Two-Phase Algorithms for a Novel Utility-Frequent Mining Model

Jieh-Shan Yeh[1], Yu-Chiang Li[2], and Chin-Chen Chang[3,4]

[1] Department of Computer Science and Information Management, Providence University, Taichung 433, Taiwan
jsyeh@pu.edu.tw

[2] Department of Computer Science and Information Engineering, Southern Taiwan University, Yung-Kang City, Tainan 71005, Taiwan
lyc@cs.ccu.edu.tw

[3] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan

[4] Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 62102, Taiwan
ccc@cs.ccu.edu.tw

Abstract. When companies seek for the combination of products which can constantly generate high profit, the association rule mining (ARM) or the utility mining will not achieve such task. ARM mines frequent itemsets without knowing the producing profit. On the other hand, the utility mining seeks high profit items but no guarantee the frequency. In this paper, we propose a novel *utility-frequent mining* model to identify all itemsets that can generate a user specified utility in transactions, in which the percentage of such transactions in database is not less than a minimum support threshold. A utility-frequent itemset indicates that such combination of products can constantly generate high profit. For finding all utility-frequent itemsets, there is no efficient strategy due to the nonexistence of "downward/upward closure property". In order to tackle such challenge, we propose a bottom-up two-phase algorithm, **BU-UFM**, for efficiently mining utility-frequent itemsets. We also introduce a novel concept, *quasi-utility-frequency*, which is upward closed with respect to the lattice of all itemsets. In fact, each utility-frequent itemset is also quasi-utility-frequent. A top-down two-phase algorithm, **TD-UFM**, for mining utility-frequent itemsets is also presented in the paper.

1 Introduction

Data Mining has made a profound impact on business practices and knowledge management in recent years. Association Rule Mining (or market basket analysis), finding interesting association or correlation relationships among data items, is one of the most important data mining strategies. Since the concept of association rules was introduced by Agrawal et al. [2] in 1993, many algorithms and techniques for mining association rules have been proposed in the literature. Traditional association rule mining (ARM) model treats all the items in the database equally by only considering if an item is present in a transaction or not. ARM focuses on deriving correlations among a set of items and their association rules.

Although finding correlations of itemsets is very important, frequent itemsets identified by ARM may only contribute a small portion of the overall utility. In many situations, people may be more interested in finding out how a set of items support a specific objective that they want to achieve. Recently, a utility mining model was defined [17]. The goal of utility mining is to identify high utility itemsets that drive a large portion of the total utility. Utility mining is useful in a wide range of practical applications. However, it does not indicate how often such itemsets appear in the database. There may be some full priced items or high margin items which are high utility, but only appear in a small number of transactions. Due to the infrequency, such items may not be beneficial to the companies consistently.

When companies seek for the combination of products which can constantly generate high profit, clearly, the association rule mining or the utility mining will not achieve such task. ARM mines frequent itemsets without knowing the producing profit. On the other hand, the utility mining seeks high profit items but no guarantee the frequency. In this paper, we propose a novel *utility-frequent mining* model to identify all itemsets that can generate a user specified utility in each of certain transactions, in which the percentage of such transactions in database is not less than a minimum support threshold. An utility-frequent itemset indicates that the combination of items (or products) can constantly generate high utility (or profit). For finding all utility-frequent itemsets, there is no efficient strategy due to the nonexistence of “downward/upward closure property” (anti-monotone property). In order to tackle such challenge, we propose a bottom-up two-phase algorithm, called Bottom-Up Utility-Frequent Mining algorithm (**BU-UFM**), for efficiently mining utility-frequent itemsets. We also introduce a novel concept, *quasi-utility-frequency*, which is upward closed with respect to the lattice of all itemsets. In fact, each utility-frequent itemset is also quasi-utility-frequent. Therefore, a top-down approach can be applied to identify utility-frequent itemsets as well. **TD-UFM**, a top-down two-phase algorithm, for mining utility-frequent itemsets is also presented in the paper.

The rest of this article is organized as follows. Section 2 overviews the related work. In Section 3, we propose the utility-frequent mining model. Section 4 presents the algorithms. Section 5 provides experimental results. Finally, we conclude in Section 6 with a summary of our work.

