THE NOVEL APPROACH FOR ONLINE MINING OF TEMPORAL MAXIMAL UTILITY ITEMSETS FROM DATA STREAMS

Mrs P. Muthu Lakshmi[1], S.Sherin Ranjitha[2], R.Shirly Myrtle[3], M.Sneha[4]

[1]*Assistant Professor, Department of Computer Science and Engineering, Francis Xavier Engineering College, vannarpettai, Tirunelveli*
[2]*UG students, Department of Computer Science and Engineering, Francis Xavier Engineering College, vannarpettai, Tirunelveli*

ABSTRACT

Data stream mining has become an evolving research topic in the data mining field, where finding periodic itemsets is an important task in data stream mining with wide variety of applications. In recent days, utility mining is getting high attentions with two issues re- considered: First, the utility (e.g., profit) of each item may be distinct in real applications; second, the frequent itemsets may not build the highest utility. In this paper, we propose a novel algorithm named *GUIDE* (***G**eneration of temporal maximal **U**tility **I**temsets from **D**ata str**E**ams*) which can find temporal maximal utility itemsets from streams of data. A novel data structure, named, *TMUI-tree* (***T**emporal **M**aximal **U**tility **I**temset tree*), is also proposed for systematically capturing the value of each itemset within one-time scanning. The major contributions of this paper are as follows: 1) GUIDE is the first one-pass utility-based algorithm for extracting temporal maximal utility itemsets from a stream of data, and 2) TMUI-tree is efficient and simple to manage. Through conducted experimental results we come to know that our approach surpasses other existing utility mining algorithms like Two-Phase algorithm under the data stream environments.

Keywords

Data stream mining, utility mining, maximal itemsets, temporal high utility itemsets

1. INTRODUCTION

Data stream analysis is an evolving topic broadly studied in recent decade. A data stream is a continuously ordered sequence of transactions that occurs sequentially in actual manner. There are many applications in data stream mining, like knowledge gaining from online e-business or transaction flows, review of network flows, observing of sensor data, and so on. Distinct from traditional databases, data streams also have some unique features like continuous, unbounded, high speed andtime-varying data distribution. Hence, some barrier are posed in data stream mining as follows [9]. First, as infinite transactions could not be stored, multi-scan algorithms are no more allowed. Second, in order to capture the details of the high speed data streams, the algorithms must be faster. or else, the precise of the mining results will be reduced. Third, the data distribution of data streams should be remain to avoid concept drifting issues. Fourth, incremental processes are required for mining data streams in order to make working with the old data as compact as possible. Therefore, efficient one-pass methods and small data structures for distinguishing the data stream are required.

Over recent years, utility mining appears as a new research problem, which is to discover the itemsets with high utilities, i.e., the itemsets whose utility values are greater than or equivalent to the user-specifiedminimum utility threshold. There are some rich applications of utility mining, like business promotion, webpage organization and catalog outline. Unlike long established association rule mining, utility mining can discover profitable itemsets which may not appear systematically in databases. By the above mentioned merits, we can see that it is necessary to thrust utility mining into data stream mining, such as for the surging data in chain hypermarkets.

From this, we aim at discovering maximal utility itemsets from data streams with the motive of minimizing the number of patternsin this paper. This is inspired by the observation that there exists no study that investigates advanced utility itemset patterns such as maximal utility itemsets in data streams. We proposed a novel algorithm called GUIDE (***G**eneration of temporal maximal **U**tility **I**temsets from **D**ata str**E**ams*) for discovering maximal utility itemsets from the critical time to the present time in data streams. The fundamental idea of GUIDE is to productively pick up the essential details about the utilities of the appeared itemsets and stock them into a concise data structure, namely TMUI-tree (***T**emporal **M**aximal **U**tility **I**temset tree*). The novel contributions of GUIDE are potrayed as follows. First, to the finest of our knowledge, it is the first one-pass algorithm for discovering maximal high utility itemsets from streams of data. Second, the proposed data structure, namely TMUI-tree, is simple to manage for stocking essential details in the mining processes and it facilitates GUIDE for discovering temporal maximal utility itemsets productively. These nice characteristics meet the needs of data stream mining as previously mentioned. By conducted experimental results, GUIDEis shown to surpasses other relevant algorithms like Two-Phase. [14] considerably since the nice characteristics of GUIDE that there is no need to develop candidate itemsets during the mining processes.

