International Journal of Innovative
Computing, Information and Control
Volume 8, Number 7(B), July 2012

ICIC International ©2012 ISSN 1349-4198
pp. 5165–5178

AN EFFICIENT GRADUAL PRUNING TECHNIQUE FOR UTILITY MINING

Guo-Cheng Lan[1], Tzung-Pei Hong[2,3,*] and Vincent S. Tseng[1,4]

[1]Department of Computer Science and Information Engineering
[4]Institute of Medical Informatics
National Cheng Kung University
No. 1, University Rd., Tainan City 701, Taiwan

[2]Department of Computer Science and Information Engineering
National University of Kaohsiung
No. 700, Kaohsiung University Rd., Nanzih District, Kaohsiung 811, Taiwan
*Corresponding author: tphong@nuk.edu.tw

[3]Department of Computer Science and Engineering
National Sun Yat-Sen University
No. 70, Lienhai Rd., Kaohsiung 80424, Taiwan

Received April 2011; revised October 2011

Abstract. *Utility mining in knowledge discovery has recently become a prominent research issue due to its many practical applications. A high utility itemset in utility mining considers not only quantities but also profits of items in transactions. Most of previous approaches were based on the traditional utility upper bound model to find high utility itemsets in databases. By using the model, however, a huge number of candidates have to be generated, and a good deal of time to count utility upper bounds of itemsets has to be needed for mining. In this paper, we thus propose a level-wise mining approach to find efficiently high utility itemsets in databases. In particular, a pruning strategy is designed to gradually cause better utility upper bounds of itemsets in passes. Also, data size could be gradually reduced to save data scan time. Finally, the experimental results on synthetic datasets and a real dataset show the proposed approach outperforms the traditional two-phase utility mining approach in pruning effect and execution efficiency.*

Keywords: Data mining, Utility mining, High utility itemsets, Level-wise mining approach, Pruning strategy

1. **Introduction.** Data mining is a critical process of knowledge discovery in databases, with the goal of extracting useful patterns or rules from data sets. Among the data mining issues, association-rule mining [1,2] is one of the most important and has a wide range of applications in areas from retailing to medicine. However, association-rule mining only considers the high occurrence of items in a transaction database [1], and thus does not reflect any other factors, such as price or profit. Also, the same significance is also assumed for all the items in a database, and thus the actual significance of itemsets cannot be easily recognized. For example, assume there is an itemset like {DVD PLAYER, LCD TV}. This itemset may be not frequent in a database, but may have high utility over the entire database due to the high prices of LCD TV. As this example illustrates, some products (items) with high profit but low frequency may not be found in databases by using association-rule mining approaches. To deal with this, a utility mining approach was proposed by Chan et al., 2003 [3]. Since both the individual profits and quantities of products (items) in transactions were considered in their proposed approach [3], the

actual utility of itemsets could be determined. The high utility itemsets, which had utility values larger than or equal to a predefined threshold, could thus be easily found.

However, since the downward-closure property in association-rule mining [1] cannot be directly used in utility mining [3]. To deal with this, Liu et al. thus proposed a two-phase utility mining (*TP*) algorithm to find high utility itemsets from databases by adopting their specific downward-closure property [12]. By this property, the utility values of all the items in a transaction were summed up as the transaction utility and used as the upper bound of any itemset in that transaction. This property was called the transaction-weighted utilization (*TWU*) model [12]. By using the *TWU* model, their proposed algorithm can effectively handle the problem of utility mining. However, when using the model, it was observed that many unpromising candidates were still generated in each pass for mining.

In this study, we thus propose an efficient utility mining algorithm, which is a level-wise pruning approach (also termed the gradual pruning approach, or *GPA*), to discover high utility itemsets from a database. In particular, a new pruning strategy can be applied to reduce the number of candidates in each pass, in which unpromising items are removed early from transactions to make more precise utility upper bounds for itemsets. Also, the data size in each pass can be gradually reduced to save the scanning time needed for mining. The experimental results show that the number of candidates required by *GPA* is obviously less than that required by the *TP* algorithm [12]. The *GPA* algorithm executes faster than the *TP* algorithm as well.

The remaining parts of this paper are organized as follows. The related works are reviewed in Section 2. The problem to be solved and the proposed mining algorithm with a pruning strategy are stated in Section 3. An example is then given to illustrate the detailed process of the proposed algorithm in Section 4. The experimental results are next shown in Section 5, and the conclusions and directions for future work are given in Section 6.

2. **Review of Related Works.** As mentioned in the introduction, the main principle of data mining is how to extract desired patterns or rules in a set of data. 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 [1,2]. However, only the occurrences of items are considered in association-rule mining, but values for both profits and quantities are usually included in real-world transaction data. Consequently, some high-profit but low-frequency products may not be found by the association-rule mining algorithms. For example, both jewels and diamonds have high utility values, but may not be a frequent product combination when compared with food and drink in a transaction database. To deal with this, Chan et al. proposed an approach called utility mining to discover high utility itemsets in a transaction database [3]. In Chan et al.'s study [3], a utility itemset is identified by considering not only the quantities of the items in transactions, but also their individual profits. Formally, both 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.

However, utility mining is much harder than traditional association-rule mining, as the former lacks the downward-closure property. Liu et al. thus proposed a two-phase utility mining algorithm to find high utility itemsets in a database by adopting a new downward-closure property [12]. The above is called the transaction-weighted utilization (abbreviated as TWU) model [12], which mainly uses the summation of the 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. With the aid of the TWU model, the whole process of the mining algorithm can be divided into two phases. In the first phase, the possible candidate itemsets are found in a transaction database by the TWU model. Then, in the second phase, the database is rescanned to find the actual utility value of each candidate itemset and identify the high utility itemsets with actual utility values larger than or equal to a pre-defined threshold (called the minimum utility threshold). Based on the principle of this two-phase mining algorithm, several other studies about utility mining have also been published [4,5,7,9-11,13-16], such as on-shelf utility mining, temporal utility mining, and incremental utility mining. However, a large number of unpromising candidates may still be produced with the TWU model [12], and thus it is desirable to reduce the number.

