the utility of entire item in a transaction which is based on the interestingness of the user's preference or frequent patterns of interest. Temporal data mining has enchanted a lot of academicians and business peoples due to its nature of practicality [4, 5]. In hypermarkets, newly arrived products are put on the stores front display and taken off multiple times in a day whenever particular items are in need. To recognize such itemsets, On-Shelf Utility Mining algorithms were proposed [6]. In order to acquire the accurate utility value of a itemset, the On-Shelf period (time period) of every product is computed. This algorithm has lined the way for the entrepreneur to make efficient and effective decisions at business.

Most of the researches based on utility mining emphasize suggestions to invent HUI's from the huge transaction database [7, 8, 9]. Most of the academicians and researchers were proposed an association rule-based technique to dynamically mine the much-needed items [1, 4, 5, 10 and 11]. An example of dynamic itemset mining is finding "the frequent patterns of On-shelf products". However, a consumable item may be placed on a shelf and taken out of the shelf numerous times in a store. High On-Shelf Utility itemset considers quantity, profit and time period of every item in a transaction as well. For example, consider the following frequent item "In the rainy season, clients habitually purchase sweaters and rain-coats jointly". The itemset {rain-coats, sweaters} maybe not frequent items all from end to end of the complete database, but maybe a HUI's in the rainy season. Hence, a three-scanning algorithm called TS-HOUN (Three-Scan Algorithm for Mining On-shelf HUI's with Negative unit profit) is commenced for proficient mining. TS-HOUN is outperformed by the algorithm FOSHU. The FOSHU is improved by invoking a list structure to store utility information of itemsets. This reduces the extra scan of a database. The candidates are generated from the created list itself. This improved algorithm is called as LOSUM (List-On-Shelf Utility Mining Algorithm). This research work implements the list structure to store the calculated utility data is retrieved from the list for further calculations. Retrieving data from list structure reduce the database scan.

This paper is ordered as follows. section II, elaborately describes the literature review. In section III discuses about the problem definition, preliminary definitions and the proposed solution. Section IV demonstrates illustrative example. Section V compares the resultant value of time and memory of both existing and proposed algorithm. Conclusions are given in Section 6.

II. RELATED WORKS

Lan et al. [13] have designed a two-phase mining algorithm for the effective discovery of On-Shelf HUI's from the transactional database. In the initial phase, within every time period the possible candidates of On-Shelf HUI's are found by the way of a level by level approach and the candidate items were generated in the succeeding phase. By a supplementary database scan, the candidates of On-Shelf HUI's are examined for the definite price of utility item. In the FIM considers only the occurrence of an item in a database. On-Shelf HUI includes weight, profit, and time periods of On-Shelf consumable products.

Lin et.al [14] has proposed a new method that assimilates utility mining's previous two-phase procedures. Condensed tree structure and downward closure property is expanded using the incorporation of traditional approach called FP-tree. Author's proposed algorithm evidence the better performance when the algorithm is evaluated against the traditional two-phase algorithm. From the valuation, it's found that execution time and generation tree node is reduced.

Liang et al. [15] proposed THUI (Temporal HUI's)-mine approach. The utility value considers the profit of the items. UM aims at recognizing the itemsets which poses high utility rate. Temporal HUI's are used for retrieve temporal HUI's from data streams proficiently. The method of identifying all temporal HUI's beneath entire-time windows of data streams can be accomplished successfully with compact amount of execution time and fewer memory space.

Wong et al. [16] have shown a new approach using an incremental mining algorithm, which is proposed to maintain discovered HUI's based on effective manner. Itemsets are first segregated into three parts by considering the factors namely have small, pre-large, or large (high) TWU in the original database or not and in inserted transactions. Finally, individual functions are then executed for every single part of the data.

Lan et. al. [17] developed an incremental mining algorithm for proficient mining of HUI's and is based on the model of the Fast-UPdate (FUP) method. ARM is also accomplished with FUP algorithm. At the beginning itemsets are partitioned in to four parts based on the High Transaction Weighted Utility (TWU) value in the original database. FUP achieves better as well as faster solution than the two-phase batch mining algorithm in the background of irregular data. ARM, which is based on the amount of occurrence values of items. FUP is entirely different from the traditional ARM approach.

