Table 1: Transactional database

tid	*items*
1	a, b, c
2	a, b, d
3	c, d, f
4	c, g
5	a, e, f
6	c, d
7	e, f, g
8	e, f
9	c, d, e
10	a, b

Table 2: Internal utility database

tid	*a*	*b*	*c*	*d*	*e*	*f*	*g*
1	2	2	1	0	0	0	0
2	1	1	0	1	0	0	0
3	0	0	3	3	0	1	0
4	0	0	1	0	0	0	1
5	1	0	0	0	3	3	0
6	0	0	2	2	0	0	0
7	0	0	0	0	3	2	1
8	0	0	0	0	4	3	0
9	0	0	2	2	1	0	0
10	1	1	0	0	0	0	0

Table 3: External utility database

item	price
a	200
b	50
c	25
d	50
e	15
f	20
g	75

1 INTRODUCTION

This paper contains the detailed explanation of the RHUI-Miner algorithm. Due to limitations on the page size the running of the algorithm, computational complexity were not mentioned in the paper therfore we try to discuss these points in detail here. Table 1 is the transactional database which we will be using to demonstrate the working of the algorithm. Table 2 and 3 are the internal utility database and external utility database.

2 PROPOSED ALGORITHM

2.1 Basic idea

The RHUIs generated by the proposed model do not satisfy the downward closure property (or apriori property). This increases the search space, which in turn increases the computational cost of finding the desired itemsets. Given this complex scenario, RHUI-Miner tries to discover RHUIs effectively by employing the following steps:

(1) Identify all items whose supersets may yield RHUIs. We call these items as **secondary items.**
(2) Construct set-enumeration tree constituting of secondary items.
(3) Recursively mine the entire set-enumeration tree by classifying each itemset as a *primary itemset, secondary itemset,* or *uninteresting itemset.* We find all RHUIs by performing a depth-first search only on the nodes of *primary itemsets.* We maintain the nodes of *secondary itemsets* for completeness purposes. Please note that we do not perform a depth-first search on these itemsets.

Overall, the above depth-first search of RHUI-Miner is efficient because it drastically reduces the search space while finding the desired itemsets in a database. We now introduce the terms of *secondary items, primary itemset, secondary itemset, and uninteresting itemset.*

2.1.1 *Step 1: Identifying secondary items in the database.*

Liu described *transaction weighted utility* (TWU) to prune all uninteresting items whose supersets may not yield HUIs. We employ the same measure to prune all uninteresting items whose supersets may not generate any RHUI. The remaining items are known as *secondary items.* The definition of secondary item is based on Definitions 1 and 2 and defined in Definition 3.

Definition 1. (**Transaction utility of a transaction**.) The *transactional utility* of a transaction T_c, denoted as $TU(T_c)$, represents the total utility of all items in T_c. That is, $TU(T_c) = \sum_{\forall i_j \in T_c} u(i_j, T_c)$.

Example 1. Consider the first transaction in Table 1. The $utility$ values of a, b and c in T_1, i.e., $u(a, T_1) = 2{\times}200 = 400$¥, $u(b, T_1) = 2{\times}50 = 100$¥ and $u(c, T_1) = 1 \times 25 = 25$¥. The *transaction utility* of T_1 (or total *revenue* generated by the first transaction), i.e., $TU(T_1) = u(a, T_1) + u(b, T_1) + u(c, T_1) = 400 + 100 + 25 = 525$¥.

Definition 2. (**Transaction weighted utility of an item**.) Let $TDB^{i_j} \subseteq TDB$ denote the set of all transactions containing i_j in TDB. The *transaction weighted utility of an item* $i_j \in I$, denoted as $TWU(i_j)$, represents the total utilities of transactions containing i_j in TDB. That is, $TWU(i_j) = \sum_{\forall T_k \in TDB^{i_j}} TU(T_k)$.

Example 2. In Table 1, the item g appears in the transactions whose $tids$ are 4 and 7. Therefore, $TDB^g = \{T_4, T_7\}$. The TWU of g in Table 1, i.e., $TWU(g) = \sum_{T_k \in TDB^g} TU(T_k) = TU(T_4) + TU(T_7) = 100 + 160 = 260$¥. The TWU of g provides the information that g and all of its supersets can at most have the $utility$ of 260¥. If $minUtil = 300$¥, then g can be pruned as neither g nor its supersets can be RHUIs.