3. **The Proposed Algorithm.**

In this paper, we propose a gradual pruning (GPA) algorithm with a pruning strategy to efficiently find all high utility itemsets from a database. A utility itemset considers the individual profits and quantities of products in transaction data. The problem of utility mining is defined as follows. Assume a database contains a number of transactions, and each transaction is recorded with the items purchased and their corresponding quantities. Also, a utility table with the profits of the items is given. The problem is to find the itemsets with utility values larger than or equal to a predefined minimum utility threshold. The pruning strategy is first described below.

3.1. **Pruning unpromising item strategy.**

The proposed strategy is mainly applied to remove unpromising items in transactions, and then improve the upper bounds of utility values of itemsets for mining. The strategy is still based on the transaction-weighted utilization (TWU) model proposed by Liu et al. [12]. Based on the downward-closure property of the TWU model, candidate $(r+1)$-itemsets with $r+1$ items (C_{r+1}) of the next pass are produced from the set of high transaction-weighted utilization r-itemsets ($HTWU_r$) in the current rth pass. That is, if one item is not a member of any of the set of $HTWU_r$ in the rth pass, then it cannot be a member of any of the high transaction-weighted utilization itemsets in the later passes. As mentioned above, some items in a transaction that do not appear in the current set of $HTWU_r$ can thus be removed from that transaction, so that the upper bound of transaction utility of that transaction can be tighter. To avoid keeping unnecessary transactions, the number of items kept in each modified transaction is checked for whether it is larger than or equal to the number of items in the itemsets to be processed in the next pass. If a transaction does not satisfy the condition, it is removed from the current set of modified transactions; otherwise, it is kept. Below, an example is given to illustrate how to improve upper bounds of utility values for itemsets by using the strategy.

For example, assume the following three transactions are to be processed: $Trans_1$: $\{1A, 1B, 2C, 1D\}$, $Trans_2$: $\{1B, 25C\}$ and $Trans_3$: $\{1B, 12C\}$, where the numbers represent quantities and the symbols represent items. There are four items in the three transactions, denoted as A to D. Also, the profits of the items are assumed to be 3, 10, 1 and 6, respectively, and the minimum utility threshold is set at 30. First, the transaction utility of each transaction and the individual utility values of items in each transaction are found. Take the first transaction $Trans_1$: $\{1A, 1B, 2C, 1D\}$ as an example. The transaction $Trans_1$ includes four items, A, B, C and D, their quantity values are 1, 1, 2 and 1, and their profits are 3, 10, 1 and 6, respectively. Their individual utility values can then be calculated as $1 * 3$ (= 3), $1 * 10$ (= 10), $2 * 1$ (= 2), and $1 * 6$ (= 6), respectively. Its transaction utility can thus be calculated as $3 + 10 + 2 + 6$, which is 21. All the other transactions can be similarly processed. The transaction utility values of the three transactions are then 21, 35 and 22.

Next, the transaction-weighted utility (twu) and the actual utility (au) of all possible items in the transactions can be found in the first pass. Take item B as an example, which appears in the three transactions, $Trans_1$, $Trans_2$ and $Trans_3$, and their transaction utility values are 21, 35 and 22, respectively. In addition, the quantity values of B in the three transactions are all 1, and its profit is 10. Then all the utility values of item B in the three transactions can be calculated as $1 * 10$ (= 10), and thus the transaction-weighted utility value and actual utility value of the 1-itemset $\{B\}$ can be calculated as $21 + 35 + 22$ (= 78) and $10 + 10 + 10$ (= 30), respectively. All the other items in the transactions can be processed in the same way. The transaction-weighted utility values of the four 1-itemsets in the transactions, $\{A\}$, $\{B\}$, $\{C\}$ and $\{D\}$, are 21, 57, 78 and 21, respectively, and their actual utility values are 3, 30, 39 and 6. In this example, since the transaction-weighted utility values of only the two 1-itemsets, $\{B\}$ and $\{C\}$, satisfy the minimum utility threshold (= 30), they are put in the set of high transaction-weighted utilization 1-itemsets ($HTWU_1$). In addition, only the 1-itemset $\{C\}$ is put into the set of high utility 1-itemsets (HU_1) as its actual utility value is larger than the minimum utility threshold.

In the second pass, only one candidate 2-itemset $\{BC\}$ is generated from the set of $HTWU_1$. The members of itemsets in the set of candidate 2-itemsets only include two items, B and C, and only they can be kept in transactions at the second pass. Take the first transaction $Trans_1$: $\{1A, 1B, 2C, 1D\}$ as an example. Since the two items A and D in $Trans_1$ are not the members of itemsets in the set of candidate 2-itemsets, they are removed from $Trans_1$. The modified transaction then becomes $\{1B, 2C\}$. All the other transactions can be processed in the same way, and the modified transactions are $\{1B, 2C\}$, $\{1B, 25C\}$ and $\{1B, 12C\}$. Next, the number of items kept for each modified transaction is checked to see whether the number of this transaction is larger than or equal to the value (= 2). If it is, the modified transaction is kept in the modified transactions, and otherwise it is removed. Continuing this example, since the first modified transaction satisfies the condition, it is kept in the set of modified transactions.

