A Three-Scan Mining Algorithm for High On-Shelf Utility Itemsets

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

[1]Department of Computer Science and Information Engineering
National Cheng-Kung University
Tainan, 701, Taiwan, ROC.
rrfoheiay@idb.csie.ncku.edu.tw; tsengsm@mail.ncku.edu.tw

[2]Department of Computer Science and Information Engineering
National University of Kaohsiung
Kaohsiung, 811, Taiwan, ROC.
tphong@nuk.edu.tw

[3]Department of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, 804, Taiwan, ROC.

ABSTRACT. *In this paper, we introduce a new kind of patterns, named high on-shelf utility itemsets, which consider not only individual profit and quantity of each item in a transaction but also on-shelf periods in a database. We have thus proposed a 3-scan mining algorithm to efficiently discover the itemsets. The proposed approach adopts a new pruning strategy and an itemset-generation mechanism to prune redundant candidate itemsets early and to systematically check the itemsets from transactions. Experimental results also show its performance.*

Keywords: Data mining; Utility mining; High utility itemsets; On-shelf data; Pruning strategy.

1. **Introduction.** Mining association rules [2] is an important issue in the fields of data mining due to its wide applications. Agrawal et al. [1] first proposed the most well-known algorithm, Apriori, for mining association rules from a transaction database. However, since the association- rule model assumes the same significance or profit for each item, the actual significance of an itemset cannot be easily recognized. Chan et al. thus proposed a new topic, namely utility mining, which discovered high utility itemsets from a transactional database [4]. It considers both individual profit and quantity of each product in a database, thus representing practical utility of an itemset. Several researches about utility mining were proposed in these years, most of which emphasized on how to efficiently find out the high utility itemsets from the databases [6, 10, 11].

Besides, many studies [1, 3, 5, 9] were proposed to dynamically mine association rules. An example for dynamical mining is to find the patterns for on-shelf products. However, a product may be put on shelf and taken off shelf multiple times in a store. If the entire database is considered for mining, the rules discovered may have a bias.

In this paper, we thus introduce a new kind of patterns, called high on-shelf utility itemsets, which do not only consider individual profit and quantity of each item in a transaction but also the on-shelf periods of all items in an itemset. For example, assume there is the following rule "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 a high utility itemset in the winter. We thus propose a 3-scan mining algorithm, called HOUI (High On-shelf Utility Itemsets)-Mine, for effectively and efficiently finding the itemsets from a database.

The remaining parts of this paper are organized as follows. The tables required in the proposed approach are described in Section 2. The proposed algorithm with the techniques adopted is explained in Section 3. The empirical evaluation is stated in Section 4. Conclusions and future work are given in Section 5.

2. **The Tables Required.** In this section, three tables used in the proposed algorithm are introduced. They are OS table, PTTU table and COUI table, which are used for increasing the execution efficiency.

2.1. **The OS table.** In this study, 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 kind of information of all items. For example, assume there are three time periods and three items for sale in a store. TABLE 1 shows the on-shelf information of the three products A, B, and C in each period, where '1' represents "on shelf" and '0' represents "off shelf".

TABLE 1. An example of the OS table.

	t_1	t_2	t_3
A	1	1	1
B	1	1	0
C	0	1	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 are represented by the bit strings (1, 1, 1), (1, 1, 0) and (0, 1, 1), respectively. By the AND operation on them, the common sale periods for the itemset {*ABC*} 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.

2.2. **The PTTU table.** The PTTU (Period Total Transaction Utility) table is designed in the algorithm to increase the execution efficiency. An entry in the table records the period 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 their transaction utilities are 16, 21, 17 and 13, respectively. The sum of the transaction utilities for the four transactions within the time period t_3 is thus 67. 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	Period Total Transaction Utility
t_1	108
t_2	48
t_3	67

2.3. **The COUI table.** Another table used in the proposed approach is the COUI (Candidate On-Shelf Utility Itemsets) table. The transaction-weighted-utility of itemset *X* is the sum of the transaction utilities of all the transactions containing *X* in a time period. If the actual utility of an itemset over the summation all transaction utilities in a time period is larger than or equal to the predefined threshold λ, it is called a high transaction-weighted-utilization itemset. The COUI table thus keeps the actual utility values of each high transaction-weighted-utilization itemset in each time period. If an itemset is not high, its actual utility value will not be found and the value 0 will be put into the table. For example, assume that the OS table and the COUI table are shown in TABLE 1 and TABLE 3, respectively. Take the itemset {A} as an example. Since the on-shelf bit string of item A is [1, 1, 1] and its corresponding utility values in the COUI table are 0, 12 and 12. It means that A is not high in t_1 and is high in t_2 and t_3. And both its actual utility values in t_2 and t_3 are 12. Only the itemsets which are high in at least one time period will be kept in the table. It can thus be used to efficiently maintain and catch the actual utility value of each itemset in its on-shelf periods. The COUI table can be easily constructed after the high transaction-weighted-utilization itemsets within each time period are found. The COUI table can be constructed after the database is scanned once.