Zida et al. [18] have come out with the EFIM approach (EFficient HUI's). EFIM has on two upper-bounds criterions named the local and sub-tree utility, search space is more efficiently pruned. A novel array-based utility counting method was named as Fast Utility Counting (FUC) which is mainly used to estimate the upper-bound values in direct space and time. Moreover, FUC is to lessen the rate of database scans. EFIM is a proficient database transaction and projection of assimilation techniques. Illustrations in this paper with several dataset's shows that EFIM is quicker and consumes lesser memory than the algorithms HUI-Miner, d2HUP, FHM, UP-Growth+, and HUP-Miner.

Hong et al. [19] have presented an innovative variety of pattern entitled as On-Shelf HUI's. An item's distinct profit, amount of entire item in an operation, and on-shelf time periods in a database are taken in to account of. Thus proposed a 3-scan mining algorithm which is efficient to find out the needed itemsets. The proposed work implements a pruning scheme and an itemset-generation mechanism to prune repeated candidate itemsets. From the transaction of itemset pruning is done to systematically check and eliminate the repeated or duplicate itemsets in a database.

III. PROPOSED ON-SHELF UTILITY MINING ALGORITHM: LOSUM

A. *Preliminary definitions*

Some of the preliminary explanations for retrieving On-Shelf HUI's [12].

Definition 1: In the data T = {t1, t2, . . . ,tj, . . . ,tn} represents the time periods (tp), where tj represents the j_{th} period in the entire set of periods, T. V = {v1, v2, . . . ,v_n} is a set of items shown in a transactions. Temporal transaction database is represented as D = {$trans_{1.1}$, $trans_{1.2}$, $trans_{j.y}$....$trans_{n.m}$} where $trans_{j.y}$ is the y^{th} transaction in the j^{th} time period. The temporal transaction utility q(i, $trans_{j.y}$).

Definition 2: The External Utility (EU) value of an item s(i) and i represents the profit and reflects the impact of every individual item in the item. Choosing the profit value as an EU is general practice and are stored in a distinct utility table.

Definition 3: The utility value u(v, $trans_{j.y}$) of an transaction v in the temporal transaction is called as

$$u(v, trans_{j.y}) = s(v)*q(v,trans_{j.y})$$

Definition 4: The transaction utility tu(v, $trans_{j.y}$) is the total value of all the items within the transaction.

$$tu(v, trans_{j.y}) = \sum_{v \in transj.y} u(v, transj.y)$$

Definition 5: The summation of the utility values of W in a itemset is periodical utility pu(W,t_j) of an itemset.

$$pu(W, t_j) = \Sigma\, u(W, trans_{j.y})$$

Definition 6: The sum of the transaction utilities of all transactions within the j^{th} of t_j is the periodical total transaction utility pttu(t_j).

Definition 7: calculate pttu for every time period (tp) from the created On-ShelfList. pttu calculated by the summing up of same periods of all the items.

$$pttu(t_j)=tu(trans_{j.y})+ \ldots\ldots + tu(trans_{n.m})$$

Definition 8: On-Shelf utility value of an itemset by summing up the values of pu.

$$ou(v)=pu(u(v, trans_{j.y}))$$

Algorithm: LOSUM algorithm
Input: Temporal Transaction Database D, minutil: //a user-specified threshold.
Output: High On-Shelf Utility Itemsets.

1. Database D, Item V, Time period T
2. Scan D to calculate Item Utility IU
3. For each period t and item v
   a. Calculate IU = u(v, $trans_{jy}$) = s(v)*q(i,$trans_{jy}$) //utility value of an item
4. Store IU values in list structure
   a. OnshelfList→list<tp,tid,item>
   b. If IU ≤ 0 then
      Delete the item from IU
   c. End if
      Update the utility list
5. End for
6. for each item W in HPU //high periodic utility value
7. pur(W,t_j)=pu(W, t_j)/pttu(t t_j) //periodic utility ratio
8. ou(v)=pu(u(v, $trans_{jy}$)) //onshelf utility value
9. our(W,t_j)=our({pur(W,t_j)/pttu(t_j)}) //onshelf utility ratio
10. if puur(itemset)=puur(itemset)>0 then //periodic utility upperbound ratio
    a. keep the list
11. else
    a. ignore the item
12. end if
13. update list
14. if HPU > MTU then //minimum tu value
    a. store the item in the list
15. end if
16. end for
17. Display High OnShelf Utility Value
    a. ignore the item
18. end if
19. update list
20. if HPU > MTU then //minimum tu value
    a. store the item in the list
21. end if
22. end for
23. Display High OnShelf Utility Value