After the above process, the transaction utility for each transaction in the set of modified transactions is updated. Take the first modified transaction $\{1B, 2C\}$ as an example; It includes the two items, B and C, and their quantity values are 1 and 2. Also, the profits for the two items are 10 and 1. The new transaction utility of the transaction can then be calculated as $(1 * 10) + (2 * 1)$, which is 12. All the other transactions in the set of modified transactions can be similarly processed, and thus the new transaction utility values of the three transactions, $\{1B, 2C\}$, $\{1B, 25C\}$ and $\{1B, 12C\}$, are found as 12, 35, and 22, respectively. Thus, the transaction-weighted utility of the candidate 2-itemset $\{BC\}$ can be calculated as $12 + 35 + 22$, which is 69. After the second pass, both the transaction-weighted utility and the actual utility of the candidate $\{BC\}$ are 69. The 2-itemset $\{BC\}$ is thus a high transaction-weighted utilization 2-itemset, and it is also a high utility 2-itemset.

By using the TWU model without the pruning strategy, the transaction-weighted utility value of the 2-itemset in this example, $\{BC\}$, is 78. Thus, the proposed algorithm can improve the utility upper bounds of itemsets in a database. Another strategy for, reduction of data sizes, is then described in the following subsection.

3.2. **Reducing data size strategy.** For some modified transactions, all the items within them may become the same after the process of pruning unpromising items. To reduce the scanning time, the transactions with the same items are merged into one transaction, and the quantity values of items in the merged transactions are also summed together. After the above step, the transaction utility values of the merged transactions are re-calculated and added together as the current total transaction utility. The mining process may be terminated early when the current total transaction utility is not larger than or equal to the minimum utility threshold. The main reason is that no itemsets with enough utility upper bounds can be found from the transactions when the above condition is satisfied. As noted above, the data size is gradually reduced in the later passes.

Continuing the example described in Section 3.1, the set of modified transactions are $\{1B, 2C\}$, $\{1B, 25C\}$ and $\{1B, 12C\}$ after the pruning strategy is performed. In this example, since the three modified transactions have the same items, B and C, they can be merged into a new one. Also, the quantity values of the item B in the transactions are all 1, and the quantity values of the item C are 2, 25 and 12, respectively. The new quantity values of the two items, B and C, in the new transaction can be then calculated as $1 + 1 + 1$ ($= 3$) and $2 + 25 + 12$ ($= 39$). The merged transaction is thus $\{3B, 39C\}$. Next, the transaction utility of each merged transaction is re-calculated. In this example, since the profit values of the above two items are 10 and 1, the new transaction utility of the merged transaction is calculated as $(3 * 10) + (39 * 1)$, which is 69. As shown in this example, the proposed algorithm can increase the efficiency of finding high utility itemsets in a database. The details of the proposed algorithm are stated in the following subsection.

3.3. **The proposed algorithm for finding high utility itemsets.** The details of the proposed algorithm (Gradual Pruning Approach, *GPA*) are as follows.

The proposed level-wise mining algorithm with a gradual pruning 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 λ.
OUTPUT: A final set of high utility itemsets (HU).
STEP 1: For each yth transaction $Trans_y$ in D, do the following substeps.

(a) Calculate the utility value u_{yj} of each jth item I_{yj} in $Trans_y$ as:

$$u_{yj} = s_{yj} * q_{yj},$$

where s_{yj} is the profit of item I_{yj} and q_{yj} is the quantity of I_{yj}.

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

$$tu_y = \sum_{j=1}^{|Trans_y|} u_{yj},$$

where $|Trans_y|$ is the number of items in $Trans_y$.

STEP 2: Calculate the total transaction utility ttu in the set of transactions as:

$$ttu = \sum_{y} tu_y.$$

STEP 3: For each item I in D, do the following substeps.

(a) Calculate the transaction-weighted utility twu_I of each item I as the summation of the transaction utility values of the transactions which include the item I. That is:

$$twu_I = \sum_{I \in Trans_y} tu_y.$$

(b) Calculate the actual utility au_I of each item Ias the summation of the utility values of the transactions which include the item I. That is:

$$au_I = \sum_{I \in Trans_y} u_{yI}.$$

STEP 4: For each item I in D, do the following substeps.

(a) Check whether the transaction-weighted utility twu_I of an item I is larger than or equal to the minimum utility threshold λ. If it is, put it in the set of high transaction-weighted utilization 1-itemsets, $HTWU_1$.

(b) Check whether the actual utility au_I of an item I is larger than or equal to the minimum utility threshold λ. If it is, put it in the set of high utility 1-itemsets, HU_1.

STEP 5: Set $r = 1$, where r represents the number of items in the current set of candidate utility itemsets (C_r) to be processed.
STEP 6: Generate the candidate set C_{r+1} from set $HTWU_r$, for which all the r-sub-itemsets in each candidate in C_{r+1} must be contained in set $HTWU_r$.
STEP 7: Acquire the items existing in set $HTWU_{r+1}$ and denote them as S_{r+1}.
STEP 8: For each transaction $Trans_y$ in D, do the following substeps.

(a) Remove the items not appearing in S_{r+1} in $Trans_y$.

(b) Check whether the number of items kept in the modified transaction is smaller than the number of items in the current set of candidate utility itemsets C_{r+1}. If it is, remove the transaction; otherwise, put it in the set of modified transactions.

STEP 9: Merge the modified transactions with the same items into a new transaction $Trans_m$ in the set of modified transactions; the quantity of each item in the merged transaction $Trans_m$ is the sum of all its quantities in the transactions to be merged.
STEP 10: Calculate the transaction utility tu_m of each modified transaction $Trans_m$. That is:

$$tu_m = \sum_{j=1}^{|Trans_m|} u_{mj},$$

where $|Trans_m|$ is the number of items in $Trans_m$.
STEP 11: Calculate the total transaction utility ttu_r in the set of currently modified transactions in the rth pass. That is:

$$ttu_r = \sum_m tu_m.$$

STEP 12: If the total transaction utility ttu_r in the set of modified transactions is larger than or equal to the threshold λ, go to the next step; otherwise, go to STEP 16.
STEP 13: Scan the set of modified transactions to find the transaction-weighted utility twu_x and actual utility au_x of each itemset x in set C_{r+1} as described in STEP 3.
STEP 14: For each candidate (r+1)-itemset x in set C_{r+1}, do the following substeps.