2 Related Works

In this section, we review the Support-Confidence Framework and the utility mining model introduced in [17].

2.1 Support-Confidence Framework

Association rule mining attempts to discover the interesting relationships among items in a given a transaction database. The presence of some items in a transaction implies the high possibility of other items also appear in the same transaction. The formal definition is as follows.

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let $DB = \{T_1, T_2, \ldots, T_n\}$, the task-relevant data, be a set of database transactions where each transaction T_j is a set of items, that is, $T_j \subseteq I$. A set of items is also referred as an *itemset*. An itemset that contains k-items is called a k-itemset. Each transaction is associated with an identifier, called TID. Let X be an itemset, a transaction T is said to contain X if and only if $X \subseteq T$. An *association rule* is an implication of the form $X \Rightarrow Y$,where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ holds in the transaction set DB with **support** s, where s is the percentage of transaction in DB that contain $X \cup Y$ (i.e. both X and Y). This is taken to be the probability, $P(X \cup Y)$. The rule $X \Rightarrow Y$ has **confidence** c in the transaction set DB, if c is the percentage of transactions in DB containing X that also contain Y. This is taken to be the conditional probabilities, $P(Y|X)$. That is, $support(X \Rightarrow Y) = P(X \cup Y)$ and $confidence(X \Rightarrow Y) = P(Y|X)$.

A rule that satisfy a minimum support threshold (*min_sup*) and a minimum confidence threshold (*min_conf*) is called *strong*. An itemset is called a *frequent* itemset, if it satisfies minimum support.

Apriori, a multiple passes algorithm [3], is the most famous method to discover frequent itemsets. The Apriori Principle indicates that each subset of a frequent itemset must be frequent; otherwise the itemset is infrequent. This property is also called *downward closure property* or *anti-monotone property*. In each pass, Apriori scans a database once and employs the downward closure property to filter out many useless candidates.

Numerous efficient methods have been proposed to discover frequent itemsets, such as level-wise algorithms [3], [5], [6], [7] and pattern-growth methods [1], [9], [8], [14].

2.2 Utility Mining Model

A utility mining model has been proposed to measure how "useful" an itemset is [17]. It overcomes the shortcomings of traditional association rule mining, which ignores the sale quantity and price (or profitability) among items in a transaction. The traditional association rule ming becomes a special case of utility mining [16].

The following is the definition of a set of terms, given in [17], that leads to the formal definition of utility mining problem.

- The *item count of item* $i_p \in I$ *in transaction* T_q, $c(i_p, T_q)$, is the number of item i_p purchased in transaction T_q. For example, $c(A, T_1) = 0$ $c(B, T_1) = 0$, and $c(C, T_1) = 18$, in Table 1 (a).
- Each item i_p has an associated set of transactions $\mathcal{T}_{i_p} = \{T_q \in DB |\ i_p \in T_q\}$.

Table 1. An example of transaction database [15]

(a) The transaction table

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

(b) The external utility table

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

(c) The transaction utility table

TID	$TU(T_q)$
T_1	23
T_2	71
T_3	12
T_4	14
T_5	14
T_6	13
T_7	111
T_8	57
T_9	13
T_{10}	72

– A k-itemset $X = \{x_1, x_2, \ldots, x_k\}$ is a subset of I, where $1 \le k \le m$, $x_i \in I$ for all $i = 1, 2, \ldots, k$.
– Each k-itemset X has an associated set of transactions $\mathcal{T}_X = \{T_q \in DB | \; X \subseteq T_q\}$.
– The *external utility* of item $i_p \in I$, $eu(i_p)$, is the value associated with item i_p in the external utility table. This value reflects the importance of an item, which is independent of transactions. For example, in Table 1 (b), the external utility of item A, $eu(A)$, is 3.
– The *utility of item* $i_p \in I$ *in transaction* T_q, $u(i_p, T_q)$, is the quantitative measure of utility for item i_p in transaction T_q, defined as $eu(i_p) \times c(i_p, T_q)$. For example, $u(A, T_8) = 3 \times 3 = 9$.
– The *utility of itemset* X *in transaction* T_q, $u(X, T_q)$, is $\sum_{i_p \in X} u(i_p, T_q)$, where $X \subseteq T_q$. For example, let $X = \{A, C\}$, $u(X, T_8) = 3 \times 3 + 25 \times 1 = 34$.
– The *utility of itemset* X, $u(X)$, is defined as $\sum_{X \subseteq T_q \in DB} u(X, T_q)$. For example, let $X = \{A, C\}$, $u(X) = u(X, T_3) + u(X, T_8) = 7 + 34 = 41$.
– The *transaction utility* of T_q, $TU(T_q)$, is equal to $u(T_q, T_q) = \sum_{i_p \in T_q} u(i_p, T_q)$. For example, $TU(T_1) = 18 \times 1 + 1 \times 5 = 23$.

