Modified GUIDE (LM) algorithm for mining maximal high utility patterns from data streams

Chiranjeevi Manike [1] [*], **Hari Om** [2]

[1,2] *Department of Computer Science and Engineering, Indian School of Mines, Dhanbad, Jharkhand 826004, India*

[1] *chiru.research@gmail.com*

[2] *hariom4india@gmail.com*

Received 19 September 2013

Accepted 24 January 2015

Abstract

High utility pattern mining is an emerging research topic in the data mining field. Unlike frequent pattern mining, high utility pattern mining deals with non-binary databases, in which the information about purchased quantities of items is maintained. Due to the non-existence of anti-monotone property among the utilities of itemsets, utility mining becomes a big challenge. Moreover, discovering useful patterns from the huge number of potential patterns is a mining bottleneck. However, the compact (Closed and Maximal) high utility pattern mining moderately lessens the number of patterns, but it does not solve it. Recently, an efficient framework called GUIDE, was proposed in the literature to address this issue. Though, GUIDE effectively reduced the number of high utility patterns, yet the quality of few mined patterns and their utilities are not exact. In view of this, we propose a modified MGUIDE$_{LM}$ algorithm to improve the quality and determine exact utilities of maximal patterns.

Keywords: High utility patterns, Data mining, Maximal Patterns, Anti-monotone property, Transaction projection.

1. Introduction

Frequent pattern mining is probably one of the most important concepts in data mining. It is a process of discovering the complete set of frequent patterns, based on the support measure [2]. The support is defined over the binary database, where the information of each item in the database is represented in the form of 0s and 1s. The occurrence frequency of an itemset only represents the statistical significance, but not the semantic significance. Moreover, in the real world scenarios customers may purchase multiple items of different quantities. Sometimes less frequently purchased items are also contribute more to the total profit. Thus, the occurrence frequency of an itemset is not sufficient to measure the importance of an itemset. Hence, utility mining model [1] was introduced, in which the statistical and semantic significance of the itemsets are considered.

For example, consider the transaction database (i.e., Table 1), in which each item that is present has some numerical value that is called purchased quantity. Also consider the utility table (i.e., Table 2), in which unit profit of each item is given. In frequent itemset mining support is defined as the portion of transactions in the data set which contain the item-

[*]Corresponding Author.

set. Suppose the minimum support threshold is set to 60%. If we apply Apriori or any other frequent itemset mining algorithm, then it discovers the item E as frequent, since, it appeared in more than 60% of the transactions in the transaction database (i.e., Table 1). Remaining items A, B, C and D are omitted, because of their low (i.e., <60%) occurrence frequencies. Among these items, item B contributes more profit (i.e., 240$) than the infrequent items A, C, D as well as frequent item E. Therefore, frequent itemset mining may neglect some infrequent items, even though they are contributing more profit to the company. To overcome above limitations in frequent itemset mining, utility mining was introduced in the year 2004.

Table 1. Transaction Database

Item/Tid	A	B	C	D	E
T_1	0	0	18	0	1
T_2	0	6	0	1	1
T_3	2	0	1	0	1
T_4	1	0	0	1	1
T_5	0	0	4	0	2
T_6	1	1	0	0	0
T_7	0	10	0	1	1
T_8	3	0	25	3	1
T_9	1	1	0	0	0
T_{10}	0	6	2	0	2

Table 2. Utility Table

Item	A	B	C	D	E
Profit($)	3	10	1	6	5

Utility mining discovers the patterns with utility more than the user specified minimum utility threshold. The utility of an itemset may be defined in terms of its purchased quantity, profit, cost and any other useful measure, which reflect the semantic significance of an itemset. The Apriori anti-monotone [2] property facilitates the process of mining frequent patterns. Due to the absence of anti-monotone property in utility mining model, the process of mining high utility patterns becomes more complex than the process of frequent pattern mining. Utility mining may be considered as an extension of frequent itemset mining. The first theoretical model of utility mining and basic definitions were given by Yao et al. [1], in their approach two pruning strategies are identified, based on the utility and support of an itemset. These strategies are overestimating the high utility patterns. Thus, authors proposed two more algorithms in order to reduce the overestimation rate. Later, an efficient pruning strategy called transaction weighted utility downward closure property, was introduced by Liu et al. [3]. Based on this property authors proposed an algorithm called Two-Phase. In Phase I of this algorithm, it overestimates some high transaction weighted utility patterns. In Phase II, it determines high utility patterns from the high transaction weighted utility patterns by calculating their actual utilities.

Though, transaction weighted utility downward closure (anti-monotone) property effectively reduces the patterns, filtering huge number of high utility patterns at low minimum utility threshold range becomes a main bottleneck. Hence, researchers addressed this issue by introducing a concept of compact form of high utility patterns. There are two compact representations of high utility patterns namely, closed and maximal. An itemset is said to be closed if it does not have any immediate proper superset with same support, whereas a maximal itemset can be an itemset with no subsets. Recently, a framework called GUIDE [18], was introduced to discover the maximal high utility patterns from data streams. Due to the incomplete information given by their procedure (i.e., namely Transaction Projection), algorithm may not identify the complete set of maximal patterns and their utilities. In view of this, in this paper, we have investigated the problem and proposed modified version of $GUIDE_{LM}$ algorithm by incorporating our procedure. Modified algorithm has improved the quality of the patterns as well as given exact utilities of patterns.

The remaining part of this paper is organized as follows. The problem statement and basic definitions of high utility pattern mining are given in Section 2. In Section 3, related work and the procedure of transaction projection is discussed. In Section 4, we discussed our proposed algorithm. Experimental results are presented in Section 5. Finally, conclusions and future enhancements are given in Section 6.

2. Basic Definitions

We have adopted the similar definitions presented in the previous works [1], [3], [4], [8]. Let $I = \{i_1, i_2, i_3, \ldots, i_m\}$ be a set of items and DB be a transaction database $\{T_1, T_2, T_3, \ldots, T_n\}$, where each transaction $T_i \in DB$ is a subset of I.