Pseudo-code for LOSUM

B. Proposed Algorithm

In LOSUM (List-On-Shelf Utility Mining) utility information's is stored in the form of the list structure. Initially, utility information's are calculated and stored in a list. A list structure is called On-ShelfList is generated which consists of three columns which stores the Time period, Transaction_id and Utility_value. List values are sorted to avoid the null transaction that is transaction with zero values. Calculate tu value by considering the same transaction id ($trans_{j.y}$) from the On-ShelfList. Generate the candidate itemsets with those items. From the generated candidate itemsets, the values namely periodic utility value and the high pu(W,t_j) of an items is calculated from the created list structure (On-ShelfList). The periodical utility ratio pur(W,t_j) of an itemset W is the pu(W,t_j) of W within the j_{th} period. Pur(W) is compared with a minimum threshold value and then a HPUI is retrieved. An itemset is called a high On-Shelf utility itemset when it satisfies the condition if puur(W,tj) greater than k (minimum_threshold value).

IV. AN ILLUSTRATED EXAMPLE

Example for Proposed work

The proposed work is elucidated with an example. Table 1 shows the temporal transaction database. Table 2 shows the utility table. Table 3 shows the On-Shelf time periods. Here {T1, T2 and T3} are Periods, {t1, t2, t3, and t4} are transaction ids, {tp1, tp2 and tp3} are On-Shelf periods and {M, N, O, P, Q and R} are items.

Table 1: Temporal Transaction Data

Period	TID	M	N	O	P	Q	R
T1	t1	6	0	2	1	1	0
	t2	0	0	4	2	0	1
	t3	3	0	1	2	0	1
	t4	5	0	0	0	0	3
T2	t1	1	2	0	3	0	0
	t2	1	0	0	3	3	0
	t3	2	2	3	0	0	1
	t4	3	1	2	5	0	3
T3	t1	0	3	4	2	1	0
	t2	0	1	0	0	3	0
	t3	0	0	0	1	5	0
	t4	0	0	0	3	0	2

Table 2: Profit Values

Profit	Item
M	4
N	1
O	2
P	6
Q	3
R	1

Table 3: On-Shelf Time Periods

Period	M	N	O	P	Q	R
T1	1	0	1	1	1	1
T2	1	1	1	1	1	1
T3	0	1	1	1	1	1

Step 1: A sorted transaction is created using the List Structure, in this step all the void transaction (transaction having zero values) will get confiscated to reduce the calculation time.

Step 2: Create a list structure called On-ShelfList with Three columns that stores Time period, Transaction_id and utility value. The following table contains the On-ShelfList of items.

On-ShelfList of M

Tp	Tid	U
1	1.1	24
1	1.3	12
1	1.4	20
1	2.1	4
1	2.2	4
1	2.3	8
1	2.4	12
SUM(OU)		84

On-ShelfList of N

Tp	Tid	U
1	2.1	2
1	2.3	2
1	2.4	1
1	3.1	3
1	3.2	1
SUM(OU)		9

On-ShelfList of O

Tp	Tid	U
1	1.1	4
1	1.2	8
1	1.3	2
1	2.3	6
1	2.4	4
1	3.1	8
1	3.4	6
SUM(OU)		38

On-ShelfList of P

Tp	Tid	U
1	1.1	6
1	1.2	12
1	1.3	12
1	2.1	18
1	2.1	18
1	3.1	12
1	3.3	6
SUM(OU)		84

On-ShelfList of Q

Tp	Tid	U
1	1.1	3
1	2.2	9
1	3.1	3
1	3.2	9
1	3.3	15
SUM(OU)		39

On-ShelfList of R

Tp	Tid	U
1	1.2	5
1	1.3	5
1	1.4	15
1	2.3	5
1	2.4	15
1	3.4	10
SUM(OU)		55

Step 3: With the help ofOn-ShelfList, intersection of individual itemsets through Transaction_id is accomplished. If the item has common transactions, then create candidate itemsets with those items. From the generated candidate itemsets, periodic utility value and the utility of an item is estimated. Calculate tu value by considering the same transaction id from the On-ShelfList.

