Advanced Engineering Informatics xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification ☆

Jerry Chun-Wei Lin [a,b,*], Wensheng Gan [a], Tzung-Pei Hong [c,d]

[a] Innovative Information Industry Research Center (IIIRC), School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, HIT Campus, Shenzhen University Town, Xili, Shenzhen, PR China
[b] Shenzhen Key Laboratory of Internet Information Collaboration, School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, HIT Campus, Shenzhen University Town, Xili, Shenzhen, PR China
[c] Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan, ROC
[d] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, ROC

ARTICLE INFO

Article history:
Received 10 December 2014
Received in revised form 11 May 2015
Accepted 12 May 2015
Available online xxxx

Keywords:
Utility mining
Pre-large concept
Transaction modification
Two-Phase procedure
Dynamic databases

ABSTRACT

High-utility itemsets mining (HUIM) is a critical issue which concerns not only the occurrence frequencies of itemsets in association-rule mining (ARM), but also the factors of quantity and profit in real-life applications. Many algorithms have been developed to efficiently mine high-utility itemsets (HUIs) from a static database. Discovered HUIs may become invalid or new HUIs may arise when transactions are inserted, deleted or modified. Existing approaches are required to re-process the updated database and re-mine HUIs each time, as previously discovered HUIs are not maintained. Previously, a pre-large concept was proposed to efficiently maintain and update the discovered information in ARM, which cannot be directly applied into HUIM. In this paper, a maintenance (PRE-HUI-MOD) algorithm with transaction modification based on a new pre-large strategy is presented to efficiently maintain and update the discovered HUIs. When the transactions are consequentially modified from the original database, the discovered information is divided into three parts with nine cases. A specific procedure is then performed to maintain and update the discovered information for each case. Based on the designed PRE-HUI-MOD algorithm, it is unnecessary to rescan original database until the accumulative total utility of the modified transactions achieves the designed safety bound, which can greatly reduce the computations of multiple database scans when compared to the batch-mode approaches.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Data mining is used to reveal meaningful or useful information from an extensive database. Discovered information or knowledge can be used to aid mangers or retailers in making efficient decisions or strategies. Association-rule mining (ARM) [2,3,7,22] is a common way to present the binary relationships between the purchased products in market basket analysis. Agarwal et al. designed the Apriori algorithm [3] to first mine frequent itemsets (FIs) in a level-wise way based on the minimum support threshold, and then combine the discovered FIs to generate association rules (ARs) based on the minimum confidence threshold. It is insufficient to mine high profitable itemsets by only considering occurrence frequency in ARM. High-utility itemsets mining (HUIM) was proposed to find rare itemsets with high profits by considering both profit and quantity factors [6,25,26]. An item/itemset is considered as a high-utility itemset (HUI) if its utility is no less than the minimum utility threshold. Liu et al. presented a Two-Phase model to maintain the transaction-weighted downward closure (TWDC) property of the designed high transaction-weighted utilization itemsets (HTWUIs) [19]. An additional database scan is required to determine the actual utilities of the remaining HTWUIs. Many algorithms have been proposed to efficiently mine HUIs from a static database in batch mode [5,10,15,23]. When the transactions are frequently changed through insertion [8], deletion [9] or modification [13], most approaches discard the previously discovered information and then perform a conventional scan on the updated database to re-mine the required information, which is not practical in real-life situations.

☆ Handled by C.-H. Chen.

* Corresponding author at: Innovative Information Industry Research Center (IIIRC), School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, HIT Campus, Shenzhen University Town, Xili, Shenzhen, PR China.

E-mail addresses: jerrylin@ieee.org (J.C.-W. Lin), wsgan001@gmail.com (W. Gan), tphong@nuk.edu.tw (T.-P. Hong).

http://dx.doi.org/10.1016/j.aei.2015.05.003
1474-0346/© 2015 Elsevier Ltd. All rights reserved.

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

Table 1
A quantitative database.

TID	Transaction
1	(*A*:2), (*B*:3), (*D*:1)
2	(*A*:2), (*B*:2), (*D*:1), (*E*:1)
3	(*C*:1), (*D*:2), (*G*:1)
4	(*A*:3), (*B*:2), (*D*:1), (*F*:1)
5	(*A*:1), (*B*:1), (*C*:1), (*D*:1)
6	(*A*:1), (*D*:1), (*E*:1)
7	(*B*:2), (*C*:1)
8	(*A*:1), (*B*:2), (*D*:1), (*G*:1)

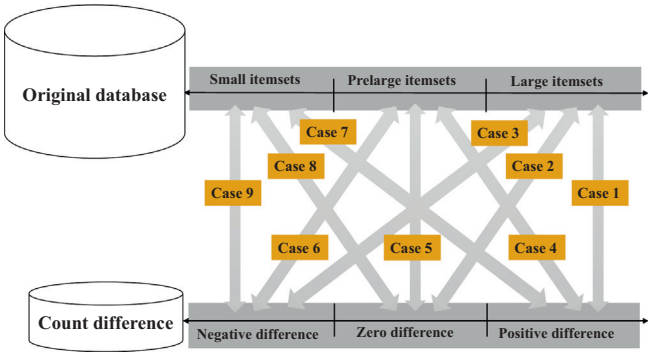

Fig. 1. Nine cases are created due to transaction modification.

Since transactions in a real-life environment may change dynamically, it is a critical issue to efficiently maintain the discovered knowledge without rescanning the original database each time. In the past, Fast UPdated (FUP) [8], FUP2 [9], and pre-large [11,12] concepts were proposed to maintain and update the discovered information for transaction insertion and transaction deletion, respectively. For HUIM, it is not an easy task to maintain the discovered HUIs compared to traditional ARM since quantity and profit factors are both concerned in HUIM. Fewer maintenance algorithms for HUIM were proposed to efficiently maintain and update the discovered HUIs in a dynamic environment with transaction insertion [5,16] and transaction deletion [14]. As one of the three common operations (transaction insertion, deletion, and modification) in databases, transaction modification is also commonly seen in real-life situations since many typos or errors may occur when the collected data from periodic transactions is inputted into a computer using a keyboard. For instance, the example database as shown in Table 1, when the transaction {6, ⟨(*A*:1), (*D*:1), (*E*:1)⟩} in the database needs to be modified as {6, ⟨(*A*:2), (*B*:1), (*F*:1), (*G*:9)⟩} due to the previous typos, the final HUIs of the updated database will be changed. Thus, some information may become invalid or new information may arise. Although the maintenance process of transaction modification can be done by two procedures which can be performed in either order: first, delete the incorrect transactions, and second, insert the correct transactions. It requires twice computations, which is very time-consuming and impractical. Thus, an efficient maintenance process for transaction modification is a critical issue and necessary in the dynamic environment. However, the issue of HUIM with transaction modification has not been proposed, to the best of our knowledge.

In this paper, a novel PRE-HUI-MOD algorithm is presented to maintain and update the discovered HTWUIs and HUIs with transaction modification. The proposed PRE-HUI-MOD algorithm adopts a Two-Phase model [19] to set the effective upper bound for discovering HTWUIs and HUIs from the original databases. The discovered HTWUIs and all transaction-weighted utilization itemsets in the modified transactions are then divided into three parts with nine cases. Each case is handled by the designed procedure to determine whether the discovered HTWUI will still remain as HTWUI or become non-HTWUI in the updated database. An additional database scan is required to determine the actual HUIs of the remaining HTWUIs. Based on the designed framework with transaction modification, the number of computations can be greatly reduced until the accumulative total utility in the modified transactions achieves the designed safety bound. In addition, previously discovered HTWUIs can be used to help the maintenance process, thus speeding up computations. Major contributions of the proposed approach are listed below.

1. We have designed a maintenance algorithm to efficiently update the discovered HTWUIs for producing HUIs with transaction modification compared to the state-of-the-art algorithms running in batch mode.
2. We extended the pre-large concept in ARM to maintain and update the discovered HUIs and HTWUIs without multiple database scans in HUIM.
3. Based on the designed safety bound mechanism, it is unnecessary to rescan the updated database each time, unlike batch-mode algorithms, until the accumulative total utility achieves the safety bound, which can greatly reduce the database scan computations.
4. Experiments are conducted to show that the proposed maintenance algorithm can efficiently handle the dynamic database with transaction modification of HUIM, and generally has better performance compared to the state-of-the-art batch-mode HUIM algorithms.

2. Related work

In this section, the high-utility itemset mining (HUIM) and the pre-large concept of ARM are briefly reviewed.

2.1. High-utility itemset mining

High-utility mining is an extension of frequent-itemset mining [5,6,10,19,25,26]. It considers both quantity and profit factors to produce more useful and profitable itemsets. An itemset is concerned as a HUI if its utility is no less than the minimum utility threshold. In the past, Chan et al. proposed a top-k mechanism to mine both positive and negative high-utility closed patterns [6]. Highly statistical patterns with a new developed pruning strategy can be discovered efficiently in a level-wise way. Yao et al. applied mathematical properties to mine HUIs based on designed utility constraints [23]. Two pruning methods were proposed to reduce the search space by the utility upper bounds and the expected utility upper bounds. Liu et al. first extended the downward closure (DC) property of ARM into the transaction-weighted downward closure (TWDC) property, and presented a Two-Phase model for mining HUIs [19]. The Two-Phase model is thus designed to efficiently speed up the computations for discovering high transaction-weighted utilization itemsets (HTWUIs) in a level-wise way. An additional database scan is then required to mine actual HUIs from the remaining HTWUIs. The Two-Phase model can also be modified as a parallelized algorithm using a Common Count Partitioned Database (CCPD) strategy on shared memory multi-process architecture [20].

Table 2
Profit table.

Item	*A*	*B*	*C*	*D*	*E*	*F*	*G*
Profit	10	3	5	25	2	3	5

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

Table 3
Transaction utilities of the given example.

