Discovery of High Utility Itemsets from On-shelf Time Periods of Products

Guo-Cheng Lan[a], Tzung-Pei Hong[b, c]*, and Vincent S. Tseng[a]

[a]Department of Computer Science and Information Engineering
National Cheng Kung University, 701, Taiwan
[b]Department of Computer Science and Information Engineering
National University of Kaohsiung, 811, Taiwan
[c]Department of Computer Science and Engineering
National Sun Yat-Sen University, 804, Taiwan
rrfoheiay@idb.csie.ncku.edu.tw, tphong@nuk.edu.tw, tsengsm@mail.ncku.edu.tw

Abstract

Utility mining has recently been an emerging topic in the field of data mining. It finds out high utility itemsets by considering both the profits and quantities of items in transactions. It may have a bias if items are not always on shelf. In this paper, we thus design a new kind of patterns, named high on-shelf utility itemsets, which considers not only individual profit and quantity of each item in a transaction but also common on-shelf time periods of a product combination. We also propose a two-phased mining algorithm to effectively and efficiently discover high on-shelf utility itemsets. In the first phase, the possible candidate on-shelf utility itemsets within each time period are found level by level. In the second phase, the candidate on-shelf utility itemsets is further checked for their actual utility values by an additional database scan. At last, the experimental results on synthetic datasets also show the proposed approach has a good performance.

Keyword: Data mining; Utility mining; High utility itemsets; On-shelf data.

---

*corresponding author

1. Introduction

Mining association rules [1] is an important issue in the field of data mining due to its wide applications. Agrawal *et al*. first proposed the most well-known algorithm, namely Apriori, for mining association rules from a transaction database [1]. Traditional association rules are, however, derived from frequent itemsets, which only consider the occurrence of items but do not reflect any other factors, such as price or profit. The actual significance of an itemset cannot be easily recognized in this way since the same significance is assumed for all the items in a database. Chan *et al*. thus proposed the utility mining to solve the problem [4]. They considered both individual profits and quantities of products (items) in transactions, and used them to find out actual utility values of itemsets. The high utility itemsets, which had their utility values larger than or equal to a predefined threshold, were then found as the desired. Several other researches about utility mining were proposed in these years, most of which emphasized on how to efficiently find out the high utility itemsets from databases [6][7][11][16][18][19].

Temporal data mining has emerged and attracted much attention in these years because of its practicality [3][5][10][14]. For example, assume there is an association rule like “In the winter, customers usually purchase overcoats and stockings together”. The itemset {overcoats, stockings} may be not frequent throughout the entire database, but may be with a high frequency in the winter. Mining time-related

knowledge is thus interesting and useful.

In this paper, we considered on-shelf time of products, which is a specific research field in temporal data mining. It is important because if an entire database, instead of the transactions in on-shelf time periods, is used for mining, some biases may exist in the association rules discovered. In the past, Ale *et al*. considered the first and the last transaction time of a product but not its actual on-shelf time period in a store [5]. Besides, Chang *et al*. discovered temporal association rules from a database by considering the exhibition period of a product in a store, which is defined as the interval between its first and last on-shelf time periods [3]. They didn't consider a product might be intermittently on and off the shelf in the time interval. Lee *et al*. then applied the above concept to an actual application of book store transactions[9].

In this paper, we thus introduce a new kind of patterns, called high on-shelf utility itemsets, which consider not only individual profits and quantities of products in transactions, but also actual on-shelf time periods of products. The whole time interval to be analyzed is split into several time periods, and any itemset with a high utility value within the union of all its on-shelf time periods is thought of as a high on-shelf utility itemset. Since utility mining is usually much more complicated than traditional association-rule mining, a two-phased mining algorithm is thus proposed for effectively and efficiently finding the high on-shelf utility itemsets from a

database.

The mining process of the proposed algorithm can be divided into two phases. In the first phase, the possible candidate on-shelf utility itemsets within each time period are found level by level. In the second phase, the candidate on-shelf utility itemsets are further checked for their actual utility values by an additional database scan. Finally, the itemsets with their actual utility values larger than or equal to a predefined threshold are output as the high on-shelf utility itemsets.

The remaining parts of this paper are organized as follows. The related works are reviewed in Section 2. The problem to be solved is defined in Section 3, with the tables required in the proposed approach being described as well. The proposed two-phased mining algorithm for finding high on-shelf utility itemsets from a database is stated in Section 4. An example to illustrate the process of the proposed algorithm is demonstrated in Section 5. The experimental evaluation is shown in Section 6. Conclusions and future works are finally given in Section 7.

2. Review of Related Works

In this section, some related researches are briefly reviewed. They are temporal association-rule mining and utility mining.