The TWU measure, as defined above, is a *utility* upper bound measure that determines the maximum *utility* a superset of an item may have in the entire database (see Property 1). Thus, we prune the items having a TWU value less than $minUtil$.

Definition 3. (Secondary item.) An item $i_j \in I$ is said to be a secondary item if $TWU(i_j) \geq minUtil$.

Example 3. The TWU values of the items a, b, c, d, e, f and g in Table 1 are 1380¥, 1075¥, 1185¥, 860¥, 750¥, 830¥ and 260¥, respectively. As the items a, b, c, d, e and f have TWU values that are no less than the user-specified $minUtil$ value, they are considered as secondary items. In contrast, g will not be considered as a secondary item because $TWU(g) \ngeq minUtil$.

2.1.2 *Step 2: Construction of set-enumeration tree.*

Let SI denote the sorted list of secondary items in TWU ascending order. This sorted list of secondary items will result in the set-enumeration tree. Additionally, sort the transactional database by considering only secondary items in TWU ascending order.

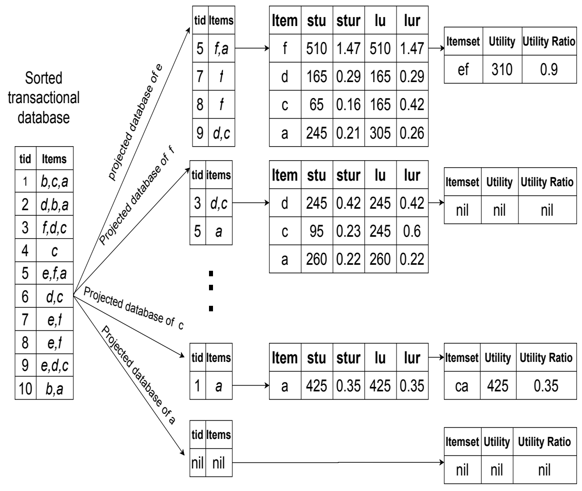
Example 4. Continuing with the previous example, the TWU ascending order of secondary items in Table 1, i.e., $SI = \{efdbca\}$. The depth-first search on the itemset lattice of secondary items will result in a set-enumeration tree. (As the set-enumeration tree generated by these secondary items is too big (or contains $2^6 - 1$ itemsets), we are not showing it in our paper.) The table on the left side of Figure 1 shows the sorted transactional database generated from Table 1. Observe that we pruned the uninteresting item g in this sorted database.

2.1.3 *Step 3: Recursively mining the set-enumeration tree by identifying primary, secondary, and uninteresting itemsets.*

In the set-enumeration tree, we will perform depth-first search if an itemset in a node represents a *primary itemset.* The definition of primary itemset is based on Definitions 4, 5, 6, 7, 8 and defined in Definition 9

Definition 4. (Items that can extend an itemset.) Let SI denote the set of secondary items. Let $\succ$ denote an order of secondary items in TWU ascending order. Let α be a (suffix) itemset. Let $E(\alpha)$ denote the set of all secondary items that can be used to extend α according to the depth-first search, i.e., $E(\alpha) = \{z | z \in SI \wedge z \succ x, \forall x \in \alpha\}$.

Example 5. If $\alpha = e$, then the set of items that can extend e in SI, i.e., $E(e) = \{fdbca\}$. Similarly, if $\alpha = eb$, then $E(eb) = \{ca\}$.



tid	Items
1	*b,c,a*
2	*d,b,a*
3	*f,d,c*
4	*c*
5	*e,f,a*
6	*d,c*
7	*e,f*
8	*e,f*
9	*e,d,c*
10	*b,a*

tid	Items
5	*f,a*
7	*f*
8	*f*
9	*d,c*

Item	stu	stur	lu	lur
f	510	1.47	510	1.47
d	165	0.29	165	0.29
c	65	0.16	165	0.42
a	245	0.21	305	0.26