TID	Transaction	Transaction utility (TU)
1	(*A*:2), (*B*:3), (*D*:1)	54
2	(*A*:2), (*B*:2), (*D*:1), (*E*:1)	53
3	(*C*:1), (*D*:2), (*G*:1)	60
4	(*A*:3), (*B*:2), (*D*:1), (*F*:1)	64
5	(*A*:1), (*B*:1), (*C*:1), (*D*:1)	43
6	(*A*:1), (*D*:1), (*E*:1)	37
7	(*B*:2), (*C*:1)	11
8	(*A*:1), (*B*:2), (*D*:1), (*G*:1)	46

The above approaches are performed in a level-wise way to generate-and-test the candidates for mining HUIs. Lin et al. proposed high utility pattern (HUP)-tree algorithm to mine HUIs without candidate generation based on the designed HUP-tree structure [15]. It keeps the sorted 1-HTWUIs in a descending order by using three sorting strategies for tree construction. Each node in the HUP-tree stores the quantities of its prefix items in a branch to actually mine HUIs without an additional database scan. Vincent et al. also designed an UP-tree structure to keep high transaction-weighted utilization 1-itemsets for tree construction [23]. Each node stores not only the occurrence frequencies of the processed item in the branch, but also the transaction utility minus the utilities of items from the previously processed item in the branch. Two mining algorithms called UP-growth and UP-growth+ were proposed to efficiently mine HUIs. Since the pattern-growth mechanism still requires very large amount of memory to keep the discovered HTWUIs, Liu et al. then designed a HUI-Miner algorithm to directly mine HUIs without HTWUIs based on the designed utility-list structure [18]. For the utility-list structure, each element stores transaction IDs (*tids*), the utility of itemset *X* in the transaction (*iutils*), and the remaining utility of itemset *X* in the transaction (*rutils*). The computations of pattern-growth mechanism can be greatly reduced without an additional database rescan. Fournier-Viger designed an improved fast high-utility miner (FHM) approach [10] by storing the item co-occurrences based on HUI-Miner structure, thus reducing join operations to speed up performance for mining HUIs. More memory is, however, required for mining HUIs based on the FHM approach. Experiments showed that the FHM algorithm outperforms existing state-of-the-art HUIM algorithms in term of runtime. Wu et al. also proposed three efficient algorithms for mining closed + HUIs [24]. From their proposed framework, the massive number of HUIs can be reduced, and more condense and concise representation of HUIs can be produced. However, the above algorithms can only discover HUIs in a static database, which is not practical in real-life applications.

In real-life situations, transactions may be inserted, deleted, or modified from the original database. New HUIs may arise and some existing HUIs may become invalid when the data is changed (transactions be inserted, deleted, or modified) in the original database. Ahmed et al. presented the IHUP-tree structures with three designed sorting strategies for mining HUIs with transaction insertion [5]. Lin et al. designed level-wise algorithms to efficiently mine HUIs with transaction insertion [17] and transaction deletion [14]. As mentioned previously, transaction modification is also commonly seen in real-life situations as typos or errors should be corrected due to intentional or unintentional input by end users. However, mining HUIs from dynamic databases with transaction modification has not yet been proposed.

2.2. Pre-large concept of ARM

Many algorithms of ARM have been proposed to mine association rules or frequent itemsets from a static database. An itemset is considered as a frequent itemset if its support ratio is no less than the minimum support threshold. The Fast UPdated (FUP) [8] and FUP2 [9] concepts were respectively proposed to update the discovered information for transaction insertion and transaction deletion. However, the original database still needs to be rescanned to handle the itemsets in case 3 [8] for transaction insertion, and the itemsets in case 4 [9] for transaction deletion.

The pre-large concept was proposed to keep more unpromising information in the buffer to avoid multiple database scans for handling transaction insertion [11], transaction deletion [12], and transaction modification [13]. A pre-large itemset is not truly large (frequent), but has a high probability to be large after the database is updated. Two support thresholds of S_u and S_l are respectively set to mine large (frequent) or pre-large itemsets in ARM. As the pre-large itemsets can be used as a buffer to reduce the movement of an itemset directly from small to large or the other way round, the very time-consuming computations of multiple database scans can be greatly reduced. Let the count difference (CD) be the changes between the transactions before and after modification, which can be calculated as a positive, zero or negative value. When transactions are modified in a database, three parts with nine cases [13] are created as shown in Fig. 1.

The itemsets in case 1 remain large when the database is updated. The itemsets in cases 2, 5, and 8 remain their original situation since the CD did not change (zero) after the database was updated, which will not affect the final frequent itemsets. Some itemsets in case 3 may become invalid, which will then be removed from the large itemsets. Some frequent itemsets thus arise in cases 4 and 7. The itemsets can be easily handled if the pre-large itemsets are kept in cases 4 and 6. The itemsets in cases 6 and 9 cannot be large since they have negative CDs. In the maintenance process, the ratio of the modified transactions in the original database is usually very small. An itemset in case 7 cannot possibly be large after the database is updated as long as the number of modified transactions is smaller than the safety bound f [13] as:

$$f = \lfloor (S_u - S_l) \times |D| \rfloor, \tag{1}$$

where S_u is the upper threshold, S_l is the lower threshold, and $|D|$ is the number of transactions in the original database.

Note that only occurrence frequency was considered in ARM, while both quantity and profit factors are considered to reveal HUIs, HUIM is more complicated than ARM. Although the pre-large concept in ARM can be used to reduce the computational cost of multiple database scans, it is insufficient to directly apply the pre-large concept to maintain and update the discovered HUIs. Hence, it is not a trivial task to maintain and update the discovered HUIs with transaction modification in the dynamic environment. In this paper, we develop an efficient method for transaction modification in HUIM, which is more practical and realistic in real-life situations.

3. Preliminaries and problem statement

In this section, the preliminaries and problem statement related to our proposed approach are defined as follows.

3.1. Preliminaries

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of items in database D, with each item i_j having corresponding profit $p(i_j)$. An itemset $X \in I$ with k distinct items has length k and is referred to as a k-itemset. The transitional database is denoted as $D = \{T_1, T_2, \ldots, T_n\}$, where $T_d \in D$. The quantity $q(i_j, T_d)$ is the sold quantity of item i_j in transaction T_d. An example database and its profit table are respectively shown in Tables 1 and 2. As an example of TID 1 (transaction 1) in

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

Table 4
Transactions before and after modification.

TID	Before modification	After modification
6	(*A*:1), (*D*:1), (*E*:1)	(*A*:2), (*B*:1), (*F*:1), (*G*:9)

Table 5
Parameters of used datasets.

|*D*|	Total number of transactions
AvgLen	Average transaction length
|*I*|	Number of distinct items
Type	Dataset type (sparse or dense)

Table 6
Characteristics of used datasets.

Dataset	|*D*|	AvgLen	|*I*|	Type
Foodmart	21,556	4.4	1559	Sparse
Retail	88,162	10.3	16,470	Sparse
Accidents	340,183	33.8	458	Dense
T10I4D100K	100,000	10.1	870	Sparse

Table 1: an item (*A*) is purchased with two quantities; an item (*B*) is purchased with three quantities; and an item (*D*) is purchased with one quantity. The profit value of (*A*) is set as 10, which indicates that the retailer gets 10 profit if one unit of item (*A*) is sold.

Definition 1 (*Utility of an item in transaction*). The utility of an item i_j in T_d is denoted as $u(i_j, T_d)$, which can be defined as:

$$u(i_j, T_d) = q(i_j, T_d) \times p(i_j), \tag{2}$$

in which $q(i_j, T_d)$ is the quantity of an itemset i_j in T_d, and $p(i_j)$ is the profit of an itemset i_j.

For the given example in Table 1, the utility of (*A*) in TID (= 1) is calculated as: $u(A,T_1) = q(A, T_1) \times p(A)$ (= 2 × 10) (= 20).

Definition 2 (*Utility of an itemset in transaction*). The utility of an itemset *X* in transaction T_d is denoted as $u(X, T_d)$, which can be defined as:

$$u(X, T_d) = \sum_{i_j \in X \wedge X \subseteq T_d} u(i_j, T_d). \tag{3}$$

For the given example in Table 1, the utility of (*AB*) in TID (= 1) is calculated as: $u(AB, T_1) = u(A, T_1) + u(B, T_1) = q(A, T_1) \times p(A) + q(B, T_1) \times p(B)$ (= 2 × 10 + 3 × 3) (= 29).