(a) Check whether the transaction-weighted utility twu_x of x is larger than or equal to the minimum utility threshold λ. If it is, put it in set $HTWU_{r+1}$.

(b) Check whether the actual utility au_x of x is larger than or equal to the minimum utility threshold λ. If it is, put it in set HU_{r+1}.

STEP 15: If $HTWU_{r+1}$ is null, do STEP 16; otherwise, set $r = r + 1$ and repeat STEPs 6 to 15.
STEP 16: Output the final set of high utility itemsets, *HU*.

4. An Example of Using the *GPA* Algorithm.

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

Table 1. The set of ten transaction data used in this example

TID	***A***	***B***	***C***	***D***	***E***	***F***
$Trans_1$	1	0	2	1	1	1
$Trans_2$	0	1	25	0	0	0
$Trans_3$	0	0	0	0	2	1
$Trans_4$	0	1	12	0	0	0
$Trans_5$	2	0	8	0	2	0
$Trans_6$	0	0	4	1	0	1
$Trans_7$	0	0	2	1	0	0
$Trans_8$	3	2	0	0	2	3
$Trans_9$	1	0	0	1	0	0
$Trans_{10}$	0	0	4	0	2	0

STEP 1: The utility value of each item occurring in each transaction is found. Take the second transaction $Trans_2$ in Table 1 as an example. The transaction includes the two items, B and C, and their quantitiesare 1 and 25. Also, the profits of items B and C are 10 and 1. Their utility values in $Trans_2$ can be then calculated as $1 * 10$ $(= 10)$ and $25 * 1$ $(= 25)$, respectively. After this process, the transaction utility (*tu*) of each transaction in Table 1 is found. Continuing the example, the transaction utility of $Trans_2$ can be calculated as $10 + 25$, which is 35. All the other transactions can be processed in the same way. After this step, the transaction utility (*tu*) values of the ten transactions in Table 1 are 18, 35, 12, 22, 24, 12, 8, 45, 9, and 14, respectively.

STEP 2: The total transaction utility (*ttu*), which is the summation of the transaction utility values of all the transactions, is calculated. Using the data in Table 1, the total transaction utility can be calculated as $18 + 35 + 12 + 22 + 24 + 12 + 8 + 45 + 9 + 14$, which is 199.

STEP 3: All the possible items are generated from the transaction data, and their transaction-weighted utility (*twu*) and actual utility (*au*) values are also found. Taking the item A in Table 1 as an example, it appears in four transactions, $Trans_1$, $Trans_5$, $Trans_8$ and $Trans_9$, and its utility values are 3, 6, 9 and 3, respectively. Its transaction-weighted utility and actual utility can then be calculated as $18 + 24 + 45 + 9$ $(= 96)$ and $3 + 6 + 9 + 3$ $(= 21)$, respectively, and the item A and its two values are then put in set C_1. All the other items in Table 1 can be similarly processed, and the results for the transaction-weighted utility (*twu*) and actual utility values (*au*) of all the 1-itemsets in set C_1 are shown in Table 2.

TABLE 2. The transaction-weighted utility and actual utility values of 1-itemsets in C_1

1-itemset	***twu***	***au***
$\{A\}$	96	21
$\{B\}$	102	40
$\{C\}$	133	57
$\{D\}$	47	24
$\{E\}$	113	45
$\{F\}$	87	12

STEP 4: In Table 2, since the transaction-weighted utility (= 47) of only $\{D\}$ does not satisfy the minimum utility threshold λ (= 55), all the other 1-itemsets, $\{A\}$, $\{B\}$, $\{C\}$, $\{E\}$, and $\{F\}$, are put in set $HTWU_1$. On the other hand, the actual utility (= 57) of only $\{C\}$ in Table 2 satisfies the minimum utility threshold λ (= 55), and thus it is put in set HU_1.

STEP 5: The variable r is currently set at 1, where r is used to represent the number of items in the current candidate itemsets to be processed.

STEP 6: Ten candidate 2-itemsets are generated from set $HTWU_1$ as follows: $\{AB\}$, $\{AC\}$, $\{AE\}$, $\{AF\}$, $\{BC\}$, $\{BE\}$, $\{BF\}$, $\{CE\}$, $\{CF\}$ and $\{EF\}$.

STEP 7: The items appearing the items of 2-itemsets in set C_2 then include A, B, C, E and F, and these are put in set S_2.

STEP 8: For each transaction in Table 1, the items in it which do not appear in set S_2 are removed. Also, it is checked to see whether its new transaction length is larger than or equal to the value (= 2). If it is, it is kept in the set of modified transactions, and otherwise it is removed. Taking the first transaction $Trans_1$ in Table 1 as an example, it includes five items A, C, D, E and F. However, since the item D does not appear in set S_2, it is removed from $Trans_1$. After this process, the number (= 4) of items kept for that transaction is larger than or equal to the value (= 2), and thus it is kept in the set of modified transactions. All the other transactions can be processed similarly, and the results are shown in Table 3.

TABLE 3. The results of all the modified transactions in this example

TID	***A***	***B***	***C***	***E***	***F***
$Trans_1$	1	0	2	1	1
$Trans_2$	0	1	25	0	0
$Trans_3$	0	0	0	2	1
$Trans_4$	0	1	12	0	0
$Trans_5$	2	0	8	2	0
$Trans_6$	0	0	4	0	1
$Trans_8$	3	2	0	2	3
$Trans_{10}$	0	0	4	2	0

STEP 9: Some modified transactions with the same items in the set of modified transactions are merged into one.