Transaction Utility (tu) for T1

t1.1 = 37 t1.2 = 25 t1.3 = 31
t1.4 = 35 t2.1 = 24 t2.2 = 31
t2.3 = 21 t2.4 = 32 t3.1 = 26
t3.2 = 10 t3.3 = 21 t3.4 = 16

Step 4: Create itemset On-ShelfList.

On-ShelfList of MN

Tp	Tid	U
1	2.1	6
1	2.3	10
1	2.4	13
SUM		29

On-ShelfList of MO

Tp	Tid	U
1	1.1	28
1	1.3	14
1	3.3	14
1	3.4	16
SUM		72

On-ShelfList of MP

Tp	Tid	U
1	1.1	30
1	1.3	24
1	2.1	22
1	2.2	22
SUM		98

On-ShelfList of MQ

Tp	Tid	U
1	1.1	27
1	2.2	13
SUM		40

On-ShelfList of MR

Tp	Tid	U
1	1.3	17
1	1.4	35
1	2.3	13
1	2.4	27
SUM		92

On-ShelfList of NO

Tp	Tid	U
1	2.3	8
1	2.4	5
1	3.1	11
SUM		24

On-ShelfList of NP

Tp	Tid	U
1	2.1	20
1	3.1	15
SUM		35

On-ShelfList of NR

Tp	Tid	U
1	2.3	7
1	2.4	16
SUM		23

On-ShelfList of OP

Tp	Tid	U
1	2.1	10
1	2.2	20
1	2.3	14
1	3.1	20
SUM		64

On-ShelfList of OQ

Tp	Tid	U
1	2.1	7
1	3.1	11
SUM		18

On-ShelfList of OR

Tp	Tid	U
1	1.2	13
1	1.3	7
1	2.3	11
1	2.4	19
1	3.4	16
SUM		66

On-ShelfList of PQ

Tp	Tid	U
1	2.2	27
1	3.1	15
SUM		43

On-ShelfList of PR

Tp	Tid	U
1	1.2	17
1	1.3	17
SUM		34

Step 5: Total utility values of W is the pu(W, t_j) which is an itemset in every transactions including W contained by the j_{th} period t. The periodic utility value computation is done from the itemset generation having the same transaction id.

MN = 29 MO = 72 MP = 98
MQ = 40 MR = 92 NO = 64
NP = 15 NR = 23 OP =64
OQ = 18 PQ = 43 PR = 34

Step 6: Estimation of the pttu for every time period from the created On-ShelfList. Pttu is calculated by the summing up of same periods of all the items.

PTTU t1 = 128 PTTU t2 = 108 PTTU t3 = 73

Step 7: The pur(W,t_j) of an itemset W is pu(W,t_j) of W within the j^{th} period $t_j$over pttu(t_j) of the time period t_j. pur(W) is compared with a minimum threshold value and then HPUI is retrieved.

MN = 26.85% NR = 21.29% PR = 33.59%

Step 8: By comparing with minimum threshold value k=30%, itemset {DF} is found as HPUI and the On-Shelf time period of the item DF is t1 and t2.

Step 9: The On-Shelf utility (OU) of an item is calculated from the On-ShelfList.

Step 10: Estimating the On-Shelf utility value of an itemset by summing up the values from step 3.

Step 11: The On-Shelf utility ratio of an itemset is estimated from the summing up of the utilities to the total of PTTU's within the same transaction id.

PR = 33.59% MO = 35.82%

Step 12: The puu(W,t_j) of an itemset W within individual time period t is the summation of the tu value.

{M}, t1 = 56 {M}, t2 = 28
{N}, t2 = 5 {N}, t3 = 4

Step 13: Summation of the tu values of all transactions together with W over the summation of all the tu values contained by the puur(W,t_j) of an itemset W.

{M}, t1 = 43.75% {M}, t2 = 25.92%

Step 14: An itemset W is called a HPU when its satisfies the upper-bound itemset (HPUU) condition which is within the j^{th} of tp, if puur(W,tj) >k (minimum_threshold value).