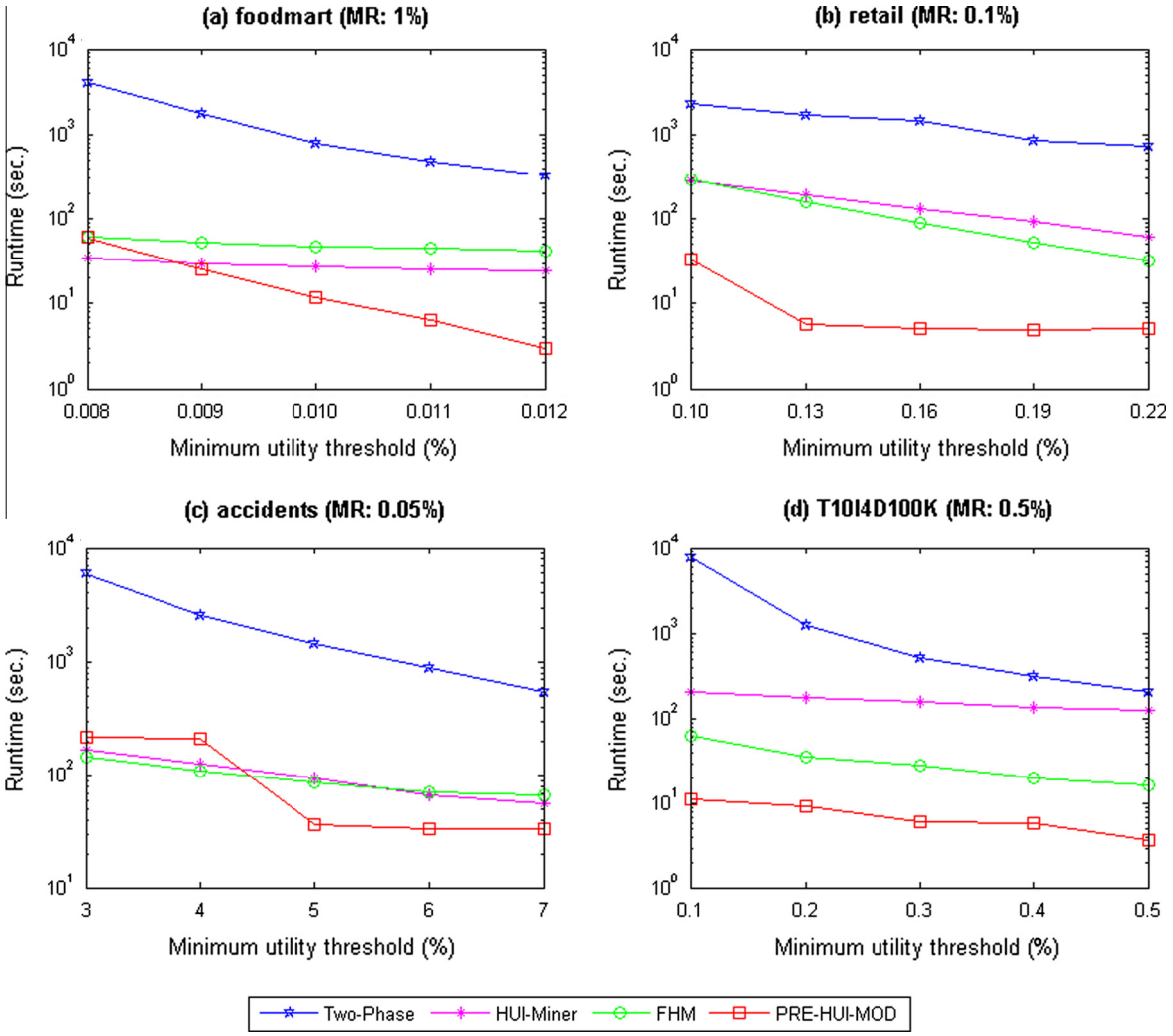

Fig. 2. Experiments of runtime under various MUs.

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

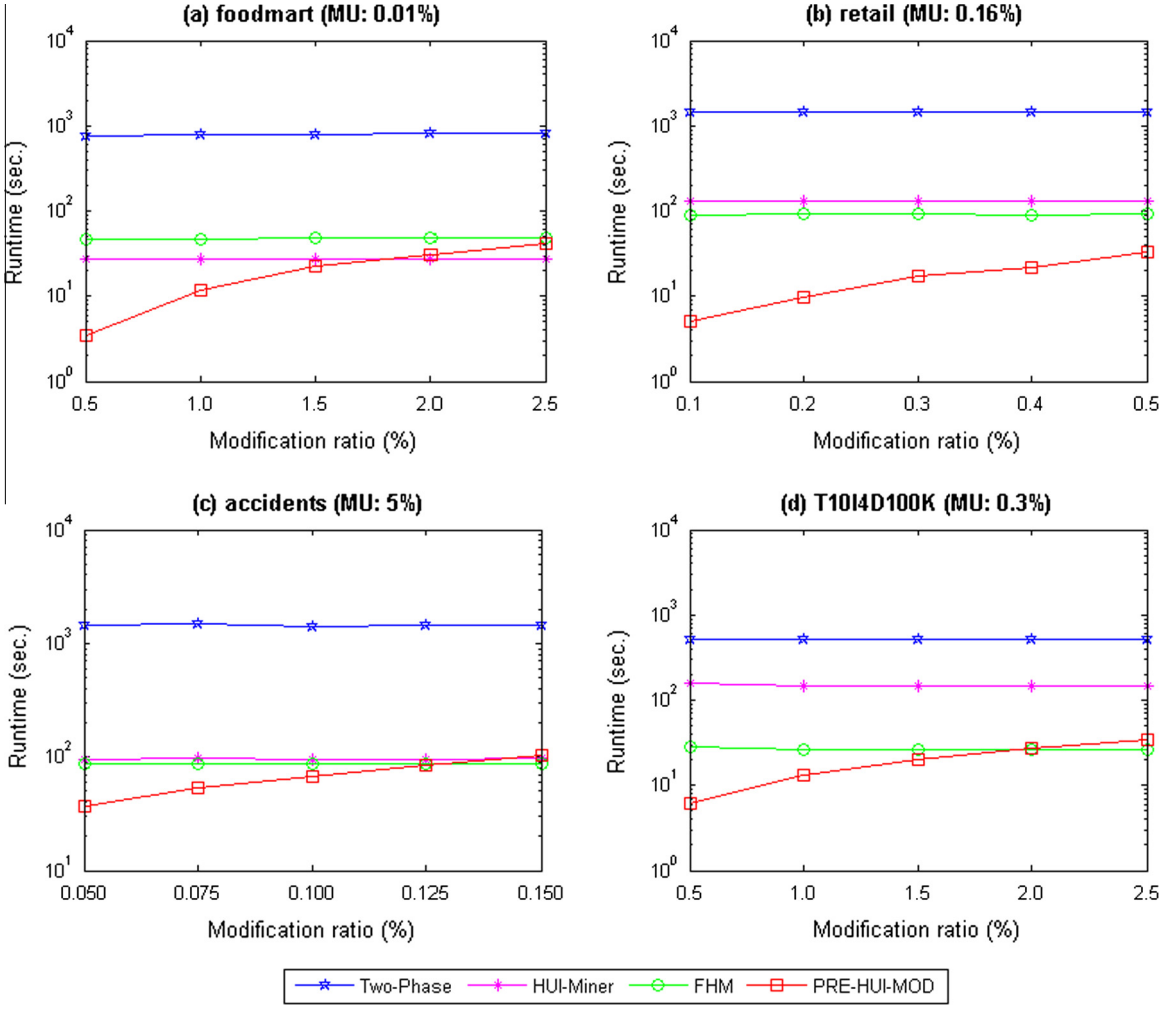

Fig. 3. Runtime experiments under various MRs.

Definition 3 (*Transaction utility*). The transaction utility of transaction T_d is denoted as $TU(T_d)$ and defined as:

$$TU(T_d) = \sum_{j=1}^{m} u(i_j, T_d), \tag{4}$$

where m is the number of items in T_d.

For the given example in Table 1, the transaction utility in TID (= 1) is calculated as: $TU(T_1) = u(A, T_1) + u(B, T_1) + u(D, T_1)$ (= 20 + 9 + 25) (= 54). The transaction utilities of the running example in Table 1 are shown in Table 3.

Definition 4 (*Total utility of a database*). The total utility TU^D is the sum of all transaction utilities in D, which can be defined as:

$$TU^D = \sum_{T_d \in D} TU(T_d). \tag{5}$$

Hence, the total utility in D is calculated as: TU^D = (54 + 53 + 60 + 64 + 43 + 37 + 11 + 46) (= 368).

Definition 5 (*High-utility itemset, HUI*). An itemset X is a high-utility itemset (HUI) in database D if its utility in D is no less than minimum utility count as:

$$HUI \leftarrow \left\{ X \middle| \sum_{X \subseteq T_d \wedge T_d \in D} u(X, T_d) \geqslant TU^D \times \varepsilon \right\}, \tag{6}$$

where ε is the minimum high utility threshold in the original database.

Taking an example that runs in both Tables 1 and 3, and supposing that the minimum utility threshold ε is set at 30%. An item (A) is not considered as a HUI since it appears in TID (=1, 2, 4, 5, 6, 8); the utility of (A) is calculated as $u(A) = u(A, T_1) + u(A, T_2) + u(A, T_4) + u(A, T_5) + u(A, T_6) + u(A, T_8)$ (= 20 + 20 + 30 + 10 + 10 + 10) (= 100), which is less than the minimum utility count as (0.3 × 368) (= 110.4). An itemset (AD) is considered as a HUI since (AD) appears in TID (= 1, 2, 4, 5, 6, 8); the utility of (AD) is calculated as $u(AD) = u(AD, T_1) + u(AD, T_2) + u(AD, T_4) + u(AD, T_5) + u(AD, T_6) + u(AD, T_8)$ (= 45 + 45 + 55 + 35 + 35 + 35) (= 250), which is larger than the minimum utility count (250 > 110.4).

From the above example, it can be observed that an item (A) is not a HUI but its superset of an itemset (AD) is concerned as a HUI. Thus, the downward closure (DC) property of ARM is not suitable for HUIM. Liu et al. presented a Two-Phase model [19] and the transaction-weighted downward closure (TWDC) property to first generate the high transaction-weighted utilization itemsets (HTWUIs), and then rescan the original database for the remaining

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

HTWUIs to produce the actual HUIs. The definition of Two-Phase model is given below.

Definition 6 (*Transaction-weighted utility of an itemset*). The transaction-weighted utility of an itemset X is the sum of all transaction utilities $TU(T_d)$ containing itemset X in D, which is defined as:

$$TWU^D(X) = \sum_{X \subseteq T_d \wedge T_d \in D} TU(T_d). \quad (7)$$

Definition 7 (*High transaction-weighted utilization itemset, HTWUI*). An itemset is called a high transaction-weighted utilization itemset (HTWUI) if its transaction-weighted utility (TWU) is no less than the minimum utility count as:

$$HTWUI \leftarrow \{X | TWU^D(X) \geqslant \varepsilon \times TU^D\} \quad (8)$$

For example, the transaction-weighted utility of (A) is calculated as $TWU^D(A) = TU(T_1) + TU(T_2) + TU(T_4) + TU(T_5) + TU(T_6) + TU(T_8)$ (= 54 + 53 + 64 + 43 + 37 + 64) (= 297); an item (A) is concerned as a HTWUI since $TWU^D(A)$ (= 297 > 110.4).