TABLE 3. An example of the COUI table.

Itemset	Utility Values
{A}	[0,12,12]
{BC}	[0,77,0]

3. **The Proposed Algorithm.** In this section, we describe the proposed mining algorithm, HOUI (High On-shelf Utility Itemsets)-Mine, for discovering high on-shelf utility itemsets in a transactional database. A high on-shelf utility itemset is the one with its sum of utilities in all on-shelf periods larger than a threshold. The pruning strategy and the filtration mechanism are first introduced below.

3.1. **The upper bound of on-shelf utility.** The pruning strategy adopts the upper bound of on-shelf utility to remove redundant candidate itemsets early during the mining process. If an itemset *X* is a high transaction-weighted-utilization itemset within at least a time period, it is a high on-shelf utility candidate itemset within the entire on-shelf periods. The candidate may have its actual utility values in some on-shelf time periods and have 0 in some others according to the COUI Table. The utility upper bound of X in the time periods without actual utility values can then be derived as follows:

$$ub(X)_y = (pttu_y * \lambda) - 1,$$

where $ub(X)_y$ is the utility upper bound of the itemset *X* in the *y*-th time period, $ptty_y$ is the period total transaction utility in the *y*-th time period, and λ is the threshold. In this way,

many unpromising candidates can be pruned.

3.2. **The filtration mechanism to generate itemsets.** In this subsection, we illustrate how to avoid generating unnecessary utility itemsets via the filtration mechanism, which is based on the high-transaction-weighted-utilization (*HTWU*) 2-itemsets. An example is first given below to illustrate the idea.

Example 1: Assume that a transaction *T* in a certain time period includes four items, 3*A*, 2*B*, 25*C* and 3*D*, where the numbers represent the quantities of the items. Also assume their profit values are 3, 10, 1 and 6, respectively. Besides, suppose {*AB*}, {*BC*} and {*CD*} are three high-transaction-weighted-utilization *2*-itemsets, which have been found. FIGURE 1 shows the process of generating itemsets by using the filtration mechanism.

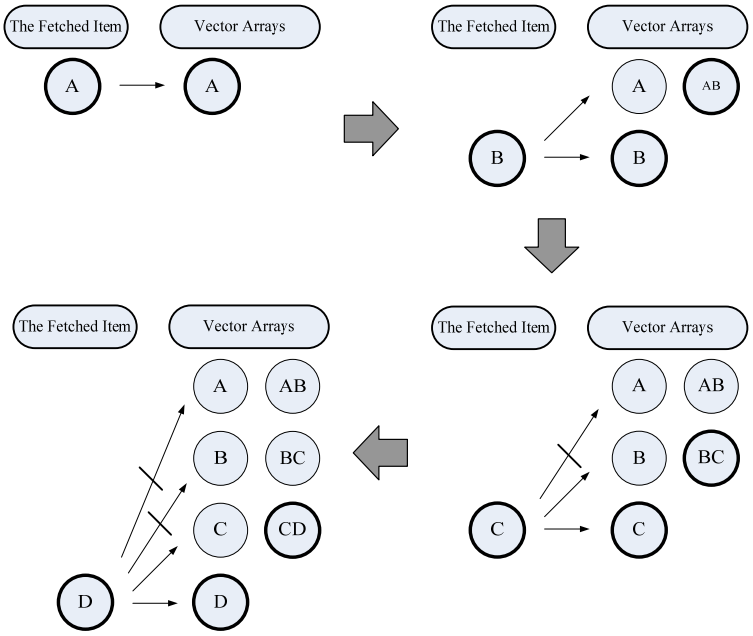

FIGURE 1. The whole process of generating itemsets by using the filtration mechanism.

In FIGURE 1, the HOUI-Mine algorithm first fetches the first item *A* in *T* and allocates it to the first row of the two dimensional vector array. The algorithm then fetches the second item *B* in *T* and allocates it to the second row of the vector array. Since there is only the item *A* in front of the fetched item *B*, the algorithm then checks whether items *A* and *B* have high-transaction-weighted-utilization relationship. In this example, *{AB}* is a high-transaction-weighted-utilization *2*-itemset. It is thus generated and put into the back of *{A}* in the first row because the first item in *{AB}* is *A*. The algorithm then continues to fetch the third item *C* and performs the same process. It first puts *{C}* in the third row of the vector array. It then forms *{AC}* and checks whether *{AC}* is a high-transaction-weighted-utilization *2*-itemset. In the example, *{AC}* is not, such that no combination of a subset in the first row with *{C}* is necessary. The algorithm then forms *{BC}* from the second row and finds it is a high-transaction-weighted-utilization *2*-itemset. *{BC}* is thus put in the back of *{B}* in the second row. Two new subsets *{BC}* and *{C}* are generated for the third item. Similarly, the fourth item *D* is fetched and the above process is repeated again. The two

new subsets *{CD}* and *{D}* are obtained and put into the vector array.