2. 1 Temporal association-rule mining

Temporal data mining is concerned with the analysis of temporal data. It is proposed to find out temporal patterns and regularity from a set of data with time. Temporal patterns can be discovered in a variety of forms, like sequential association rules [2], periodical association rules [15], cyclic association rules [13], and calendar association rules [12]. Temporal data mining is also called dynamic data mining.

As to consideration of on-shelf time of products, Ale *et al*. proposed a mining approach to find temporal association rules. They, however, only considered the transaction periods of products [3], but not their exhibition periods. In the association rules discovered, the products might be not frequent itemsets if only their transaction periods were considered. The rules might thus provide not so accurate information to decision makers such that they could not make appropriate promotion strategies.

Chang *et al*. discovered temporal association rules from a database by considering the common exhibition period of a product in a store, which was defined as the interval between its first and last on-shelf time periods [5]. Here the common exhibition period indicates that all items within a product combination in a store have to be put simultaneously on shelf within the period. Lee *et al*. then applied the concept of common exhibition periods to publication databases to discover general temporal association rules [9]. They did not consider that a product might be put on shelf and taken off shelf multiple times in a store.

As described above, a bias might exist in the temporal association rules. As a matter of fact, the on-shelf and off-shelf actions of products are usually done multiple times in a store throughout the whole period of gathering data. In this paper, the individual on-shelf time periods are considered, instead of the whole exhibition period.

2. 2 Utility Mining

In association-rule mining, only binary itemsets are considered. In real-world applications, however, products bought in transactions may contain profits and quantities. Especially, some high-profit products may occur with low frequencies in a database. For example, jewel and diamond are high utility items but may not be frequent when compared to food or drink in a database. The high-profit but low-frequency itemsets may not be found in traditional association-rule mining approaches. Hence, Chan *et al.* proposed utility mining to discover high utility itemsets [4]. They considered not only the quantities of the items in a product combination but also their 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 dataset, like the quantity of the item sold in a transaction. The external utility of an item is given by

users, like its profit. External utility often reflects user preference 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 will better match a user's expectations than by only considering the transaction dataset itself.

Traditional association-rule mining keeps the downward-closure property, but utility mining does not. Therefore, utility mining is much harder than traditional association-rule mining. Liu *et al.* proposed a two-phase algorithm to discover high utility itemsets from a database by adopting the downward-closure property [11]. They named their approach as the transaction-weighted-utilization (TWU) model. It used the whole utility of a transaction as the upper bound of an itemset in that transaction to keep the downward-closure property. It consisted of two phases. In phase 1, the model found out possible candidate itemsets from a database. In phase 2, the database was rescanned again to update the actual utility of the possible candidate itemsets and found the ones with their actual utility values larger than or equal to a predefined threshold (called the minimum utility threshold). Several other algorithms for utility mining were also proposed [7][18][19], and some related studies are still in progress [6][16].

As mentioned above, there has been no study for discovering high on-shelf utility itemsets in databases yet. This motivates our exploration of the issue of

efficiently mining high on-shelf utility itemsets in databases.

3. Problem Definition

In this paper, we introduce a new kind of patterns, called high on-shelf utility itemsets, which consider not only individual profits and quantities of products in transactions, but also actual on-shelf time periods of products. The problem of mining high on-shelf utility itemsets is defined as follows. Assume a database contains a number of transactions, and each transaction is recorded with the occurring time and the purchased items with corresponding quantities. Also assume that the whole time interval to be analyzed is split into several time periods. Besides, a utility table with the profits of the items and an on-shelf table with the on-shelf information of the items are also given. The problem is to find the itemsets with their high utility values within the union of their on-shelf time periods are larger than or equal to a predefined minimum utility threshold. We thus propose an effective algorithm to solve the above problem.

Next the OS table is described below. In this paper, we assume that the information about whether an item is on shelf of a store within a period is known. A table, called the OS (On Shelf) table, is used to keep this information for all items. For example, assume there are three time periods and six products for sale in a store.

Table 1 shows the on-shelf information of the six products *A*, *B*, *C*, *D*, *E* and *F* in the periods, where '1' represents "on shelf" and '0' represents "off shelf".

Table 1: An example of the OS table.

Item \ ***Period***	t_1	t_2	t_3
A	1	1	1
B	1	1	0
C	1	1	1
D	0	1	1
E	1	0	1
F	1	0	1

It is very easy to extend the OS table from an item to an itemset. The AND operation can be used to achieve the purpose. For example, the on-shelf and the off-shelf time periods of products *A*, *B* and *D* are represented by the bit strings as (1, 1, 1), (1, 1, 0) and (0, 1, 1), respectively. By the AND operation on them, the common sale periods for the itemset {*ABD*} on the shelf at the store is (0, 1, 0). It represents the time period for all the three products on the shelf is only t_2.