3.2. Problem statement

Given a transactional quantitative database D, a user-specified profit table *p-table*, and a minimum high utility threshold ε, the changed parts in D before and after modification are respectively denoted as T and T'. The maintenance problem with transaction modification in HUIM is to efficiently maintain and update the set of the discovered high-utility k-itemsets as:

$$HUI \leftarrow \left\{ X \middle| \sum_{X \subseteq T_d \wedge T_d \in D} u(X, T_d) \geqslant \left(TU^D + TU^{T'} - TU^T \right) \times \varepsilon \right\} \quad (9)$$

4. Proposed maintenance algorithm with transaction modification

In this section, a new pre-large strategy is designed to maintain and update the discovered HUIs in HUIM. Since both profit and quantity factors are concerned in HUIM, it is a different way to maintain the discovered information when compared to the pre-large concept in ARM. The notations used in the proposed algorithm are given below.

4.1. Notations

D	Original quantitative database, $D = \{T_1, T_2, \ldots, T_n\}$, in which each transaction T_j contains several items with sold quantities
I	A set of m items, $I = \{i_1, i_2, \ldots, i_m\}$, in which each item i_j has its own profit value $p(i_j)$ and quantity $q(i_j, T_d)$ in transaction T_d
T	Transactions before modification
T'	Transactions after modification
U	Entire updated database, i.e., $D - T + T'$
TU^D	Total utility of the transactions in D
TU^T	Total utility of the transactions before modification in T
$TU^{T'}$	Total utility of the transactions after modification in T'
TU^M	Total utility of the transactions before and after modification
TU^U	Total utility of the transactions in U
buf	Stores the total utility of the last processed transactions for transaction modification. It is set at 0 when the database is rescanned
X	An itemset containing k distinct items
S_u	Upper utility threshold for large (high) transaction-weighted utilization and high-utility itemsets. It is the same as the high utility threshold in traditional utility mining
S_l	Lower utility threshold for prelarge transaction-weighted utilization and prelarge utility itemsets, $S_u > S_l$
f	Safety transaction utility bound for transaction modification
C_r	Set of candidate r-itemsets
$TWU^D(X)$	Transaction-weighted utilization of an itemset X in the original database
$TWU^M(X)$	Transaction-weighted utilization of an itemset X before and after modification;
$TWU^U(X)$	Transaction-weighted utilization of an itemset X in the updated database
$HTWUI_r^D$	Set of large (high) transaction-weighted utilization r-itemsets in the original database
$HTWUIs^D$	Set of large (high) transaction-weighted utilization itemsets in the original database
$PTWUI_r^D$	Set of prelarge transaction-weighted utilization r-itemsets in the original database
$PTWUIs^D$	Set of prelarge transaction-weighted utilization itemsets in the original database
$HTWUI_r^U$	Set of large (high) transaction-weighted utilization r-itemsets in the updated database
$HTWUIs^U$	Set of large (high) transaction-weighted utilization itemsets in the updated database
$PTWUI_r^U$	Set of prelarge transaction-weighted utilization r-itemsets in the updated database
$PTWUIs^U$	Set of prelarge transaction-weighted utilization itemsets in the updated database
$u(X)$	Actual utility of an itemset X in the updated database
$HUIs^U$	Set of high-utility itemsets in the updated database

4.2. Maintenance strategy of HUIM

In real-life applications, transactions do not remain in a static state, but instead they are dynamically inserted, deleted, or modified in their original databases. Intuitively, transaction modification [13] can be simply considered as transaction insertion followed by transaction deletion. A conventional way for handling transaction modification requires two steps, which is inefficient in maintaining and updating the discovered information. The pre-large concept of transaction modification in ARM indicates that the small itemsets could not be large, and or the other way around, when the number of accumulative modified transactions is less than the designed safety bound. Since the pre-large concept of ARM cannot be directly applied to HUIM for transaction modification, it is necessary to develop a new approach to maintain and update the discovered HUIs for transaction modification in HUIM. The designed safety bound of the total utility in the modified transactions can be obtained as:

$$TU^M = f \leqslant (S_u - S_l) \times TU^D, \quad (10)$$

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

where TU^M is the total utility for the modified transactions, and TU^D is the total utility in the original database.

Definition 8. Transactions before modification are projected and denoted as T; Transactions after modification are also projected and denoted as T'.

In this example, the transactions before and after modification are then shown in Table 4.

Definition 9. The total utility in the modified transactions is denoted as TU^M, which is defined as:

$$TU^M = TU^{T'} - TU^T. \tag{11}$$

In this example, the total utility before transaction modification is calculated as TU^T (= 1 × 10 + 1 × 25 + 1 × 2) (= 37). The total utility after transaction modification is calculated as $TU^{T'}$ (= 2 × 10 + 3 × 3 + 1 × 3 + 9 × 5) (= 77). The TU^M is then calculated as TU^M (= 77 − 37) (= 40).

Definition 10. The modified TWU^M of an itemset X for the modified transactions is the TWU difference of an itemset X before (in T) and after modification (in T'), which is defined as:

$$TWU^M(X) = TWU^{T'}(X) - TWU^T(X). \tag{12}$$

Definition 11. The updated TWU^U of an itemset X for the updated database is the TWU^D in the original database plus TWU^M, which is defined as:

$$TWU^U(X) = TWU^D(X) + TWU^M(X). \tag{13}$$

Definition 12. The total utility for the updated database (TU^U) is calculated as the total utility in the original database (in D) plus the total utility for the modified transactions, which is defined as:

$$TU^U = TU^D + TU^M = TU^D + TU^{T'} - TU^T. \tag{14}$$

Lemma 1. *An itemset X is small in the original database (D) (not existing in HTWUI^D), but is HTWUI^M in the modified transactions (M), then it will not be a HTWUI^U in the updated database (U).*

Proof. For an original database, an itemset X is small transaction-weighted utilization itemset if $\frac{TWU^D(X)}{TU^D} < S_l$. For the modified transactions, an itemset X is a HTWUI in M if $\frac{TWU^M(X)}{TU^M} \geqslant S_u$. For the updated database, the TWU value of an itemset X becomes $TWU^U(X) = TWU^D(X) + TWU^M(X)$. Therefore:

$$\frac{TWU^U(X)}{TU^U} = \frac{TWU^D(X) + TWU^M(X)}{TU^D + TU^M}, \quad \text{since} \quad TWU^D(X) < S_l \times TU^D,$$

$$\frac{TWU^D(X) + TWU^M(X)}{TU^D + TU^M} < \frac{S_l \times TU^D + TWU^M(X)}{TU^D + TU^M}.$$

□

From the above definitions, it can be obtained that:

$$TU^M = f \leqslant (S_u - S_l) \times TU^D \Rightarrow TU^M \leqslant S_u \times TU^D - S_l \times TU^D$$
$$\Rightarrow TU^M + S_l \times TU^D \leqslant S_u \times TU^D$$

Since $TWU^M(X) < TU^M$, thus:

$$\frac{TWU^D(X) + TWU^M(X)}{TU^D + TU^M} < \frac{S_l \times TU^D + TU^M}{TU^D + TU^M} < \frac{S_u \times TU^D}{TU^D + TU^M} < S_u.$$

Lemma 2. *An itemset X is small both in the original database (D) and in the modified transactions (M), then it will still remain small in the updated database (U).*

Proof. For an original database, an itemset X is small transaction-weighted utilization itemset if $\frac{TWU^D(X)}{TU^D} < S_l$. For the modified transactions M, an itemset X is a small transaction-weighted utilization itemset if $\frac{TWU^M(X)}{TU^M} < S_l$. For the updated database, the TWU value of an itemset X becomes $TWU^U(X) = TWU^D(X) + TWU^M(X)$. Therefore:

$$\frac{TWU^U(X)}{TU^U} = \frac{TWU^D(X) + TWU^M(X)}{TU^D + TU^M} \leqslant S_u$$

Since $TWU^D(X) < S_l \times TU^D$, and $TWU^M(X) < S_l \times TU^M$, thus:

$$\frac{TWU^D(X) + TWU^M(X)}{TU^D + TU^M} < \frac{S_l \times TU^D + S_l \times TU^M}{TU^D + TU^M}.$$

□

From the above definitions, it can be obtained that $TU^M + S_l \times TU^D \leqslant S_u \times TU^D$, and $S_l \times TU^M < TU^M$, thus:

$$\frac{TWU^D(X) + TWU^M(X)}{TU^D + TU^M} < \frac{S_l \times TU^D + TU^M}{TU^D + TU^M} < \frac{S_u \times TU^D}{TU^D + TU^M} < S_u.$$

Based on the developed pre-large strategy for HUIM, the small transaction-weighted utilization itemsets could not be HTWUIs in the updated database until the number of accumulative total utility for the modified transactions achieves the designed safety bound.

5. Proposed PRE-HUI-MOD algorithm

Details of the proposed algorithm for transaction modification in HUIM is described in Algorithm 1.

