Tightening Upper Bounds of Utility Values in Utility Mining

Guo-Cheng Lan\textsuperscript{a}, Tzung-Pei Hong\textsuperscript{b,c} and Vincent S. Tseng\textsuperscript{a,d}

\textsuperscript{a}Department of Computer Science and Information Engineering
National Cheng Kung University, 701, Taiwan
\textsuperscript{b}Department of Computer Science and Information Engineering
National University of Kaohsiung, 811, Taiwan
\textsuperscript{c}Department of Computer Science and Engineering
National Sun Yat-Sen University, 804, Taiwan
\textsuperscript{d}Institute of Medical Informatics
National Cheng Kung University, 701, Taiwan
rrfoheiay@gmail.com; tphong@nuk.edu.tw; tsengsm@mail.ncku.edu.tw

Abstract

Utility mining in data mining has recently been an emerging research issue due to its practical applications. In this paper, with the concept of projection technique, we propose an efficient algorithm for finding high utility itemsets in databases. In particular, an improved upper-bound strategy in the proposed algorithm is designed to further tighten the upper bounds of the utility values for itemsets. The experimental results also show the proposed algorithm has a good performance in comparison with the traditional two-phase utility mining algorithm.

1 Introduction

Utility mining considers both the individual profits and quantities of products (items) in transactions to obtain actual utility values of itemsets [3]. The high utility itemsets, which have their utility values larger than or equal to a predefined threshold, are then found. The downward-closure property in association-rule mining does not, however, exist in utility mining. It is thus quite difficult to find high utility itemsets in databases. To deal with this, Liu \textit{et al.} proposed a two-phase utility mining algorithm (abbreviated as \textit{TP}) [9] to discover high utility itemsets in a database by adopting the downward-closure property, known as the transaction-weighted utilization (abbreviated as \textit{TWU}) model [9]. Their proposed model is that the utility values of all the items in a transaction are summed up as the transaction utility and used as the upper bound of any itemset in that transaction. The transaction-weighted utility of an itemset is then defined as the total transaction utility value of the transactions in which the itemset appears. However, lots of unpromising candidates are still generated in the mining process by the model [9]. It is thus desirable to reduce the number of candidates.

In this study, the concept of projecting tuples from an original database is applied to our proposed mining algorithm to help the execution efficiency. In the past, a similar idea has been applied to the field of data mining. For example, Han \textit{et al.} proposed several projection-based algorithms [10] to achieve the goal of sequential pattern discovery. Using the projection-based technique, the algorithms could achieve a good performance for discovering sequential patterns in databases.

With the concept of projection technique, we propose an efficient projection-based algorithm for finding high utility itemsets in databases. In particular, an improved upper-bound strategy in the proposed algorithm is designed to further tighten the upper bounds of the utility values for itemsets. Thus, there is no need to generate a huge number of candidate itemsets to find high utility ones when compared to the traditional two-phase utility mining algorithm (\textit{TP}). Finally, the experimental results show the proposed algorithm not only has a good effect in terms of pruning unpromising candidate itemsets, but it also executes faster than the traditional \textit{TP} algorithm when working with several synthetic datasets generated by the public IBM data generator [6].

The remaining parts of this paper are organized as follows. Some related works are reviewed in Section 2. The problem to be solved and the proposed mining algorithm with an improved upper-bound strategy for finding high utility itemsets from a set of transactions are stated in Section 3. An example is given to illustrate the execution of the proposed algorithm in Section 4. The experimental results are then shown in Section 5, while the conclusions and suggestions
for future work are given in Section 6.

2 Review of related works

One common type of data mining is to derive association rules from transaction data, such that the presence of certain items in a transaction will imply the presence of some other items [1]. For association-rule mining, Agrawal et al. first proposed several mining algorithms based on the concept of large itemsets to find association rules from transaction data [1-2]. In real-world applications, however, not only products bought in transactions but also the profits and quantities of the products involved in the transactions can be known. In addition, some high-profit products may occur with low frequency in a transaction database. Thus, high-profit but low-frequency itemsets may not be found by using association-rule mining approaches.

Chan et al. proposed utility mining to deal with this problem [3]. In their study [3], a high utility itemset considers not only the quantities of items in transactions, but also their individual profits. Formally, local transaction utility and external utility are used to measure the utility of an item. The local transaction utility of an item is directly obtained from the information stored in a transaction database, like the quantity of the item sold in a transaction. The external utility of an item, like its profit, is given by users. External utility thus often reflects user preferences, and can be represented by a utility table or a utility function. By using a transaction dataset and a utility table together, the discovered itemset is able to better match a user’s expectations than if found by considering only the transaction dataset itself.