4. The Proposed Mining Algorithm

In this section, we describe the proposed two-phased mining algorithm for discovering high on-shelf utility itemsets in a database. A high on-shelf utility itemset

is the one with its sum of utilities in all on-shelf periods larger than or equal to a threshold. Besides, the PTTU table is used in the proposed algorithm to help the execution of the algorithm. The PTTU table is first described below.

4. 1 The PTTU Table

The PTTU (Periodical Total Transaction Utility) table is designed in the algorithm to increase the execution efficiency. Let the transaction utility of a transaction is the sum of the profits of the items contained in it multiplied by their quantities. An entry in the table records the periodical total transaction utility of all the transactions occurring within a time period. For example, assume there are four transactions which all occur at the time period t_3. Also assume the transaction utility values of the transactions are 15, 18, 16 and 16, respectively. The sum of the transaction utilities for the four transactions within the time period t_3 is thus 65. It is then filled into the corresponding entry ($pttu_3$) of the PTTU table. An example of the PTTU table is shown in Table 2.

Table 2: An example of the PTTU table.

Period	***Periodical Total Transaction Utility***
t_1	104
t_2	48
t_3	65

4. 2 The proposed algorithm for finding high on-shelf utility itemsets

The proposed algorithm for finding high on-shelf utility itemsets is described in this section. Its execution process can be divided into two phases. Phase 1 first finds all the possible candidate on-shelf utility itemsets within each time period from a database. Next, phase 2 scans the database again to find the actual utility values of the possible candidate on-shelf utility itemsets within the union of their on-shelf time periods and judge whether they are high on-shelf utility itemsets (HOUI). The details of the algorithm are listed below.

The proposed two-phased mining algorithm for high on-shelf utility itemsets:

INPUT: A transaction database D with n transactions, each of which consists of transaction identification, transaction occurring time and items purchased, m items in D, each with a profit value, an on-shelf table of items with k desired time periods, and the minimum utility threshold λ.

OUTPUT: The set of high on-shelf utility itemsets (*HOUI*).

Phase 1:

Step 1: Transform the occurring time of each transaction into the corresponding time period.

Step 2: Initialize the PTTU (Periodical Total Transaction Utility) table as a zero table,

in which the row number is the time period number and each entry in the table is set as 0.

Step 3: For each *y*-th transaction $Trans_{jy}$ in the set of transactions D_j in each time period t_j, do the following substeps.

(a) Calculate the utility value u_{jyz} of each *z*-th item I_{jyz} in $Trans_{jy}$ as:

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

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

(b) Calculate the transaction utility tu_{jy} of the transaction $Trans_{jy}$ as:

$$tu_{jy} = \sum_{z} u_{jyz} .$$

Step 4: Calculate the periodical total transaction utility $pttu_j$ in each time period t_j as:

$$pttu_j = \sum_{y} tu_{jy} .$$

Step 5: Find in the PTTU table the entry which has the same occurring time period as each t_j, and add $pttu_j$ to the entry.

Step 6: Find the itemsets with high utility values in each time period t_j by the Finding-Individual-HUI procedure. Let the set of returned high utility itemsets as HUI_j and the union of all HUI_j's for all time periods as *HUI.*

Phase 2:

Step 7: Initially set the set of high on-shelf utility itemsets *HOUI* as empty.

Step 8: For each itemset *X* existing in *HUI*, find its actual on-shelf utility by the following substeps:

(a) Use the AND operation to obtain the set of common on-shelf periods (COS_X) of all the items in the itemset *X* from the OS table.

(b) Sum the actual utilities $u(X)^{Appearing}$ of *X* appearing in *HUI* as follows:

$$u(X)^{Appearing} = \sum_{X \in HUI \wedge t_j \in COS_X} u_j(X),$$

where $u_j(X)$ is the actual utility of itemset *X* in the time period t_j.

Step 9: For each itemset *X* in *HUI*, scan the database to find its actual utility value $u(X)_j^{scan}$ in a time period $t_j \in COS_x$ if *X* does not appear in that time period.

Step 10: Calculate the actual on-shelf utility $u(X)^{actual}$ of each itemset *X* in *HUI* as:

$$u(X)^{actual} = u(X)^{Appearing} + \sum_{X \in HUI \wedge X \notin HUI_j \wedge t_j \in COS_X} u(X)_j^{scan}.$$

Step 11: If $u(X)^{Actual} / pttu(X) \geqq \lambda$, then *X* is a high on-shelf utility itemset; set $HOUI = HOUI \cup \{X\}$ and $HUI = HUI - \{X\}$; Otherwise, set $HUI = HUI - \{X\}$.

Step 12: Output the set of high on-shelf utility itemsets in *HOUI*.