This paper is arranged systematically as follows. In section 2, we indicate some related researches. In section 3, we explain some definitions and introduce the proposed technique. In section 4, the execution of the proposed method is appraised in the experiments. The conclusions are given in section 5.

2. RELATED WORKS

A number of researches have been proposed on discovering data streams in the over last several years. In [4, 5, 9], the problems, limitations and applications of data stream mining were demonstrated and analysed in details. Further, many algorithms for discovering distinct patterns from data streams were proposed like Moment [3] and CFI-Stream [8] for discovering closed frequent itemsets, estDec+ [10] and DSM-MFI [12] for discovering maximal frequent itemsets, a time-sensitive model [13] and a two phase mining process [17] for mining frequent itemsets. However, none of the above mentioned researches reveal the expandability and flexibility for the utility mining. THUI-Mine [16] was the earliest algorithm for mining high utility itemsets from streams of data. It increase the Two-Phase algorithm [14] for data stream mining. lately, Li et al. [11] proposed two algorithms, named MHUI- BIT and MHUI-TID, for discovering high utility itemsets from data streams by a three-phase mechanism. However, the mechanisms in the above two researches will still examine the data in the sliding windows more than once that could not meet the limitations of data stream mining thoroughly.

In these kind of algorithms, three models, i.e., landmark, damped and sliding window, are used for seizing the data in data streams in distinct ways [9, 13]. Landmark model stock the total data from the unique time point called landmark and discovers patterns within the data [12]. In damped model, the total data is also stocked from the landmark time point. Though, the weight is given for decreasing the necessary of the out-of-date data [10]. In sliding window model, a window which slides with time is held to seize the data value within a fixed time or a fixed number of transactions. When the mining operation is requested, just the information kept in the window is used [3, 8, 11, 13, 16, 17]. The three models are suitable to distinct situations according to users' requirement..

On the other hand of stream data mining, largescale researches on traditional databases mining have been done with rich kinds of patterns discovered. Apriori algorithm [1] was the explorer for efficiently mining association rules from large databases. The tree-based association rule mining algorithms such as FP-growth [7] were proposed later. FP-growth improved the efficiency of mining association rules than Apriori considerably since it does not have to execute candidate itemsets and it examines database twice. Many more algorithms for discovering different patterns were put forward after Apriori, like GenMax [6] for mining maximal frequent itemsets by a backtrack search tree, and CLOSET [15] for extracting closed frequent itemsets by applying a compressed tree structure. In addition, some new techniques were proposed for extracting high utility itemsets like [2] and [14]. Liu et al [14] proposed the Two-Phase algorithm to overcome the main issue of utility mining, i.e., the downward closure property is no longer applicable in utility mining, by applying the transaction-weighted downward closure property. The Two-Phase algorithm was shown to efficiently reduce the number of the candidate itemsets and reduce the computations. However, Two-Phase is a multi-pass algorithm and it could not fit the limitation, i.e., one-pass, of data stream mining.

3. PROPOSED METHOD

Here, we propose a novel algorithm for extracting temporal maximal utility itemsets from data streams. At first we define some terminologies used in this paper and then introduce the proposed algorithm, GUIDE. The important idea of GUIDE is to use a small summary data structure, which is TMUI-tree, for storing the utility values and the potential TMUIs in a new sequential transaction. hence, the actual TMUIs are found and generated in the datastructure simultaneously.

Preliminary Concepts

In this section, we further expand the definitions of utility mining in [14] for the motive of utility mining in data stream.

- $I = \{i_1, i_2, ..., i_m\}$ is a set of literals, called items. An itemset is a subset of I. Each item in the data flow has its own unit prize. The unit prize of the item x is represented by $pr(x)$.
- A data stream DS is represented by $\{(Tid_1, t_1), (Tid_2, t_2), ..., (Tid_n, t_n), ...\}$, where Tid_k ($k \geqq 1$) is a transaction and t_k ($k \geqq 1$) is the time when Tid_k appeared in DS.
- A transaction Tid_k is collection of a set of items and their purchased numbers, which is represented by $\{(x_1, q_1), (x_2, q_2), ..., (x_p, q_p)\}$, where x_r ($1 \leqq r \leqq p$, $x_r \in I$) is the purchased item and q_r ($1 \leqq r \leqq p$) is the purchased number of x_r in Tid_k.
- The utility of the item x in Tid_k, which is represented as $trans\_u_k(x)$, is defined as $pr(x)*q_r$. The utility of an itemset X in Tid_k, $trans\_u_k(X)$, i.e., the summarization of the items which are belonged to X, is defined as $\sum_{x \in X} trans\_u_k(x)$.
- The utility of the transaction Tid_k, represented as $trans\_u_k(Tid_k)$, is defined as $\sum_{x \in Tid_k} trans\_u_k(x)$. The utility of an itemset X in DS, $u(X)$, is defined as $\sum_{k \in DS \wedge X \subseteq k} \sum_{x \in X} trans\_u_k(x)$
- . The total utility of a stream of data, represented as $totalU$, is defined as the summation of the utility of all transactions in the data stream which was gathered gradually since the landmark was set.