Definition 1. The local transaction utility value, denoted as $lu(i_p, T_q)$, represents the purchased quantity of item i_p in a transaction T_q. For example, $lu(D, T_2) = 1$, in Table 1.

Definition 2. The external utility, denoted as $eu(i_p)$, is the unit profit of item i_p. For example, $eu(B) = 10$, in Table 2.

Definition 3. The utility, denoted as $u(i_p, T_q)$, is the quantitative measure of utility for an item i_p in a transaction T_q, is defined by $u(i_p, T_q) = lu(i_p, T_q) \times eu(i_p)$. For example, $u(C, T_1) = 18 \times 1 = 18$, in Table 1 & Table 2.

Definition 4. The utility of an itemset X in a transaction T_q, denoted as $u(X, T_q)$, is defined by $u(X, T_q) = \Sigma_{i_p \in X} u(i_p, T_q)$, where $X = \{i_1, i_2, i_3, \ldots i_k\}$ is a k-itemset, $X \subseteq T_q$ and $1 \leqslant k \leqslant m$. For example, $u(AB, T_6) = 1 \times 3 + 1 \times 10 = 13$, in Table 1 & Table 2.

Definition 5. The utility of an itemset X in DB is defined as $u(X) = \Sigma_{T_q \in DB \wedge X \subseteq T_q} u(X, T_q)$. For example, $u(AB) = u(AB, T_6) + u(AB, T_9) = 13 + 13 = 26$, in Table 1 & Table 2.

Definition 6. The transaction utility of a transaction T_q, denoted as $tu(T_q)$, describes the total profit of that transaction and it is defined by $tu(T_q) = \Sigma_{i_p \in T_q} u(i_p, T_q)$. For example, $tu(T_5) = u(C, T_5) + u(E, T_5) = 4 + 10 = 14$, in Table 3.

Definition 7. The transaction weighted utility of an itemset X is denoted as $twu(X)$ and it is the sum of the transaction utilities of all transactions containing X, i.e., $twu(X) = \Sigma_{X \subseteq T_q \in DB} tu(T_q)$. For example, $twu(AB) = tu(T_6) + tu(T_9) = 13 + 13 = 26$, in Tables 1, 2 & 3.

Definition 8. The minimum utility threshold δ, is given by the percentage of total transaction utility values of the transaction database(Table 1). In Table 3, the summation of all the transaction utility values is 400. If δ is 35% (or we can also express it as 0.35), then the minimum utility value can be defined as $minUtil = \delta \times \Sigma_{T_q \in DB} tu(T_q)$. Therefore, $minUtil = 0.35 \times 400 = 140$.

Definition 9. An itemset X is high utility itemset, if $u(X) \geqslant minUtil$.

Definition 10. An itemset X is high transaction weighted utility itemset, if $twu(X) \geqslant minUtil$.

The high utility pattern mining is a process of finding patterns with utility more than the specified minimum utility threshold(δ).

3. Related Work

In this section, the related works are then described below. They are high utility pattern mining, compact form of high utility pattern mining in the context of traditional databases and data streams respectively.

3.1. High Utility Pattern Mining

A theoretical model and basic definitions of high utility mining were given by Yao et al. [1], in which the authors have also identified two properties to limit the upper bounds of potential high utility patterns, based on the values of utility and support of an itemset. However, these upper bounds are too big to identify actual high utility patterns. Hence, authors have modified these properties and introduced two more properties called utility upper bound and expected utility upper bound. They have also proposed two algorithms, called UMining and UMining_H [4], by incorporating these properties. Though, these algorithms minimized the upper bounds value, still they generate candidate patterns in each level and check every candidate by calculating its actual utility value. Thus, process of discovering the useful patterns from the large number of potential patterns is more complex and it needs several database scans. Afterwards, an efficient pruning strategy was introduced by Liu et al. [3], it follows the anti-monotone property. It is also called

Table 3. Transaction Utility

Tid	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
twu	23	71	12	14	14	13	111	57	13	72

as transaction weighted utility(twu) downward closure property. The correlation between the transaction weighted utility value and the actual utility value of itemset is more positive than the earlier upper bounds. According to this strategy, the transaction weighted utility of a super itemset is high, only if its sub itemsets transaction weighted utilities are high. This strategy never underestimates any low utility itemsets, it rather overestimates (i.e., discovers the patterns, where the actual utility of those patterns is not more than the specified utility threshold) some itemsets. This approach also suffers from the level-wise candidate generation and test problem like earlier approaches.

Later on, an efficient strategy called, IIDS (Isolated Itemset Discarding Strategy) was introduced by Li et al. [7]. Authors have adopted two efficient share mining models [20, 21, 23], and applied IIDS to these models to improve the performance of high utility pattern mining algorithms, which are based on the level-wise candidate generation-and-test approach. Although, the above algorithms effectively reduce the number of potential patterns, still suffers from the level-wise candidate generation-and-test problem. To overcome the limitations of level-wise approach, Erwin et al. [5] proposed an approach called CTU-Mine, which is based on the pattern growth approach. In this approach, first each item transaction weighted utility is accumulated by scanning the database once, and then low transaction weighted utility items are discarded, these items will not be considered for further calculations. Since, superset of a low twu itemset can not become high utility itemset (i.e., according to the anti-monotone). After processing all transactions with another scan of the database, the information on remaining items is kept in a CTU-Tree. During this step items in each transaction are processed excluding discarded items. Thus, the problem with level-wise candidate generation-and-test approach is overcome, since, it needs no candidate generation process and requires maximum two database scans. Afterwards, extension works of CTU-Mine called CTU-PRO [22] and CTU-PROL [6], were proposed to improve the performance of high utility pattern mining by reducing the number of candidate patterns. Later, many approaches were proposed in high utility pattern mining based on the pattern growth approach.