After Step 12, all the high on-shelf utility itemsets are found. The Finding-Individual-HUI procedure used in Step 6 is described below. It is based on Liu *et al.*'s approach for finding high utility itemsets [11].

Let the transaction-weighted-utility of an itemset X in a time period t is the sum of the transaction utilities of all the transactions containing X in t. If the transaction-weighted-utility of an itemset over the summation all transaction utilities in t is larger than or equal to the predefined threshold λ, it is called a high transaction-weighted-utilization (HTWU) itemset in t. Since the transaction-weighted-utility of an itemset is always larger than or equal to the actual utility value of the itemset, a high transaction-weighted-utilization itemset is only a possible high utility itemset in t. Moreover, a high utility itemset in at least one time period is a possible candidate for being a high on-shelf utility itemset. The actual utility values of a possible high on-shelf utility itemset in all its on-shelf periods are further obtained to determine whether it is. Based on the above concepts, the procedure is stated below.

The Finding-Individual-HUI procedure:

Input: The set of transactions D_j in a time period t_j.

Output: The high utility itemsets HUI_j.

Step 1: Set $r = 1$, where r represents the number of items in the current set of

candidate transaction-weighted-utilization itemsets (C_{jr}) to be processed.

Step 2: Initially set C_{jr} as the set of all the items in the time period t_j.

Step 3: For each appearing candidate r-itemset X_{jyz} in C_{jr} in a transaction $Trans_{jy}$, set the transaction utility tu_{jyz} of X as:

$$tu_{jyz} = tu_{jy} .$$

where tu_{jy} is the transaction utility of the transaction $Trans_{jy}$.

Step 4: Calculate the transaction-weighted-utility twu_{jz} of each r-itemset X_{jz} in each time period t_j as the sum of the transaction utility values of X_{jz} in all the transactions in t_j. That is,

$$twu_{jz} = \sum_{y} tu_{jyz}.$$

Step 5: Calculate the actual utility u_{jz} of each candidate r-itemset X_{jz} in each time period t_j as:

$$u_{jz} = \sum_{y} u_{jyz} ,$$

where u_{jyz} is the utility value of the r-itemset X_{jz} in the transaction $Trans_{jy}$, which is the sum of the actual utility values of the items in X_{jz}.

Step 6: Check whether the transaction-weighted-utility twu_{jz} of each candidate r-itemset X_{jz} in each time period t_j exceeds or equals to the threshold of $\lambda * pttu_j$ within the time period t_j. If it is, put it into the set of high

transaction-weighted-utilization *r*-itemsets $HTWUI_{jr}$ for t_j. That is:

$$HTWUI_{jr} = \{X_{jz} \mid twu_{jz} \geq \lambda * pttu_j \ \ for\ the\ time\ period\ t_j\}.$$

Step 7: Check whether the actual utility u_{jz} of each candidate *r*-itemset X_{jz} in each time period t_j exceeds or equals to the threshold of $\lambda * pttu_j$ within the time period t_j. If it is, put it with its actual utility u_{jz} into the set of HUI_{jr}. That is:

$$HUI_{jr} = \{(X_{jz}, u_{jz}) \mid u_{jz} \geq \lambda * pttu_j \ \ for\ the\ time\ period\ t_j\}.$$

Step 8: Generate the candidate set $C_{j(r+1)}$ from $HTWUI_{jr}$ in the current time period t_j. The *r*-subitemsets in each candidate in $C_{j(r+1)}$ must exist in $HTWUI_{jr}$.

Step 9: If $HTWUI_r$ is not null, set $r = r + 1$ and repeat Steps 1 to 9; otherwise, return the set of high utility itemsets HUI_j *with their actual utility values.*

5. An Example

In this section, an example is given to show how the proposed algorithm can be easily used to find out the high on-shelf utility itemsets in a database *D*. Assume there are twelve transactions, each of which consists of three features, transaction identification (*TID*), transaction occurring time and items purchased. The occurring time of each transaction is first transformed into the corresponding time period. Assume the data after the transformation are shown in Table 3 and are used for mining, where the numbers represent the purchased quantities. Also, assume the on-shelf

information is the same as that in Table 1 and the individual profit of each item is shown in Table 4. Moreover, the predefined threshold λ is set at 30%. To find the high on-shelf utility itemsets from *D* in Table 3, the proposed mining algorithm proceeds as follows.

Table 3: The set of transaction data in the example.

Period	***TID***	***A***	***B***	***C***	***D***	***E***	***F***
t_1	$Trans_1$	1	0	0	0	1	0
	$Trans_2$	0	2	25	0	0	2
	$Trans_3$	1	1	0	0	0	1
	$Trans_4$	0	2	12	0	0	0
t_2	$Trans_5$	2	0	8	0	0	0
	$Trans_6$	0	0	4	1	0	0
	$Trans_7$	1	0	2	1	0	0
	$Trans_8$	1	1	0	0	0	0
t_3	$Trans_9$	3	0	0	1	0	0
	$Trans_{10}$	0	0	0	1	2	1
	$Trans_{11}$	0	0	4	1	0	3
	$Trans_{12}$	1	0	2	1	1	0

Table 4: The utility table for this example.

