Time-Efficient Tree-Based Algorithm for Mining High Utility Patterns

Chiranjeevi Manike and Hari Om

Abstract. High utility patterns mining from transaction databases is an important research area in the field of data mining. Due to the unavailability of downward closure property among the utilities of the itemsets it becomes great challenge to the researchers. Even though, efficient pruning strategy called, transaction weighted utility downward closure property used to reduce the number of candidate itemsets, total time to generate and test candidate itemsets is more. In view of this, in this paper we have proposed a time-efficient tree-based algorithm (TTBM) for mining high utility patterns from transaction databases. We construct conditional pattern bases to generate high transaction weighted utility patterns in the second pass of our algorithm. We used an efficient tree structure called, HP-Tree and tracing method to keep high transaction weighted utility patterns and for discovering high utility patterns respectively. We have compared the performance against Two-Phase and HUI-Miner algorithms. The experimental results show that the execution time of our approach is better.

1 Introduction

The problem of mining frequent patterns have been playing an important role in different stages of data mining and knowledge discovery process such as association rule mining [1], classification [4], clustering [7] etc. Big challenge faced by researchers is finding efficient pruning strategy to reduce the large number of candidate patterns. Agrawal et al.[1] was introduced an efficient pruning strategy called, Apriori. This strategy says that the support count of a superset of low support subsets is low. In frequent pattern mining the patterns with support count above the

Chiranjeevi Manike · Hari Om
Indian School of Mines, Dhanbad - 826 004, Jharkhand, India
e-mail: `chiru.research@gmail.com, hari.om.cse@ismdhanbad.ac.in`

El-Sayed M. El-Alfy et al. (eds.), *Advances in Intelligent Informatics*,
Advances in Intelligent Systems and Computing 320, DOI: 10.1007/978-3-319-11218-3_37

specified threshold are generated as frequent patterns. In real time scenario all items have different profits or importance, and to answers the few queries like the customers whose contributing more to the profit of the company, the itemsets contributing more to the total profit, only the knowledge of frequent patterns is not sufficient. In view of this utility mining was introduced with the assumption of different utility values of items in transaction database and objective of usefulness [17]. Utility of an item in a transaction is different measures of items like profit, cost, and quantity sold etc. Pruning strategy like Apriori cannot be applied in utility pattern mining because of the absence of downward closure property among the utilities of itemsets. Therefore, finding efficient pruning strategy becomes a great challenge to the researchers. Liu et al. [11] proposed an efficient algorithm called, Two-Phase by incorporating a pruning strategy namely, transaction weighted utilization (TWU) downward closure property. This strategy says that the transaction weighted utilization of itemsets hold downward closure property. Many researchers have been proposed different algorithm to efficiently mine high utility patterns with limited resources. In this paper we have proposed a time-efficient tree based algorithm for mining high utility patterns with the objective of reducing execution time.

The remaining part of this paper is organized as follows. In Sect. 2, we describe the related work. Problem definition is given in Sect. 3. In Sect. 4, we describe our proposed algorithm with example. In Sect. 5, experimental results are presented and analyzed. Conclusions and future works are given in Sect. 6.

2 Related Work

Utility mining theoretical model and fundamental definitions are given by Yao et al. [16]. This model is based on the level-wise candidate generation and test methodology, in each level low utility itemsets are pruned by using two properties called Utility bound property, and support bound property. To overcome the drawbacks, Liu et al. [11] proposed an efficient two-phase algorithm with a pruning strategy called transaction weighted utilization downward closure property. This property says that transaction weighted utility of any superset of a low transaction weighted utility itemset is low. This algorithm suffers from level-wise candidate generation and test problem. Yao et al. [15] proposed two algorithms namely UMining and Umining_H with two efficient pruning strategies by analyzing the utility upper bound property which is introduced in [16]. These algorithms are also suffers from level-wise candidate generation and test problem and loss some patterns. Another algorithm was proposed by Li et al. [9] called isolated itemset discarding strategy (IIDS) based on the level-wise candidate generation and test problem. To overcome the drawbacks with level-wise candidate generation and test problem of above algorithms Erwin et al. [6] was proposed an efficient algorithm called CTU-Mine using pattern growth approach [8]. Even though the runtime efficiency of CTU-Mine is better for low utility values on dense datasets, overall performance is better than the Two-Phase algorithm. Another efficient algorithm was proposed by same authors called CTU-PRO [5] that is performed well on both sparse and dense datasets. Ahmed et al.