By the above definitions, we introduce *temporal utility itemset*, represented as *TUI*, as the itemset whose utility is greater than or equivalent to a user-specified minimum utility threshold s ($0 \leqq s \leqq 1$), where s is the percentage of the utility over total utility in *DataStream*. An itemset X is a TUI if its utility $u(X) \geqq totalU*s$. furthermore, by the basic definition of maximal frequent itemset in [6], we define *temporal maximal utility itemset*, represented as *TMUI*, as the TUI which is not a subset of any other TUI. We mean the maximal utility itemsets found from data streams temporal maximal utility itemsets because every pattern is emerging by time in data streams.

Problem Statement. For a given specified data stream, a pre-defined utility table and a user-specified minimum utility threshold, the problem of extracting temporal maximal high utility itemsets using the landmark model is to discover the set of TMUIs from the landmark timestamp over the entire record of the data stream.

Definitions of Terminologies

Here we introduce some basic definitions of the terminologies in the following.

Definition 1. The components of TMUI-tree. A TMUI-tree is collection of nodes and links. An itemset will be stocked as a node in TMUI-tree. The node of TMUI-tree, represented as *<id: utility:*

time>, is a triple which includes identity of the itemset, the utility of the itemset, and the time when the itemset was attached asa node into the TMUI-tree. Every node has two kinds of links, one node is parent-link which points to its parent node and another node ischildren-link which points to its children node.

In the GUIDE algorithm, when a new transaction occurs, we must decompose the transaction into several estimation and then stock them into TMUI-tree. The definition of the technique, i.e. transaction-projection, is as follows.

Definition 2. Transaction-projection. Without loss of generality, in this paper, we presume that the itemsets in all transactions are classify in a certain fixed order, e.g. in alphabetical order. When a transaction {(i1, q1) (i2, q2) ... (in, qn)} occurs, we produce the projections of this transaction. Rather than generating all sub-itemsets of this transaction, we have proposed the transaction-projection which is depicted as follows. First, the transaction is projected into the postfixes by cutting off the first item in the transaction in turn, i.e., {(i1, q1) (i2, q2) ... (in, qn)}, {(i2, q2) ... (in, qn)}, ..., and {(in, qn)}. Then, for each postfix, it is projected into the prefixes by cutting off the last item in turn. Taking {(i2, q2) (i3, q3) ... (in, qn)} for example, we obtain {(i2, q2) (i3, q3) ... (in, qn)}, {(i2, q2) (i3, q3) ... (in-1, qn-1)}, ..., and {(i2, q2)}.

When a new TUI was produced from TMUI-tree, it will be tested whether it is maximal or not. If it is maximal, it will be stocked into a temp list which is called TMUI-list. When a user queries GUIDE for all TMUIs from the landmark time to the present, we will yield the list. The TMUI-list is defined as follows.

Definition 3. TMUI-list. A TMUI-list is a collection of all TMUIs fromthe landmark time to the present. Each TMUI in the TMUI-list is represented by a pair, represented as <*id: utility*>, where id stands for the identity of the TMUI, utility stands the summation of the utility value of this TMUI from the landmark time to the present in the stream of data.