Algorithm 1. Proposed PRE-HUI-MOD algorithm

INPUT: D, T, T', TU^D, $HTWUIs^D$, $PTWUIs^D$, S_u and S_l.
OUTPUT: U (= $D - T + T'$), $HTWUIs^U$, $PTWUIs^U$, and $HUIs^U$.
1. set buf := 0.
2. compute $f := (S_u - S_l) \times TU^D$.
3. scan T and T' to compute TU^T and $TU^{T'}$.
4. compute $TU^M := TU^{T'} - TU^T$.
5. compute $TU^U := TU^D + TU^M$.
6. **if** ($buf + TU^M$) $\leqslant f$ **then**
   $buf := buf + TU^M$.
   for each X 1-itemset in T and T' **do**
   find $TWU^T(X)$ and $TWU^{T'}(X)$.
   compute $TWU^M(X) := TWU(X)^{T'} - TWU(X)^T$.
   end for
   $C_1 \leftarrow X$.
   else
   set $U \leftarrow D - T + T'$.
   scan U to find $HTWUIs$, $PTWUIs$, and $HUIs$ in a level-wise way.
   set $HTWUIs^D \leftarrow HTWUIs^U$.
   set $PTWUIs^D \leftarrow PTWUIs^U$.
   set $HUIs^D \leftarrow HUIs^U$.
   set buf := 0.
   terminate algorithm.
   end if

(*continued on next page*)

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

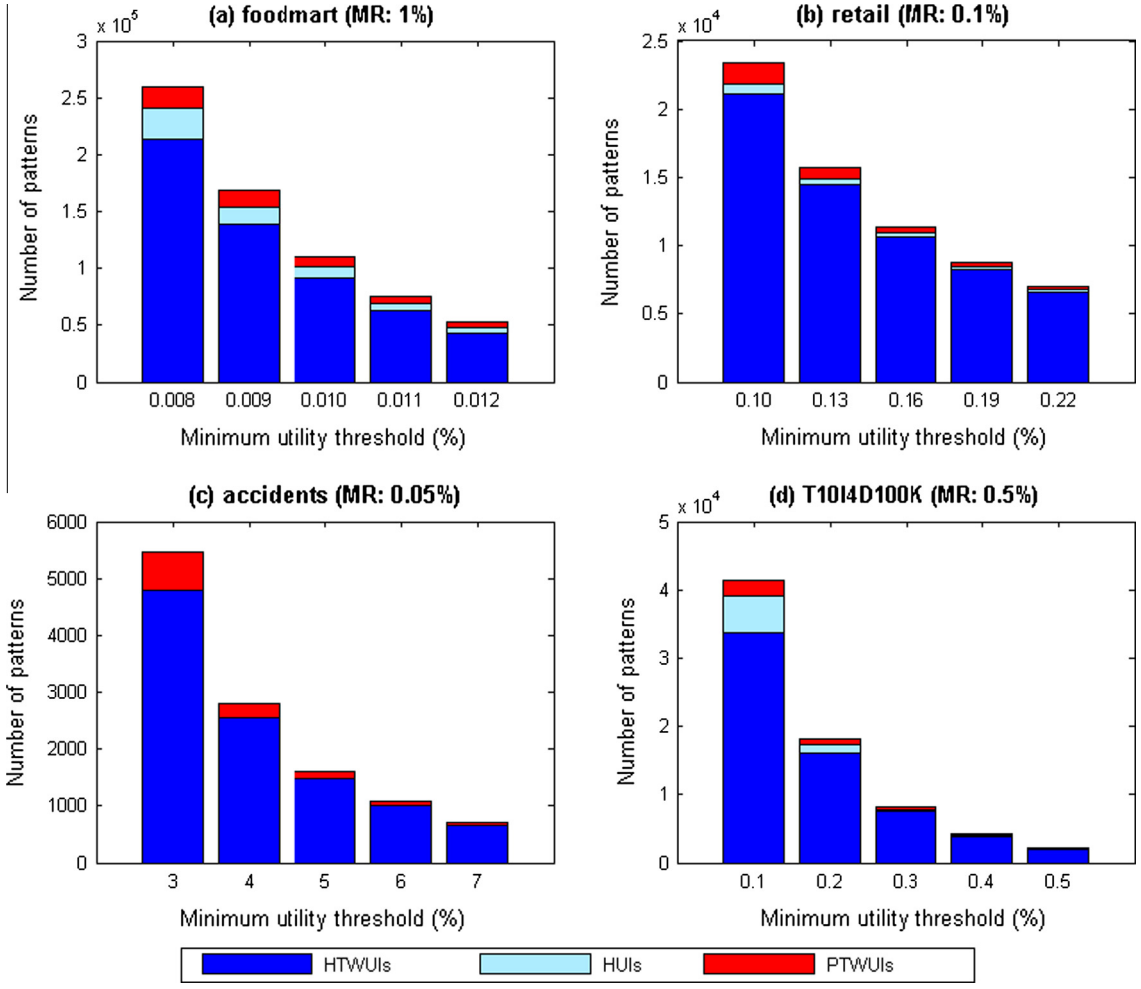

Fig. 4. Experiments of number of patterns under various MUs.

7. set $r := 1$.
8. **for** each r-itemset of X in C_r **do**
   if $(X \in HTWUI_r^D, PTWUI_r^D)$ **then**
   compute $TWU^U(X) := TWU^D(X) + TWU^M(X)$.
   if $(TWU^U(X) \geqslant TU^U \times S_u)$ **then**
   $HTWUI_r^U \leftarrow X$.
   else if $(TU^U \times S_l \leqslant TWU^U(X) < TU^U \times S_u)$ **then**
   $PTWUI_r^U \leftarrow X$.
   end if
   else
   $R\_set \leftarrow X$.
   end if
   end for
9. **if** $(R\_set \neq \textbf{\textit{null}})$ **then**
   for each X in R_set **do**
   scan D to find $TWU^D(X)$.
   compute $TWU^U(X) := TWU^D(X) + TWU^M(X)$.
   if $(TWU^U(X) \geqslant TU^U \times S_u)$ **then**
   $HTWUI_r^U \leftarrow X$.
   else if $(TU^U \times S_l \leqslant TWU^U(X) < TU^U \times S_u)$ **then**
   $PTWUI_r^U \leftarrow X$.
   end if
   end for
   end if
10. set $r := r + 1$.
11. $C_r \leftarrow HTWUI_{r-1}^U \cup PTWUI_{r-1}^U$.
12. repeat STEPs 8 to 11 until $C_r :=$ ***null***.
13. **for** $(HTWUI_k^U \leftarrow k := 1, r)$ **do**
    scan U to compute $u(X)$ from $HTWUI_k^U$.
    if $(u(X) \geqslant TU^U \times S_u)$ **then**
    $HUI_k^U \leftarrow X$.
    end if
    end for
14. $HTWUIs^U \leftarrow \cup\, HTWUIs_k^U$.
15. $PTWUIs^U \leftarrow \cup\, PTWUIs_k^U$.
16. $HUIs^U \leftarrow \cup\, HUIs_k^U$.
17. set $HTWUIs^D \leftarrow HTWUIs^U$.
18. set $PTWUIs^D \leftarrow PTWUIs^U$.
19. set $HUIs^D \leftarrow HUIs^U$.

In the proposed PRE-HUI-MOD algorithm, it takes input as D, T, T', TU^D, $HTWUIs^D$, $PTWUIs^D$, S_u and S_l. The developed safety bound is first calculated to determine whether the updated database needs to be rescanned (Step 2). The total utility before and after transaction modification is also calculated (Step 4). Afterwards, the total utility in the updated database is then found (Step 5).

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

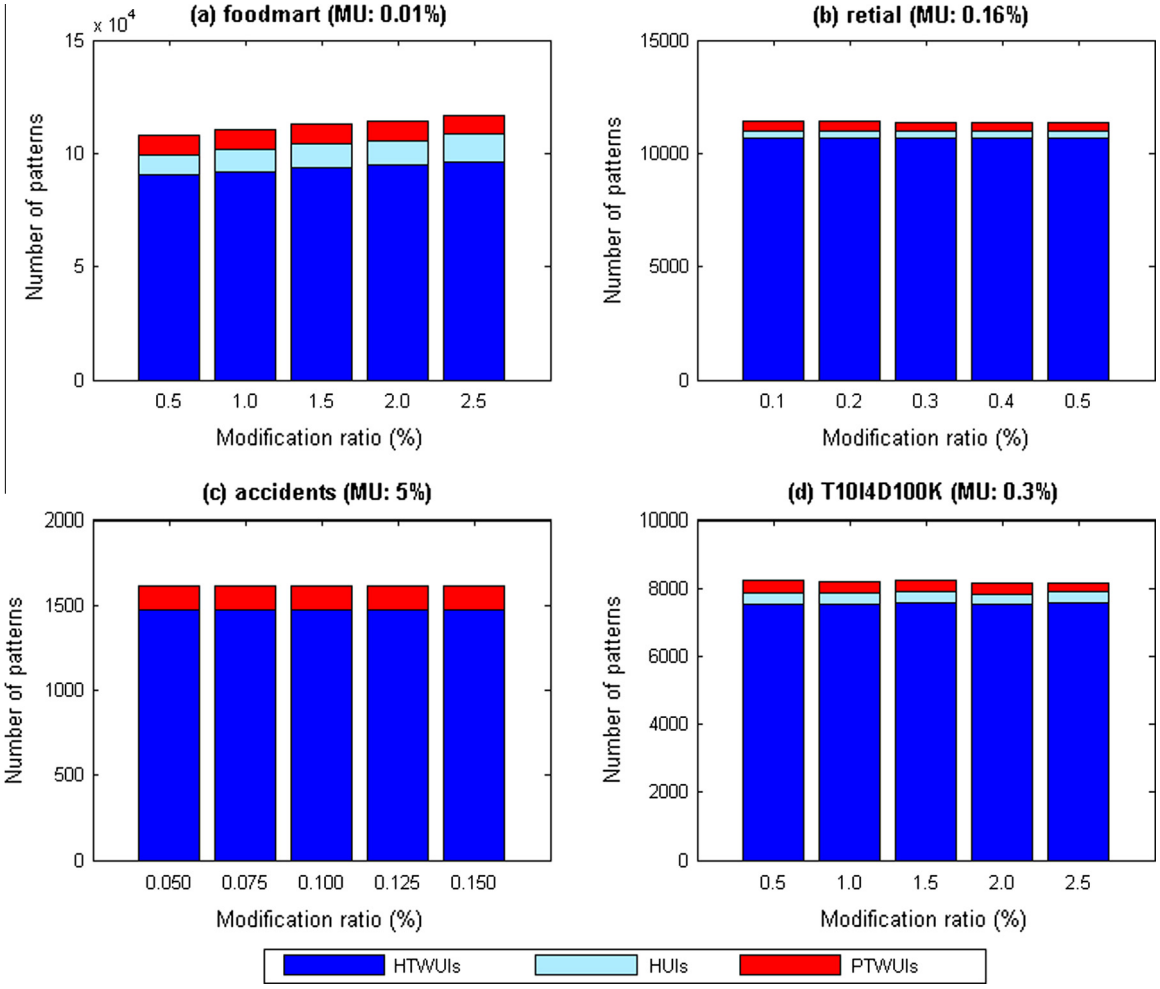

Fig. 5. Experiments of number of patterns under various MRs.