A novel tree-based candidate pruning technique called HUC-Prune, was proposed by Ahmed et al. [8], this technique finds the cumulative sum of transaction utilities of items and removes the items that have low (i.e., twu is not more than the minimum utility threshold) transaction weighted utility(twu). The remaining items are sorted in descending order of their twu values. In second scan, the items in each transaction are processed in the same order and inserted into the tree. By using pattern growth approach, it finds all high transaction weighted utility patterns. In the final scan, it calculates the actual utilities of high transaction weighted utility patterns and determines high utility patterns. Afterwards, Tseng et al. [9] proposed an efficient approach called UP-Growth, in which several pruning strategies have been applied to reduce the number of candidate patterns in phase I of the Two-Phase algorithm. Lin et al. [10] proposed an efficient tree structure called HUP-Tree, and applied the HUP-Growth mining algorithm. The HUP-Growth algorithm achieved better performance over the Two-Phase algorithm. Subsequently, many approaches were proposed in the literature to improve the efficiency of high utility pattern mining [11, 13].

On the other hand, the problem of mining high utility patterns from evolving data streams becomes more complex. Since, the transactions in the transaction data stream are arriving continuously with rapid rate, and limited amount of memory and execution time available to process. Tseng et al. [14] first addressed the problem of high utility pattern mining in the context of data streams and proposed an efficient algorithm called THUI-Mine, using the sliding window model. This approach, processes the transactions in the data stream batch by batch fashion. For each batch in the window, it finds the transaction weighted utility of each item and then discards the items which have low transaction weighted utility. Next, it generates the candidate 2-itemstes and maintains their transaction weighted utility value and information of window (i.e., in which it has appeared) in a tree like structure. With filtering threshold, this algorithm prunes the itemsets during window sliding. Next, from high transaction weighted utility 2-itemsets, it generates a set of candidate k-itemsets $(k \geqslant 2)$. Finally, it does one more database scan to find the actual high utility patterns.

Like earlier approaches in the context of traditional databases, THUI-Mine also overestimates many patterns, this leads to the increase in usage of memory and processing time. Li et al. [15], proposed another two efficient approaches namely MHUI-TID and MHUI-BIT, in which the transaction information is represented in the form of Tidlist and Bitvector respectively. Then high transaction weighted utility patterns (i.e., 1-itemsets) are identified by processing these lists, and the high transaction weighted candidate 2-itemsets are generated and updated in a tree called LexTree. The Tidlist, Bitvector representations and construction of LexTree for 2-itemsets facilitate optimal utilization of resources, but for processing the itemsets of length more than 2, they still depend on the level-wise candidate generation-and-test approach. Moreover, mining process becomes more complex to manage (i.e., Tidlist, Bitvector), when the number of transactions or items becomes more.

Afterwards, an approach called HUPMS [17], was proposed based on the pattern growth approach, in which the transaction utility of each transaction in a batch is calculated, and items in each transaction are sorted according to lexicographic order. Next, the items in each transaction are inserted in HUS-Tree along with transaction utility value. Each node in the HUS-Tree contains transaction utilities in decreasing sequence to facilitate the easy deletion when the window slides. To mine the high utility patterns, pattern growth approach is applied to generate all high transaction weighted utility patterns. Finally, it performs one more scan to calculate the actual utilities of patterns. Although, the HUPMS achieved better performance as compared to the MHUI-TID, still it generates many false patterns. Therefore, in the context of traditional databases and data streams, algorithms still suffers from the problem of huge number of patterns when the minimum utility threshold is set to low.

3.2. *Maximal High Utility Patterns*

To overcome the problem of discovering useful patterns from the huge number of low utility patterns, algorithms have adopted interesting alternates, which are already been used in the frequent itemset mining [19]. Alternatives which are used earlier in frequent itemset mining are closed and maximal patterns. So for, very few approaches are available in the literature namely GUIDE [16], UMMI [12], and GUIDE [18] for mining the maximal high utility patterns. Based on the context of traditional databases, an approach called UMMI, was proposed by Lin et al. [12]. UMMI algorithm used a maximal itemset property and lexicographic tree structure in mining high utility patterns. Like earlier approaches, UMMI also accumulates transaction weighted utility of each item and discards the low transaction weighted utility items. Next, remaining items are sorted according to their transaction weighted utility ascending order. In the next phase, items in each transaction excluding discarded items are processed in the above order. These items are updated in HTP-Tree along with transaction utilities. Next, by applying pattern growth approach maximal high transaction weighted utility patterns are discovered. By inserting all maximal high transaction weighted utility patterns it builds MLexTree, then by doing last scan of the database it updates the actual utility of patterns. Finally, by tracing MLexTree, algorithm generates all maximal high utility patterns.

To mine the non-redundant high utility patterns from the data streams, an efficient algorithm called GUIDE [16], was proposed, based on landmark window model. To generate potential maximal high utility patterns, a procedure called transaction projection(TP), was used in this approach. The experimental results shown that the GUIDE outperforms the Two-Phase algorithm. Later, same authors were proposed an efficient framework namely GUIDE [18], for each window model of the data stream. In this framework, three algorithms are designed: $GUIDE_{LM}$, $GUIDE_{SW}$ and $GUIDE_{TF}$ for landmark window, sliding window, and time fading window respectively. A common procedure called transaction projection(TP), is used for generating potential patterns from the transactions. First, from each transaction, algorithms generates projections and also calculates actual utilities in parallel. Next, these projections are updated in a tree structure (i.e., MUI-Tree). To mine the patterns, algorithms use an efficient bottom-up approach. During tree tracing, if any node utility is more than the user specified minimum utility threshold, then it will send the pattern corresponding to that node to the output, and it skips checking the remaining nodes along this path. Though, algorithm efficiently finds the maximal high utility patterns with single database scan, it generates only approximate patterns. Therefore, GUIDE framework failed to achieve the 100 percent quality patterns.