In Table 3, since only the two transactions, $Trans_2$ and $Trans_4$, have the same items, namely B and C, they are merged into one transaction. The quantity values of the item B in $Trans_2$ and $Trans_4$ are both 1, and the quantity values of the item C are 25 and 12. Their quantity values in the merged transaction can then be calculated as $1 + 1$ (= 2)

Table 4. All the merged transactions in this example

MID	***A***	***B***	***C***	***E***	***F***
$Trans_{m1}$	1	0	2	1	1
$Trans_{m2}$	0	2	37	0	0
$Trans_{m3}$	0	0	0	2	1
$Trans_{m4}$	2	0	8	2	0
$Trans_{m5}$	0	0	4	0	1
$Trans_{m6}$	3	2	0	2	3
$Trans_{m7}$	0	0	4	2	0

and $25 + 12$ $(= 37)$, respectively. The results for all the merged transactions are shown in Table 4. Note that the identifier number of each transaction is the merged one and not the original one.

STEP 10: The transaction utility (tu) of each transaction in the set of modified transactions is re-calculated. The process is the same as that mentioned in STEP 1. After this step, thus, the transaction utility values of the seven merged transactions in Table 4 are 12, 57, 12, 24, 6, 45 and 14.

STEP 11: The current total transaction utility in Table 4 can then be calculated as $12 + 57 + 12 + 24 + 6 + 45 + 14$, which is 170.

STEP 12: The new total transaction utility for the seven merged transactions is larger than or equal to the minimum utility threshold λ, which is 55. Thus, the next step is done.

STEP 13: The merged transactions in Table 4 is scanned to find transaction-weighted utility (twu) value and actual utility (au) value of each candidate 2-itemset in set C_2. Take the candidate itemset $\{AE\}$ in set C_2 as an example. This 2-itemset appears in three transactions, $Trans_{m1}$, $Trans_{m4}$ and $Trans_{m6}$, in Table 5, and their transaction utility values are 12, 24 and 45, respectively. Also, the utility values of the item A in the three transactions are 3, 6, and 9, and the utility values of the item E are 5, 10, and 10, respectively. The transaction-weighted utility (twu) of $\{AE\}$ can then be calculated as $12 + 24 + 45$, which is 81, and the utility values of $\{AE\}$ in the transactions can be calculated as $(3+5)+(6+10)+(9+10)$, which is 43. All the other candidate 2-itemsets in set C_2 can be similarly processed. The results for the calculated transaction-weighted utility (twu) and actual utility (au) of each candidate 2-itemset in set C_2 are then shown in Table 5.

Table 5. The transaction-weighted utility and actual utility of each 2-itemset in set C_2

2-itemset	***twu***	***au***	***2-itemset***	***twu***	***au***
$\{AB\}$	45	29	$\{BE\}$	45	30
$\{AC\}$	36	19	$\{BF\}$	45	26
$\{AE\}$	81	43	$\{CE\}$	50	39
$\{AF\}$	57	20	$\{CF\}$	18	10
$\{BC\}$	57	57	$\{EF\}$	69	35

STEP 14: In Table 5, the transaction-weighted utility values of only the four 2-itemsets in set C_2, $\{AE\}$, $\{AF\}$, $\{BC\}$ and $\{EF\}$, satisfy the minimum utility threshold λ $(= 55)$, and they are put in set $HTWU_2$. In addition, the actual utility of only the 2-itemset $\{BC\}$ in set C_2 satisfies the minimum utility threshold λ $(= 55)$, and so it is then put in set HU_2.

STEP 15: In this example, since set $HTWU_2$ is not null, r is incremented to 2 and STEPs 6 to 15 are repeated. The candidate 3-itemset $\{AEF\}$ is then generated from set $HTWU_2$, and the total transaction utility of the merged transactions in the 3th pass is 35 after the STEP 13 is done. The mining task can be terminated early since the total transaction utility in the 3th pass is smaller than the minimum utility threshold λ, which is 55. The reason is that the 3-itemset $\{AEF\}$ does not still have enough transaction-weighted utility and actual utility to satisfy the minimum utility threshold, even if the merged transactions in the 3th pass are scanned, and thus the next step is STEP 16.

STEP 16: In this example, set *HU* only includes two high utility itemsets, $\{C\}$ and $\{BC\}$, and their actual utility values are both 57, and these are then output as the decision makers' auxiliary information. Note that the three itemsets, $\{AEF\}$, $\{CE\}$ and $\{EF\}$, are the high transaction-weighted utilization itemsets obtained by using the traditional *TWU* model without the pruning strategy [12], but these itemsets are not by using *GPA*. As a result, the *GPA* algorithm can improve the execution efficiency with the strategy for mining.

5. Experimental Evaluation.

A series of experiments were conducted to compare the performances of the proposed *GPA* algorithm and the traditional two-phase utility mining (*TP*) algorithm [12] with different parameter values. They were implemented in J2SDK 1.5.0 and executed on a PC with 3.0 GHz CPU and 1 GB memory.