V. Result and Discussion

Experimentation is carried out using Java and accomplished in a PC with 3.20 GHz CPU. Performance of LOSUM and FOSHU is compared using the performance metrics namely memory space occupied and execution time.

A. *Dataset Description*

B. Table 4: Characteristics of Dataset

Dataset Name	Instances	Attributes
Mushroom	8,124	22
Accidents	40,183	39
Kosarak	45,678	54
Chess	28,056	6

Table 5: Memory Usage for LOSUM and FOSHU algorithm

Dataset	Threshold	Memory Usage for FOSHU	Memory Usage for LOSUM	Time Consumption for FOSHU	Time Consumption for LOSUM
Mushroom	0.8	16.65	14.45	1735	1564
	0.7	18.49	17.22	1922	1752
	0.6	15.87	16.61	2641	2499
	0.5	31.84	27.81	4578	3921
	0.4	19.39	18.23	15704	14891
Accident	0.8	24.1	19.56	2981	1954
	0.7	54.09	43.98	2671	1673
	0.6	56.68	49.77	2659	1561

	0.5	39.02	28.51	2545	1397
	0.4	73.87	63.65	2661	1489
Kosarak	0.8	5.89	4.67	3145	2667
	0.7	7.69	5.56	4932	3577
	0.6	8.72	6.91	5676	4311
	0.5	118.59	105.11	20282	15678
	0.4	142.36	125.38	17658	12887
Chess	0.8	15.47	5.91	656	6487
	0.7	15.81	18.2	918	12984
	0.6	18.47	24.32	1063	25874
	0.5	27.15	31.25	1782	51025
	0.4	24.89	41.58	5988	121865

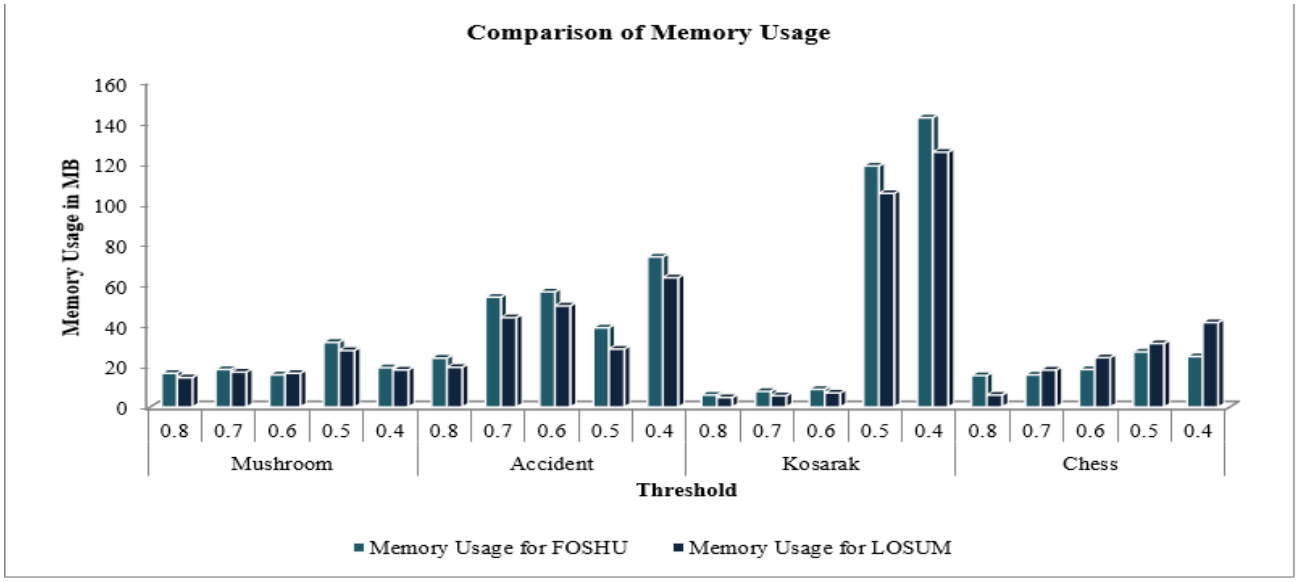
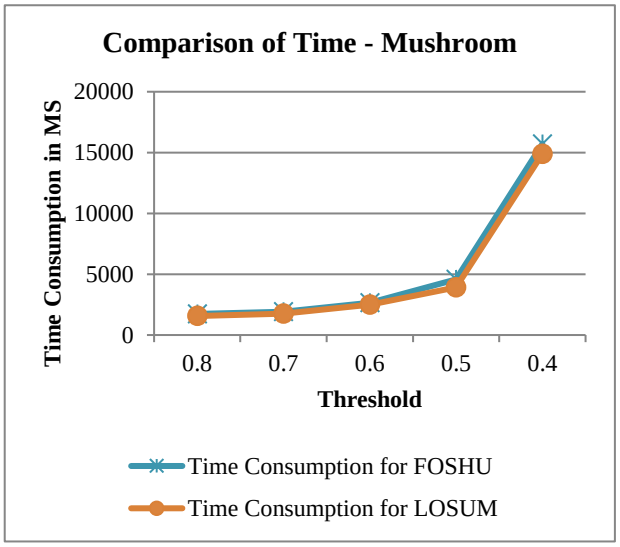
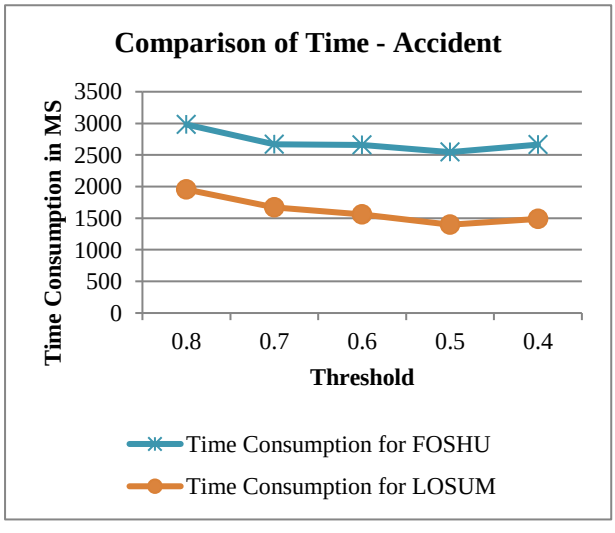