The 1-itemsets in the transactions before and after modification are then derived to find their TWU values (Step 6). When the total utility in the modified transactions archives the designed safety bound, the database is then required to be rescanned with batch-mode processing for mining HUIs (Step 6). The derived candidate 1-itemsets are then determined with the discovered $HTWUI_1^D$ and $PTWUI_1^D$ in a level-wise way to check whether they are larger than or equal to the minimum utility count, or lie between the two minimum utility counts (Step 8). If the derived candidate 1-itemset from step 6 does not satisfy the condition in step 8, it is then put into the set of *R_set* for later processing. If *R_set* is not ***null***, it is necessary to rescan the original database to find the *TWU* values of itemsets existing in *R_set* (Step 9). A similar determination procedure to step 6 is then processed to determine whether the processed itemsets satisfy the condition as the high transaction-weighted utilization 1-itemsets (HTWUIs) and pre-large transaction-weighted utilization 1-itemsets (PTWUIs). After the 1-itemsets of HTWUIs and PTWUIs are completely determined, the Apriori-like algorithm is then processed to level-wisely produce the HTWUIs (Steps 11 to 12). After that, the remaining HTWUIs are then processed to find their actual HUIs (Step 13). The discovered $HTWUIs^U$, $PTWUI^Us$ and $HUIs^U$ at each level are then merged (Steps 14 to 16). Finally, the discovered information will be used as next iteration for transaction modification (Steps 17 to 19).

5.1. Experimental evaluation

Note that mining HUIs from dynamic databases with transaction modification has not yet been proposed before. In this section, the proposed PRE-HUI-MOD algorithm is compared to three batch-mode algorithms, including the well-known Two-Phase algorithm [19], the HUI-Miner [18], and the state-of-the-art FHM [10] algorithms, in terms of runtime, memory consumption, and the number of the discovered patterns. Since the Two-Phase algorithm, HUI-Miner, and FHM algorithms were designed to handle a static database, they performed in a conventional way to rescan an updated database each time after transactions are consequentially modified from an original database (our dynamic environment setting). The Two-Phase algorithm used in the experiments has been improved by our designed granular mechanism, which can be used to speed up the computations [14]. The HUI-Miner and FHM are both the state-of-the-art algorithms that mine HUIs in batch mode, so it is unnecessary to perform other HUIM algorithm comparisons in our experiments such as $IHUP_{TWU}$ [5] and UP-growth+ [23]. The experimental descriptions and results are described below.

5.2. Experimental environment

The four compared algorithms of the Two-Phase model, the HUI-Miner, the FHM, and the proposed PRE-HUI-MOD in the

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

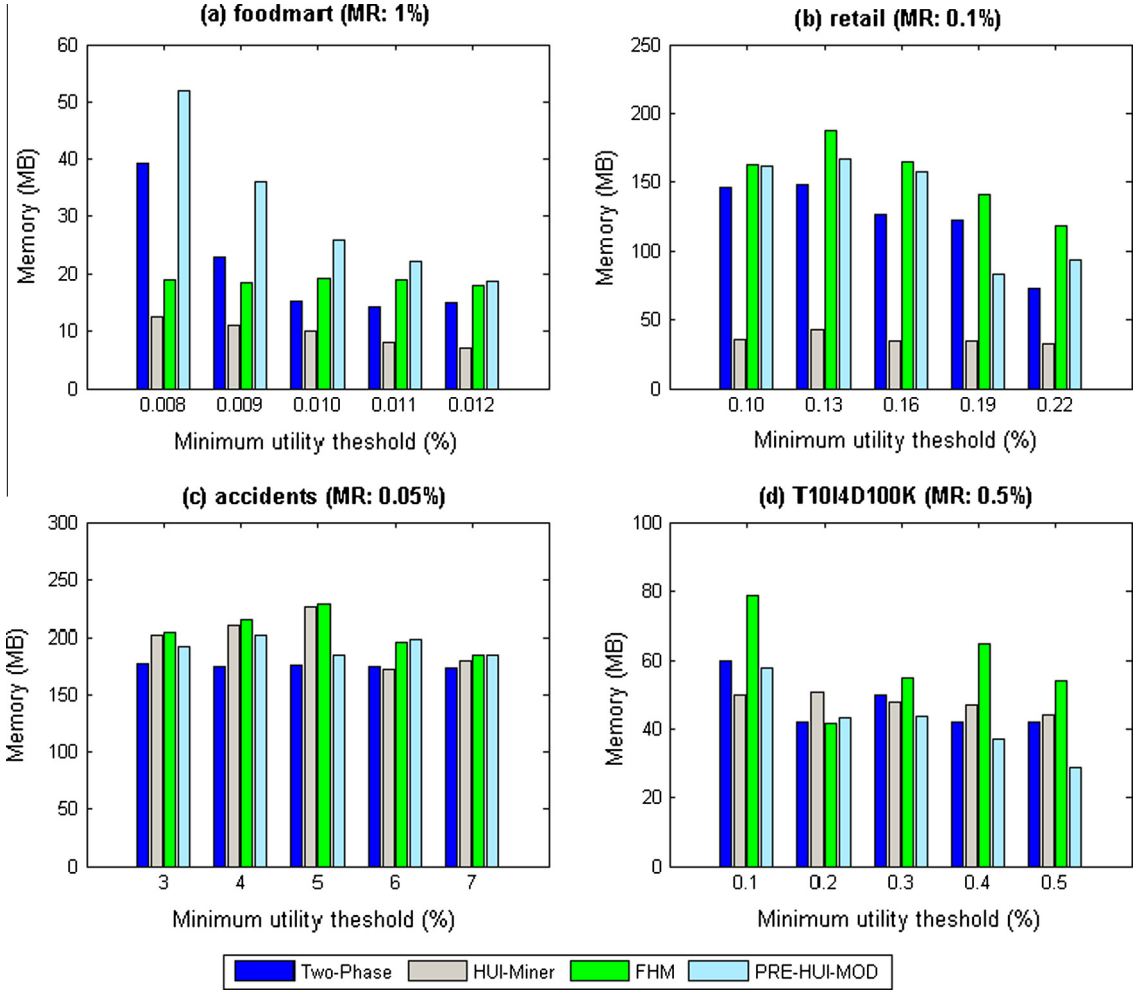

Fig. 6. Experiments of memory consumption under various MUs.

experiments were implemented in the Java computer programming language on a personal computer with an Intel 2.6-GHz dual-core processor and 4 GB of RAM, running the 32-bit Microsoft Windows 7 operating system. The three real-life foodmart [21], accidents [1], and retail [1] datasets, and one synthetic T10I4D100K dataset [4], were used to evaluate the performance of the proposed PRE-HUI-MOD algorithm when compared to those of the Two-Phase, HUI-Miner and the FHM algorithms. The foodmart dataset was collected from an anonymous chain store. It is a quantitative dataset of the products sold by the chain store. The accidents dataset contains anonymous traffic accident data on a public road in Belgium. The retail dataset contains anonymous retail market basket data from a retail store in Belgian. The synthetic T10I4D100K and T10I4N4KD|X|K datasets were generated by IBM Quest Synthetic Data Generation Code. With the exception of the foodmart dataset, both quantity (internal) and profit (external) values are assigned to the items in the accident, retail, T10I4D100K and T10I4N4KD|X|K datasets by Liu's simulation model [19]. The parameters and characteristics are respectively shown in Tables 5 and 6.

In the conducted experiments, a modification ratio (MR) was set for the number of modified transactions in the datasets. Note that the related MR of each dataset was given at the top of all figures in experiments. The transactions for modification ($|D| \times$ MR) were consequentially selected bottom-up from the original dataset, which will be replaced and modified to top-down transactions from the original dataset. For example, the dataset size of foodmart is 21,556, and its MR is set as 1% (shown in Fig. 3(a)). The bottom-up (21,556 × 1%) (= 215) transactions from the foodmart dataset are selected to be modified as top-down (21,556 × 1%) (= 215) transactions. The next 215 bottom-up transactions from 21125th to 21340th are consequentially selected to be modified as the top-down 216th to 431th transactions. Five iterations of transaction modification are performed for each algorithm. Note that the number of iterations can be defined according to the user's preference. The other datasets adopted the given MR settings for the results of Figures in the dynamic environment.