Item	***Profit***
A	3
B	10
C	1
D	6
E	5
F	2

Step 1: The occurring time of each transaction is first transformed into the

corresponding time period. In Table 3, we assume the transaction time has been transformed.

Step 2: The PTTU (Periodical Total Transaction Utility) table is initialized as the zero table, in which the value of each entry is 0. Since there are three time periods in the example, only three rows are generated in the PTTU table and all the *pttu* values are 0.

Step 3: For each transaction in the data set D_j within each time period t_j, the utility value of each item in it is first calculated. Take the first transaction $Trans_1$ in the time period t_1 as an example. The transaction includes the two items {*A*, *E*}. Since the profit of the item *A* is 1 and its quantity is 3, its utility is thus 1*3, which is equal to 3. Similarly, the utility value of the item *E* can be calculated as 5. The transaction utility of the first transaction is then the sum of the utility values of the two items, which is 3 + 5 (= 8). The results for all the transaction utility values of all the transactions are shown in Table 5.

Table 5: The transaction utility values of all the transactions.

Period	***TID***	***A***	***B***	***C***	***D***	***E***	***F***	***tu(*$Trans_{jy}$*)***
t_1	$Trans_1$	1	0	0	0	1	0	8
	$Trans_2$	0	2	25	0	0	2	49
	$Trans_3$	1	1	0	0	0	1	15
	$Trans_4$	0	2	12	0	0	0	32
	$Trans_5$	2	0	8	0	0	0	14
	$Trans_6$	0	0	4	1	0	0	10

t_2	$Trans_7$	1	0	2	1	0	0	11
	$Trans_8$	1	1	0	0	0	0	13
t_3	$Trans_9$	3	0	0	1	0	0	15
	$Trans_{10}$	0	0	0	1	2	1	18
	$Trans_{11}$	0	0	4	1	0	3	16
	$Trans_{12}$	1	0	2	1	1	0	16

Step 4: The periodical total transaction utility of each time period is calculated as the summation of the transaction utility values in the period. Take the first time period t_1 as an example. It includes four transactions, and their transaction utility values are 8, 49, 15 and 32, respectively. The periodical total transaction utility of t_1 is then 8 + 49 + 15 + 32, which is 104. Similarly, the periodical total transaction utility values of t_2 and t_3 are 48 and 65, respectively.

Step 5: The periodical total transaction utility in each time period t_j is added to the corresponding entry ($pttu_j$) in the PTTU table. The completed PTTU table is shown in Table 6.

Table 6: The completed PTTU table.

Period	***Periodical Total Transaction Utility***
t_1	104
t_2	48
t_3	65

Step 6: The itemsets with high utility values in each time period t_j are found by

the Finding-Individual-HUI procedure. Take the four transactions in the time period t_1 as an example. The candidate transaction-weighted-utilization *1*-itemsets in the time period t_1 include the five items $\{A\}$, $\{B\}$, $\{C\}$, $\{E\}$ and $\{F\}$. The transaction-weighted-utility value and the actual utility value of each of the *1*-itemsets are simultaneously calculated, with the results shown in Table 7.

Table 7: The results of the *1*-itemsets in t_1.

1-itemset	***Transaction-weighted Utility***	***Actual Utility***
{A}	23	6
{B}	96	50
{C}	81	37
{E}	8	5
{F}	64	6

The periodical total transaction utility of t_1 is 104. The utility threshold in the time period t_1 can thus be calculated as 30%*104, which is about 32. The transaction-weighted-utility values of the above candidate *1*-itemsets are then checked against the threshold. Only the three candidate *1*-itemsets $\{B\}$, $\{C\}$ and $\{F\}$ satisfy the threshold condition and are thought of as high transaction-weighted-utilization *1*-itemsets. The actual utility values of the three high transaction-weighted-utilization *1*-itemsets $\{B\}$, $\{C\}$ and $\{F\}$ are further checked against the threshold. Only the two high transaction-weighted-utilization *1*-itemsets $\{B\}$ and $\{C\}$ satisfy the threshold condition and are thought of as high utility *1*-itemsets. They are then put in the set of

HUI_{11} for later usage. Keeping high transaction-weighted-utilization *1*-itemsets, instead of only high utility*1*-itemsets, is for the downward closure property. Next, the candidate *2*-itemsets are generated from the high transaction-weighted-utilization *1*-itemsets, and the above process is performed again to find the high transaction-weighted-utilization *2*-itemsets. In this example, the high transaction-weighted-utilization *2*-itemsets are {*BC*}, {*BF*} and {*CF*}. Among them, only {*BC*} and {*BF*} are high utility *2*-itemsets and are put in the set of HUI_{12}. The above process is repeatedly executed level by level until no candidate itemsets are generated. In this example, there is only a high transaction-weighted-utilization *3*-itemset {*B*, *C*, *F*} and it is also a high utility itemset. It is thus put in the set of HUI_{13}. In this example, no candidate *4*-itemset is generated. The final *HUI* set from all the three time periods is shown in Table 8, where the subscript in the field of the periodical utility represents the time period in which the corresponding itemset is a high utility itemset.