Utility mining is to find all the itemsets whose utility values are beyond a user specified threshold. An itemset X is a *high utility itemset* if $u(X) \ge \epsilon$, where $X \subseteq I$ and ϵ is the minimum utility threshold, otherwise, it is a *low utility itemset.* For example, in Table 1, $u(\{A, D, E\}) = u(\{A, D, E\}, T_4) + u(\{A, D, E\}, T_8) = 14 + 32 = 46$. If $\epsilon = 120$, $\{A, D, E\}$ is a low utility itemset.

There is no efficient strategy to find all the high utility itemsets due to the nonexistence of "downward closure property" in the utility mining model. The challenge of utility mining is in restricting the size of the candidate set and simplifying the computation for calculating the utility.

An exhaustive search method can be applied to identify all high utility itemsets. However, such method is too time-consuming and space-consuming to work well in a large dataset environment. Several heuristic methods have been proposed to accelerate discovering high utility itemsets (or share frequent itemsets), such as MEU [17], SIP, CAC, and IAB [4] methods. Nevertheless, they may loss some high utility itemsets.

Recently, Li, et al. developed some efficient approaches, including the FSM, SuFSM, ShFSM, and DCG methods for share mining [11], [12], [13]. In the meanwhile, Liu, et al. [15] also presented a Two-Phase (TP) algorithm for fast discovering all high utility itemsets. In fact, under appropriate adjustment on item count and external utility of items, share mining is equivalent to utility mining.

3 Utility-Frequent Mining Model

As defined in the previous section, the utility of itemset X in transaction T_q, $u(X, T_q)$, is defined as $\sum_{i_p \in X} u(i_p, T_q)$, where $X \subseteq T_q$.

For a given utility μ, each itemset X is associated with a set of transactions $\mathcal{T}_{(X,\mu)} = \{T_q \in DB | \ X \subseteq T_q \text{ and } u(X, T_q) \geq \mu\}$.

$\mathcal{T}_{(X,\mu)}$ can be seen as the set of transactions that contain X and generate at least utility μ on X. The ratio of the size of $\mathcal{T}_{(X,\mu)}$ and the total number of transactions is denoted as

$$support(X, \mu) = \frac{|\mathcal{T}_{(X,\mu)}|}{|DB|} \tag{1}$$

Definition 1. *For a given utility μ and a given minimum support threshold s, an itemset X is utility-frequent (U-frequent), if $support(X, \mu)$ is not less than s. Otherwise, X is utility-infrequent.*

For example, in Table 1, let $X = \{A, C\}$, $\mu = 20$ and $s = 20\%$, then $\mathcal{T}_{(X,\mu)} = \{T_8\}$, where $X \subseteq T_8$ and $u(X, T_8) = 34 > 20$. $support(X, \mu) = 10\% < s$, therefore, X is utility-infrequent.

Utility-frequent mining is to obtain all itemset X, in which the percentage of transactions, containing X and generating utility value on X beyond a user specified threshold μ, is greater than or equal to a user specified threshold s.