5.1. Experimental datasets.

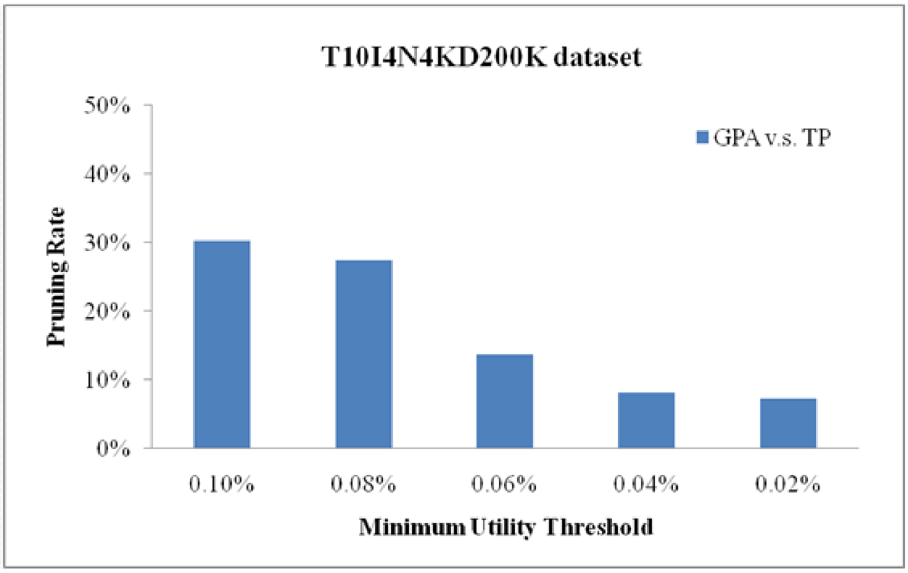
In the experiments, the public *IBM* data generator was utilized to produce the required testing datasets [8]. Since our purpose was to find out high utility itemsets, we developed a simulation model, which was similar to that used in Liu et al. [12]. The quantity values generated by using the simulation model are attached to items in the transactions. Each quantity ranged from 1 to 5 following the described way in that study [12]. 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 each item. Besides, the related parameters included T, I, N and D, which represented the average length of items per transaction, the average length of maximal potentially frequent itemsets, the total number of different items, and the total number of transactions, respectively.

To show the practical performance, on the other hand, a real public dataset (called *BMS-POS*) was also used in the experiments [6]. The real dataset was used in the KDDCUP 2000 competition, and it collected several years of point-of-sale data from a large electronics retailer, making it very suitable as a test dataset. The details of this dataset are follows. Each transaction consisted of all the product categories purchased by a customer at one time. There were a total of 515,597 transactions in this dataset and the number of different items was 1,657. In addition, the maximal length of a transaction was 164, and the average length was 6.5. However, since there were no quantitative values and utility values for the items in transactions in *BMS-POS*, the quantity values and the utility values forming a utility table were generated by our model and were appended to the items in transactions.

5.2. Effectiveness of the pruning strategy.

Experiments were first made on the synthetic datasets to evaluate the difference in the number of candidate itemsets required by the two algorithms, *GPA* and *TP*. To evaluate the effect of pruning unpromising candidates (abbreviated as C) for the proposed pruning strategy, a measure was defined below to evaluate the pruning effect:

$$Pruning\ Effect = \frac{|C_{TP}| - |C_{GPA}|}{|C_{TP}|},$$



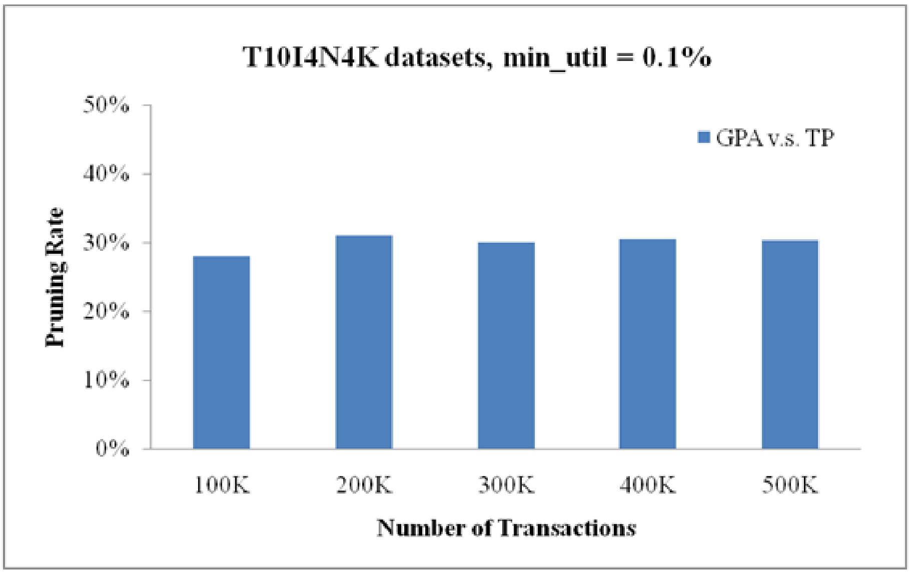


FIGURE 1. The pruning effects of *GPA* along with different thresholds and data sizes

where $|C_{TP}|$ and $|C_{GPA}|$ represent the number of candidate itemsets generated by the two algorithms. Figure 1 showed the pruning effects of the *GPA* algorithm for the T10I4N4KD200K dataset with the thresholds varying from 0.10% to 0.02% and for the T10I4N4K datasets with the sizes varying from 100K to 500K when the minimum utility threshold λ was set at 0.1%, respectively.

It could be observed from these figures that the pruning effect decreased along with the minimum utility thresholds. Note that it was not easy to prune the unpromising candidate itemsets when the values of the minimum utility threshold were set at a lower level, but at least an average of 18% of the whole pruning rates in Figure 1 was achieved.

5.3. **Efficiency evaluation.** Figure 2 showed the execution time for the T10I4N4KD20 0K dataset with the thresholds varying from 0.10% to 0.02% and for the T10I4N4K datasets with the sizes varying from 100K to 500K when λ was set at 0.1%, respectively.

