High Utility Rare Itemset Mining over Transaction Databases

Vikram Goyal[1], Siddharth Dawar[1], and Ashish Sureka[2]

[1] Indraprastha Institute of Information Technology-Delhi (IIIT-D), India
{vikram,siddharthd}@iiitd.ac.in
[2] Software Analytics Research Lab (SARL), India
ashish@iiitd.ac.in

Abstract. High-Utility Rare Itemset (HURI) mining finds itemsets from a database which have their utility no less than a given minimum utility threshold and have their support less than a given frequency threshold. Identifying high-utility rare itemsets from a database can help in better business decision making by highlighting the rare itemsets which give high profits so that they can be marketed more to earn good profit. Some two-phase algorithms have been proposed to mine high-utility rare itemsets. The rare itemsets are generated in the first phase and the high-utility rare itemsets are extracted from rare itemsets in the second phase. However, a two-phase solution is inefficient as the number of rare itemsets is enormous as they increase at a very fast rate with the increase in the frequency threshold. In this paper, we propose an algorithm, namely UP-Rare Growth, which uses UP-Tree data structure to find high-utility rare itemsets from a transaction database. Instead of finding the rare itemsets explicitly, our proposed algorithm works on both frequency and utility of itemsets together. We also propose a couple of effective strategies to avoid searching the non-useful branches of the tree. Extensive experiments show that our proposed algorithm outperforms the state-of-the-art algorithms in terms of number of candidates.

Keywords: Data Mining, Pattern Mining, Rare Itemset Mining, Rare Utility Itemset, Utility Mining.

1 Introduction

High-Utility Rare Itemset mining finds those itemsets from the database which are rare as well as of high utility. An itemset is defined as a high utility itemset if its utility value is no less than a given minimum utility threshold. The utility of an itemset is a function of its quantity and the profit value associated with it. An itemset is rare if its support is no greater than a given maximum support threshold. Mining high-utility rare itemsets (HURI) [1] from a database may be interesting for business organizations. For example, identifying HURI in a retail store will help the retail owner to focus on items that should be marketed well to earn more profit. High utility rare itemset mining also finds its use in applications of anomaly detection such as identifying fraudulent credit card transactions, medicine [2], molecular biology [3] and security [4]. A lot of work has been done in the areas of Frequent Itemset Mining (FIM) [5–7] and high-utility itemset mining [8–11] separately. FIM mines those itemsets from the database which are frequent without considering the profit value or quantity value associated with the items. High utility itemset mining removes this limitation, but still does not consider the frequency of itemsets into account. Itemsets which are interesting in utility as well as frequency aspects may not be identified if only utility or frequency objective is considered.

In this paper, we focus on mining high-utility rare itemsets from transaction databases. Jyothi et al. [1] proposed a two-phase algorithm to find high utility rare itemsets from transaction databases. The rare itemsets are mined in the first phase and utility of rare itemsets are computed in the next phase to find high utility rare itemsets. Jyothi et al. [12] proposed an approach similar to their previous work [1] for finding profitable transactions along with high utility rare itemsets from a transaction database. However, the two-phase approach to mine high utility rare itemsets is not efficient as the amount of rare itemsets increase rapidly with the increase in frequency threshold resulting in longer excecution time.

We propose an algorithm called, UP-Rare Growth, which uses a UP-Tree data structure [13] to find high utility rare itemsets. Our proposed algorithm works on both utility and frequency dimensions together and generates candidate high utility rare itemsets in the first phase which are then verified in the second phase. Our approach for high-utility rare itemset mining is efficient because of following reasons:

1. UP-Tree allows for a compressed representation of the database and allow to develop an efficient algorithm which takes both frequency and utility dimensions simultaneously into account,
2. Our approach is a pattern-growth approach which allows for the generation of a significantly lesser number of candidates as compared to a level-wise approaches like Apriori [5].

Our novel research contributions can be summarized as follows:

1. We propose an efficient algorithm, UP-Rare growth, to find high-utility-rare itemsets from a transaction database.
2. We propose effective pruning strategies which help in computation of results faster by pruning the non-promising search space.
3. We conduct extensive experiments on Mushroom dataset to show that our proposed algorithm outperforms state-of-the-art algorithms in terms of the number of candidates.