Proposed Algorithm: GUIDE

We introduce the GUIDE algorithm which we have proposed for discovering TMUIs from stream of data in detail as follows. The GUIDE algorithm is shown in Figure 1. When a new transaction occurs into the stream of data, we execute transaction-projection on this transaction and get its projections. Then, every item in the projections is examined whether there is a node in TMUI-tree for the item. If the node did not exist, it is generated. Orelse, its utility is gathered into the node. At the same time, the utility of every modified node *X* is examined to see whether it is greater than or equivalent to the minimum utility threshold. If a new TUI was created, i.e. $u(X) \geqq totalU*s$, it is also examined whether it is maximum. If it is not a subset of any other TMUI in TMUI-list, it is attached into the TMUI-list. At last, an additional check is executed. Some TMUIs in the TMUI-list may be inaccurate since they would be run out as the time goes by. In this checking operation, each TMUI in the TMUI-list is examined whether its utility is above than or equivalent to the new threshold, i.e. $totalU_{now}*s$. Because $totalU_{now}*s$ will be expanded when a new transaction occurs, the TMUIs in TMUI-list whose utility become smaller than the new threshold must be cut off. If the utility of a TMUI *Y* is above than or equivalent to the threshold, i.e., $u(Y) \geqq totalU_{now}*s$, itis kept; otherwise, it is cut off.

To demonstrate by an example, presume that there is a data stream *DS* and a utility table for the items in *DS* as shown in Figure 2. In Figure 2(a), the first column "*Tid, Time*" is the time when the transaction comes into *DS*. We assign the minimum utility threshold *s* as 20% and the landmark time is at time 0. By the transaction time, the first transaction Tid_1, that is {(*C, 12*) (*E, 2*)}, is arriving into *DS* at time 2. After transaction-projection, we are having three projections, such as{*C*}, {*E*} and {*CE*}. The utility of these projections are 12, 10 and 22, respectively. Because TMUI-tree isnull now, we add all nodes with regard to all of these projections. The node <*C: 12: 2*> is first added as a child node of the root of the TMUI-tree. Then the node <*E: 22: 2*> is added as a child of the node <*C: 12: 2*>. It stands for the itemset {*CE*}. Finally, the node <*E: 10: 2*> is added as another child of the root. The TMUI- tree now is shown in Figure 3(a). During adding the first transaction into the TMUI-tree, the modified nodes are examined whether they are new TMUIs. At present, the *totalU* of *DS* is 22 and the threshold *totalU*s* is 4.4. So, the itemsets <*C: 12*>, <*CE: 22*> and <*E: 10*> are all new TUIs which are above the threshold. After examined the maximal property of the three TUIs, only <*CE: 22*> is a TMUI among them and it is added into the TMUI-list at time 2. Figure 3 shows the TMUI-tree when all transactions in *DS* were examined by GUIDE. The gray nodes stand for the nodes which were altered or generated at that time.

Procedure: GUIDE
Input: A data stream *DS*, a pre-defined utility table and a minimum utility threshold *s*
Output: A list of currently TMUIs
1. **while** a new transaction Tid_k arrives into *DS* **do**
2. $totalU = totalU + trans\_u_k(Tid_k)$
3. perform transaction-projection on Tid_k and get its projections $proj_k$
4. **for** each item $i \in p$, $p \in proj_k$ **do**
5. $tmp\_util = tmp\_util + trans\_u_k(i)$ //a temp utility value of *i*
6. trace TMUI-tree to the node of *i* by the identity of *p*
7. **if** the node do not exist **do**
8. create a new node <*i*'s id: *tmp_util*: t_k> in TMUI-tree
9. **else** modify the node's utility by adding *tmp_util*
10. **if** the node's utility $\geq$ *totalU*s* **do**
11. create a new TUI *X* <*node's id*: *node's utility*>
12. **if** X is maximum **do** add *X* into the TMUI-list
13. **end while**
14. **for** each TMUI in TMUI-list **do**
15. **if** its utility is less than *totalU*s* **do** prune the TMUI

Figure 1. The procedure of GUIDE

Tid, Time	Transaction
(1,2)	(C,12)(E,2)
(2,3)	(B,6)(D,1)(E,1)
(3,6)	(A,1)(B,4)(E,1)
(4,8)	(A,1)(C,1)(D,15)(E,1)
(5,9)	(A,1)(B,1)(C,27)

(a)

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

(b)