For utility mining, it is difficult to find high utility itemsets in databases in comparison with traditional association-rule mining because it does not have the downward-closure property in association-rule mining. Liu et al. subsequently proposed a two-phase utility mining algorithm to discover high utility itemsets in a database by adopting the downward-closure property [9], and this approach is known as the transaction-weighted utilization (abbreviated as TWU) model. This TWU model uses the summation of utility values of all the items in a transaction as the upper bound of any itemset in that transaction to keep the downward-closure property. Several other studies about utility mining have then been published [4-5, 7-8, 11], and most of them were based on the principle of the two-phase utility mining approach to discover high utility itemsets from a transaction database. However, when using the TWU model a large number of candidate itemsets need to be generated for mining. It is thus desirable for reducing the number of candidates in the mining process.

3 The proposed algorithm

The problem of mining high utility itemsets is defined as follows. Assume a database contains a number of transactions, each of which is recorded with the items purchased and their corresponding quantities. A utility table keeping the profits of the items is also given. The problem is to find the itemsets with their high utility values larger than or equal to a predefined minimum utility threshold. An improved utility upper-bound strategy is used in the algorithm to help its execution. The strategy is first described below.

The Improved Upper-bound Strategy

The main concept of the strategy is based on the projection-based technique to improve upper-bound utility values for itemsets. For the proposed projection technique to find high utility itemsets in a database, items in transactions have to be processed in a specific data order. In this study, we thus assume the data have been sorted in an alphabetical order. In addition, the proposed algorithm is based on the transaction-weighted utilization (TWU) model [9] to find high utility itemsets. In the recursive mining process, the alphabetical order is also applied to the processing order of high transaction-weighted utilization itemsets (HTWU) of a prefix itemset.

When a prefix $x$ is processed, only the items appearing in the set (HTWU) of itemsets with $x$ as their prefix can be kept in the projected transactions for $x$ to obtain tighter upper bounds of utility for Itemsets. The transaction utility of each projected transaction is then re-calculated. The transaction utility values of these transactions will decrease since the item numbers in transactions are reduced in the projected database. Tighter upper bounds of utility of itemsets can thus be obtained by the strategy above.

The Projection-based Mining Algorithm with Improved Upper-Bounds

In the proposed algorithm, a recursive procedure called Find-HU($x$, $d$, $r$) is designed to recursively find the high utility itemsets with the prefix itemset $x$, which includes $r$ items. In addition, the improved upper-bound strategy is applied to effectively reduce the number of unpromising itemsets in mining. The details of the proposed algorithm are stated as follows.
**The mining approach with improved upper-bound strategy:**

**INPUT:** A set of items, each with a profit value; a transaction database $D$, in which each transaction includes a subset of items with quantities; the minimum utility threshold $\lambda$.

**OUTPUT:** A final set of high utility itemsets ($HU$).

**STEP 1:** For each $y$-th transaction $Trans_y$ in $D$, do the following substeps.

(a) Calculate the utility value $u_{ij}$ of each $j$-th item $i_j$ in $Trans_y$ as:

$$u_{ij} = s_{ij} \cdot q_{ij},$$

where $s_{ij}$ is the profit of item $i_j$ and $q_{ij}$ is the quantity of $i_j$ in $Trans_y$.

(b) Calculate the transaction utility $tu_y$ of the transaction $Trans_y$ as:

$$tu_y = \sum_{r=1}^{[Trans_y]} u_{jr},$$

where $[Trans_y]$ is the number of items in $Trans_y$.

**STEP 2:** Scan the database $D$ to find the transaction-weighted utility ($twu$) and the actual utility ($au$) of each item.

**STEP 3:** If the transaction-weighted utility ($twu$) of an item is larger than or equal to $\lambda$, put it in the set of $HTWU$; if its actual utility ($au$) is larger than or equal to $\lambda$, put it in the set of $HU$.

**STEP 4:** Process each item $I$ in the set of $HTWU$, in an alphabetical order by the following substeps.

(a) Set $r = 1$, where $r$ represents the number of items in the current itemset to be processed.

(b) Project the required transactions in which the item $I$ appears by keeping $I$ and all the items located after $I$, and denote the projected transactions as $d_I$.

(c) Check whether each item located after the item $I$ in each projected transaction in $d_I$ is a high transaction-weighted utilization $I$-itemset or not. If it is, keep it in the projected transaction; otherwise, remove it.

(d) Check whether the number of items kept in each transaction in $d_I$ is larger than or equal to $r + 1$. If it is, keep it in $d_I$; otherwise, remove it.

(e) Re-calculate the transaction utility of each projected transaction kept in $d_I$.

(f) Sum up the total transaction utility ($ttu_I$) of all projected transactions in $d_I$.

(g) Check whether the total transaction utility $ttu_I$ is larger than or equal to the minimum utility threshold $\lambda$. If it is, do the next substep; otherwise, stop the processing for the next item $I$.