As can be seen easily in these figures, the execution time of the *GPA* algorithm was obviously better than that of the traditional *TP* algorithm, especially when the value of

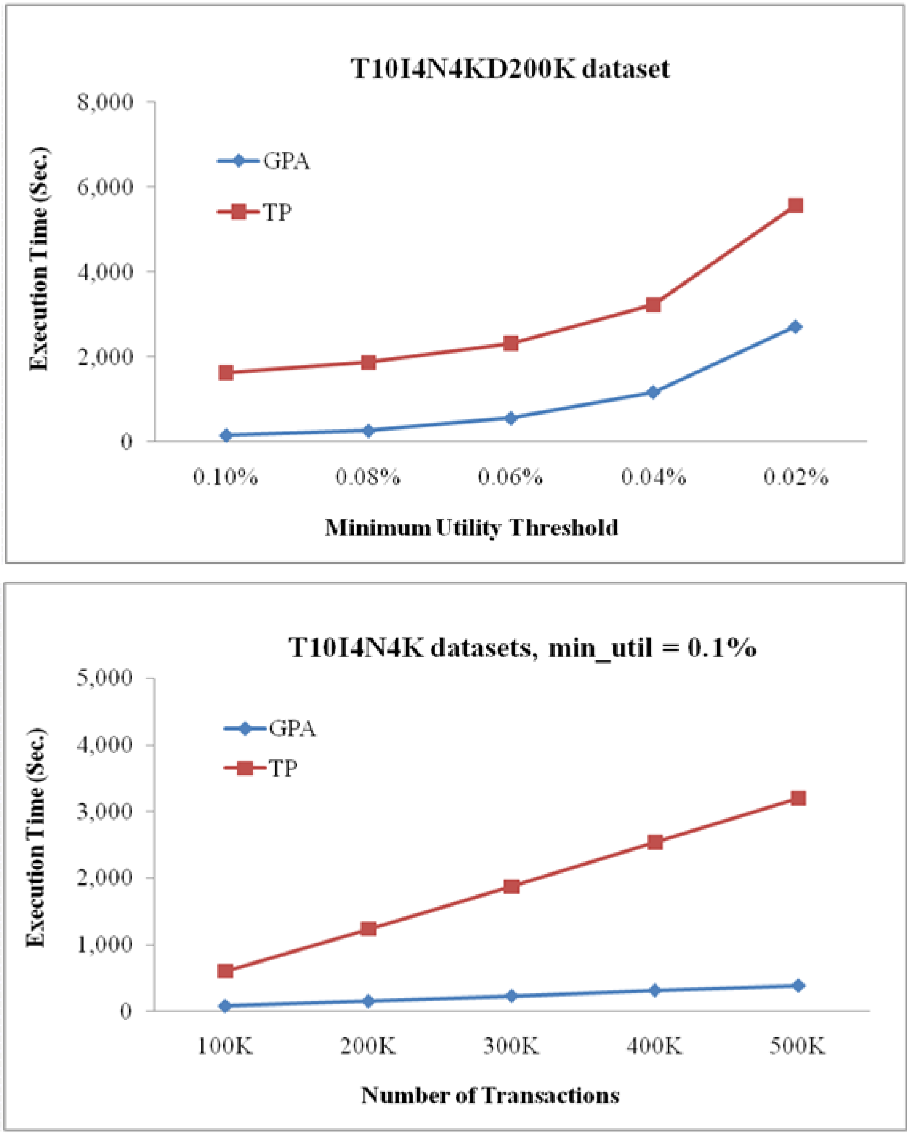


FIGURE 2. Efficiency of the two algorithms along with different thresholds and data sizes

the minimum utility threshold was small or the data size was large, for a similar reason as mentioned in Section 5.2. Also, the data size could be gradually reduced in each pass to rescue the time cost of data scan with the pruning strategy. The *GPA* algorithm thus outperformed better than the *TP* algorithm in discovering high utility itemsets.

5.4. **Evaluation on a real dataset.** The real dataset *BMS-POS* was also used to evaluate the performance of *GPA* and *TP*. Figure 3 showed the effect of pruning unpromising candidates for the strategy with the threshold varying from 5% to 1%. Figure 4 showed the execution time for the real dataset *BMS-POS* with the threshold varying from 5% to 1%.

It can be seen that the *GPA* algorithm outperformed the traditional *TP* algorithm for the real dataset *BMS-POS* with regard to both pruning effectiveness and execution efficiency, and the reason for this is the same as mentioned previously.

6. **Conclusions.** In this paper, we proposed an efficient mining algorithm for finding high utility itemsets in databases. The main contributions of the paper are summarized

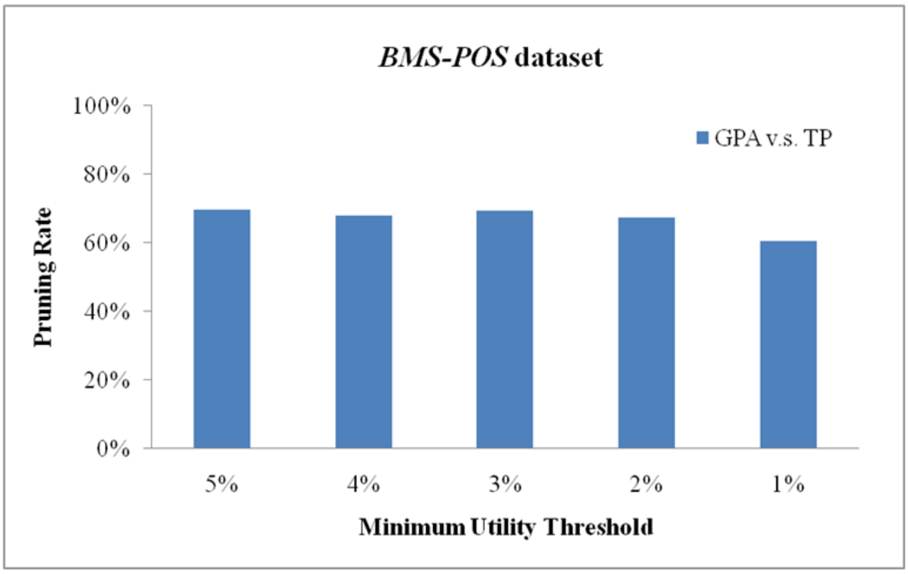


FIGURE 3. The pruning effects of *GPA* along with different thresholds and data sizes