[2] was proposed an algorithm called HUC-Prune based on the pattern growth approach. Tseng et al. [14] proposed an efficient two pass algorithm called UP-Growth by using a data structure called UP-Tree. Shi et al. [12] proposed modified version of Two-Phase algorithm to improve the performance. Above algorithms [16, 15, 9] suffers from the problem of level-wise candidate generation and test problem and algorithms which are implemented based on the pattern growth require more memory hence, recently Liu et al. [10] proposed an efficient algorithm called HUI-Miner. Even though it needs no candidate generation process and requires very less amount of memory, takes more time to construct utility-lists and generate high utility patterns. Yen et al. [18] proposed an efficient algorithm for mining high utility patterns with out need of generating candidate itemsets. Ahmed et al. [3] proposed three efficient data structures for interactive and incremental high utility pattern mining. Tseng et al. [13] proposed two algorithms called UP-Growth and UP-Growth$^+$ with more efficient strategies to generate candidate itemsets with two scans of the database. In view of the runtime performance and memory consumed by above algorithms, we proposed a tree based algorithm to improve the runtime performance of mining high utility patterns.

Table 1 Transaction Database

Tid	Item A	Item B	Item C	Item D	Item E
T_1	0	1	2	3	1
T_2	2	1	1	0	0
T_3	1	2	1	1	1
T_4	0	1	0	14	1
T_5	1	0	0	1	0

Table 2 Utility Table

Item Name	A	B	C	D	E
Profit($)	3	5	1	1	10

3 Problem Definition

We have followed the identical definitions conferred in the preceding works [16, 11, 15]. Let $I = \{i_1, i_2, i_3, \ldots, i_m\}$ be a finite set of items and TD be a transaction database $\{T_1, T_2, T_3, \ldots, T_n\}$ in which each transaction $T_i \in TD$ is a subset of I. each item in a transaction associated with purchased quantity that is also called as internal utility of item defined by $iu(i_p, T_q)$, for example, $iu(A, T_2) = 2$, in Table 1. External utility of an item is the unit profit value defined by $eu(i_p)$, for example, $eu(B) = 5$, in Table 2.

Definition 1. The utility of an item i_p in a transaction T_q is the product of its internal utility and external utility and it is defined by $u(i_p) = iu(i_p, T_q) \times eu(i_p)$, for example, $u(C, T_1) = 2 \times 1 = 2$, in Table 1 and Table 2.

Definition 2. The database utility of an item i_p is the sum of all utilities in the database defined by $du(i_p) = \Sigma_{i_p \in T_q \in TD} u(i_p)$, for example, $du(D) = u(D, T_1) + u(D, T_3) + u(D, T_4) + u(D, T_5) = 3 + 1 + 14 + 1 = 19$, in Table 1 and Table 2.

Definition 3. The utility of an itemset X in transaction T_q is defined as $u(X, T_q) = \Sigma_{i_p \in X} u(i_p, T_q)$, for example, $u(AB, T_2) = 11$, in Table 1 and Table 2.

Definition 4. The database utility of an itemset X in a database TD is defined by $u(X, TD) = \Sigma_{i_p \in X \in TD} u(i_p, T_q)$, for example $u(AB, TD) = u(AB, T_2) + u(AB, T_3) = 11 + 13 = 24$, in Table 1 and Table 2.

Definition 5. Transaction utility can be defined as the sum of the individual utilities of all items in that transactionT_q, defined by $tu(T_q) = \Sigma_{i_p \in T_q} u(i_p)$,$tu(T_1) = 20$, in Table 1 and Table 2.

Definition 6. The total database utility is the sum of the all transaction utilities in the database, defined by $tdu(TD) = \Sigma_{T_q \in DB} tu(T_q)$, $tdu(TD) = tu(T_1) + tu(T_2) + tu(T_3) + tu(T_4) + tu(T_5) = 20 + 12 + 25 + 29 + 4 = 90$, in Table 1 and Table 2.

Definition 7. Transaction weighted utility of an item i_p in the transaction database TD is defined as $twu(i_p, TD) = \Sigma_{i_p \in T_p \in TD} tu(T_q)$, $twu(A, TD) = tu(T_1) + tu(T_3) + tu(T_5) = 12 + 25 + 4 = 41$, in Table 1 and Table 2.

Definition 8. Minimum utility threshold is given by the percentage of the total transaction database utility, defined by $minUtil = tdu(TD) \times \delta$.

Definition 9. High utility itemset is an itemset with utility $\geq minUtil$.

Definition 10. High utility pattern mining is the process of finding patterns with utility more than the specified minimum utility threshold.