Table 8: The final set of *HUI*.

Itemset	***Periodical Utility***	***Itemset***	***Periodical Utility***
{*B*}	[50_1]	{*BF*}	[36_1]
{*C*}	[37_1]	{*CD*}	[18_2]
{*D*}	[24_3]	{*DE*}	[27_3]
{*AC*}	[19_2]	{*DF*}	[20_3]
{*AD*}	[24_3]	{*BCF*}	[49_1]
{*BC*}	[77_1]		

Step 7: The set HOUI is initialized as empty, which is used to maintain the high on-shelf utility itemsets and their actual utility values.

Step 8: For each itemset X in the *HUI* set, its appearing actual utility is first calculated in all its common on-shelf time periods. Take the itemset {AC} as an example. The common periods of both the items in {AC} are t_1, t_2 and t_3 from the OS table by the AND operation on the tuples of A and C, but it is a high utility *2*-itemset only in t_2. Its appearing actual utility is thus 19. The appearing actual utility values for the other itemsets in the set of *HUI* can be similarly found.

Step 9: This step is to find all the high on-shelf utility itemsets from the *HUI set*. Appropriate database rescan is needed for finding the actual on-shelf utility values of the itemsets kept in the set of *HUI* if necessary. Take the itemset {BC} in the set of *HUI* as an example. Its on-shelf time periods are t_1.and t_2, but only its actual utility value in t_1 is known. The four transactions in t_2 are thus rescanned to find the actual utility value in this period. In this example, the actual utility value of {BC} in t_2 is 0. All the other itemsets in the *HUI* set can be similarly processed. After this step, all the actual on-shelf utility values of a candidate itemset in its on-shelf time periods are known.

Step 10: The total actual on-shelf utility value of each candidate itemset in *HUI*

is calculated. Take the itemset {*BC*} in the set of *HUI* as an example. Its total on-shelf utility is easily calculated as 77 + 0, which is 77.

Step 11: The total on-shelf utility ratio of each candidate itemset in *HUI* is then compared with the utility threshold. Again, take {*BC*} as an example. Its common on-shelf periods are t_1 and t_2 with periodical total utility values being 104 and 48, respectively. The total on-shelf utility ratio of {*BC*} is thus calculated as 77/(104+48), which is 50.66%. Since the ratio is larger than the utility threshold (= 30%), the itemset {*BC*} is put into the *HOUI* set with its actual utility values in its on-shelf time periods. The final results are shown in Table 9.

Table 9: The final *HOUI* set.

Itemset	***Part of the Actual On-shelf Utility Ratio***	***Periodical Utility***
B	0.3871	[50, 10]
D	0.3130	[12, 24]
BF	0.3364	[36]
BCF	0.4205	[49]
CD	0.3826	[18, 18]
BC	0.4967	[77, 0]

Step 12: In this example, the six high on-shelf utility itemsets in the *HOUI* set are output as decision makers' auxiliary information.

6. Experimental Evaluation

A series of experiments was conducted to compare the effects of the traditional and the proposed patterns and to evaluate the performance of the proposed mining algorithm with different parameter values. The experiments were implemented in J2SDK 1.5.0 and executed on a PC with 3.0 GHz CPU and 1GB memory.

6. 1 Experimental Datasets

The public IBM data generator was first used in our experiments [8]. However, since our objective was to discover high on-shelf utility itemsets, we also developed a simulation model, which was similar to that used in [11], to generate the quantities of the items in the transactions. Each quantity ranged among 1 to 5 according to the way in [11]. Besides, an integer value among 1 to the total number of given time periods was randomly assigned to each transaction in the generated datasets. Moreover, for each dataset generated, a corresponding utility table was also generated in which a profit value lying among 0.01 to 10.00 was randomly assigned to an item. Figure 1 shows the profit value distribution of all the items generated by the simulation model in the utility table.