3.3. Analysis of Patterns Generated by Transaction Projection(TP)

Transaction Projection(TP) is a procedure which is used for generating potential patterns in GUIDE framework. Let us assume that a transaction contains a set of items $\{i_1, i_2, i_3, \dots, i_n\}$. Procedure TP projects these items into all its postfixes i.e., $\{\{i_1, i_2, i_3, \dots, i_n\}, \{i_2 i_3, \dots, i_n\}, \{i_3, \dots, i_n\}, \dots, \{i_n\}\}$. Then each postfix is projected again to get all its prefixes, such as $\{i_2, i_3, \dots, i_n\}$ is projected into $\{\{i_2, i_3, \dots, i_n\}, \{i_2, i_3, \dots, i_{(n-1)}\}, \dots, \{i_2, i_3\}, \{i_2\}\}$. Therefore, the total number of projections generated by TP from a transaction (if the transaction contains n items) is $n(n+1)/2$. Consider Fig. 1, that represents the total search space for 5 items (i.e., A, B, C, D and E). All patterns within the dotted line region are the total number of projections generated by TP.

Example. Let us consider a transaction T, which contains 5 items A, B, C, D, and E. The itemset (i.e., a set of these five items) can be represented as {A, B, C, D, E}/ {ABCDE}. According to the above procedure (i.e., TP), first all postfixes of itemset {A, B, C, D, E} are generated, i.e., {A, B, C, D, E}, {B, C, D, E}, {C, D, E}, {D, E} and {E}. Next, from each postfix set of all prefixes are generated, for example, prefixes {A, B, C, D}, {A, B, C}, {A, B} and {A} are generated from postfix {A, B, C, D, E}. The same procedure is applied to the remaining postfixes to get the complete set of projections/itemsets from transaction T. All patterns within the dotted line region (i.e., in Fig. 1) are the projections generated by TP from transaction T. The term pattern, itemset and projection are used interchangeably in this paper. For our convenience, we use the notation of itemset as {ABC} instead of {A,B,C} in this paper. Suppose two transactions $T_1 = \{(i_1, u_1), (i_2, u_2), (i_3, u_3)\}$ and $T_2 = \{(i_1, u_4), (i_3, u_5)\}$ with items utility(i.e.,$u_1, u_2 \dots, u_5$) values are updated in the MUI-Tree after generating the projections. The projections generated from the transactions T_1 and T_2, are the following:

$\{\{i_1, i_2, i_3\} - (u_1 + u_2 + u_3), \{i_2, i_3\} - (u_2 + u_3), \{i_3\} - (u_3)\}$ $\{\{i_1, i_2\} - (u_1 + u_2), \{i_1\} - (u_1)\}$ $\{\{i_2\} - (u_2)\}$ $\{\{i_1, i_3\} - (u_4 + u_5), \{i_3\} - (u_5)\}$ $\{\{i_1\} - (u_4)\}$

Among these projections/patterns, the total utility of the pattern $\{i_1, i_3\}$ in the MUI-Tree is $(u_4 + u_5)$ instead of $(u_1 + u_3 + u_4 + u_5)$. Because the pattern $\{i_1, i_3\}$ is not generated from the transaction T_1, it is only generated from transaction T_2. Therefore, the GUIDE loss some patterns of this type while tracing the MUI-Tree for high utility patterns. Hence the quality of generated maximal patterns automatically decreases.

4. Proposed algorithm $MGUIDE_{LM}$

In this section, we discuss our proposed modified version of $GUIDE_{LM}$ algorithm. Here we mainly focus on the quality of patterns that are generated by our procedure. To resolve the problem with transaction projection of $GUIDE_{LM}$ algorithm, we have modified the procedure of transaction projection(TP) in such a way that it generates some additional patterns including the actual projections.