2 Related Work

Frequent-itemset mining [5–7] has been studied extensively in the literature. Agrawal et al. [5] proposed an algorithm named Apriori, for mining association rules from market-basket data. Their algorithm was based on the downward closure property [5]. The downward closure property states that every subset of a frequent itemset is also frequent. Park et al. [14] proposed a hash based algorithm for mining association rules which generates less number of candidates compared to Apriori algorithm. Zaki et al. [15] proposed an algorithm, namely ECLAT, for mining association rules which used itemset clustering to find the set of potentially maximal frequent itemsets. Han et al. [6] proposed a pattern-growth algorithm to find frequent itemsets by using FP-tree data structure. Other variants of itemset mining problem that have been proposed in the literature are high utility itemset mining, high utility-frequent itemset mining and high-utility rare itemset mining. However, frequent-itemset mining algorithms can't be used to find high utility itemsets as it is not necessarily true that a frequent itemset is also a high utility itemset in the database. On the other hand, mining high-utility patterns is challenging compared to the frequent-itemset mining, as there is no downward closure property [5], like we have in frequent-itemset mining scenario.

Several algorithms have also been proposed to find high utility itemsets. Liu et al.[10] proposed a two-phase algorithm which generates candidate high utility itemsets in the first phase and verification is done in the second phase. Ahmed et al.[16] proposed another two-phase algorithm, which uses a data structure named IHUP-Tree, to mine high utility patterns incrementally from dynamic databases. The problem with the above mentioned algorithms is the generation of a huge amount of candidates in the first phase which leads to longer execution times. In order to reduce the number of candidates, Tseng et al.[13] proposed a new data structure called UP-Tree and algorithms, namely UP-Growth [13] and UP-Growth+ [9]. The authors proposed effective strategies like DGU, DGN, DLU and DLN to compute better utility estimates.

Some work has also been done on high frequency-high utility [17] and high utility-rare itemset [1], [12]. Yeh et al. [17] proposed a bottom-up and top-down two phase algorithms to find frequent high utility itemsets. They introduced the concept of quasi-utility-frequency which is upward closed with respect to the lattice of all itemsets. The top-down algorithm finds quasi-utility-frequency candidates in the first phase, which are verified in the second phase. The problem of finding rare itemsets have been investigated by some authors [18], [19], [20]. Koh et al. [18] proposed an algorithm Apriori-Inverse for discovering sporadic rules by discarding all the itemsets which have their support greater than the maximum frequency threshold. Troiano et al. [20] proposed a top-down algorithm which used power set lattice to find rare itemsets. Pillai et al. [1] proposed an algorithm HURI for finding high utility rare itemsets. Their proposed algorithm used the concept of Aprori-Inverse. Pillai et al. [12] proposed a modified HURI algorithm to find profitable transactions which contained rare itemsets and the share of such items in the overall profit of transactions. However, the above mentioned algorithm generates rare itemsets in the first phase, which are verified in the second phase.

Table 1. *Example Database*

TID	Transaction	TU
T_1	$(C:5)\,(D:20)$	70
T_2	$(C:1)\,(F:40)$	42
T_3	$(A:1)\,(B:1)\,(C:2)\,(G:10)$	20
T_4	$(A:1)\,(B:1)\,(C:2)$	10
T_5	$(A:5)\,(C:10)$	45
T_6	$(B:1)\,(C:1)\,(E:1)$	5
T_7	$(B:1)\,(C:1)\,(E:1)\,(G:10)$	15
T_8	$(B:1)\,(C:1)\,(E:1)\,(H:1)$	6
T_9	$(C:10)\,(E:10)$	40
T_{10}	$(A:1)\,(B:1)\,(C:1)$	8

Table 2. *Profit Table*

Item	A	B	C	D	E	F	G	H
Profit	5	1	2	3	2	1	1	1

3 Background

In this section, we present some definitions given in the earlier works and describe the problem statement formally. We also discuss the UP-Tree data structure briefly.

3.1 Preliminary

We have a set of m distinct items $I = \{i_1, i_2, ..., i_m\}$, where each item has a profit $pr(i_p)$ $(external\,utility)$ associated with it. An itemset X of length k is a set of k items $X = \{i_1, i_2, ..., i_k\}$, where for $j \in 1.....k$, $i_j \in I$. A transaction database $D = \{T_1, T_2, ..., T_n\}$ consists of a set of n transactions, where every transaction has a subset of items belonging to I. Every item I_p in a transaction T_d has a quantity $q(i_p, T_d)$ associated with it.