Figure 2. (a) A data stream *DS* (b) A utility table

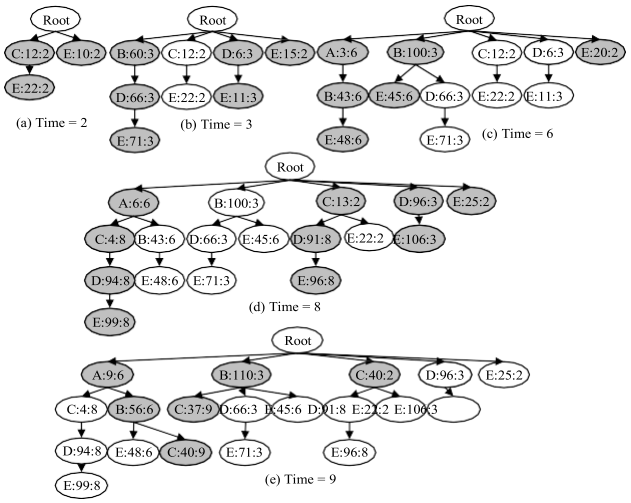


Figure 3. An example of the construction of TMUI-tree

The Pruning Method of TMUI-tree

Since the data streams are limitless, we could not stock all data into TMUI-tree for a long period of time. so, we propose a pruning method for TMUI-tree to stop the memory exhaustion. Before explaining the pruning technique, an additional definition for a notation *PT(X)* is described as follows.

- *PT(X)*, the ratio of pass time of the itemset *X* appeared in data stream *DS*, is depicted as $(t_{now}\text{-}t_X)/(t_{now}\text{-}t_{LM})$, where t_{now} is current time of *DS*, t_{LM} is the landmark time of *DS*, and t_X is the time when the itemset *X* was firstly appeared in *DS*. Without loss of majority, we also defined the pass time of the landmark time and the current time, i.e., *PT(LM)* and *PT(now)*, as 1 and 0.

We define the proposed pruning technique of the TMUI-tree in detail as follows. presume that the minimum utility threshold is *s*. as there is no downward closure in utility mining, we must examine each node in TMUI-tree without skipping any node. When the pruning process is called, each node in TMUI-tree is examined orderly to decide whether it will be cutoff. The pruning threshold for a node *x* is *s*totalU*PT(x)*. If the utility of *x* was less than the pruning threshold, *x* is cutoff. The threshold is described by the time when the node was added into TMUI-tree. In similar words, if a node was added into TMUI-tree lately, the pruning threshold of this node is small; orelse the threshold is large. Thus, the nodes added early with less utility are cutoff; on the other hand, the nodes added lately with larger utility are held.

4. EXPERIMENTAL EVALUATIONS

To estimate the performance of GUIDE, we have conducted various experiments by using synthetic datasets generated from the IBM synthetic data generator as in [1]. For other parameters co related to utility mining, we use the settings the same to [14]. The explanation of parameters and the default settings are shown in Table 1. The simulation was executed in Java and conducted in a machine with 3.0GHz CPU and 1GB memory. To the best of our knowledge, although there are several past researches which have dealt with utility mining or data stream mining, there is still no single-pass algorithm which can discover maximal temporal utility itemsets from streams of data. Hence, we compare our technique with Two-Phase, which is the state-of-the-art algorithm for discovering high utility itemsets in long established databases. In the following paragraphs, we conduct four parts of experiments to estimate the performance of the GUIDE algorithm.

Table 1. Parameter settings

Parameter Descriptions	Default
D: Number of transactions in data stream	10k, 100k
T: Average size of the transactions	10, 20
I: Average size of the potentially high utility itemsets	6
L: Number of potentially high utility itemsets	500
N: Number of distinct items	1000
q: Number of purchased items in transactions	1~5
pr: Unit price of each item	1~1000

Experiment I. Effects of varying minimum utility thresholds. In our first part of the experiments, we compare the execution time limit of GUIDE and Two-Phase. As shown in Figure 4(a), we can see the output of the two algorithms in the dataset T10I6D10k. Though the execution time of Two-Phase decreases clearly with the minimum utility threshold increased, GUIDE still performs better than Two-Phase when minimum utility threshold is between 0.2% and 1%. It can be also noted that GUIDE is more secure than Two-Phase because the execution time of GUIDE does not increase as clearly as Two-Phase when the minimum utility threshold is decreased.