(h) Find all the high utility itemsets with $I$ as their prefix item by using the $Finding-HU(I, d_I, r)$ procedure. Denote the set of returned high utility itemsets as $HUI$.

**STEP 5:** Output the set of high utility itemsets in all the $HUI$.

After STEP 5, all the high utility itemsets are found. The $Finding-HU(s, d_s, r)$ procedure used in the above algorithm is stated as follows.

**The Finding-$HU(x, d_x, r)$ procedure:**

**Input:** A prefix $r$-itemset $x$ and its projected transactions $d_x$.

**Output:** The high utility itemsets with the prefix $x$ ($HUIx$).

**PSTEP 1:** Initialize the temporary itemset table as an empty table, in which each tuple consists of three fields: itemset, transaction-weighted utility ($twu$), and actual utility ($au$).

**PSTEP 2:** For each projected transaction in $d_x$, do the following substeps.

(a) Get each item $I$ located after the prefix $x$ in the projected transaction.

(b) Generate the $(r+1)$-itemset $s$ composed of the prefix $r$-itemset $x$ and $I$, put it in the temporary itemset table, and add its transaction-weighted utility ($twu$) and actual utility ($au$) in the corresponding fields in the temporary itemset table.

**PSTEP 3:** If the transaction-weighted utility ($twu$) of an itemset $s$ in the temporary itemset table is larger than or equal to $\lambda$, put it in the set of $HTWU_{r+1}$ with the prefix $x$; if its actual utility ($au$) is larger than or equal to $\lambda$, put it in the set of $HU_{r+1}$ with the prefix $x$.

**PSTEP 4:** Set $r = r + 1$.

**PSTEP 5:** For each high transaction-weighted utilization itemset $s$ in the set of $HTWU_{r}$ with the prefix $x$ in an alphabetical order, do the following substeps.

(a) Project the required transactions in which the itemset $s$ appears by keeping $s$ and all the items located after $s$, and denote the newly projected transactions as $d_s$.

(b) Check whether each item located after the prefix $x$ in each projected transaction in $d_s$ is a member in $r$-itemsets appearing in the set of
HTWUᵢ, with x as a prefix. If it is, keep the item in the projected transaction; otherwise, remove it.

(c) Check whether the number of items kept of each projected transaction in dᵢ is larger than or equal to r + 1. If it is, keep it in dᵢ; otherwise, remove it.

(d) Re-calculate transaction utility of each projected transaction kept in dᵢ.

(e) Compute the total transaction utility (ttuᵢ) of dᵢ.

(f) Check whether the total transaction utility (ttuᵢ) is larger than or equal to the minimum utility threshold λ. If it is, do the next step; otherwise, stop the processing for the next itemset s.

(g) Find all the high utility itemsets with x as their prefix itemset by using the Finding-HU(s, dᵢ, r+1) procedure. Let the set of returned high utility itemsets be denoted as HUᵢ.

PSTEP 6: Return the set of high utility itemsets in all the HUᵢ.

4 An Example

In this section, an example is given to demonstrate how the proposed algorithm can easily be used to find the high utility itemsets from a set of transactions. Assume the ten transactions shown in Table 2 are used for mining. Each transaction consists of two features, transaction identification (TID) and items purchased. There are six items in the transactions, respectively denoted A to F, and their profit values are 3, 10, 1, 6, 5 and 2. The value attached to each item is the quantity sold in the transactions. Moreover, the minimum utility threshold is set at 55. To find high utility itemsets from the transactions in Table 1, the proposed algorithm proceeds as follows.

Table 1: The set of ten transaction data

<table>
<thead>
<tr>
<th>TID</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trans₁</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Trans₂</td>
<td>0</td>
<td>1</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Trans₃</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Trans₄</td>
<td>0</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Trans₅</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Trans₆</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Trans₇</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Trans₈</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Trans₉</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Trans₁₀</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

For each transaction in the database, the utility value of each item in it and its transaction utility are found. All the possible items are produced from the transaction data in Table 2, and their transaction-weighted utility (tuᵢ) and actual utility (auᵢ) values are found at the same time. Then, the high transaction-weighted utilization I-itemsets (HTWUᵢ) and the high utility I-itemsets (HUᵢ) are found from the set of I-itemsets. Each I-itemset in the set of HTWUᵢ is processed in their prefix order. The I-itemset {A} in the set of HTWUᵢ is processed first. Since the I-itemset {A} in Table 2 appears in the four transactions, Trans₁, Trans₂, Trans₃, and Trans₄, the four transactions, {1A, 2C, 1D, 1E, 1F}, {2A, 8C, 2E}, {3A, 2B, 2E, 3F} and {2A, 1D}, are projected and put in the set of projected transactions dᵢ of {A}. Next, the number of items in each projected transaction in dᵢ is checked to see whether it is larger than or equal to the value 2 (= r + 1). If it is, the transaction is kept in dᵢ and otherwise it is removed. In the example, since all the transactions in dᵢ satisfy the condition, they are kept in dᵢ.