4 Proposed Algorithm

In this section, we describe our proposed algorithm called TTBM, this algorithm requires three database scans. During the first scan it finds the high transaction weighted utility of each item, in second scan constructs the tree based on the high transaction weighted utility patterns. Next it updates the actual utilities of pattern in tree by scanning database again. Brief description of the three scans of the algorithm for the above transaction database(Table 1) is given below.

4.1 First Scan

During the first scan, algorithm loads each transaction one by one and accumulates the transaction weighted utility of the items. For example, consider the transaction

Algorithm 1. TTBM Algorithm

```
Input: Transaction Database TD; Minimum Utility Threshold $\delta$
Output: High utility Patterns
while $input \neq null$ do
    for each item do
        claculate TWU value
        if $TWU > \delta$ then
            add item to the itemsList
    sort itemsList TWU desc order
while $input \neq null$ do
    for each transaction do
        if item $TWU > \delta$ then
            add item to the oList
            TU = TU + itemUtility
        sort oList items as in itemsList
        tree.addTrans(oList, TU)
HuiList = mfpgrowth(tree)
tree.addHui(HuiList) while $input \neq null$ do
    for each transaction do
        load items and generate subitemsets
        calculate Utility of each subset
        tree.addUtilities(subitemsets, listutility)
hui_list = tree.tracing($\delta$)
return hui_list
```

database and utility table in Table 1 and Table 2, after first scan algorithm finds the following transaction weighted utilities of items, A:41, B:86, C:57, D:78, E:74. If any item transaction weighted utility is lower than the minimum utility that will be discarded, and that will not be consider for further computation so number of candidate patterns effectively reduced.

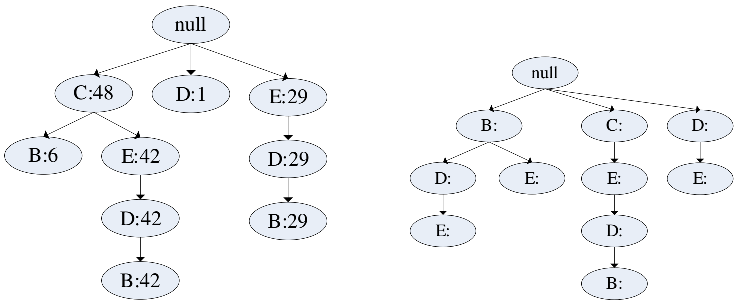
4.2 Second Scan

During second scan, algorithm loads all items of a transaction except the items which are discarded in the first scan. No item will be discarded if minimum utility is set to 30, item A will be discarded for minimum utility 50 (i.e. twu(A) = 41). Transaction utility of each transaction will also be calculated in parallel, if any item is discarded that utility is subtracted from transaction utility. The below table (Table 3) represents the transaction database during second scan, items are ordered in transaction weighted utility ascending. Next by reading each transaction TWU-Tree is constructed that is shown in Fig. 1(a). HP-Tree in Fig. 1(a) is constructed for high transaction weighted utility patterns {CEDB}, {DE}, {BE}, {BD}, and

{BDE}. While loading first transaction root node is null so a new node with item C will be constructed and transaction utility is assigned to the utility of that node. Next we check child list of current node for the next item E, as the item C do not have any child so E will be attached as child to the node C, in this fashion all items will be processed. While processing next transaction first we will check first item exists in the children list of root node, if not it will be created else transaction utility is accumulated to existing. Next modified pattern growth is applied to find high transaction weighted utility patterns. HP-Tree is constructed for all high transaction weighted utility patterns, Fig. 1(b) represent the HP-Tree after adding all high transaction weighted utility patterns.

Table 3 Updated Transaction Database

Tid	Item C	Item E	Item D	Item B	TU
T_1	2	1	3	1	20
T_2	1	0	0	1	6
T_3	1	1	1	2	22
T_4	0	1	14	1	29
T_5	0	0	1	0	1



(a). Transaction weighted utilities (b). High Transaction weighted utility itemsets

Fig. 1 HP-Tree

4.3 Third Scan

In third scan, candidate patterns are generated and utility also calculated in parallel. These candidate utilities are updated in HP-tree only if candidate is exist in the tree, Fig. 2 represents the HP-Tree after adding all high transaction weighted utility patterns. After adding all candidate utilities by traversing HP-Tree all high utility patterns with utility more than the specified utility are generated. Algorithm generates the high utility patterns {BDE:68} and {BE:50} for minimum utility 50.