Itemset	Utility	Utility Ratio
ef	310	0.9

tid	items
3	*d,c*
5	*a*

Item	stu	stur	lu	lur
d	245	0.42	245	0.42
c	95	0.23	245	0.6
a	260	0.22	260	0.22

Itemset	Utility	Utility Ratio
nil	nil	nil

tid	Items
1	*a*

Item	stu	stur	lu	lur
a	425	0.35	425	0.35

Itemset	Utility	Utility Ratio
ca	425	0.35

tid	Items
nil	nil

Itemset	Utility	Utility Ratio
nil	nil	nil

Figure 1: Mining RHUIs using RHUI-Miner

Definition 5. (Projected transaction.) Let $TDB^{SI} \subseteq TDB$ be the sorted transactional database constituting of only secondary items. The projection of a transaction $T_{tid} \in TDB^{SI}$ using an itemset α is denoted as PT^{α}_{tid} and defined as $PT^{\alpha}_{tid} = \cup_{\forall i \in T_{tid} \wedge i \in E(\alpha)} i$.

Example 6. The table on the left side of Figure 1 represents the sorted transactional database generated from Table 1. In order words, the table on left side of Figure 1 represents TDB^{SI}. Let the (suffix) itemset $\alpha = e$. The projected transaction of e in T_5, i.e., $PT^{e}_{5} = \{5 : fa\}$.

Definition 6. (Projected transactional database.) The projection of entire database using an itemset α is denoted as $PTDB^{\alpha}$ and defined as the multiset, i.e., $PTDB^{\alpha} = \{PT^{\alpha}_{a}, PT^{\alpha}_{b}, \cdots, PT^{\alpha}_{c}\} \subseteq TDB^{SI}$, where $1 \leq a \leq b \leq c \leq n$.

Example 7. In the sorted transactional database, the item e appears in the transactions whose *tids* are 5, 7, 8 and 9. Therefore, the projected database of e, denoted as $PTDB^{e} = \{\{5 : fa\}, \{7 : f\}, \{8 : f\}, \{9 : dc\}\}$. The same can be observed in Figure 1.

Definition 7. (Sub-tree utility.) Let α be an itemset and $z \in E(\alpha)$ be an item that can extend α according to the depth-first search in the itemset lattice. The sub-tree utility of z with respect to α, denoted as $stu(\alpha, z) =$

$$\sum_{\forall T_c \in PTDB^{\alpha \cup z}} \left[u(\alpha, T_c) + u(z, T_c) + \sum_{i \in T_c \wedge i \in E(\alpha \cup z)} u(i, T_c) \right].$$

Example 8. In the projected database of e, the item f appears in the transactions whose *tids* are 5, 7 and 8. Therefore, the sub-tree utility of $\alpha \cup f$, denoted as $stu(\alpha, f) = (u(e, T_5) + u(f, T_5) + u(a, T_5)) + (u(e, T_7) + u(f, T_7)) + (u(e, T_8) + u(f, T_8)) = (45 + 60 + 200) + (45 + 40) + (60 + 60) = 305 + 85 + 120 = 510$¥.

Definition 8. (Sub-tree utility ratio.) The sub-tree utility ratio of z with respect to α, denoted as $stur(\alpha, z) = \frac{stu(\alpha,z)}{\sum_{i_j \in \{\alpha \cup z\}} u(i_j)}$.

Example 9. The sub-tree utility ratio of ef, i.e., $stur(e, f) = \frac{stu(e,f)}{(u(e)+u(f))} = 1.47$.

The *sub-tree utility* and *sub-tree utility ratio* are a tighter *utility* upper-bound constraints that can determine whether depth-first search on an itemset results any RHUI. Thus, the itemsets satisfying both of these constraints are known as primary itemsets.

Definition 9. (Primary itemset.) A itemset $X = \{\alpha \cup z\}$ is said to be a primary itemset if $stu(\alpha, z) \geq minUtil$ and $stur(\alpha, z) \geq minUR$.