Definition 1. *The utility of an item I_p in a transaction T_d is the product of the profit of the item and its quantity in the transaction i.e. $u(i_p, T_d) = q(i_p, T_d) * pr(i_p)$.*

Definition 2. *The utility of an itemset X in a transaction T_d is denoted as $u(X, T_d)$ and defined as $\sum_{X \subseteq T_d \wedge i_p \in X} u(i_p, T_d)$.*

Definition 3. *The utility of a transaction T_d is denoted as $TU(T_d)$ and defined as $\sum_{i_p \in T_d} u(i_p, T_d)$.*

Let us consider the example database shown in Table 1 and the profit values in Table 2. The utility of item $\{A\}$ in $T_3 = 1 \times 5 = 5$ and the utility of itemset $\{A, B\}$ in T_3 denoted by $u(\{A, B\}, T_3) = u(A, T_3) + u(B, T_3) = 5 + 1 = 6$.

Definition 4. *The utility of an itemset X in database D is denoted as $u(X)$ and defined as $\sum_{X \subseteq T_d \wedge T_d \in D} u(X, T_d)$.*

For example, $u(A, B) = u(\{A, B\}, T_3) + u(\{A, B\}, T_4) + u(\{A, B\}, T_{10}) = 7 + 7 + 7 = 21$.

Definition 5. *An itemset is called a high utility itemset if its utility is no less than a user-specified minimum threshold denoted by min_util.*

For example, $u(A, C) = u(\{A, C\}, T_3) + u(\{A, C\}, T_4) + u(\{A, C\}, T_5) + u(\{A, C\}, T_{10}) = 9 + 9 + 45 + 7 = 70$. If *min_util* $= 30$, then $\{A, C\}$ is a high utility itemset. However, if *min_util* $= 75$, then $\{A, C\}$ is a low utility itemset.

Table 3. *Rare Itemset Table*

Itemsets	**List of rare itemsets**
1-itemset	{ D }, { F }, { H }
2-itemset	{ AG }, { BH }, { CD }, { CF }, { CH }, { EG }, { EH }
3-itemset	{ ABG }, { ACG }, { BCH },{ BEH }, { BEG }, { CEG }, { CEH }
4-itemset	{ ABCG }, { BCEG }, { BCEH }

Definition 6. *The support of an itemset X denoted by sup(X) is the number of transactions in database D which contain itemset X.*

For example, the $sup(\{A, C\}) = 4$.

Definition 7. *An itemset X is called a rare itemset, if* $sup(X) < max\_sup\_threshold$.

Let *max_sup_threshold* $= 2$. The rare itemsets are shown in Table 3.

Table 4. *Rare High Utility Itemset Table*

Itemsets	**List of high utility rare itemsets**
1-itemset	$\{D\}$, { F }
2-itemset	{ CD }, { CF }
3-itemset	$\{\emptyset\}$
4-itemset	$\{\emptyset\}$

Problem Statement. Given a transaction database D, a minimum utility threshold *min_util* and maximum support threshold *max_sup_threshold* , the aim is to find all the itemsets which are rare as well as of high utility, i.e. itemsets which have utility no less than *min_util* and support value less than *max_sup_threshold*.

Table 5. *MIU Table*

Item	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
MIU	5	2	2	60	2	40	10	1

Let $min\_util = 30$ and $max\_sup\_threshold = 2$. The set of high utility rare itemsets are shown in Table 4.

We will now describe the concept of transaction utility and transaction weighted downward closure(TWDC)[8].

Definition 8. *The transaction utility of a transaction T_d is denoted by $TU(T_d)$ and defined as $u(T_d, T_d)$.*

For example, the transaction utility of every transaction is shown in Table 1.

Definition 9. *Transaction-weighted utility of an itemset X is the sum of the transaction utilities of all the transactions containing X, which is denoted as $TWU(X)$ and defined as $\sum_{X \subseteq T_d \wedge T_d \in D} TU(T_d)$.*

Definition 10. *An itemset X is called a high-transaction-weighted utility itemset (HTWUI), if $TWU(X)$ is no less than min_util.*

Property 1 (Transaction-weighted downward closure). For any itemset X, if X is not a (HTWUI), any superset of X is not a HTWUI.
For example, $TU(T_1) = u(\{CD\}, T_1) = 70; TWU(\{A\}) = TU(T_3) + TU(T_4) + TU(T_5) + TU(T_{10}) = 83$. If $min\_util = 80$, $\{A\}$ is a HTWUI. However, if $min\_util = 100$, $\{A\}$ and any of its supersets are not HTWUIs.