However, U-frequency is neither upward nor downward closed with respect to the lattice of all itemsets. For example, in Table 1, let $\mu = 20$ and $s = 20\%$, $\mathcal{T}_{(\{C\},\mu)} = \{T_8\}$, $\mathcal{T}_{(\{C,E\},\mu)} = \{T_1, T_8\}$ and $\mathcal{T}_{(\{C,D,E\},\mu)} = \{T_8\}$. Since $|DB| = 10$, $support(\{C\}, \mu) = support(\{C, D, E\}, \mu) = 10\%$ and $support(\{C, E\}, \mu) = 20\%$. $\{C, E\}$ is utility-infrequent, however, neither $\{C\}$ nor $\{C, D, E\}$. Therefore, U-frequency is neither upward nor downward closed.

Similar to the utility mining model, the challenge of utility-frequent mining is also in restricting the size of the candidate set and simplifying the computation for calculating the supports.

First, we observe the following important property on utility-frequent itemsets.

Theorem 1. *If itemset X is utility-frequent, X is also frequent.*

Proof: For a given utility μ and a threshold s, clearly, $\mathcal{T}_{(X,\mu)} = \{T_q \in DB|\ X \subseteq T_q \text{ and } u(X,T_q) \geq \mu\} \subseteq \{T_q \in DB|\ X \subseteq T_q\} = \mathcal{T}_X$. $support(X) \geq support(X,\mu)$. If X is U-frequent, $support(X,\mu) \geq s$, so, $support(X) \geq s$. Therefore, X is frequent. □

On the other hand, we give an extended definition of utility-frequent, called *quasi-utility-frequent* in the follows.

For a given utility μ, each itemset X is associated with a set of transactions $\mathcal{T}'_{(X,\mu)} = \{T_q \in DB|\ \sum_{i_p \in X} u(i_p, T_q) \geq \mu\}$. Note that, X contained in T_q is not necessary. In Table 1, consider $X = \{A, C, E\}$ and $\mu = 20$, $\sum_{i_p \in X} u(i_p, T_1) = u(A,T_1) + u(C,T_1) + u(E,T_1) = 0 \times 3 + 18 \times 1 + 1 \times 5 = 23 > \mu$. Accordingly, $T_1 \in \mathcal{T}'_{(\{A,C,E\},20)}$. $\mathcal{T}'_{(X,\mu)}$ can be seen as the set of transactions that generate at least utility μ on X. The ratio of the size of $\mathcal{T}'_{(X,\mu)}$ and the total number of transactions is denoted as

$$Qsupport(X,\mu) = \frac{|\mathcal{T}'_{(X,\mu)}|}{|DB|} \qquad (2)$$

Definition 2. *For a given utility μ, an itemset X is quasi-utility-frequent (QU-frequent), if $Qsupport(X,\mu) \geq s$, where s is the user specified threshold.*

Theorem 2. *QU-frequency is upward closed with respect to the lattice of all itemsets.*

Proof: For a given utility μ and a threshold s, let X and Y be subsets of I, $X \subseteq Y$ and X be QU-frequent. For any T_q in $\mathcal{T}'_{(X,\mu)}$, $\sum_{i_p \in Y} u(i_p, T_q) = \sum_{i_p \in Y-X} u(i_p, T_q) + \sum_{i_p \in X} u(i_p, T_q) \geq \mu$, then T_q is also in $\mathcal{T}'_{(Y,\mu)}$. Therefore, $|\mathcal{T}'_{(Y,\mu)}| \geq |\mathcal{T}'_{(X,\mu)}|$, $Qsupport(Y,\mu) \geq Qsupport(X,\mu) \geq s$. By definition, Y is QU-frequent. □

Moreover, we have the following theorem.

Theorem 3. *If itemset X is utility-frequent, X is also quasi-utility-frequent.*

Proof: Given an utility μ and a threshold s, for any $T_q \in \mathcal{T}_{(X,\mu)}$, by definition, $T_q \supseteq X$, so that $\sum_{i_p \in X} u(i_p, T_q) = u(X, T_q) \geq \mu$. Accordingly, $T_q \in \mathcal{T}'_{(X,\mu)}$. In addition, $Qsupport(X,\mu) \geq support(X,\mu)$. If X is U-frequent, $support(X,\mu) \geq s$, then $Qsupport(X,\mu) \geq s$. Therefore, X is QU-frequent. □

For example, in Table 1, let $\mu = 20$ and $s = 20\%$, $\mathcal{T}'_{(\{C,D,E\},\mu)} = \{T_1, T_8\}$, $\mathcal{T}'_{(\{C,E\},\mu)} = \{T_1, T_8\}$ and $\mathcal{T}'_{(\{C\},\mu)} = \{T_8\}$. $Qsupport(\{C,D,E\},\mu) = Qsupport(\{C,E\},\mu) = 20\%$ and $Qsupport(\{C\},\mu) = 10\%$. $\{C,D,E\}$ and $\{C,E\}$ is QU-frequent, however, $\{C\}$ is not.