Example 10. The itemset ef is a primary itemset because $stu(e, f) \geq minUtil$ and $stur(e, f) \geq minUR$. We perform depth-first search on this itemset to find ef as a RHUI (see Figure 1). The next item in the projected database of e is d. For this item, $stu(e, d) \ngeq minUtil$ and $stur(e, d) \ngeq minUR$ (see in Figure 1). Thus, we do not perform depth-first search on the itemset ed. Similarly, we will not perform depth-first search on the items c and a that exist in the projected database of e.

Since RHUIs do not satisfy the anti-monotonic property, we cannot simply prune all itemsets whose *stu* and *stur* values are less than *minUtil* and *minUR*, respectively. It is because there supersets may generate RHUIs. In this context, we further classify the non-primary itemsets into secondary itemsets and uninteresting itemsets, and prune all uninteresting itemsets to reduce the search space effectively. The definition of secondary itemsets is based on Definitions 10 and 11 and defined in Definition 12. Any itemset that is not a secondary itemset is an uninteresting itemset. Henceforth, we are not particularly defining this itemset.

Definition 10. (Local utility of an itemset in a database) Let α be an itemset and $z \in E(\alpha)$ be an item that can extend α according to the depth-first search in the itemset lattice. The Local utility of z with respect to α, denoted as $lu(\alpha, z) = \sum_{\forall T_c \in PTDB^{\alpha \cup z}} [u(\alpha, T_c) + \sum_{i \in T_c \wedge i \in E(\alpha)} u(i, T_c)]$, where $PTDB^{\alpha \cup z}$ represent the projected database of $\alpha \cup z$.

Example 11. The projected database of e contains the items f, d, c and a in TWU ascending order. In this database, the item f appears in the transactions whose *tids* are 5, 7 and 8. The *local utility* of $\alpha \cup f$, denoted as $lu(\alpha, f) = (u(e, T_5) + u(f, T_5) + u(a, T_5)) + (u(e, T_7) + u(f, T_7)) + (u(e, T_8) + u(f, T_8)) = (45 + 60 + 200) + (45 + 40) + (60 + 60) = 305 + 85 + 120 = 510$¥.

Definition 11. (local utility ratio of an itemset in a database) The *local utility ratio* of an itemset $\alpha \cup z$ in a database, denoted as $lur(\alpha, z)$, represents the ratio of *local utility* of $\alpha \cup z$ to the sum of *utilities* of all items in $\alpha \cup z$. That is, $lur(\alpha, z) = \frac{lu(\alpha,z)}{\sum_{\forall i_j \in \alpha \cup z} u(i_j)}$.

Example 12. The sum of individual utilities of the items e and f in ef, i.e., $u(e) + u(f) = 165 + 180 = 345$. The *local utility ratio* of ef, i.e., $lur(ef) = \frac{lu(ef)}{u(e)+u(f)} = \frac{510}{345} = 1.47$.

Local utility and *local utility ratio* are *utility* upper bound measures that determine the maximum utility ratio a superset of an itemset can have in the entire database (see Lemma A.5). Thus, these measures can used to prune those uninteresting itemsets whose *supersets* may not yield any RHUI. The correctness of the *secondary itemset* is shown in Theorem A.7.

Definition 12. (Secondary itemset.) An itemset X is said to be a secondary itemset if $lu(X) \geq minUtil$ and $lur(X) \geq minUR$.

Example 13. The itemset ef is a secondary itemset because $lu(ef) \geq minUtil$ and $lur(ef) \geq minUR$. In contrast, the itemsets ed, ec and ea are not secondary itemsets as their lu and lur values do not satify the $minUtil$ and $minUR$, respectively (see Figure 1). Henceforth, the items d, c and a can be pruned from the projected database of e. Thus, reducing the search space effectively.