We will now describe the concepts [9] for computing ovestimated utility of an item.

Definition 11. *Minimum item utility of item i_p in database D, denoted as $miu(i_p)$ is i_p's utility in transaction T_d if there does not exist a transaction T_d' such that $u(i_p, T_d') < u(i_p, T_d)$.*

The minimum item utility of the items in database D is shown in the Table 5

Definition 12. *Assume that N_x is the node which records the item x in the path p in a UP-Tree and N_x is composed of items x from the set of transactions $TIDSET(T_X)$. The minimum node utility of x in p is denoted as $mnu(x, p)$ and defined as $min_{\forall T \in TIDSET(T_X)}(u(x, T))$.*

3.2 UP-Tree

Each node N in UP-Tree [13] consists of a name *N.item*, overestimated utility *N.nu*, support count *N.count*, a pointer to the parent node *N.parent* and a pointer *N.hlink* to the node which has the same name as *N.name*. The root of

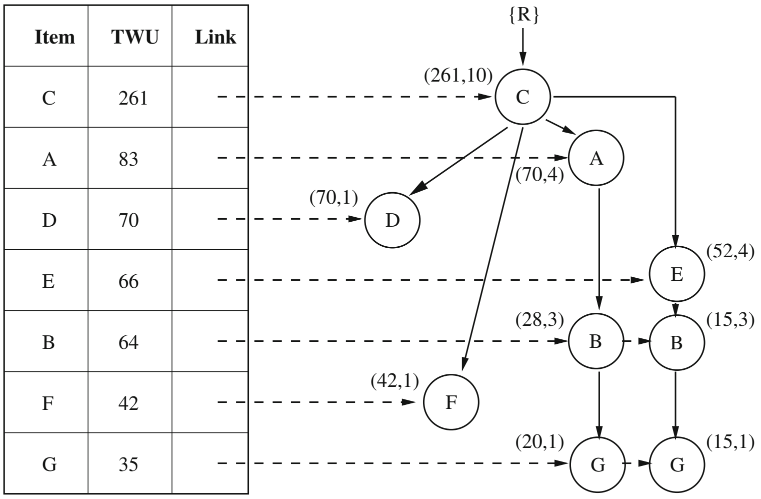

Fig. 1. Global UP-Tree

the tree is a special empty node which points to its child nodes. The support count of a node N along a path is the number of transactions contained in that path that have the item $N.item$. $N.nu$ is the overestimated utility of an itemset along the path from node N to the root. In order to facilitate efficient traversal, a header table is also maintained. The header table has three columns, *Item*, TWU and *Link*. The nodes in a UP-Tree along a path are maintained in descending order of their TWU values. All nodes with the same label are stored in a linked list and the link pointer in the header table points to the head of the list.

4 Mining High Utility Rare Itemsets

In this section, we will describe our algorithm UP-Rare Growth for mining high utility rare itemsets. We will illustrate the working of our algorithm with an example and will formally prove its correctness.

4.1 Construction of a Global UP-Tree

In this subsection, we will discuss how to construct the global UP-Tree from the database D. The global UP-Tree is constructed in two scans of the database. In the first scan, the TWU value of every item is computed. The unpromising items are removed from the transaction database and transactions are reorganized in decreasing order of their TWU values. Unpromising items are the items which have their TWU value less than the minimum utility threshold. The removal of unpromising items from the database is called Discarding Global Unpromising items (DGU).

For example, consider the example database as shown in the Table 1 and the profit value associated with each item in Table 2. Let the minimum utility threshold be 30 and maximum frequency threshold be 2. Item $\{H\}$ is an unpromising item as TWU($\{H\}$) is 6 which is less than the minimum utility threshold. The reorganized transactions are shown in Table 6.