Figure 1 and Figure 2 illustrate the inclusion relationships between frequent itemsets and utility-frequent itemsets, and between quasi-utility-frequent itemsets and utility-frequent itemsets, respectively. The set of utility-frequent itemsets, $\{B, BD, BE, CE, BDE\}$, is a subset of both the set of frequent itemsets

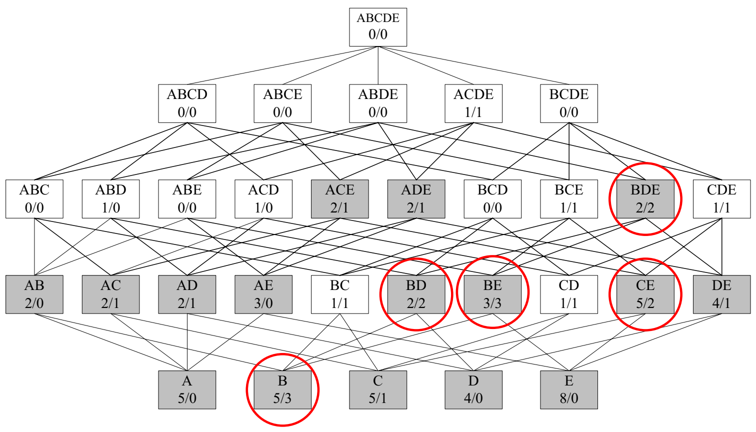

Fig. 1. Itemsets lattice related to the example in Table 1 with $\mu = 20$ and $s = 20\%$. Itemsets in gray-shaded boxes are frequent. Itemsets in circles are utility-frequent. Numbers in each box are "support count/utility support count ($|\mathcal{T}_X|/|\mathcal{T}_{(X,\mu)}|$)".

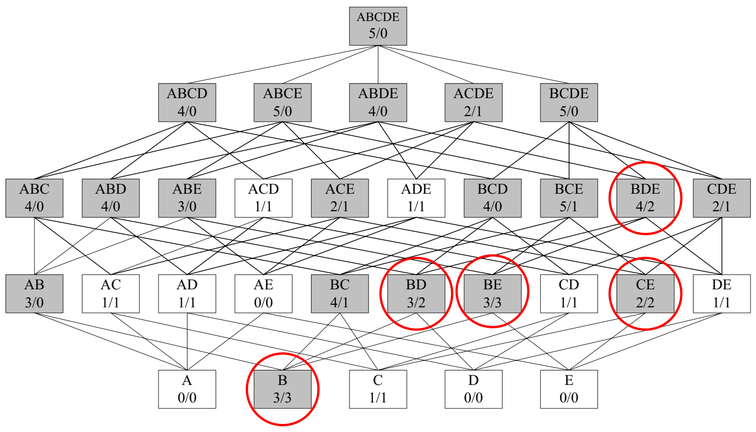

Fig. 2. Itemsets lattice related to the example in Table 1 with $\mu = 20$ and $s = 20\%$. Itemsets in gray-shaded boxes are quasi-utility-frequent. Itemsets in circles are utility-frequent. Numbers in each box are "quasi-utility support count/utility support count ($|\mathcal{T}'_{(X,\mu)}|/|\mathcal{T}_{(X,\mu)}|$)".

and the set of quasi-utility-frequent itemsets. Based on the above observation, we are able to develop a bottom-up and a top-down two-phase utility mining algorithms in the next section.

4 Algorithms

Intuitively, an exhaustive search algorithm can extract all utility-frequent itemsets. For a transaction database with n distinct items, the algorithm must generate 2^n possible itemsets that is impractical.