2.2 RHUI-Miner

Initially, we set $\alpha = \emptyset$ and $su_\alpha = 0$ (i.e., the sum of *utilities* of all items in α equal to zero). Next, we scan the database and calculate the TWU and *utility* values for each item in Table 1. The list of TWU values is $\{\{a : 1380\}, \{b : 1075\}, \{c : 1185\}, \{d : 860\}, \{e : 750\}, \{f : 830\}, \{g : 260\}\}$. Next, secondary items with respect to α is generated by comparing TWU of each item with $minUtil$. The generated secondary items are sorted in TWU ascending order $\succ$. Thus, $Secondary(\alpha) = \{e, f, d, b, c, a\}$. Next, a sorted transactional database as shown in Figure 1 was constructed with only secondary items. Next, we calculate stu for each item in $Secondary(\alpha)$. The stu values for the ordered secondary items is $\{\{e : 675\}, \{f : 605\}, \{d : 825\}, \{b : 1025\}, \{c : 625\}, \{a : 1000\}\}$. The items that have stu value greater than $minUtil$ are considered as *primary items*. That is, $Primary(\alpha) = \{e, f, d, b, c, a\}$. Next, we call RecursiveSearch function to find all RHUIs from the database.

The RecursiveSearch works as follows. Start with any item e in the $Primary(\alpha)$. Since $\alpha = \emptyset$, $\beta = e$ and $su_\beta = u(e)$. We scan the sorted transactional database and calculate $u(\beta) = u(e) = 165$. Next, we construct the projected database of e. In the next step, we calculate the stu, $stur$, lu and lur values of all the items that exist in the projected database of e (see Figure 1). As the stu, $stur$, lu and lur values of f in the projected database of e satisfy the $minUtil$ and $minUR$ constraints, depth-first search will be performed on ef to discover ef as a RHUI. The lu values of d and c are less than $minUtil$ value. Therefore, both of these items are pruned from the projected database of e. Though lu of a is more than the $minUtil$ value, lur is less than the user-specified $minUR$. Henceforth, a is also pruned from the projected database of e. Similar process is repeated for the remaining secondary items in the database to find all RHUIs. The mining procedure of RHUI-Miner is shown in Figure 1.

2.3 Complexity of RHUI-Miner algorithm

The complexity of **RHUI-Miner** can be analyzed as follows. If n is the number of transactions and w is the average transaction length, then in terms of time, a $O(nw \log nw)$ sort is performed initially. This cost is however negligible since it is performed only once. Then, to process each primary itemset α encountered during the depth-first search, **RHUI-Miner** performs database projection, transaction merging and upper-bound calculation. These three operations are each carried out in linear time $O(nw)$. Let l be the number of itemsets in the search space. The global time complexity of **RHUI-Miner** is thus $O(nw \log(nw) + l(nw + nw + nw))$. Since the sort is only performed once, the term $O(nw \log nw)$ can be ignored and the complexity is actually closer to $O(lnw)$.

Thus, the performance of the algorithm is proportional to the number of itemsets in the search space. The number of itemsets in the search space is determined by the upper-bounds which are used to prune the search space. In the **RHUI-Miner** algorithm, four tight upper-bounds are used among which two tight upper-bounds based on *utility ratio* are proposed in this paper. Another aspect that increase the efficiency of **RHUI-Miner** is the use of a pattern-growth approach, that only consider itemsets appearing in the database.

Based on the above time complexity analysis, it can also be observed that the number of transactions n depends on how the actual size of projected databases,and in particular how many transactions can be merged. Thus, the more transactions are merged, the smaller n will be, and the faster the algorithm will be.

In terms of space complexity, the main cost is the space used by utility-bin arrays (simply, lists to record the itemsets) and the space for storing projected databases. Utility-bin arrays are created once and require $O(I)$ space. The database projection operation is performed for each primary itemset α and requires at most $O(nw)$ space for each projected database. In practice, this is small considering that projected databases become smaller as larger itemsets are explored, and that database projections are implemented using offset pointers. Globally, the space complexity of **RHUI-Miner** is $O(lnw + |I|)$ because the number of projected databases is determined by the number of itemsets in the search space l.

A APPENDIX: PROPERTIES AND PROOFS

Property 1. **(Pruning items using TWU value.)** If $i_j \subseteq X$, then $u(X) \leq TWU(i_j)$. Thus, $u(X) < minUtil$ if $TWU(i_j) < minUtil$.

Property 2. If $X \subset Y$, then $u(X) \geq u(Y)$ or $u(X) \leq u(Y)$.

Property 3. If $X \subset Y$, then $ur(X) \geq ur(Y)$ or $ur(X) \leq ur(Y)$.