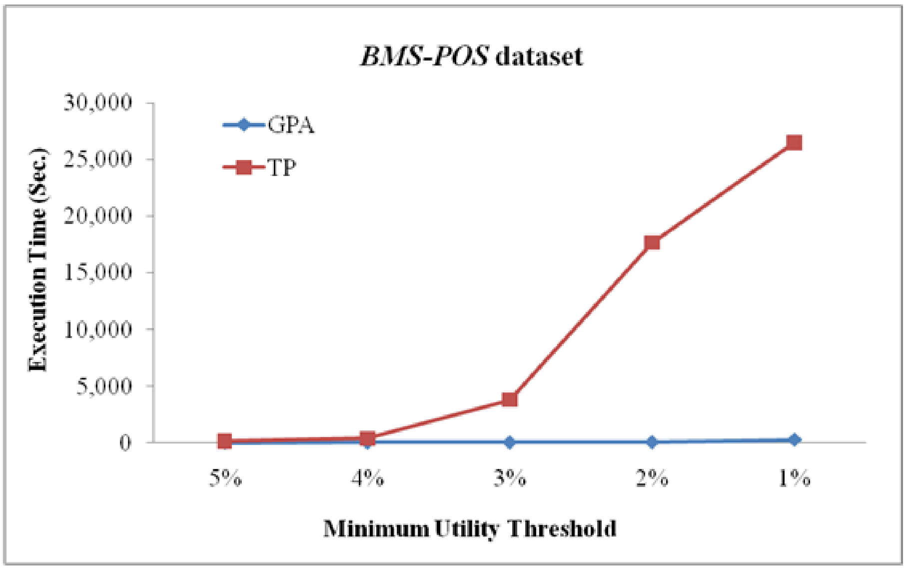


FIGURE 4. Efficiency of the two algorithms along with different thresholds

as follows. Firstly, a level-wise pruning algorithm is proposed to obtain more accurate upper bounds of utility values for itemsets, thus reducing the number of unpromising candidates for mining. Secondly, the data size in each pass can be gradually reduced, and the data scan cost can thus be saved. Finally, the upper bounds of utility values and actual utility values of candidates can be simultaneously found in the process, and thus an extra additional scanning is not needed. The experimental results also show that the pruning strategy has a good effect in pruning unpromising itemsets, and the proposed algorithm also outperforms the two-phase utility mining algorithm for both the synthetic datasets and a real dataset.

REFERENCES

[1] R. Agrawal and R. Srikant, Fast algorithm for mining association rules, *Proc. of the International Conf. on Very Large Data Bases*, pp.487-499, 1994.

[2] R. Agrawal, T. Imielinksi and A. Swami, Mining association rules between sets of items in large database, *Proc. of the ACM SIGMOD International Conf. on Management of Data*, pp.207-216, 1993.
[3] R. Chan, Q. Yang and Y. Shen, Mining high utility itemsets, *Proc. of the 3rd IEEE International Conf. on Data Mining*, pp.19-26, 2003.
[4] C.-J. Chu, V. S. Tseng and T. Liang, Mining temporal rare utility itemsets in large databases using relative utility thresholds, *International Journal of Innovative Computing, Information and Control*, vol.4, no.11, pp.2775-2792, 2008.
[5] A. Erwin, R. P. Gopalan and N. R. Achuthan, CTU-mine – An efficient high utility itemset mining algorithm using the pattern growth approach, *Proc. of the 7th IEEE International Conf. on Computer and Information Technology*, pp.71-76, 2007.
[6] *Frequent Itemsets Mining Dataset Repository*, http://fimi.cs.helsinki.fi/data/.
[7] J. Hu and A. Mojsilovic, High-utility pattern mining: A method for discovery of high-utility item sets, *Pattern Recognition*, vol.40, no.11, pp.3317-3324, 2007.
[8] IBM Quest Data Mining Project, *Quest Synthetic Data Generation Code*, http://www.almaden.ibm.com/cs/quest/syndata.html.
[9] G. C. Lan, T. P. Hong and V. S. Tseng, Discovery of high utility itemsets from on-shelf time periods of products, *Expert Systems with Application*, vol.38, no.5, pp.5851-5857, 2011.
[10] G.-C. Lan, T.-P. Hong and V. S. Tseng, Enhancing the pruning performance in on-shelf utility mining, *ICIC Express Letters*, vol.5, no.10, pp.3749-3754, 2011.
[11] Y. C. Li, J. S. Yeh and C. C. Chang, Isolated items discarding strategy for discovering high utility itemsets, *Data & Knowledge Engineering*, vol.64, no.1, pp.198-217, 2008.
[12] Y. Liu, W. Liao and A. Choudhary, A fast high utility itemsets mining algorithm, *Proc. of the Utility-Based Data Mining Workshop*, pp.90-99, 2005.
[13] V. S. Tseng, C. J. Chu and T. Liang, Efficient mining of temporal high utility itemsets from data streams, *Proc. of the ACM KDD Workshop on Utility-Based Data Mining*, pp.1105-1117, 2006.
[14] H. Yao and H. J. Hamilton, Mining itemset utilities from transaction databases, *Data & Knowledge Engineering*, vol.59, no.3, pp.603-626, 2006.
[15] J. S. Yeh, C. Y. Chang and Y. T. Wang, Efficient algorithms for incremental utility mining, *Proc. of Ubiquitous Information Management and Communication*, pp.229-234, 2008.
[16] G. Yu, K. Li and S. Shao, Mining high utility itemsets in large high dimensional data, *Proc. of the International Workshop on Knowledge Discovery and Data Mining*, pp.17-20, 2008.