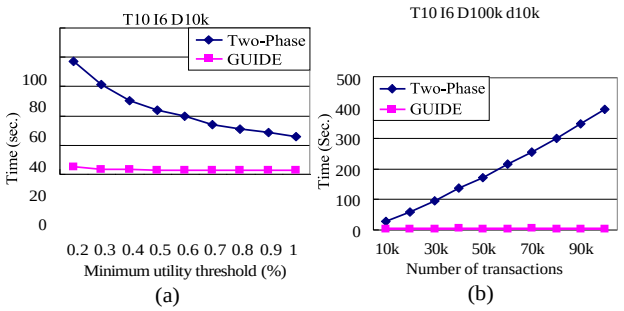
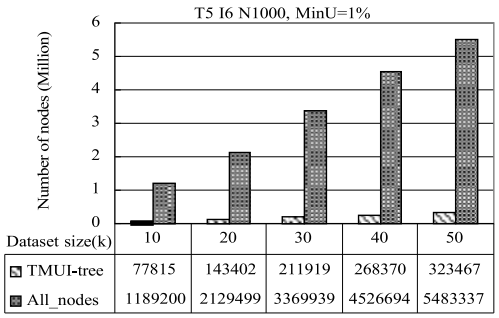


Figure 4. Execution time
(a) under varied minimum utility thresholds
(b) under varied number of transactions

Experiment II. Performance comparisons under stream environments. In our second part of the experiments, we increase the number of transactions by loading new transactions to replicate the streaming of data mining environment, i.e., limiting the processes to one-scan, to see the differences of performance between GUIDE and Two-Phase. In this experiment, we set up 10,000 transactions as a batch and input a batch at a time, i.e., the size of the incremental transactions is once 10,000. since Two-Phase is a multi-pass algorithm which is not created for mining data streams, it re-scans the entire data when a new batch occurs. However, GUIDE just generates the new inserted 10,000 transactions without re-scanning other old data. The experimental outputs are shown in Figure 4(b). We can see that the execution time of Two-Phase increases frequently with the number of total transactions since it needs to re-scan the entire data when a new batch occurs. On the other hand, GUIDE just deals with the loaded transactions when a new batch occurs. Its execution time remains steady at every batch. It shows that one-pass algorithm is important for mining data streams. Multi-pass algorithms which must re-scan the entire database could not perform effectively, that is, they cannot fit the limitations of streaming data environments.

Experiment III. Compactness of TMUI-tree. In this part, we show the closeness of TMUI-tree. The output are shown in

Figure 5. In this figure, *All_nodes* means for the number of nodes in the trees whose projections are all combinations of the transactions and, on contrary, *TMUI-tree* stands for the number of nodes in the trees whose projections are created by the transaction-projection of GUIDE. In this figure, we can see that the number of nodes of *TMUI-tree* is consecutively increased with the dataset size increasing. For now, the number of nodes in *All_nodes* held about 15 times of those in *TMUI-tree*. By this experiment, we can see that the TMUI-tree is a dense and scalable tree structure.



Dataset size(k)	10	20	30	40	50
TMUI-tree	77815	143402	211919	268370	323467
All_nodes	1189200	2129499	3369939	4526694	5483337

Figure 5. Number of nodes in TMUI-tree under varied number of transactions

Experiment IV. Completeness of GUIDE. Since GUIDE is an imprecise algorithm, it may lose some patterns. In the following experiments, the completeness of GUIDE is checked. Without loss of principle, we first define the measurement named *recall of patterns*, shortened as *ROP*, as the percentage of the patterns seized by GUIDE. The formula of ROP is given below. ROP reflects the completeness of patterns found.

$$ROP(\%) = \frac{\text{number of patterns found by GUIDE}}{\text{number of all patterns should be found}} \times 100\%$$

We process the experiment by differing the size of datasets. Table 2 is the experiment output of GUIDE under the dataset T5I6N1000. The minimum utility threshold has been set up as 1% and D is set up from 10,000 to 50,000. In this table, the second column *#patterns* is the number of all TMUIs which can be found from dataset and, on the contrary, the third column *#GUIDE* is the number of TMUIs which is actually set up by GUIDE. It can be seen that from this as the size of dataset increases, the ROP of GUIDE is gradually decreases. It shows that GUIDE is more applicable for small datasets. Besides, in the experiments, the ROPs of the distinct datasets are almost above 97%. This means that GUIDE has high completeness of patterns for mining TMUIs in streams of data.

Table 2. The recall of patterns under different dataset sizes

D(k)	#Patterns	#GUIDE	ROP (%)
10	100	99	99
20	104	103	99.04
30	95	94	98.95
40	115	113	98.26
50	87	85	97.70

By the experiments conducted above, we can show that although GUIDE misses a few patterns, it spends much lower execution time than Two-Phase. Furthermore, TMUI-tree is shown for its completeness and compactness. Thus GUIDE is an efficient and effective algorithm for discovering TMUIs in streams of data.