5.3. Runtime

The runtime of the proposed PRE-HUI-MOD algorithm is then compared to those of the Two-Phase, HUI-Miner and the FHM algorithms in the four datasets. Since the Two-Phase model [19], HUI-Miner [18] and the FHM [10] algorithms are performed in batch mode, they have to rescan the original dataset and re-mine for HTWUIs or HUIs after each transaction modification. The proposed PRE-HUI-MOD only needs to maintain and update the discovered HTWUIs and PTWUIs for the later HUI mining process. The PTWUIs are kept for maintenance, and hence, it is unnecessary to rescan the original dataset until the accumulative total utility for transaction modification achieves the safety bound. The conducted

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

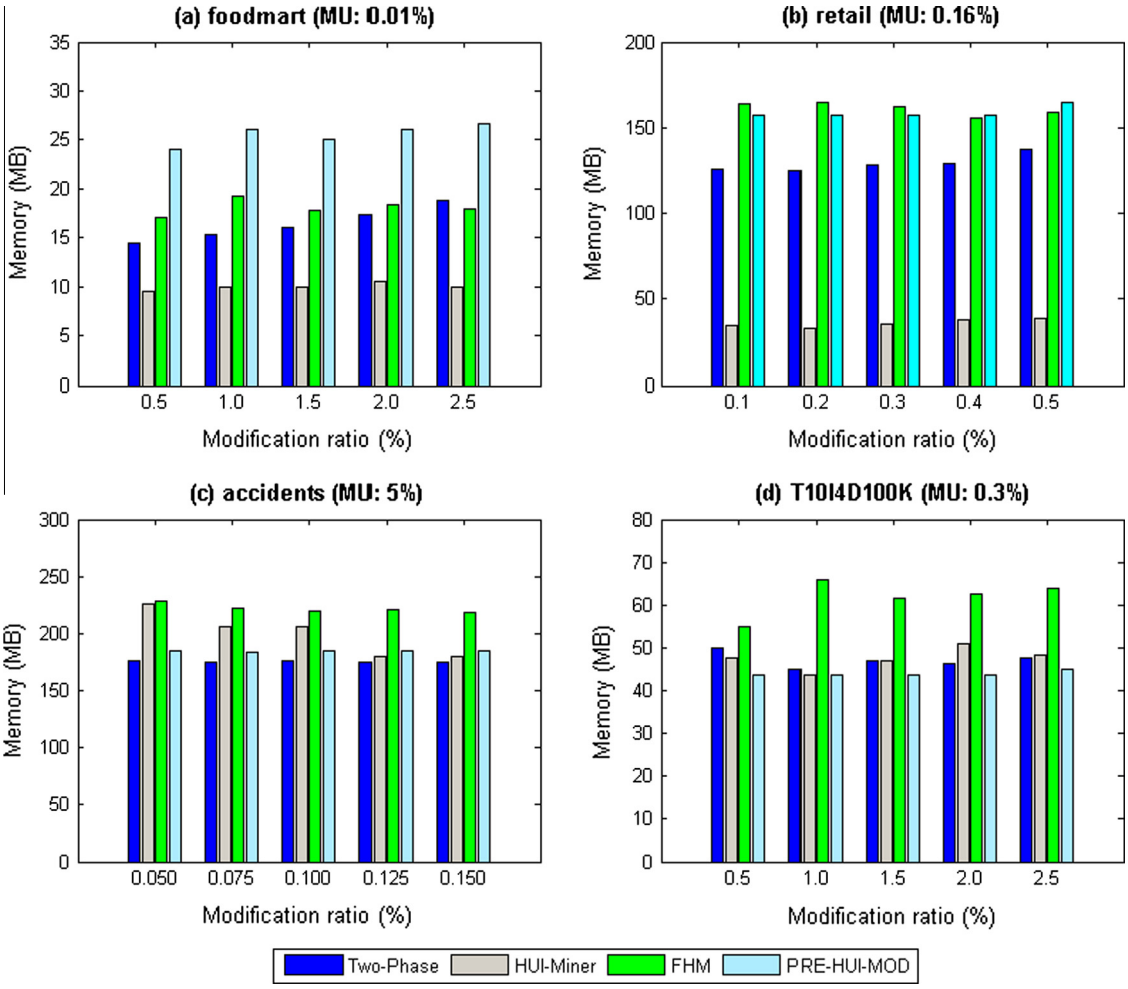

Fig. 7. Experiments of memory consumption under various MRs.

experiments for runtime under various minimum utility thresholds (MUs) with a fixed MR are shown in Fig. 2.

From Fig. 2, it can be observed that the proposed PRE-HUI-MOD algorithm has better performance compared to those of Two-Phase, HUI-Miner and FHM algorithms in the four datasets. The batch-mode FHM algorithm has better performance compared to those of other two batch-mode algorithms, except in the foodmart dataset as shown in Fig. 2(a). For the accidents dataset shown in Fig. 2(c), the proposed PRE-HUI-MOD algorithm requires more runtime under the lower minimum utility threshold compared to the HUI-Miner and FHM algorithms. The PRE-HUI-MOD algorithm is almost two orders of magnitude faster than the other algorithms as it is unnecessary to rescan the original dataset until the accumulative total utility in the modified transactions achieves the designed safety bound based with the pre-large concept. The conducted experiments for runtime under various modification ratios with a fixed minimum utility threshold (MU) are shown in Fig. 3. Since the datasets have different characteristics, the various MUs are respectively set at 0.01% for foodmart dataset, 0.15% for retail dataset, 5% for accidents dataset and 0.3% for T10I4D100K dataset.

In Fig. 3, is can be observed that the proposed PRE-HUI-MOD algorithm still has a better performance than the Two-Phase, HUI-Miner and FHM algorithms under various modification ratios in all datasets. Also, another very interesting result can be observed from Fig. 3 that the runtime of the proposed PRE-HUI-MOD algorithm increases whereas the runtime of the Two-Phase, HUI-Miner and FHM algorithms remains steady, and so the gap between the proposed PRE-HUI-MOD algorithm becomes smaller compared to the other three algorithms along with increasing MR. When the size of transactions for modification is increased, more computations are required to partition the discovered HTWUIs and PTWUIs into several cases for maintenance based on the proposed PRE-HUI-MOD algorithm, which can be observed from Fig. 3; this result indicates that the proposed PRE-HUI-MOD algorithm is more efficient in handling small size transactions for modification. Since only few transactions require daily or monthly modifications, the proposed PRE-HUI-MOD algorithm is much more suitable for real-world applications compared to the other batch-mode algorithms. Besides, it is unnecessary to rescan original dataset until the accumulative total utility number for the modified transactions achieves the safety bound based on the designed PRE-HUI-MOD algorithm. The accumulative value increases along with the increasing transactions size modifications, which indicates that the updated dataset may require rescanning each time.

5.4. Number of discovered patterns

The number of patterns of the proposed PRE-HUI-MOD algorithm is then compared to those of the Two-Phase, HUI-Miner and the FHM algorithms in the four datasets. The Two-Phase algorithm uses HTWUIs to reduce the number of candidates in the mining process, but the HUI-Miner and FHM algorithms adopted the utility-list structure for directly mining HUIs without candidate generation. The proposed PRE-HUI-MOD algorithm adopted not only HTWUIs based on the Two-Phase model, but also the

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

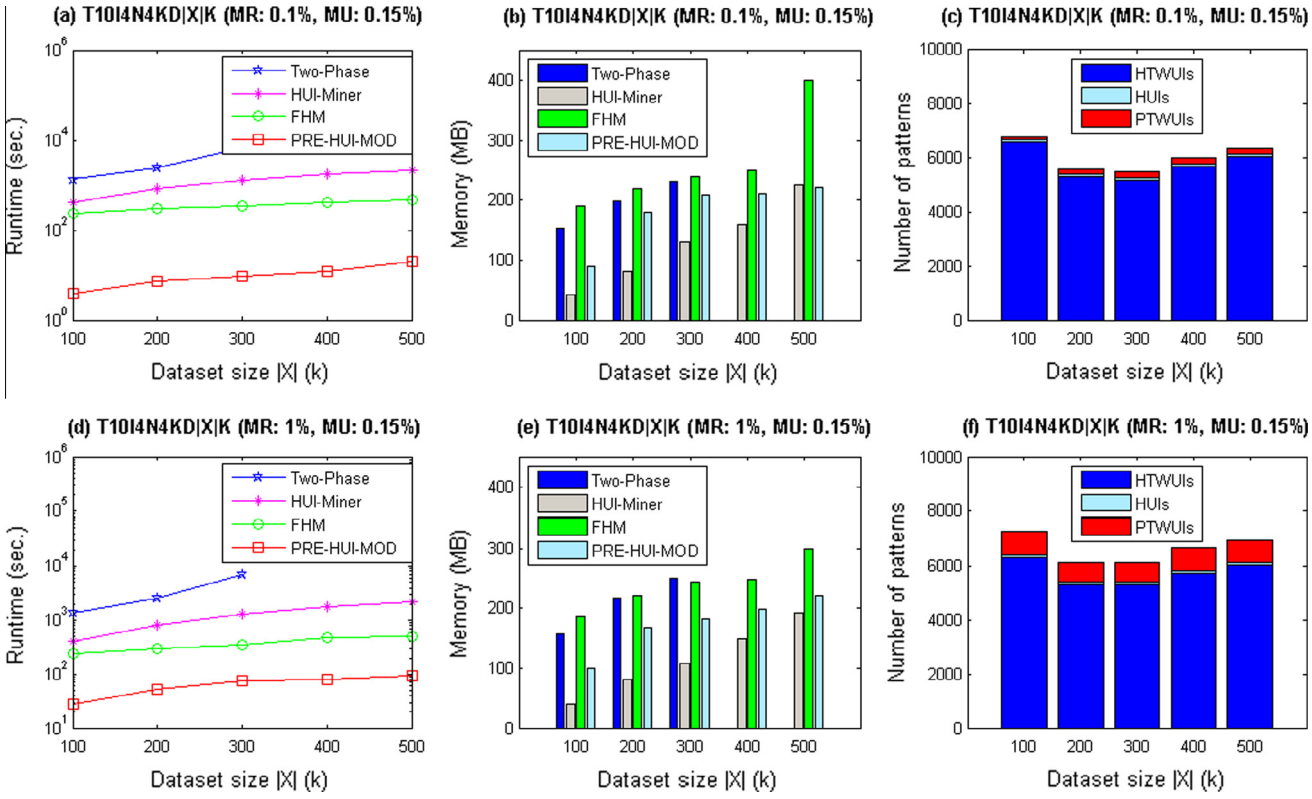

Fig. 8. Experiments of scalability under various dataset sizes.