To avoid generating too many candidates, this study proposes a bottom-up two-phase algorithm to discover utility-frequent itemsets called Bottom-Up Utility-Frequent Mining **BU-UFM** algorithm. According to Theorem 1, each utility-frequent itemset is also frequent. In the first phase, Apriori algorithm is employed to discover all frequent itemsets. In the second phase, **BU-UFM** scan database once to check whether each frequent itemset is utility-frequent. The pseudo-code of **BU-UFM** is as follows:

Algorithm: BU-UFM. Discover all utility-frequent itemsets.
Input: Database DB; minimum utility threshold μ;
minimum support threshold s.
Output: UFI, utility-frequent itemsets in DB.
Method:

```
//Phase I
(1)  $UFI = \emptyset$;
(2)  $CandidateSet$ = Apriori($DB, s$);
//Phase II
(3)  foreach candidate $c \in CandidateSet$ {
(4)      foreach transaction $T \in DB$ { //scan database
(5)          if( $c \subseteq T$ and $u(c,T) < \mu$){
(6)              $c.count - -$;} } }
(7)  foreach candidate $c \in CandidateSet$ {
(8)      if ($c.count \geq s$) {
(9)          $UFI := UFI + c$;} }
(10) return $UFI$;
```

On the other hand, according to Theorem 2, we can utilize the upward closure property on the quasi-utility-frequent itemsets. We first propose an Apriori-like algorithm, **QUF-Apriori**, for the quasi-utility-frequent itemset mining. The following is the detailed algorithm.

Algorithm: QUF-Apriori. Find quasi-utility-frequent itemsets using an iterative level-wise approach based on candidate generation.

Input: Database DB; minimum utility threshold μ;
minimum support threshold s.
Output: $QUFI$, quasi-utility-frequent itemsets in DB.
Method:

```
(1)  $L_{m-1}$ = find_quasi-utility-frequent_($m-1$)-itemsets($DB, \mu, s$);
             // m is the size of $I$.
(2)  for($k = 2; L_{m-k+1} \neq \emptyset; k++$){
(3)      $C_{m-k}$ = apriori_gen($L_{m-k+1}$);
(4)      foreach candidate $c \in C_{m-k}$ { //scan $DB$ for count
(5)          foreach transaction $T \in DB$ {
(6)              if( $u(c,T) \geq \mu$) {
(7)                  $c.count + +$; } } }
(8)      $L_{m-k} = \{c \in C_{m-k} \mid c.count \geq s\}$; }
(9)  return $QUFI = \bigcup_{m-k} L_{m-k}$;
```

```
procedure find_quasi-utility-frequent_$(m-1)$-itemsets (
            $DB$ : database; $\mu$ : utility; $s$ : minimum support threshold)
(1) foreach item $i \in I$ {
(2)     $\overline{\{i\}} = I - \{i\}$; //$(m-1)$-itemset
(3)     foreach transaction $T \in DB$ {
(4)         if( $u(\overline{\{i\}}, T) \geq \mu$) {
(5)             $\overline{\{i\}}.count++$; } }  }
(6) $L_{m-1} = \{\overline{\{i\}} \mid \overline{\{i\}}.count \geq s\}$;
(7) return $L_{m-1}$;
```

According to Theorem 3, the complete set of utility-frequent itemsets is a subset of quasi-utility-frequent itemsets. Now, we propose Top-Down Utility-Frequent Mining algorithm (**TD-UFM**). In Phase I, we utilize **QUF-Apriori** algorithm to discover the complete set of quasi-utility-frequent itemsets. We prune those overestimated itemsets in Phase II. The detailed algorithm is as follows.

```
Algorithm: TD-UFM. Discover all utility-frequent itemsets.
Input: Database $DB$; minimum utility threshold $\mu$;
        minimum support threshold $s$.
Output: $UFI$, utility-frequent itemsets in $DB$.
Method:
//Phase I
(1) $CandidateSet$ = QUF-Apriori($DB, \mu, s$);
//Phase II
(2) foreach candidate $c \in CandidateSet$ { //scan $DB$ for count
(3)     foreach transaction $T \in DB$ {
            //calcuate the utility value of $c$ in $T$ for $c \subseteq T$
(4)         if( $c \subseteq T$ and $u(c, T) \geq \mu$){
(5)             $c.count++$; } } }
(6) $UFI = \{c \in CandidateSet \mid c.count \geq s\}$;
(7) return $UFI$;
```