5. CONCLUSIONS

In this paper, we have proposed a unique algorithm, called GUIDE, forefficiently mining temporal maximal utility itemsets from the landmark time to the present in data streams. We have also proposed a unique data structure, called TMUI-tree, for storing data inthe processes of mining utility patterns from data streams. The major contributions of GUIDE and TMUI-tree are that GUIDE is the first one-pass algorithm for extracting maximal utility itemsets indata streams and TMUI-tree is simple to manage and it can help GUIDE finding TMUIs efficiently. The conducted experimental results on different datasets showed that GUIDE is considerably efficient than the state-of-the-art utility mining algorithm, i.e. the Two- Phase algorithm, under distinct conditions. Furthermore, the results describes that GUIDE has high compactness and high completenessof patterns. For future works, we will apply and expand the GUIDE algorithm to the sliding window model and time-fading model for data stream mining. Advanced estimations and experimental estimations on GUIDE will also be conducted.

6. ACKNOWLEDGMENTS

This research was supported by National Science Council, Taiwan, R.O.C. under grant no. NSC 96-2221-E-006-143-MY3 and NSC 98-2631-H-006 -001.

7. REFERENCES

[1] Agrawal, R. and Srikant, R. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conf. on Very Large Data Bases*, 1994, 487-499.
[2] Chan, R., Yang, Q. and Shen, Y. Mining high utility itemsets. In *Proc. of Third IEEE Int'l Conf. on Data Mining*, Nov., 2003, 19-26.
[3] Chi, Y., Wang, H., Yu, P. S. and Muntz, R. R. Moment: maintaining closed frequent itemsets over a stream sliding window. In *Proc. of the IEEE Int'l Conf. on Data Mining*, 2004.
[4] Gaber, M. M., Zaslavsky, A. B. and Krishnaswamy, S. Mining data streams: a review. *ACM SIGMOD Record*, Vol.34, No.2, 2005, 18-26.
[5] Golab, L. and Ozsu, M. T. Issues in data stream management. In *ACM SIGMOD Record*, Vol. 32, No. 2, 2003.
[6] Gouda, K. and Zaki, M. J. Efficiently mining maximal frequent itemsets. In *Proc. of the IEEE Int'l Conf. on Data Mining*, 2001.
[7] Han, J., Pei, J., and Yin, Y. Mining frequent patterns without candidate generation. In *Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data*, 2000, 1-12.
[8] Jiang, N. and Gruenwald, L. CFI-Stream: mining closed frequent itemsets in data streams. In *Proc. of the Utility-Based Data Mining Workshop, ACM KDD*, 2006.
[9] Jiang, N. and Gruenwald, L. Research issues in data stream association rule mining. In *ACM SIGMOD Record*, Vol. 35, No. 1, 2006.
[10] Lee, D. and Lee, W. Finding maximal frequent itemsets over online data streams adaptively. In *Proc. of 5th IEEE Int'l Conf. on Data Mining*, 2005.
[11] Li, H.-F., Huang, H.-Y., Chen, Y.-C., Liu, Y.-J., Lee, S.-Y. Fast and Memory Efficient Mining of High Utility Itemsets in Data Streams. In Proc. of *the 8th IEEE Int'l Conf. on Data Mining*, 2008, 881-886.
[12] Li, H.-F., Lee, S.-Y., and Shan, M.-K. Online mining (recently) maximal frequent itemsets over data streams. In *Proc. of the 15th IEEE Int'l Workshop on Research Issues on Data Engineering*, 2005.
[13] Lin, C. H., Chiu, D. Y., Wu, Y. H. and Chen, A. L. P. Mining frequent itemsets from data streams with a time-sensitive sliding window. In *Proc. of SDM*, 2005.
[14] Liu, Y., Liao, W. and Choudhary, A. A fast high utility itemsets mining algorithm. In *Proc. of the Utility-Based Data Mining Workshop*, 2005.
[15] Pei, J., Han, J., and Mao, R. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000, 11-20.
[16] Tseng, V. S., Chu, C. J. and Liang, T. Efficient mining of temporal high utility itemsets from data streams. In *ACM KDD Workshop on Utility-Based Data Mining Workshop*, Philadelphia, USA, 2006.
[17] Tanbeer, S. K., Ahmed, C. F., Jeong, B.-S. and Lee,Y.-K. Efficient frequent pattern mining over data streams. In *Proc. of the ACM 17th Conference on Information and Knowledge Management*, 2008.