Each transaction is now processed and inserted to form the global UP-Tree as shown in the Figure 1. Let us consider the insertion of the reorganized transaction T_1' in the global tree. The global UP-Tree is initially empty. The item $\{C\}$ is processed and a new node N_C is created with $N_C.item = C$ and $N_C.count = 1$. The utility value of each node in the UP-Tree is reduced further by applying Discarding Global Node Utilities (DGN) strategy. DGN strategy is that the node utility of a node in the global tree can be reduced further by removing the node utilities of the descendant nodes. The rationale behind removing the node utilities of descendant nodes from the utility of a node N is that the local tree of N will not include its descendants as the transactions are ordered according to TWU values. The utility of node N_C is computed by subtracting the utilities of its descendants, i.e. the items after C in the reorganized transaction. In our example, $N_C.nu$=$RTU(T_1') - u(\{D\}, T_1')$=70-60=10. Next, item $\{D\}$ is processed. A new node N_D is created with $N_D.item = D$ and $N_D.count = 1$. Since, there is no item after $\{D\}$ in the transaction T_1', its utility is equal to $RTU(T_1')$ i.e. 70. If the node of the item to be inserted in the tree is already present along that path, the support count and node utilities are simply incremented. Similarly, other reorganized transactions are inserted to construct the global UP-Tree. The strategies DLU and DLN are similar to DGU and DGN, but are applied to the local UP-Tree. Since, exact utilities are not stored in the global UP-Tree, utilities of unpromising items are estimated using the minimum item utility and minimum node utility as per Definition 11 and 12.

Table 6. *ReorganizedTransactions*

TID	**Reorganized Transaction**	**RTU**
T_1'	$(C:5)\,(D:20)$	70
T_2'	$(C:1)\,(F:40)$	42
T_3'	$(C:2)\,(A:1)\,(B:1)\,(G:10)$	20
T_4'	$(C:2)\,(A:1)\,(B:1)$	10
T_5'	$(C:10)\,(A:5)$	45
T_6'	$(C:1)\,(E:1)\,(B:1)$	5
T_7'	$(C:1)\,(E:1)\,(B:1)\,(G:10)$	15
T_8'	$(C:1)\,(E:1)\,(B:1)$	5
T_9'	$(C:10)\,(E:10)$	40
T_{10}'	$(C:1)\,(A:1)\,(B:1)$	8

4.2 UP-Rare Growth

The algorithm UP-Rare Growth takes as input a UP-Tree, a header table, an itemset, a minimum utility threshold, a maximum support threshold and returns the candidate rare high utility itemsets. The steps of the algorithm are shown in Algorithm 1. The algorithm starts with an empty prefix and extends the prefix with item i_k of the header table. In this process of extension (growth), a conditional pattern base(CPB) is constructed from the prefix. The CPB of the prefix extended with item i_k consists of all the paths through which i_k is reachable from the root of the tree. A local UP-Tree is constructed and strategies DLU and DLN are applied. We apply the following strategies while processing for a prefix X just extended with item i_k to prune the search space:

1. If X has low TWU i.e. estimated utility of any itemset containing X is less than the minimum utility threshold, X prefix is not processed further and the algorithm proceeds with the next alternative of the header table. Else, X is processed further.
2. If the support count of every leaf node of the UP-Tree is greater than the given support threshold, it is guaranteed that no rare itemset can be found using item i_k as a prefix. In this case, prefix i_k is not processed further.

Algorithm 1. UP-Rare Growth(T_x, H_x, X)

Input: A UP-Hist tree T_x, a header table H_x for T_x, an itemset X, a minimum utility threshold min_util and maximum support threshold $max\_sup_thresh old$.
Output: All candidate rare High Utility Itemsets in T_x.

```
1: for entry $i\_k$ in $H\_x$ do
2:     Traverse the linked list associated with $i\_k$ and accumlate sum of node utilities $nu\_sum(i\_k)$.
3:     if $nu\_sum(X) \geq min\_util$ then
4:         if ( then sup($i\_k$) $< max\_sup\_threshold$))
5:             Consider $Y = X \cup i\_k$ as a candidate and construct CPB of $Y$.
6:         else
7:             Construct the CPB of $Y$.
8:         end if
9:         Put local promising items in $Y - CPB$ into $H\_Y$ and apply DLU to reduce path utilities.
10:        Insert every reorganized path into $T\_Y$ after applying DLN.
11:        if $T_Y \neq null$ then
12:            if support of every leaf node of $T\_Y > max\_sup\_threshold$ then
13:                continue
14:            else
15:                Call UP-Rare Growth($T\_Y$,$H\_Y$,$Y$)
16:            end if
17:        end if
18:    end if
19: end for
```