After the above process, the new transaction utility of each projected transaction in dᵢ is found, and the transaction utility values of the four projected transactions are 18, 24, 45 and 12, respectively. All the high utility itemsets with the prefix {A} are found by using the Finding-HU(s, dᵢ, r+1) procedure with the parameters x = {A} and r = 1 as follows.

The temporary itemset table is initialized as an empty table, in which each tuple consists of three fields: itemset, transaction-weighted utility and actual utility of the itemset. For each transaction in the set of projected transactions dᵢ, all possible 2-itemsets with the prefix {A} in it are produced. In addition, the transaction utility (= 12) and the individual utility value of each 2-itemset in the transaction are added to their transaction-weighted utility and actual utility in temporary itemset table. After the step, all the possible 2-itemsets with the prefix {A} include {AB}, {AC}, {AE} and {AF}, and their transaction-weighted utility values are 45, 36, 81 and 57, respectively, and their actual utility values are 29, 19, 43 and 20.

In this example, since the transaction-weighted utility values of the two 2-itemsets, {AE} and {AF}, satisfy the minimum utility threshold (= 55), they are put in the set of HTWUᵢ of the prefix {A}. However, no 2-itemsets are put in the set of HUᵢ of the prefix {A}. The current variable r is then increased to 2. Each (r+1)-itemset in the current set of HTWUᵢ is recursively processed in the alphabetical order. After the above process, the number of items in each projected transaction in dᵢ is checked for whether it is larger than or equal to 3. If it is, the transaction is kept in dᵢ, and otherwise it is removed. In this example, the second projected transaction {2A, 2E} does not satisfy the above condition, the newly projected
transactions in \( d_{\{AE\}} \) then include the following two transactions, \( \{1A, 1E, 1F\} \) and \( \{3A, 2E, 3F\} \). Next, transaction utility of each projected transaction in \( d_{\{AE\}} \) is re-calculated. Since the total transaction utility of \( d_{\{AE\}} \) is smaller than the minimum utility threshold (= 55), the procedure of finding high utility itemsets with the prefix \( \{AE\} \) is terminated, and the processing is executed for the next itemset \( \{AF\} \). All the itemsets in the current set of HTWU_2 of \( \{A\} \) have similarly been done, and all the high utility itemsets with the prefix \( \{A\} \) are returned. The above process is recursively processed until all the other \( 1 \)-itemsets in the set of HTWU_1 have been processed.

In this example, only the two high utility itemsets, \( \{C\} \) and \( \{BC\} \), and their actual utility values are all 57. Finally, they are output as the results.

5 Experimental Evaluation

The experiments were implemented in J2SDK 1.5.0 and executed on a PC with 3.0 GHz CPU and 1GB memory. The public IBM data generator was used in our experiments to produce the data sets [6]. Since our purpose was to find high utility itemsets, we developed a simulation model, which was similar to that used in Liu et al. [9], to generate the quantity values of the items in the transactions. Each quantity ranged among 1 to 5 following the way described in [9]. Moreover, for each dataset generated, a corresponding utility table was also produced in which a profit value in the range from 0.01 to 10.00 was randomly assigned to an item. Figure 1 showed the pruning rates of the PBI algorithm for the T10I4N4KD200K dataset with the thresholds varying from 0.10% to 0.02%, and Figure 2 showed the execution time for the T10I4N4KD200K dataset under different thresholds, varying from 0.10% to 0.02%.

![Figure1: The pruning effect of the PBI algorithm with the strategy along with different minimum utility thresholds.](image1)

![Figure2: Execution time of the two algorithms along with different minimum utility thresholds.](image2)

In the figures, it could be observed that the numbers of candidate itemsets required by the proposed algorithm PBI were obviously less than those required by the algorithm TP. Also, the efficiency of the proposed PBI algorithm was better than that of the TP algorithm, especially when the value of the minimum utility threshold was small. The main reason for this was that the PBI algorithm adopted the prefix concept to improve utility upper bounds for itemsets.

6 Conclusions

In this paper, we have proposed an efficient utility mining approach, called the projection-based mining approach with improved upper bounds (PBI), for finding high utility itemsets in a database. In particular, an improved upper-bound strategy based on a projection-based technique is proposed to obtain more accurate upper bounds of utility values for itemsets. The experimental results show that the proposed algorithm has a good effect in terms of pruning unpromising candidates. Moreover, it also executes faster than the traditional two-phase utility mining algorithm TP for the synthetic datasets generated by the public IBM data generator.

References