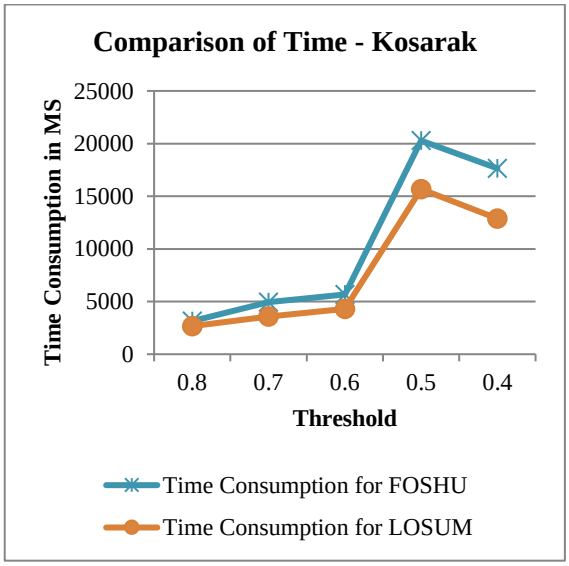
Figure 2: The graphical representation of LOSUM and FOSHU algorithm's performance. From the result, it is found that LOSUM outperforms FOSHU in terms of memory usage.



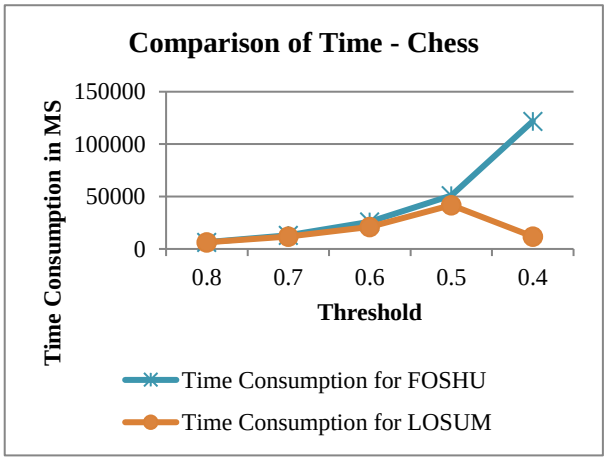
(a). Mushroom



(b). Accident



(c). Kosarak



(d). Chess

Figure 3: The graphical representation of the performance of LOSUM and FOSHU algorithms. (a). Mushroom (b). Accident (c). Kosarak (d). Chess From the result, it is found that LOSUM outperforms FOSHU in terms of Time Consumption.