Now, we will illustrate the working of our algorithm with an example. Consider the global UP-Tree shown in the Figure 1. Let the minimum utility threshold be 30 and maximum frequency support be 2. The algorithm picks the lowest entry from the header table i.e. G and accumulates its node utility. Since the accumulated node utility $nu_{sum}(G)$ i.e. 35 is greater than the maximum utility threshold, item $\{G\}$ will be processed further. However, $\{G\}$ is not added to the set of candidate itemsets as the sup(G) is not less than the maximum frequency threshold. The conditional pattern base of $\{G\}$ is constructed as shown in the Table 7. The TWU values of the items in the CPB of $\{G\}$ are computed and the

Table 7. $\{G\} - CPB\, after\, applying\, DGU\; DGN\; and\, DLN$

Retrieved Path: Path utility	**Reorganized Path: Path utility (after DLU)**	**Support**
$< CAB >$: 20	$< CB >$: 15	1
$< CEB >$: 15	$< CB >$: 13	1

unpromising items are removed to get the reorganized paths. In the CPB of $\{G\}$, Item $\{A\}$ and $\{E\}$ are unpromising as $TWU(\{A\}) = 20$ and $TWU(\{E\}) = 15$ is less than the minimum utility threshold. The reorganized path utilities are computed using minimum node utilities similar to UP-Growth+ i.e.,
$pu(< CAB >, \{G\} - CPB) = 20$ - $A.mnu \times < CAB > .support$=20-5×1=15.
$pu(< CEB >, \{G\} - CPB) = 15$ - $E.mnu \times < CEB > .support$=15-2×1=13.
The algorithm is called recusively for the itemset $\{GC\}$, $\{GB\}$ and $\{GCB\}$. However, all the itemsets which have G as a prefix are low utility itemsets as their TWU value is less than the minimum utility threshold. Similarly, the next item $\{F\}$ is processed from the header table.

Table 8. $\{B\} - CPB\, after\, applying\, DGU\; DGN\; and\, DLN$

Retrieved Path: Path utility	**Reorganized Path: Path utility (after DLU)**	**Support**
$< CA >$: 28	$< CA >$: 28	3
$< CE >$: 15	$< CE >$: 15	3

We will now focus on the processing of item $\{B\}$ in the global header table. The linked list associated with B is traversed from the header table and the sum of node utilities is accumulated. Since the TWU of $\{B\}$ is greater than the minimum utility threshold, it is processed further. However, $\{B\}$ is not added to the candidates as sup(B) is 6 which is greater than the maximum frequency threshold. The conditional pattern base of $\{B\}$ is constructed as shown in the Table 8. There are no unpromising items in the CPB of $\{B\}$ and a local UP-Tree is constructed as shown in the Figure 2. However, there is no rare itemset containing $\{B\}$ as the support of every leaf node in the local UP-Tree is greater than the maximum frequency threshold. After processing the remaining items in the global header table, the complete set of candidate high utility rare itemsets are generated. The candidate itemsets obtained from UP-Rare Growth are $\{F\}$, $\{CF\}$, $\{D\}$ and $\{CD\}$. The generated candidates are verified by scanning the original database again and computing the exact utilities of the candidate itemsets.

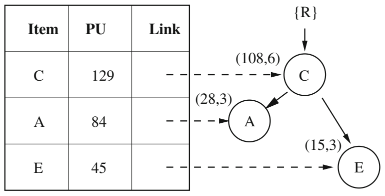

Fig. 2. Local UP-Tree

Claim 1. *Algorithm UP-Rare Growth does not generate any false negatives.*

Proof. The algorithm UP-Rare Growth prunes an itemset and its supersets on the basis of two rules: (1) If the overestimated utility of an itemset is less than the minimum utility threshold, (2) the support count of all leaf nodes in the tree constructed from the CPB of the itemset, is greater than the maximum support threshold. We know that the TWU value associated with every itemset is an upper bound on the exact utility value and satisfies the downward closure property. Therefore, if an itemset is marked off as low utility, it is guaranteed that this itemset and its supersets will be of low utility. So, rule 1 will not generate any false negatives. The algorithm also prunes the supersets of an itemset if all the leaf nodes have their support count greater than the maximum frequency threshold. Since the transactions are reorganized in the decreasing order of TWU values, the leaf nodes have the least support value compared to its ancestors in the tree. Therefore, if the support of all the leaf nodes of the UP-Tree constructed from the conditional pattern base of an itemset is greater than the maximum support threshold, it is guaranteed that there can't be any superset of that itemset which is rare. This proves the claim. □