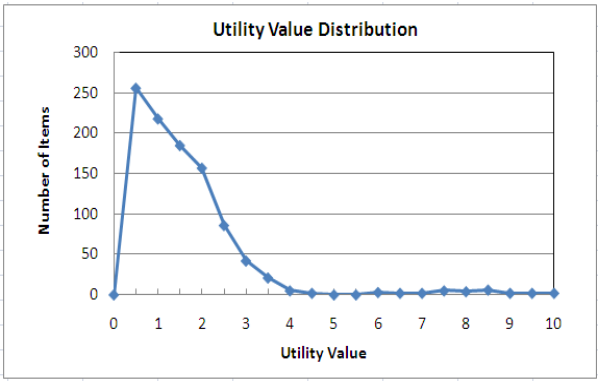

Figure 1: The profit value distribution in the synthetic data set.

6. 2 Experimental Results

Experiments were made on the synthetic T10I4N4KD200K dataset to evaluate the difference between the high utility itemsets with and without considering common on-shelf time periods. The results were shown in Figure 2, where the proposed high on-shelf utility itemsets were represented as *HOUI* and the traditional high utility itemsets were represented as *HUI.* Besides, *HOUI_P5* represented using the proposed approach with 5 time periods.

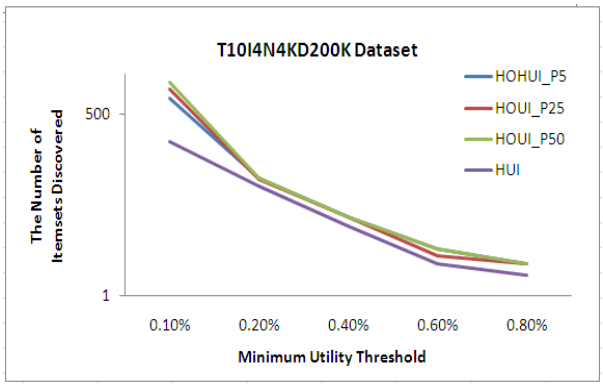

Figure 2: The number of high on-shelf utility itemsets discovered under different minimum utility thresholds

It could be observed from the figure that the number of high utility itemsets was always less that of high on-shelf utility itemsets. It could be easily explained as follows. A high on-shelf utility itemset always had the same utility value as its corresponding traditional high utility itemset, but the former considered a less number of transactions than the latter. A high on-shelf utility itemset was thus easier to satisfy the minimum utility threshold than its corresponding traditional high utility itemset. The traditional high utility itemsets were thus included in the set of the proposed high on-shelf utility itemsets under the same parameter settings. Besides, the number of high on-shelf utility itemsets would increase along with the number of time periods. We then evaluated the number change rate, which was defined as follows:

$$Number\ Change\ Rate = \frac{|THUI| - |HOUI|}{|OHUI|}.$$

The experimental results for different minimum support thresholds are shown in Figure 3.

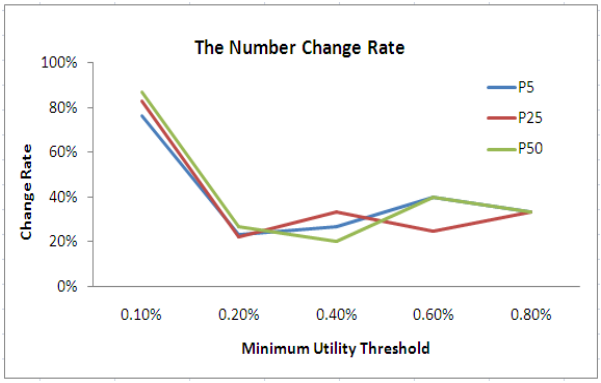

Figure 3: The number change rates along with different minimum support thresholds

It could be observed from the figure that the number change rate varied along with different minimum utility thresholds and different numbers of time periods, but at least 20% change rates were achieved. It meant that considering common on-shelf periods could discover a bigger number of high utility itemsets with more centralized utility values than not considering.

Experiments were at last made to evaluate the efficiency of the proposed mining algorithm. Figure 4 showed the execution time on the T10I4N4KD200K dataset for different thresholds varying from 0.10% to 0.80%, and Figure 5 showed the execution time on the T10I4N4K dataset for different sizes varying from 100K to 500K when λ was set at 0.2%.

It could be easily observed that the execution time of the proposed mining algorithm increased nearly linearly with the value of threshold or the size of the database in those two figures.

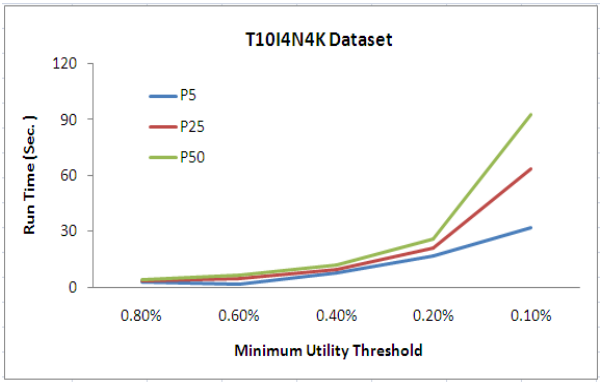

Figure 4: The execution time of the proposed algorithm under different thresholds.