5 Experimental Results

All the experiments were performed on an AMD K8 3500+ (2200 MHz) PC with 1 GB main memory, running the Windows XP Professional operating system. The **BU-UFM** algorithm was implemented in Visual C++ 6.0 and applied to several synthetic datasets. The two datasets T10.I6.D1000k.N1000 and T20.I6.D1000k.N1000 was generated from the IBM synthetic data generator [10].

The item count of each item in the two datasets were randomly generated between one and four. Observed from real world databases, most items are in the low profit range. Therefore, the external utility of each item was heuristically chosen between 0.01 and 10 and randomly generated with a log-normal distribution.

To choose appropriate minimum utility thresholds in the experiments, instead of randomly selecting μ, we pick different ratios of the average of transaction

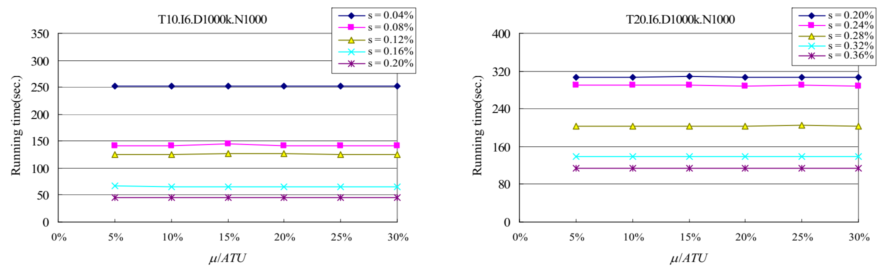

Fig. 3. Running times of **BU-UFM** on T10.I6.D1000k.N1000 and T20.I6.D1000k.N1000 for five minimum support thresholds

Table 2. Itemsets number comparison between utility-frequent itemsets and frequent itemsets

Dataset	T10.I6.D1000k.N1000						
Support threshold (s)	0.16%						
μ/ATU	5%	10%	15%	20%	25%	30%	F_k
Frequent itemset size	UF_k	UF_k	UF_k	UF_k	UF_k	UF_k	
$k = 1$	616	402	255	157	109	84	848
$k = 2$	1575	830	416	212	90	45	2106
$k = 3$	38	18	5	1	0	0	49
$k = 4$	5	5	4	2	0	0	5
$k \geq 5$	0	0	0	0	0	0	0
Dataset	T20.I6.D1000k.N1000						
Support threshold (s)	0.28%						
μ/ATU	5%	10%	15%	20%	25%	30%	F_k
Frequent itemset size	UF_k	UF_k	UF_k	UF_k	UF_k	UF_k	
$k = 1$	428	185	93	54	29	16	859
$k = 2$	4587	1573	503	176	45	14	9037
$k = 3$	109	23	1	0	0	0	205
$k = 4$	253	62	4	0	0	0	339
$k = 5$	422	161	15	0	0	0	462
$k = 6$	462	217	14	0	0	0	462
$k \geq 7$	0	0	0	0	0	0	0

utility (ATU) in database. For example, in Table 1, $\mu = 20$ is equal to 50% of the average of transaction utility (ATU), where the total utility of the database is 400 and the average of transaction utility is 40.

Figure 3 appears the running time performances of **BU-UFM** on both datasets T10.I6.D1000k.N1000 and T20.I6.D1000k.N1000. The x-axis value indicates the percentage of utility threshold to the average of transaction utility in database. A high minimum support threshold resulted in a short running time. For a certain minimum support threshold, the process of Phase I was identical over different utility thresholds, since Phase I is a traditional frequent itemset mining. The parameter of utility threshold almost had no influence the performance, since Phase II checked which candidates were utility-frequent. The running time of Phase II was relatively small. Therefore, the distinction of running time performances over various utility thresholds was insignificant.