5 Experiments and Results

In this section, we compare the performance of our proposed algorithm UP-Rare Growth against the state-of-the-art algorithm HURI [1]. We implemented all the algorithms in Java on Eclipse 3.5.2 platform with JDK 1.6.0_24. The experiments were performed on an Intel Xeon(R) CPU=26500@2.00 GHz with 64 GB RAM. We ran our experiments on the Mushroom dataset which was obtained from FIMI repository [21]. The quantity and external utility for the Mushroom datasets was generated using log-normal distribution.

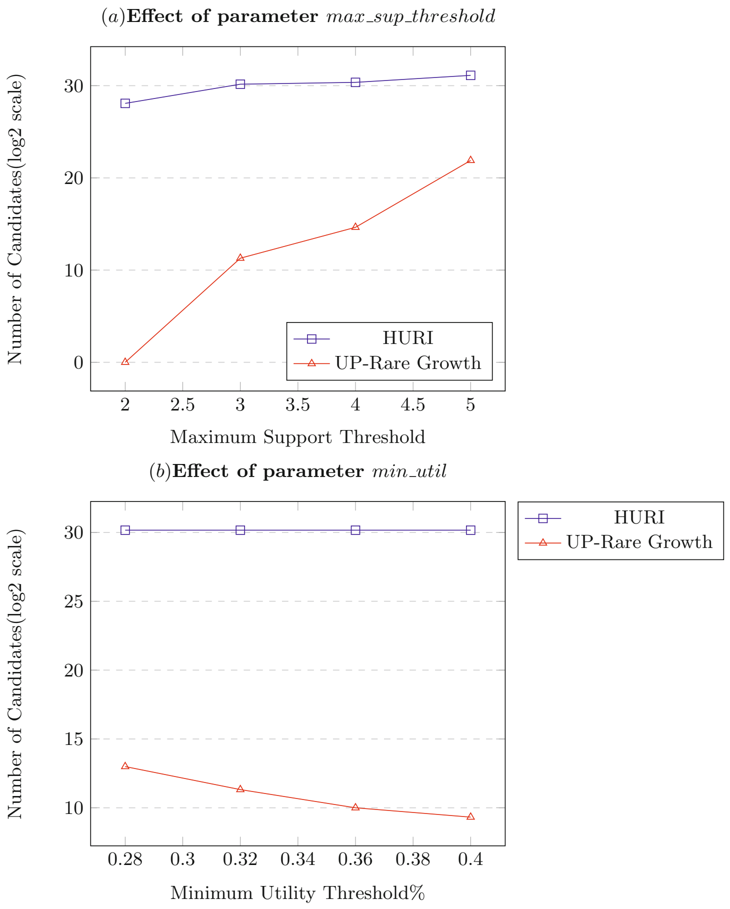

Fig. 3. Performance Evaluation on Mushroom dataset

We studied the impact of the parameters, maximum support threshold and minimum utility threshold on the performance of the algorithms. We observed that there were no high-utility itemsets in our Mushroom database for the utility threshold 2,50,00,000. Therefore, we set this value as the maximum utility threshold and represent the different values of *min_util* as percent with respect to this value. We compare the performance of the algorithms in terms of the number of candidates generated after the first phase. The number of candidates is represented on a log scale with base 2. In order to study the effect of the maximum support threshold, we fixed $min\_util = 0.32\%$ and the results are shown in Figure 3(a).

We expect the number of rare itemsets to increase exponentially with varying support threshold and the results meet our expectation. The results show that our algorithm generates a significantly lesser number of candidates compared to HURI and the verification time taken by HURI will be very large as the verification time depends upon the number of candidates. In order to study the effect of minimum utility threshold, we fixed the maximum support threshold to 3 and the results are shown in Figure 3(b). The number of candidates generated by HURI remains constant with the varying minimum utility threshold as the algorithm doesn't take the utility dimension into account while computing the number of candidates. We also observe that the number of candidates generated by our algorithm decrease with an increase in the minimum utility threshold. The results clearly demonstrate the importance of taking the utility dimension into account when computing the candidate high-utility rare itemsets.