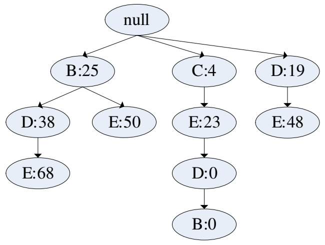


Fig. 2 HP-Tree with utilities

5 Experimental Evaluation

All algorithms are implemented in java and experiments are preformed on a PC with processor Intel $Core^{TM}$ i7 2600 CPU @ 3.40 G_{HZ}, 2GB Memory and the operating system is Microsoft Windows 7 32-bit.

5.1 Synthetic Datasets

IBM synthetic dataset generator was used for generating datasets. The parameter settings for generating datasets are followed from [1]. IBM generator generates the binary database, so fit this into the scenario of utility pattern mining purchased quantities of items in every transaction is generated randomly ranging from 1 to 5. External utility (unit profit) value is also generated randomly ranging from 1 to 20 by following lognormal distribution as shown in Fig. 3.

5.2 Experimental Results

Experiments are done over synthetic datasets to evaluate the performance of proposed approach in different cases. In first case, we have performed experiments with varying minimum utility thresholds (low range 0.01 to 0.09 and high range 0.1 to 0.9). In next and subsequent cases, experiments are done to check the scalability of algorithms with varying number of distinct items, number of transactions, and transaction lengths respectively. In the Fig. 4(a) we have shown only the performance comparison of our approach with HUI-Miner, because Two-Phase algorithm

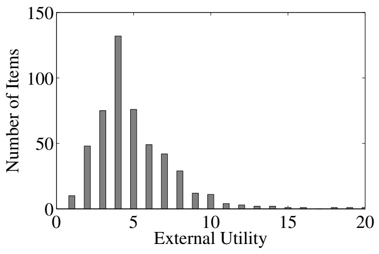
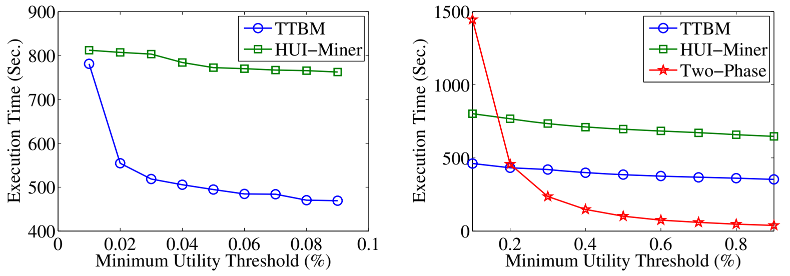


Fig. 3 External utility distribution for 300 items

Table 4 Number of HUPs on T8I4D50K

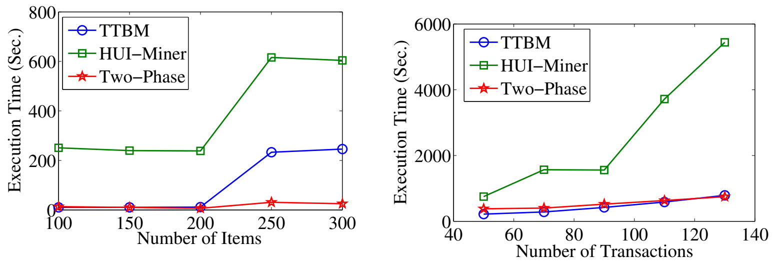
MinUtility(%)	# of HUP	MinUtility(%)	# of HUP
0.01	596954	0.1	12350
0.02	199106	0.2	3435
0.03	104845	0.3	1574
0.04	64322	0.4	879
0.05	42997	0.5	555
0.06	31073	0.6	376
0.07	23597	0.7	271
0.08	18729	0.8	199
0.09	14983	0.9	149



(a). Low δ range (b). High δ range

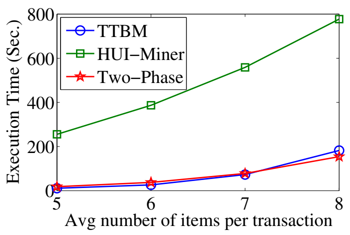
Fig. 4 Varying minimum utility thresholds

performance is not good at low minimum utility thresholds that can also be observed from Fig. 4(b) that the execution time is exponentially increasing with the decreasing utility threshold. Fig. 4(a) & (b) shows that performance of our proposed algorithm is far better than the performance of HUI-Miner for different utility ranges. From Fig. 4(b) we can also observe that execution time of Two-Phase algorithm is more efficient than proposed and HUI-Miner.