Table 2 lists the number of utility-frequent itemsets and frequent itemsets with different item sizes on both T10.I6.D1000k.N1000 and T20.I6.D1000k.N1000 for several several distinct percentages of utility thresholds to ATU.

6 Conclusion

Utility measures the total utility derived from itemsets in a database. It does not indicate how often such itemsets appear in the database. We proposed a novel *utility-frequent mining* model to identify all itemsets that can generate a user specified utility in transactions, in which the percentage of such transactions in database is greater than or equal to a user specified threshold. We proposed a bottom-up two-phase algorithm, **BU-UFM**, for efficiently mining utility-frequent itemsets. We also introduced *quasi-utility-frequent* which is upward closed with respect to the lattice of all itemsets. Since each utility-frequent itemset is also quasi-utility-frequent, therefore, a top-down approach can be applied to identify utility-frequent itemsets as well. In the future, we will investigate more on the relationship among frequent, utility-frequent, and quasi-utility-frequent itemsets. We believe that a combination of top-down and bottom-up approach can more accelerate the mining process on utility-frequent itemsets.

References

1. Agarwal, R.C., Aggarwal, C.C., Prasad, V.V.V.: A tree projection algorithm for generation of frequent item sets. Journal of Parallel and Distributed Computing 61(3), 350–371 (2001)
2. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., USA, pp. 207–216 (1993)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, pp. 487–499 (1994)
4. Barber, B., Hamilton, H.J.: Extracting share frequent itemsets with infrequent subsets. Data Mining and Knowledge Discovery 7(2), 153–185 (2003)
5. Berzal, F., Cubero, J.C., Marín, N., Serrano, J.M.: Tbar: An efficient method for association rule mining in relational databases. Data and Knowledge Engineering 37(1), 47–64 (2001)
6. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: Peckham, J. (ed.) Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 255–264 (1997)
7. Chang, C.C., Lin, C.Y.: Perfect hashing schemes for mining association rules. The Computer Journal 48(2), 168–179 (2005)
8. Grahne, G., Zhu, J.: Fast algorithms for frequent itemset mining using fp-trees. IEEE Transactions on Knowledge and Data Engineering 17(10), 1347–1362 (2005)

9. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent pattern without candidate generation: A frequent pattern tree approach. Data Mining and Knowledge Discovery 8(1), 53–87 (2004)
10. IBM. Synthetic data generation code for associations and sequential patterns (accessed Jan. 2007), `http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/mining.shtml`
11. Li, Y.C., Yeh, J.S., Chang, C.C.: Direct candidates generation: A novel algorithm for discovering complete share-frequent itemsets. In: Wang, L., Jin, Y. (eds.) FSKD 2005. LNCS (LNAI), vol. 3614, pp. 551–560. Springer, Heidelberg (2005)
12. Li, Y.C., Yeh, J.S., Chang, C.C.: Efficient algorithms for mining share-frequent itemsets. In: Fuzzy Logic, Soft Computing and Computational Intelligence - 11th World Congress of International Fuzzy Systems Association (IFSA 2005), pp. 534–539 (2005)
13. Li, Y.C., Yeh, J.S., Chang, C.C.: A fast algorithm for mining share-frequent itemsets. In: Zhang, Y., Tanaka, K., Yu, J.X., Wang, S., Li, M. (eds.) APWeb 2005. LNCS, vol. 3399, pp. 417–428. Springer, Heidelberg (2005)
14. Liu, J., Pan, Y., Wang, K., Han, J.: Mining frequent item sets by opportunistic projection. In: Proceedings of the 8th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 229–238 (2002)
15. Liu, Y., Liao, W.-K., Choudhary, A.: 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)
16. Yao, H., Hamilton, H.J., Geng, L.: A unified framework for utility based measures for mining itemsets. In: Proceedings of ACM SIGKDD 2nd Workshop on Utility-Based Data Mining, pp. 28–37 (2006)
17. Yao, H., Hamilton, H.J., Butz, C.J.: A foundational approach to mining itemset utilities from databases. In: Proceedings of the 4th SIAM International Conference on Data Mining, pp. 428–486 (2004)