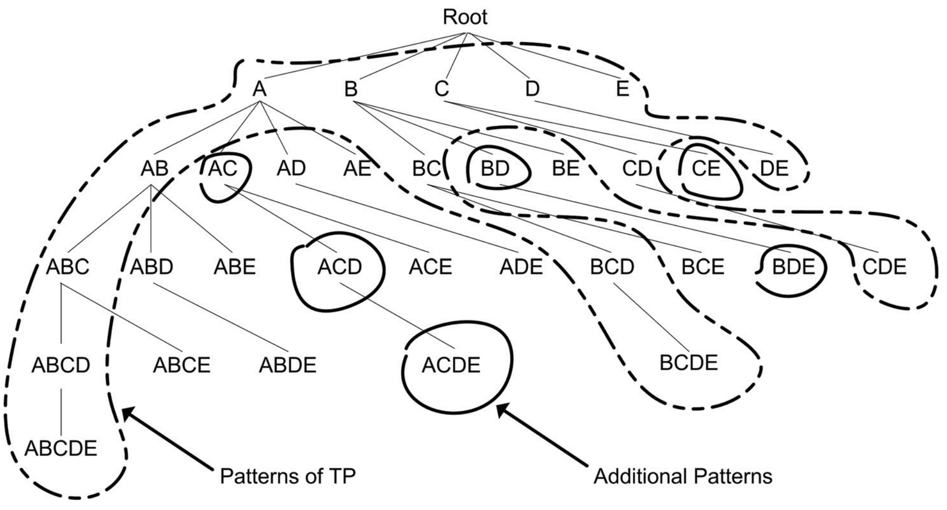


Fig. 1. Total search space for 5 items

4.1. Procedure of Generating Additional Patterns

Let $S = \{i_1, i_2, i_3, \dots, i_n\}$,$(n > 2)$ be a set of items in a transaction, assuming the items in lexicographic order. The following sets of itemsets of length 2 to n-1 are generated from S.
2-itemsets $\{\{i_1\} \bigcup \{i_3\}, \{i_2\} \bigcup \{i_5\}, \{i_3\} \bigcup \{i_6\}, \dots, \{i_{(n-2)}\} \bigcup \{i_n\}\}$
3-itemsets $\{\{i_1\} \bigcup \{i_3, i_4\}, \{i_2\} \bigcup \{i_4, i_5\}, \{i_3\} \bigcup \{i_5, i_6\}, \dots \{i_{(n-3)}\} \bigcup \{i_{(n-1)}, i_n\}$,
4-itemsets $\{\{i_1\} \bigcup \{i_3, i_4, i_5\}, \{i_2\} \bigcup \{i_4, i_5, i_6\}, \{i_3\} \bigcup \{i_5, i_6, i_7\}, \dots, \{i_{(n-4)}\} \bigcup \{i_{(n-2)}, i_{(n-1)}, i_n\}\}$
k-itemsets $\{\{i_1\} \bigcup \{i_3, i_4, i_5 \dots i_{(k+1)}\}, \{i_2\} \bigcup \{i_4, i_5, i_6 \dots i_{(k+2)}\}, \{i_3\} \bigcup \{i_5, i_6, i_7 \dots i_{(k+3)}\}, \dots, \{i_{(n-k)}\} \bigcup \{i_{(n-k+2}, \dots, i_{(n-1)}, i_n\}\}$
Example. Consider the transaction database and utility table in Tables 1 & 2 respectively. The transaction T_2 contains the set of items {B, D, E}. So, the generated additional pattern is {B, E} with utility 65. The actual projections generated by TP are {B, D, E}, {D, E}, {E}, {B, D}, {B}, {D}. In Fig. 1, the rounded content with dark line represents the additional patterns generated. The complete information about the patterns after updating all transactions of the transaction database (i.e., Table 1) in MUI-Tree and MMUI-Tree are shown in Figs. 2 & 3, each ellipse contains a pattern and its utility.

The patterns which are represented by white ellipses in Fig. 2, contains the incomplete information. The patterns {AD}, {ADE}, {ACE} and {CE} are updated with utilities 9, 14, 12 and 55 instead of actual utilities (i.e., in Fig. 3) 36, 46, 51, and 85 respectively. In Fig. 3, the MMUI-Tree contains actual projections and the additional patterns with exact utility values. Following is our modified procedure of transaction projection.
Procedure: Modified Transaction Projection
Input: A transaction T_i, n is the number of items
Output: Set of patterns of Tid_k including transaction projections, $P_i + Proj_k$.

```
1. $P_i := \phi$
2. Stack $Proj_k := \phi$
3. If $n > 2$
4. **for** $i := 1$ to $n-1$
5.         additional patterns of length 2 to $n-1$
6.         add patterns to $P_i$
7. **end for**
8. **while** $Tid_k \neq \phi$
9.         add $Tid_k, uTid_k$ into $Proj_k$
10.     $Tid_{k\_temp} = Tid_k$
11.         **while** $Tid_{k\_temp} \neq \phi$
12.         prune the last item of $Tid_{k\_temp}$
13.         add $Tid_{k\_temp}$ into $Proj_k$
14.     **end while**
15. prune the first item of $Tid_k$
16. **end while**
```

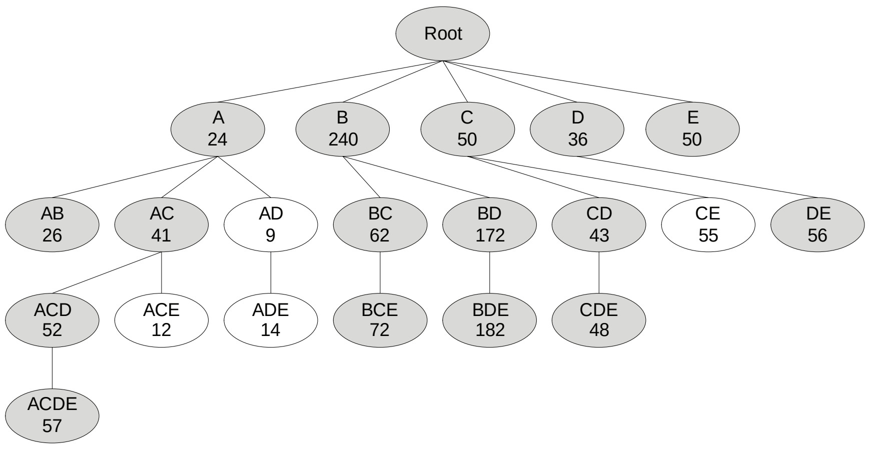


Fig. 2. Updated MUI-Tree

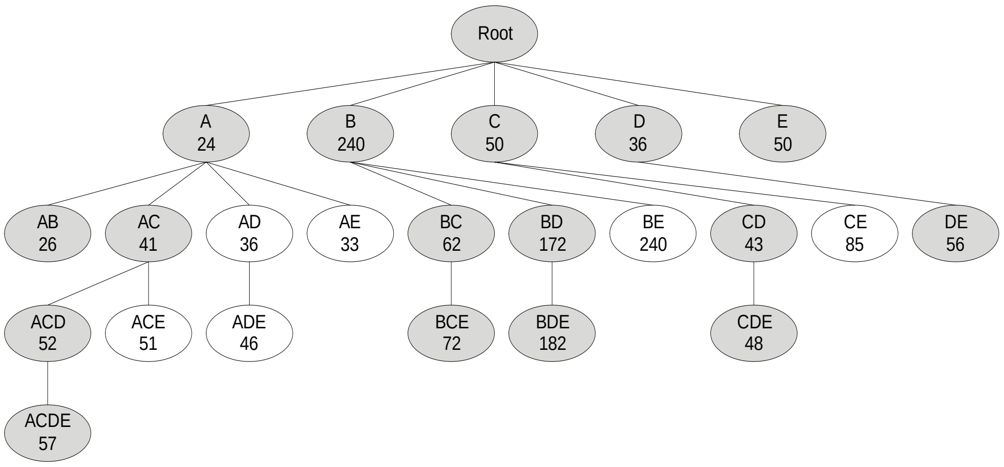


Fig. 3. Updated MMUI-Tree

Above procedure generates additional patterns, only if the number of items in a transaction is more than 2. After generating patterns/projections from a transaction, MMUI-Tree is constructed by inserting all these patterns. In a header table [10], the items are maintained in alphabetical order, each item linked to its node in the MMUI-Tree. Thus, the process of updating patterns utilities becomes more efficient. That is, the patterns with the same prefix as the items will directly be identified from the header table. Additional patterns with the same prefix as in the projections will be processed in sequence. While updating the MMUI-Tree, new node will be created if that pattern does not exits, otherwise the utility of that particular pattern will be incremented with new utility.