3.3. **The HOUI-Mine algorithm.** Based on the above mechanism and data structures, the proposed mining algorithm is stated as follows.

STEP 1: Initialize the *PTTU* (Period Total Transaction Utility) table as the zero table.

STEP 2: Initially set *COUI* = ϕ.

STEP 3: Scan the database once to construct the *PTTU* table and to find the high-transaction-weighted-utilization 2-itemsets in each time period.

STEP 4: Scan the database once to find all the high-transaction-weighted-utilization (*HTWU*) itemsets in each time period by the filtration mechanism based on the high-transaction-weighted-utilization *(HTWU)* 2-itemsets in STEP 3.

STEP 5: Find the utility upper bound ratio of each itemset in the *COUI* table. If the utility upper bound ratio of an itemset is larger than or equal to λ and all its utility values in the on-shelf time periods are actual values, then put it in the final set; If one of its utility values is not the actual value, then put it in the rescan set.

STEP 6: Scan the database again to find the actual utility value of each itemset in the rescan set. If the utility ratio of an itemset is larger than or equal to λ, then put it in the final set.

STEP 7: Output the final set of high on-shelf utility itemsets.

4. **Experimental Evaluation.** We conducted a series of experiments to compare the difference between the traditional and the proposed patterns and to evaluate the performance of proposed HOUI-Mine algorithm with various parameter values. They were implemented in J2SDK 1.5.0 and executed in a PC with 3.0 GHz CPU and 1GB memory.

The datasets in the experiments were generated by IBM data generator [8]. However, since our objective is to discover high on-shelf utility itemsets, we also develop a simulation model which is similar to the model used in [11]. Two measures are used to evaluate the change status of the traditional and the proposed patterns. The type-*A* change is to measure the average utility difference and the type-*B* change is to measure the average number difference. The results for the synthetic dataset T10.I4.N2K.D200K with the threshold varying from 1% to 0.2% are shown in FIGURE 2. It can be observed that the difference is quite obvious when the factor of periods is considered.

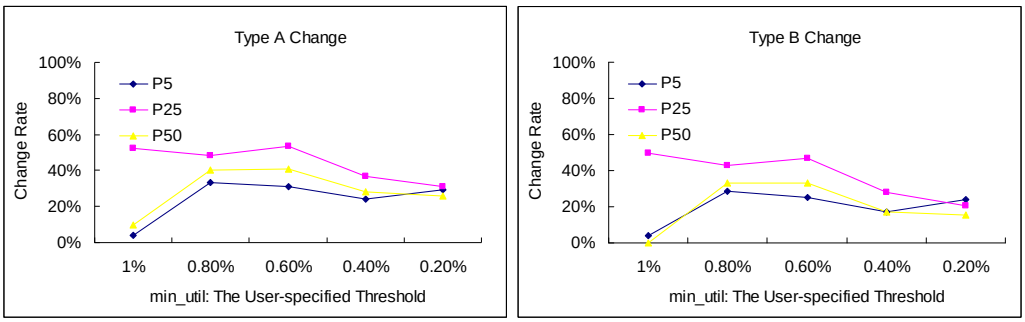

FIGURE 2. The comparison of the traditional and the proposed patterns.

5. **Conclusions.** In this paper, we handle the mining of high on-shelf utility itemsets, which consider the common on-shelf periods for items as an important factor. We have also proposed the HOUI-Mine algorithm to efficiently discover the desired itemsets from the databases. The pruning strategy based on the on-shelf utility upper bound and the filtration mechanism for generating itemsets is also designed to prune redundant candidate itemsets early and to systematically check the itemsets. The experimental results also show that the proposed approach has a good performance under various conditions. As to future work, we would apply the proposed pattern, mining approach, and pruning strategy into other practical applications, such as the data stream and supermarket promotion application.

Acknowledgment. This work is partially supported by the National Science Council of the Republic of China under contract NSC 97-2221-E-390-023. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

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, T. Imielinksi, and A. Swami, Mining association rules between sets of items in large database, *The ACM SIGMOD International Conference on Management of Data*, pp.207-216, 1993.

[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, issue.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. 1996. Quest Synthetic Data Generation Code. From: http://www.almaden.ibm.com/cs/quest/syndata.html.

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

[11] H. Yao, and H.J. Hamilton, Mining itemset utilities from transaction databases, *Data & Knowledge Engineering*, vol.59, no.3, pp.603-626, 2006.