6 Conclusion and Future Work

In this paper, we proposed a novel algorithm UP-Rare Growth, for mining high-utility rare itemsets. Our algorithm considers both utility and frequency dimensions simultaneously and uses effective strategies to reduce the search space. Experimental results show that our proposed algorithm outperforms the state-of-the-art algorithm in terms of number of candidates.

References

1. Pillai, J., Vyas, O.P., Muyeba, M.: Huri–a novel algorithm for mining high utility rare itemsets. In: Advances in Computing and Information Technology, pp. 531–540. Springer, Heidelberg (2013)
2. Medici, F., Hawa, M.I., Giorgini, A.N.G.E.L.A., Panelo, A.R.A.C.E.L.I., Solfelix, C.M., Leslie, R.D., Pozzilli, P.: Antibodies to gad65 and a tyrosine phosphatase-like molecule ia-2ic in filipino type 1 diabetic patients. Diabetes Care 22(9), 1458–1461 (1999)
3. Shi, W., Ngok, F.K., Zusman, D.R.: Cell density regulates cellular reversal frequency in myxococcus xanthus. Proceedings of the National Academy of Sciences 93(9), 4142–4146 (1996)
4. Saha, B., Lazarescu, M., Venkatesh, S.: Infrequent item mining in multiple data streams. In: Seventh IEEE International Conference on Data Mining Workshops, ICDM Workshops 2007, pp. 569–574 (2007)

5. Agrawal, R., Ramakrishnan, Srikant, o.: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB, vol. 1215, pp. 487–499 (1994)
6. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD, vol. 29, pp. 1–12. ACM (2000)
7. Leung, C.K.-S., Khan, Q.I., Li, Z., Hoque, T.: Cantree: a canonical-order tree for incremental frequent-pattern mining. Knowledge and Information Systems 11(3), 287–311 (2007)
8. Liu, Y., Liao, W.-k., Choudhary, A.: A fast high utility itemsets mining algorithm. In: International Workshop on Utility-Based Data Mining, pp. 90–99. ACM (2005)
9. Tseng, V.S., Shie, B.-E., Wu, C.-W., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Transactions on Knowledge and Data Engineering 25(8), 1772–1786 (2013)
10. 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)
11. Shie, B.-E., Tseng, V.S., Yu, P.S.: Online mining of temporal maximal utility itemsets from data streams. In: ACM Symposium on Applied Computing, pp. 1622–1626. ACM (2010)
12. Pillai, J., Vyas, O.P.: Transaction profitability using huri algorithm [tphuri]. International Journal of Business Information Systems 2(1) (2013)
13. Tseng, V.S., Wu, C.-W., Shie, B.-E., Yu, P.S.: Up-growth: an efficient algorithm for high utility itemset mining. In: ACM SIGKDD, pp. 253–262. ACM (2010)
14. Park, J.S., Chen, M.-S., Yu, P.S.: An effective hash-based algorithm for mining association rules, vol. 24. ACM (1995)
15. Zaki, M.J., Parthasarathy, S., Ogihara, M., Wei, Li, o.: New algorithms for fast discovery of association rules. In: KDD, vol. 97, pp. 283–286 (1997)
16. 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)
17. Yeh, J.-S., Li, Y.-C., Chang, C.-C.: Two-Phase Algorithms for a Novel Utility-Frequent Mining Model. In: Washio, T., Zhou, Z.-H., Huang, J.Z., Hu, X., Li, J., Xie, C., He, J., Zou, D., Li, K.-C., Freire, M.M. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4819, pp. 433–444. Springer, Heidelberg (2007)
18. Koh, Y.S., Rountree, N.: Finding sporadic rules using apriori-inverse. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 97–106. Springer, Heidelberg (2005)
19. Szathmary, L., Napoli, A., Valtchev, P.: Towards rare itemset mining. In: 19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2007, vol. 1, pp. 305–312. IEEE (2007)
20. Troiano, L., Scibelli, G., Birtolo, C.: A fast algorithm for mining rare itemsets. ISDA 9, 1149–1155 (2009)
21. Goethals, B., Zaki, M.J.: The fimi repository (2012)