When the system gets a query from a user, the MMUI-Tree will be traced (traversed) using the bottom-up approach, to efficiently identify high utility maximal patterns. Tracing starts from the leftmost node and moves towards the root. While tracing the nodes, if any node utility is more than the user specified minimum utility threshold, the pattern corresponding to that node is generated as a high utility pattern. In that case, checking of remaining nodes along that path will be skipped and the pointer will be move to the leaf node of the next path. After generating all high utility patterns, the maximal high utility patterns are identified using the subset checking.

5. Experimental Results

In this Section, we show the performance of our approach under different parameters. We have analyzed the performance of our approach $MGUIDE_{LM}$ by comparing with that of the $GUIDE_{LM}$, the MHUI-TID and THUI-Mine algorithms. All algorithms are implemented in Java and the experiments were performed on a PC with processor Intel $Core^{TM}$ i7 2600 CPU @ 3.40 G_{HZ}, 2GB Memory and Microsoft Windows 7 32-bit operating system.

Data sets were generated using IBM Synthetic Dataset Generator [24], parameters used for data sets generation are given in Table 4. The IBM data set generator generates only the binary database, in order to fit this in the real scenario, items purchased quantities are generated randomly ranging from 1 to 5. The unit profits of all items are also generated randomly ranging from 1 to 20. In the real scenario, lower profit items purchased quantities are more, in other wards most items are in the low profit range. Hence, the items unit profits are generated by following the lognormal distribution, that is shown in Fig. 4.

We have generated several data sets by varying the parameters, and another Retail data set is also used in our experiments, which is obtained from the frequent itemset mining(FIMI) repository [25].

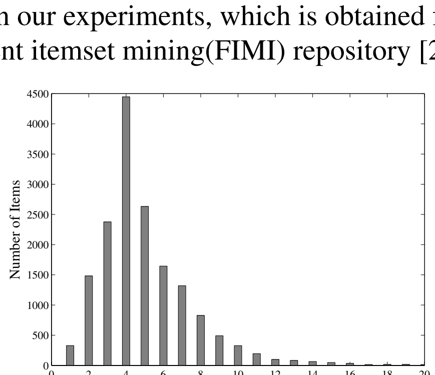


Fig. 4. External utility distribution (Lognormal distribution)

5.1. *Scalability of Modified* $MGUIDE_{LM}$

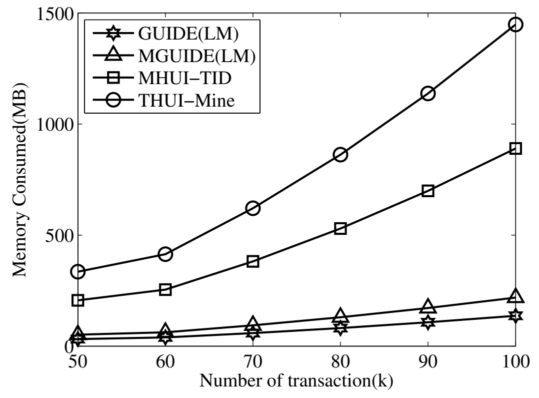
In this subsection, we show the scalability performance of algorithms. In this experiment, we have used a data set T10I8DxK, in which, the number of transactions (i.e., x) varies from 50K to 100K. The minimum utility threshold is set to 0.5%. We have analyzed the execution time and usage of memory, by increasing the number of transactions. The experimental results shown in Fig. 5, the scalability of the $GUIDE_{LM}$ and $MGUIDE_{LM}$ increases linearly in similar fashion as shown in Fig. 5. Our method is more efficient than the THUI-Mine and MHUI-TID, but it is slightly inferior as compared to the $GUIDE_{LM}$. Since, additional patterns are generated to improve the quality of the patterns.

In the second experiment, we have used a data set TxI8D50K, in which, the average number of items per transaction (i.e., x) varies from 5 to 25 and the minimum utility threshold is set to 0.8%. As the average number of items per transaction increases, the density of the data set becomes increase; so, the execution time and the memory requirement increases, that is shown in Fig. 6. From Fig. 6, we can also observe that the execution time of the $GUIDE_{LM}$ and $MGUIDE_{LM}$ algorithms increases linearly, whereas in the case of THUI-Mine and MHUI-TID algorithms, increases at much faster rate. Moreover, memory usage and execution time of THUI-Mine increases exponentially with increasing number of transactions.

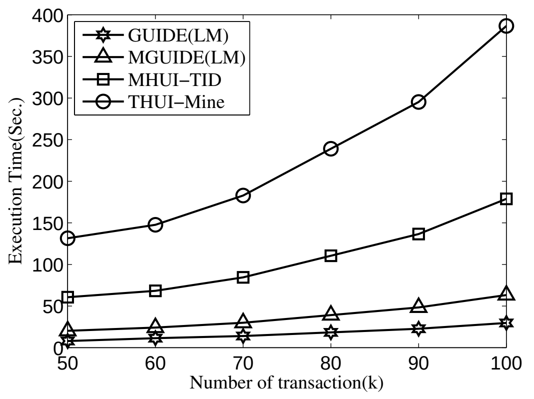
In the third experiment, we have used the data set T10I8D50K, where the number of items has increased from 0.5k to 2.5k and the minimum utility threshold is set to 0.5%. As the number of items in a data set increases, the distribution of items becomes sparser. So the density of itemset becomes lower. From the experimental results which are in Fig. 7, we can observe that the increasing number of items has made no significant effect on the execution time of any algorithm.