VI. CONCLUSION

The display period of every individual item in a hypermarket may vary likely newly arrived product, season based product and so on. Such itemsets are not retrieved by available utility mining algorithms. Thus, On-Shelf utility mining algorithms are proposed to discover such itemsets. On-Shelf HUIM considers individual profit, the quantity in addition to that it considers On-Shelf periods of all items temporal databases. In this work, a new algorithm was proposed. On-Shelf HUI's were retrieved from a transaction database. To overcome the repeated database scan, items are stored in the created List-Structure. This reduces the extra database scan of the transaction database. The candidates were generated from the list itself. The LOSUM is compared with the existing algorithm FOSHU using the datasets namely Chess, Accident, Kosarak, and Mushroom. Memory usage and utilization of time is efficient in LOSUM when compared to FOSHU and found through the illustrated experiment's result. In the future, the negative profit items and data streams can be incorporated for finding On-Shelf HUI's.

REFERENCES

1. R. Agrawal, R. Srikant, et al., —Fast algorithms for mining association rules," in Proc. 20th int. conf. very large databases, 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. 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. C.J. 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. Lan, G. C., Hong, T. P., Huang, J. P.& Tseng, V. S. (2014). On-shelf utility mining with negative item values. ExperSysytems with Applications, 41(7),3450-3459
13. Chu, C. J., Tseng, V. S., & Liang, T. (2008). An efficient algorithm for mining temporal high utility itemsets from data streams. Journal of Systems and Software, 81(7), 1105-1117.
14. 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.
15. 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.
16. Lan, G. C., Hong, T. P., & Tseng, V. S. (2011). Discovery of high utility itemsets from on-shelf time periods of products. Expert Systems with Applications, 38(5), 5851-5857.
17. Zida, S., Fournier-Viger, P., Lin, J. C. W., Wu, C. W., & Tseng, V. S. (2015, October). EFIM: a highly efficient algorithm for high-utility itemset mining. In Mexican International Conference on Artificial Intelligence (pp. 530-546). Springer, Cham.
18. Lin, C. W., Hong, T. P., Lan, G. C., Wong, J. W., & Lin, W. Y. (2014). Incrementally mining high utility patterns based on the pre-large concept. Applied Intelligence, 40(2), 343-357.
19. Lin, C. W., Hong, T. P., Lan, G. C., Wong, J. W., & Lin, W. Y. (2015). Efficient updating of discovered high-utility itemsets for transaction deletion in dynamic databases. Advanced Engineering Informatics, 29(1), 16-27.

AUTHORS PROFILE

V. Jeevika Tharini received the B.C.A degree from Mother Teresa Women's University, Tamilnadu, India, in 2016, and the M.Sc. Computer Science from Bharathiar University, Coimbatore, Tamilnadu, India, in 2018. She is a full-time M.Phil. Research Scholar with the Department of Computer Science from Bharathiar University, Coimbatore, Tamilnadu, India. Her current research interests include utility mining, privacy preserving, and optimization techniques. She has published papers in international journals and conferences.

Dr.S. Vijayarani M.C.A., M.Phil., Ph.D is an Assistant Professor, Department of Computer Science, Bharathiar University, Coimbatore, India. She has 11 years of teaching/research and technical experience. Her research interests include data mining, privacy data mining, image mining, text mining, web mining, data streams and information retrieval. She has published more than 100 research articles in national/international journals and conferences.

C. Sivamathi, is pursuing her Ph.D in Department of Computer Science, Bharathiar University, Coimbatore, Tamilnadu, India. She has completed M.Sc(CS & IT) and M.Phil(CS) in Madurai Kamaraj University, Tamilnadu, India. Her research area includes Utility Mining, Soft computing, Privacy Preserving and Optimization techniques. She has published research articles in international journals and conferences.