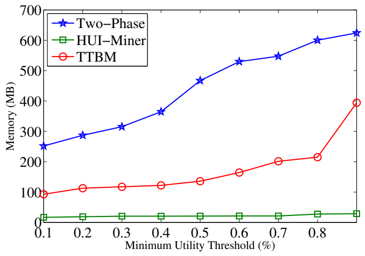


(a). Varying number of items (b). Varying number of transactions

Fig. 5 Varying number of items and transactions



(a). Varying average length of transactions



(b). Memory consumption

Fig. 6 Varying average length, minimum utility

From Fig. 5(a) we can observe that effect of increasing number of distinct items on the execution time of the algorithms. From Fig. 5(a) we can also observe that there is slight variation on the performance of proposed and Two-Phase algorithm, but meager effect on the HUI-Miner algorithm. Fig. 5(b) and Fig. 6(a) shows that effect of execution performance on varying number of transactions, and transaction length is similar on proposed and Two-Phase algorithm and smaller than that of HUI-Miner. Fig. 6(b) shows memory consumption of three algorithms.

6 Conclusion

In this paper we have proposed a time-efficient algorithm for effectively mine the high utility patterns from transaction database. Our algorithm only need maximum three database scans compare to the existing few algorithm those require multiple database scans. Experimental performance analysis shows that proposed algorithm execution time is better in all cases. However, the execution time of our algorithm is efficient, memory consumption is more. So in our future work we make our algorithm more efficient in both time and memory in finding the complete set of high utility patterns.

References

1. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th Int. Conf. VLDB, vol. 1215, pp. 487–499 (1994)
2. Ahmed, C.F., Tanbeer, S.K., Jeong, B.-S., Lee, Y.-K.: An efficient candidate pruning technique for high utility pattern mining. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS (LNAI), vol. 5476, pp. 749–756. Springer, Heidelberg (2009)
3. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Lee, Y.K.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE Transactions on Knowledge and Data Engineering 21(12), 1708–1721 (2009)
4. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: generalizing association rules to correlations. ACM SIGMOD Record 26, 265–276 (1997)

5. Erwin, A., Gopalan, R.P., Achuthan, N.: A bottom-up projection based algorithm for mining high utility itemsets. In: Proc. 2nd Int'l Workshop on Integrating Artificial Intelligence and Data Mining, vol. 84, pp. 3–11. Australian Computer Society, Inc. (2007)
6. Erwin, A., Gopalan, R.P., Achuthan, N.: Ctu-mine: An efficient high utility itemset mining algorithm using the pattern growth approach. In: Proc. 7th IEEE Int'l Conf. on CIT, pp. 71–76 (2007)
7. Fung, B.C., Wang, K., Ester, M.: Hierarchical document clustering using frequent itemsets. In: Proc. of SIAM Int'l Conf. on Data Mining, pp. 59–70 (2003)
8. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. ACM SIGMOD Record 29, 1–12 (2000)
9. Li, Y.C., Yeh, J.S., Chang, C.C.: Isolated items discarding strategy for discovering high utility itemsets. Data & Knowledge Engineering 64(1), 198–217 (2008)
10. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: Proc. 21st ACM Int'l Conf. on Information and Knowledge Management, pp. 55–64 (2012)
11. Liu, Y., Liao, W.-k., Choudhary, A.K.: A two-phase algorithm for fast discovery of high utility itemsets. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 689–695. Springer, Heidelberg (2005)
12. Shi, Y., Liao, W.K., Choudary, A., Li, J., Liu, Y.: High utility itemsets mining. Int'l Journal of Information Technology & Decision Making 09(06), 905–934 (2010)
13. Tseng, V., Shie, B.E., Wu, C.W., Yu, P.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Transactions on Knowledge and Data Engineering 25(8), 1772–1786 (2013), doi:10.1109/TKDE.2012.59
14. Tseng, V.S., Wu, C.W., Shie, B.E., Yu, P.S.: Up-growth: an efficient algorithm for high utility itemset mining. In: Proc. 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pp. 253–262 (2010)
15. Yao, H., Hamilton, H.J.: Mining itemset utilities from transaction databases. Data & Knowledge Engineering 59(3), 603–626 (2006)
16. Yao, H., Hamilton, H.J., Butz, C.J.: A foundational approach to mining itemset utilities from databases. In: Proc. 4th SIAM Int'l Conf. on Data Mining, pp. 482–486 (2004)
17. Yao, H., Hamilton, H.J., Geng, L.: A unified framework for utility-based measures for mining itemsets. In: Proc. ACM SIGKDD 2nd Workshop on Utility-Based Data Mining, pp. 28–37 (2006)
18. Yen, S.J., Chen, C.C., Lee, Y.S.: A fast algorithm for mining high utility itemsets. In: Behavior Computing, pp. 229–240 (2012)