Table 4. Parameter settings for synthetic data generation

Parameter	Description	Values
nitems	Number of distinct items	0.5k,1k,1.5k,2k,2.5k
tlen	Average items per transaction	5,10,15,20,25
patlen	Average length of maximal pattern	4,6,8,10,12,14
ntrans	Number of transactions	50k,60k,70k,80k,90k,100k
quantity	Purchased quantity of an item	1-5
profit	Unit profit of each item	1-20



(a) Memory Consumption



(b) Execution Time

Fig. 5: Performance with varying number of transactions

5.2. *Performance of Modified* $MGUIDE_{LM}$ *With Minimum Utility Variation*

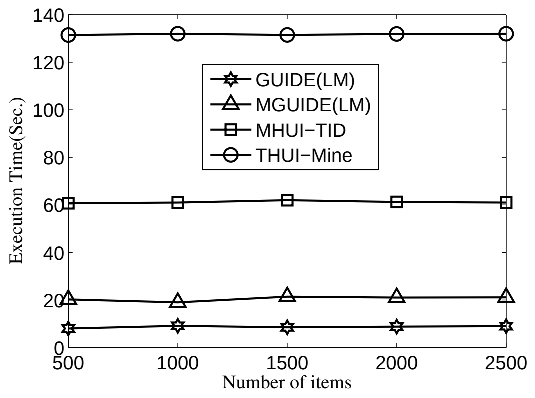
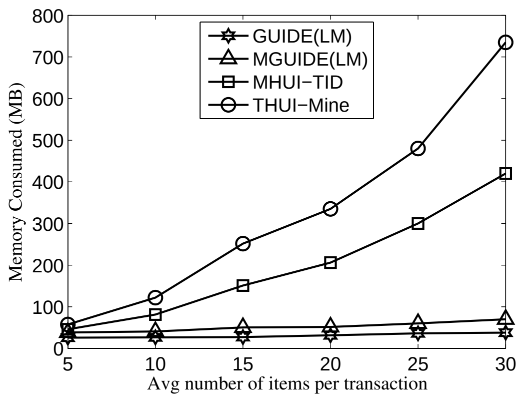


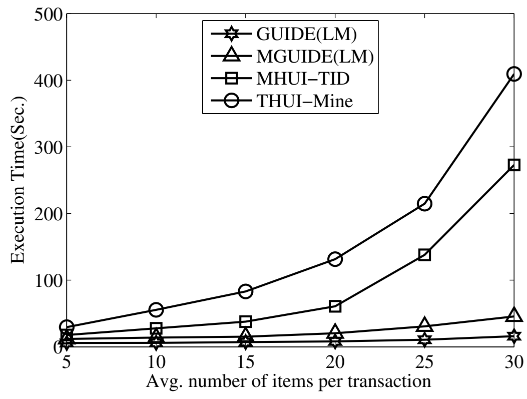
Fig. 7. Execution time vs. Number of items

In this experiment, we have used the Retail data set, in which the number of transactions, number of distinct items, average length of items per transaction and maximal pattern length are 88162, 16470, 10.3, and 76, respectively. This data set contains too many patterns at low utility ranges, thus, execution time and memory requirements of all algorithms becomes high at minimum utility threshold 0.1%, that we can observe from Fig. 8. The performance of all algorithms increases as we increase the minimum utility thresholds. There is slight variation in execution time and memory requirements of the $GUIDE_{LM}$ and $MGUIDE_{LM}$. Since, when the data set becomes sparser the performance of the $MGUIDE_{LM}$ matches that of the $GUIDE_{LM}$.

In the next experiment, we have investigated the quality of generated patterns. To evaluate the performance of proposed algorithm we have performed several experiments on synthetic data set (i.e., T10I4D100K) with varying minimum utility threshold. Results are shown in Table 5, from this table we can observe that the number of HUIs, Max-HUIs and Actual HUIs are decreasing with increas-

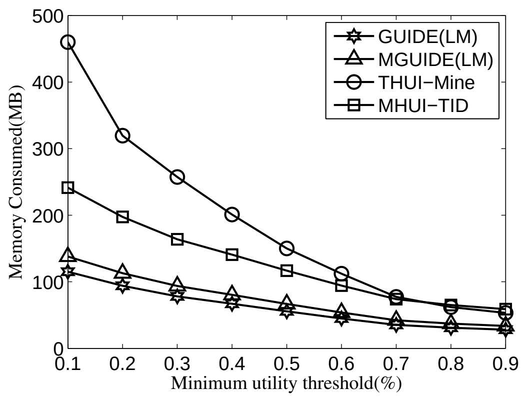


(a) Memory Consumption

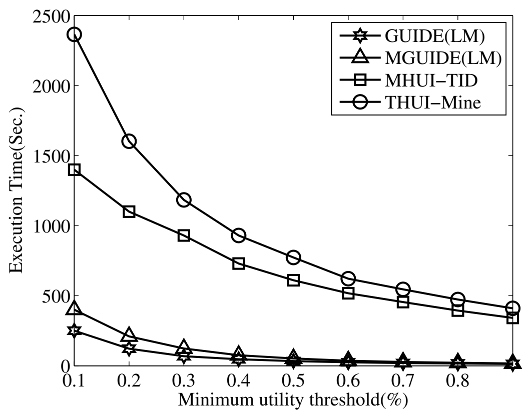


(b) Execution Time

Fig. 6: Performance with varying average number of items per transaction



(a) Memory Consumption



(b) Execution Time

Fig. 8: Performance on Retail dataset

ing minimum utility thresholds. The quality of patterns is measured using a measure called precision, equation (1) is used to calculate this value. From Table 5, we can observe that the quality of generated patterns, increasing with minimum utility threshold, since, at the lowest minimum utility threshold, number of patterns will be more. At minimum utility threshold 0.1%, our approach generated 910 MaxHUIs among 12350 HUIs. It effectively reduces 92.63% redundant patterns. Furthermore, with our modified procedure we have achieved the average precision about 91.03%(i.e., the accuracy of the generated patterns is 91.03%), which is comparatively better than the average precision of $GUIDE_{LM}$ (i.e. 83%). Therefore, inclusion of additional patterns significantly improved the quality of generated maximal patterns.

$$precision = \frac{|MaxHUI \cap \text{Actual MaxHUI}|}{|MaxHUI|} \times 100\% \tag{1}$$

In the above all experiments, the $GUIDE_{LM}$ and $MGUIDE_{LM}$ performed better than the THUI-Mine and MHUI-TID methods. Since, the THUI-Mine and MHUI-TID methods need to store all transactions in the window and require more than 2

Table 5. Quality of Patterns

MinUtil	#HUI	#Actual MaxHUI	#MaxHUI	Precision
0.1	12350	1111	910	81.90
0.2	3435	309	278	89.96
0.3	1574	141	128	90.78
0.4	879	79	72	91.13
0.5	555	49	45	91.83
0.6	376	33	30	90.90
0.7	271	24	23	95.83
0.8	199	17	16	94.11
0.9	149	14	13	92.85

database scans.