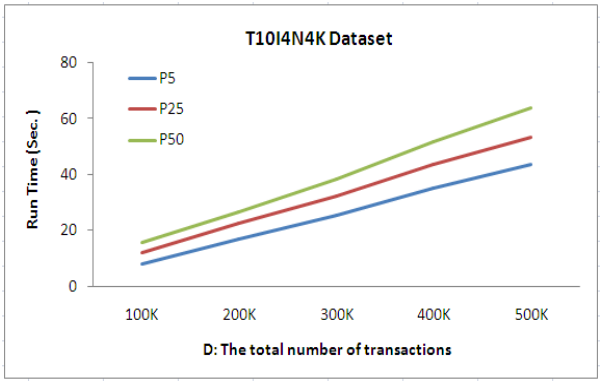

Figure 5: The execution time of the proposed algorithm under different data sizes.

7. Conclusions

In this paper, we have proposed the high on-shelf utility itemsets, which consider the common on-shelf time periods for items. We have also proposed the mining algorithm to efficiently discover the desired itemsets from a database. The experimental results show that the proposed high common on-shelf utility patterns have a good effect when compared to the traditional utility patterns. In the future, we would apply the proposed knowledge type and approach to some practical applications, such as data stream, supermarket promotion applications, and among others.

References

[1] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules," *The International Conference on Very Large Data Bases*, pp. 487-499, 1994.

[2] R. Agrawal, R. Srikant and Q. Vu, “Mining association rules with item constraints,” *The 3th International Conference on Knowledge Discovery in Databases and Data Mining*, Newport Beach, California, 1997.

[3] J. M. Ale and G. H. Rossi, “An approach to discovering temporal association rules,” *The 2000 ACM Symposium on Applied Computing*, pp. 294-300, 2000.

[4] R. Chan, Q. Yang, and Y. Shen, “Mining high utility Itemsets,” *The Third IEEE International Conference on Data Mining*, pp. 19-26, 2003.

[5] C. Y. Chang, M. S. Chen, and C. H. Lee, “Mining general temporal association rules for items with different exhibition periods,” *The Third IEEE International Conference on Data Mining*, pp. 59-66, 2002.

[6] C. J. Chu, Vincent 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, 8, 2008.

[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, 1996.

[9] C. H. Lee, C. R. Lin, and M. S. Chen, “On mining general temporal association

rules in a publication database," *The 2001 IEEE International Conference on Data Mining*, pp. 337-344, 2001.

[10] Y. Li, P. Ning, X. S. Wang, and S. Jajodia, "Discovering calendar-based temporal association rules," *Data & Knowledge Engineering*, Vol. 44, No. 2, pp. 193-218, 2003.

[11] Y. Liu, W. Liao, and A. Choudhary, "A fast high utility itemsets mining algorithm," *The Utility-Based Data Mining Workshop*, pp. 90-99, 2005.

[12] H. Mannila, H. Toivonen and A. I. Verkamo, "Efficient algorithm for discovering association rules," *The AAAI Workshop on Knowledge Discovery in Databases*, pp. 181-192, 1994.

[13] B. Ozden, S. Ramaswamy and A. Silberschatz. "Cyclic association rules," *The 14th International Conference on Data Engineering*, Orlando, Florida, USA, pp. 12-421, 1998.

[14] J. F. Roddick and M. Spiliopoilou, "A survey of temporal knowledge discovery paradigms and methods," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 14, No. 4, pp. 750-767, 2002.

[15] R. Srikant and R. Agrawal, "Mining generalized association rules," *The 21st International Conference on Very Large Data Bases*, Zurich, Swizerland, pp. 407-419, 1995.

[16] Vincent S. Tseng, C. J. Chu, and T. Liang, “Efficient mining of temporal high utility itemsets from data streams,” *The ACM KDD Workshop on Utility-Based Data Mining*, 2006.

[17] C. Y. Wang, S. S. Tseng, and T. P. Hong, “Flexible online association rule mining based on multidimensional pattern relations,” *Information Science*, Vol. 176, No. 12, pp. 1752-1780, 2006.

[18] H. Yao and H. J. Hamilton, “Mining itemset utilities from transaction databases,” *Data & Knowledge Engineering*, Vol. 59, No. 3, pp. 603-626, 2006.

[19] H. Yao, H.J. Hamilton, and C.J. Butz, “A foundational approach to mining itemset utilities from databases,” *The 4th SIAM International Conference on Data Mining*, pp. 482-486, 2004.