LEMMA A.1. *The RHUIs generating using minUtil and minUR constraints, do not satisfy the monotonic or anti-monotonic property.*

PROOF. The correctness is straight forward to prove from Properties 2 and 3. □

Property 4. For an itemset X, $lu(X) \geq u(X)$.

Property 5. If $X \subset Y$, then $lu(X) \geq lu(Y)$.

Property 6. If $X \subset Y$, then $\sum_{i_j \in X} u(i_j) \leq \sum_{i_j \in Y} u(i_j)$

Property 7. If $X \subset Y$, then $lu(X) \geq u(Y)$.

Property 8. For a pattern X, if $lu(X) < minUtil$, then X cannot be a high utility itemset.

Property 9. Let X and Y be two itemsets such that $X \subset Y$. If the *local utility* of X is less than $minUtil$, then Y cannot be a high utility itemset. That is, if $lu(X) < minUtil$, then $u(Y) < minUtil$.

Property 10. For an itemset X, $stu(X) \leq lu(X)$.

Lemma A.2. *The* local utility ratio *of an itemset X will always be greater than or equal to its utility ratio value. That is, $lur(X) \geq ur(X)$.*

Proof. According to Property 4,

$$\begin{aligned} lu(X) &\geq u(X) \\ = \frac{lu(X)}{\sum_{i_j \in X} u(i_j)} &\geq \frac{u(X)}{\sum_{i_j \in X} u(i_j)} \\ = lur(X) &\geq ur(X). \end{aligned} \tag{1}$$

Hence proved. □

Lemma A.3. *If $X \subset Y$, then $lur(X) \geq lur(Y)$.*

Proof. According to Property 5, if $X \subset Y$, then

$$\begin{aligned} lu(X) &\geq lu(Y) \\ = \frac{lu(X)}{\sum_{i_j \in X} u(i_j)} &\geq \frac{lu(Y)}{\sum_{i_j \in X} u(i_j)} \end{aligned} \tag{2}$$

As $\sum_{i_j \in X} u(i_j) \geq \sum_{i_j \in Y} u(i_j)$ (see Property 6), Equation 2 can be rewritten as follows:

$$\frac{lu(X)}{\sum_{i_j \in X} u(i_j)} \geq \frac{lu(Y)}{\sum_{i_j \in Y} u(i_j)} = lur(X) \geq lur(Y).$$

Hence proved. □

Lemma A.4. *If the* local utility ratio *of an itemset X is less than the user-specified $minUR$ value, then X cannot be a high utility itemset. That is, if $lur(X) < minUR$, then $ur(X) < minUR$.*

Proof. According to Lemma A.2, $lur(X) \geq ur(X)$. As a result, $ur(X) < minUR$ if $lur(X) < minUR$. □

Lemma A.5. *If the* local utility ratio *of an itemset X is less than the user-specified $minUR$ value, then all supersets of X cannot be high utility itemsets. That is, if $lur(X) < minUR$, then $ur(Y) < minUR$, where $X \subset Y$.*

Proof. According to Lemma A.2, $lur(X) \geq lur(Y)$ if $X \subset Y$. As a result, $lur(Y) < minUR$ if $lur(X) < minUR$. Based on Lemma A.2, it turns out that $ur(Y) \leq lur(Y) < minUR$ if $lur(X) < minUR$. Thus, Y cannot be a high utility itemset. Hence proved. □

Theorem A.6. *For a pattern X, if $lur(X) < minUR$, then neither X nor its supersets can be high utility itemsets.*

Proof. The correctness is straight forward to prove from Lemmas A.4 and A.5. □

Theorem A.7. *For a pattern X, if $lu(X) < minUtil$ or $lur(X) < minUR$, then neither X nor its supersets can be high utility itemsets.*

Proof. Correctness is straight forward to prove from Property 8 and Theorem A.6. □

Lemma A.8. *For an itemset X, $stur(X) \leq lur(X)$.*

Proof. According to Property 10,

$$stu(X) \leq lu(X) = \frac{stu(X)}{\sum\limits_{i_j \in X} u(i_j)} \leq \frac{lu(X)}{\sum\limits_{i_j \in X} u(i_j)} = stur(X) \leq lur(X).$$

Hence proved. □