6. Conclusions

In this paper, we have proposed the modified version of $GUIDE_{LM}$ to overcome the problem with projections generated by the transaction projection of the $GUIDE_{LM}$ algorithm to improve the quality of patterns with exact utilities. Our extended procedure generates few additional patterns, including the projections that are used effectively to maintain the complete information of patterns in the tree. Our approach provides the quality of patterns as 91.03% as compared to the 83% provided by the $GUIDE_{LM}$ algorithm. We understand that the quality of patterns may be correlated with distribution of items in transactions that will be the future work.

References

1. H. Yao, H. J. Hamilton and C. J. Butz, "A foundational approach to mining itemset utilities from databases," *SIAM Int. conf. on data mining*, 482-486 (2004).
2. R. Agrawal, R. Srikant,"Fast algorithms for mining association rules," *In Proc. Int. Conf. Very Large Data Bases, VLDB*,**1215**, 487-499 (1994).
3. Y. Liu, W.-k. Liao and A. Choudhary,"A two-phase algorithm for fast discovery of high utility itemsets," *In Advances in Knowledge Discovery and Data Mining*, 689-695 (2005).
4. H. Yao, H. J. Hamilton,"Mining itemset utilities from transaction databases," *Data and Knowledge Engineering*, **59**(3), 603-626 (2006).
5. A. Erwin, R. P. Gopalan and N. Achuthan, "CTU-Mine: An efficient high utility itemset mining algorithm using the pattern growth approach," *In Proc. of IEEE Int. Conf. on Computer and Information Technology*, 71-76 (2007).
6. A. Erwin, R. P. Gopalan and N. Achuthan, "Efficient mining of high utility itemsets from large datasets," *Advances in Knowledge Discovery and Data Mining*, 554-561 (2008).
7. Y.-C. Li, J.-S. Yeh and C.-C. Chang, "Isolated items discarding strategy for discovering high utility itemsets," *Data and Knowledge Engineering*, **64**(1), 198-217 (2008).
8. C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong and Y.-K. Lee, "An efficient candidate pruning technique for high utility pattern mining," *Advances in Knowledge Discovery and Data Mining*, 749-756 (2009).
9. V. S. Tseng, C.-W. Wu, B.-E. Shie and P. S. Yu, "UP-Growth: an efficient algorithm for high utility itemset mining," *Proc. of 16th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, 253-262 (2010).
10. C.-W. Lin, T.-P. Hong and W.-H. Lu, "An effective tree structure for mining high utility itemsets," *Expert Systems with Applications*, **38**(6), 7419-7424 (2011).
11. C.-W. Lin, G.-C. Lan and T.-P. Hong, "An incremental mining algorithm for high utility itemsets," *Expert Systems with Applications*, **39**(8), 7173-7180 (2012).
12. M.-Y. Lin, T.-F. Tu and S.-C. Hsueh, "High utility pattern mining using the maximal itemset property and lexicographic tree structures," *Information Sciences*, **215**, 1-14 (2012).
13. M. Liu, J. Qu, "Mining high utility itemsets without candidate generation," *Proc. of ACM Int. Conf. on Information and knowledge management*, 55-64 (2012).
14. V. S. Tseng, C.-J. Chu and T. Liang, "Efficient mining of temporal high utility itemsets from data streams," *Second Int. Workshop on Utility-Based Data Mining*, 18, (2006).
15. H.-F. Li, H.-Y. Huang, Y.-C. Chen, Y.-J. Liu and S.-Y.

Lee, “Fast and memory efficient mining of high utility itemsets in data streams,” *In Proc. of IEEE Int. Con. on Data Mining*, 881-886. (2008).

16. B.-E. Shie, V. S. Tseng and P. S. Yu, “Online mining of temporal maximal utility itemsets from data streams,” *In Proc. of the 2010 ACM Symposium on Applied Computing*, 1622-1626 (2010).
17. C. F. Ahmed, S. K. Tanbeer and B.-S. Jeong, “Efficient mining of high utility patterns over data streams with a sliding window method,” *In Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 99-113 (2010).
18. B.-E. Shie, P. S. Yu and V. S. Tseng, “Efficient algorithms for mining maximal high utility itemsets from data streams with different models,” *Expert Systems with Applications*, **39**(17), 12947-12960 (2012).
19. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, ”Discovering frequent closed itemsets for association rules,” *In Database TheoryICDT99*, 398-416 (1999).
20. Y. C. Li, J. S. Yeh and C. C. Chang, ”A fast algorithm for mining share-frequent itemsets,” *In Web Technologies Research and Development-APWeb*, 417-428 (2005).
21. Y. C. Yeh, L. J. S., and C. C. Chang, ”Efficient algorithm for mining share-frequent itmesets,” *In Proc.of the 11th Int. Fuzzy Systems Association World Congress*, **1**,(2005).
22. A. Erwin, R. P. Gopalan, and N. R. Achuthan, ”A bottom-up projection based algorithm for mining high utility itemsets,” *In Proc. of the 2nd Int. workshop on Integrating artificial intelligence and data mining*, **84**, 3-11 (2007).
23. Y. -C. Li, J. S. Yeh and C. C. Chang, ”Direct candidates generation: a novel algorithm for discovering complete share-frequent itemsets,” *Fuzzy Systems and Knowledge Discovery*, 551-560, (2005).
24. http://www.almaden.ibm.com/software/projects/hdb /resources.shtml.
25. http://fimi.ua.ac.be/data/.