PTWUIs based on the pre-large concept. The number of HTWUIs, HUIs, and PTWUIs under various MUs with a fixed MR, and also under various MRs with a fixed MU, are then compared and shown in Figs. 4 and 5 respectively.

From Figs. 4 and 5, it can be observed that the number of HTWUIs is very large compared to the number of PTWUIs in the four datasets. From the numerous HTWUIs produced by the Two-Phase and the proposed PRE-HUI-MOD algorithms in the experiments, rare HUIs are then produced from the remaining HTWUIs. Since the HUI-Miner and FHM algorithms can directly generate HUIs, it is unnecessary to generate the candidates of HTWUIs. It can also be observed that fewer PTWUIs are kept based on the proposed PRE-HUI-MOD algorithm, but the computations can be greatly improved compared to those of the Two-Phase, HUI-Miner and FHM algorithms.

5.5. Memory consumption

The memory consumption of the proposed PRE-HUI-MOD algorithm is then compared to those of the Two-Phase, HUI-Miner and the FHM algorithms in the four datasets. The results under various MUs with a fixed MR, and under various MRs with a fixed MU, are respectively shown in Figs. 6 and 7.

From Figs. 6 and 7, it can be observed that the proposed PRE-HUI-MOD algorithm requires a similar amount of memory consumption as the Two-Phase and HUI-Miner algorithms, especially in the accidents and T10I4D100K datasets shown in Figs. 6(c) and (d) and 7(c) and (d). More memory is required of the proposed PRE-HUI-MOD algorithm compared to those of the Two-Phase and HUI-Miner algorithms in the foodmart and retail datasets, which can be observed from Figs. 6(a) and (b), 7(a) and (b). The reason is that the proposed PRE-HUI-MOD algorithm is required to keep the additional PTWUIs for maintenance, especially for a sparse dataset. The memory consumption gap between the proposed PRE-HUI-MOD and HUI-Miner algorithms reduces along with increasing MUs, which can be observed from Fig. 6(a) and (b). From the above results, it can be concluded that the proposed PRE-HUI-MOD algorithm is acceptable to achieve the trade-off between the runtime, the number of patterns, and memory consumption.

5.6. Scalability

The scalability of the proposed PRE-HUI-MOD algorithm is then compared to those of the Two-Phase, HUI-Miner, and the FHM algorithms in the T10I4N4KD|X|K dataset. The MU was set at 0.15%, and the MR were respectively set at 0.1% and 0.5%. The dataset size $|X|$ was set at 100 to 500, with increments 100 (K) each time. The results are shown in Fig. 8.

From Fig. 8(a) and (c), it can be observed that the proposed PRE-HUI-MOD algorithm is two orders of magnitude faster than other algorithms in terms of runtime. The runtime of the Two-Phase algorithm is sharply increased whereas those of the HUI-Miner, FHM and PRE-HUI-MOD algorithms steadily increased along with increasing dataset size and MR. The number of generated PTWUIs with the proposed PRE-HUI-MOD algorithm is generally smaller than the number of HUIs. Less memory consumption is required for the proposed PRE-HUI-MOD algorithm when compared to the FHM algorithm in batch mode, as shown in Fig. 8(b) and (e). The memory consumption gap between the proposed PRE-HUI-MOD algorithm and the other algorithms became smaller when the MR was set at 1%. The proposed PRE-HUI-MOD algorithm is still one order of magnitude faster than the other algorithms. Hence, the proposed PRE-HUI-MOD algorithm has acceptable scalability results.

In the experiments, the proposed PRE-HUI-MOD algorithm performed five maintenance iterations. In a real-life situation, the dataset would be then consequentially and continuously changed. Since the Two-Phase, HUI-Miner and FHM algorithms all performed in batch-mode approach, the proposed PRE-HUI-MOD

Please cite this article in press as: J.-C.W. Lin et al., A fast updated algorithm to maintain the discovered high-utility itemsets for transaction modification, Adv. Eng. Informat. (2015), http://dx.doi.org/10.1016/j.aei.2015.05.003

algorithm can thus achieve better performance compared to batch-mode algorithms in a dynamic environment, especially when the number of modification is increased.

6. Comparative analysis

It is more complicated to maintain the discovered HUIs since the discovered information may become invalid or the invalid information may be arisen in dynamic databases. Most algorithms [6,19,23,25,26] of HUIM only handle the static databases and some algorithms are further designed to handle the dynamic environments of transaction insertion [5,16] and transaction deletion [14]. Although transaction modification can be concerned as the insertion operation followed by deletion operation or vice versa, it takes twice operations and time-consuming process to maintain the discovered information. To the best of our knowledge, transaction modification of HUIM is also common seen in real-life situations but has not been proposed yet. The proposed maintenance algorithm can efficiently maintain the discovered information when some transactions are modified, which is different than the past two operations (insertion and deletion) in dynamic databases and cannot be compared in the conducted experiments. From the conducted experiments of the past batch-mode algorithms [10,18,19], it can be found that the proposed PRE-HUI-MOD has best results among them. The proposed maintenance algorithm is thus acceptable and reasonable in real-life situations.

7. Conclusion and future work

In a real-life environment, a database is usually found in a dynamic environment as the transactions in the database are frequently inserted, deleted or modified. Transaction modification is also commonly seen in real-world applications, especially as errors or typos may frequently occur when information is inputted into a computer using a keyboard. Although transaction insertion is followed by transaction deletion, or deletion followed by insertion, can be concerned as a batch process for transaction modification, it takes twice operations and time-consuming processes to maintain and update the discovered HUIs. Thus, it is an important task to design a maintenance algorithm for transaction modification, and thus efficiently update the discovered HUIs in dynamic databases.

In this paper, based on a new developed pre-large strategy, an efficient algorithm called PRE-HUI-MOD algorithm is proposed to maintain and update the discovered HUIs of HUIM for handling transaction modification. When transactions are modified, the proposed PRE-HUI-MOD algorithm partitions the discovered HTWUIs into three parts, creating nine cases according to whether they are HTWUIs or small transaction-weighted utilization itemsets in the original database and in the modified transactions. Each part is then executed by the designed procedure to respectively maintain and update the discovered HTWUIs. A safety bound is also designed to reduce database rescan computations when transactions are modified. From the experimental results, it can be observed that the proposed PRE-HUI-MOD algorithm outperforms existing batch-mode algorithms in several databases in terms of execution time and scalability.

Since the proposed PRE-HUI-MOD algorithm applied the generate-and-test mechanism and the Two-Phase model for level-wisely mining HUIs, extra memory is required to maintain the HTWUIs and PTWUIs. In the near future, the utility-list structure will be extended to handle the maintenance approach whether it is insertion, deletion, or modification. The computations can thus be greatly improved without candidate generations for directly producing HUIs.

Acknowledgement

This research was partially supported by the Tencent Project under Grant CCF-TencentRAGR20140114, by the Shenzhen Peacock Project, China, under Grant KQC201109020055A, by the Natural Scientific Research Innovation Foundation in Harbin Institute of Technology under Grant HIT.NSRIF.2014100, and by the Shenzhen Strategic Emerging Industries Program under Grant ZDSY20120613125016389.

References

[1] Frequent Itemset Mining Dataset Repository, 2012. <http://fimi.ua.ac.be/data/>.
[2] R. Agrawal, T. Imielinski, A. Swami, Database mining: a performance perspective, IEEE Trans. Knowl. Data Eng. 5 (1993) 914–925.
[3] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: The International Conference on Very Large Data Bases, 1994, pp. 487–499.
[4] R. Agrawal, R. Srikant. Quest Synthetic Data Generator, 1994. <http://www.Almaden.ibm.com/cs/quest/syndata.html>.
[5] C.F. Ahmed, S.K. Tanbeer, B.S. Jeong, Y.K. Lee, Efficient tree structures for high utility pattern mining in incremental databases, IEEE Trans. Knowl. Data Eng. 21 (2009) 1708–1721.
[6] R. Chan, Q. Yang, Y.D. Shen, Mining high utility itemsets, in: IEEE International Conference on Data Mining, 2003, pp. 19–26.
[7] M.S. Chen, J. Han, P.S. Yu, Data mining: an overview from a database perspective, IEEE Trans. Knowl. Data Eng. 8 (1996) 866–883.
[8] D.W.L. Cheung, J. Han, V. Ng, C.Y. Wong, Maintenance of discovered association rules in large databases: an incremental updating technique, in: International Conference on Data Engineering, 1996, pp. 106–114.
[9] D.W.L. Cheung, S.D. Lee, B. Kao, A general incremental technique for maintaining discovered association rules, in: The International Conference on Database Systems for Advanced Applications, 1997, pp. 185–194.
[10] P. Fournier-Viger, C.W. Wu, S. Zida, V.S. Tseng, Fhm: faster high-utility itemset mining using estimated utility co-occurrence pruning, Found. Intell. Syst. 8502 (2014) 83–92.
[11] T.P. Hong, C.Y. Wang, Y.H. Tao, A new incremental data mining algorithm using pre-large itemsets, Intell. Data Anal. 5 (2001) 111–129.
[12] T.P. Hong, C.Y. Wang, Maintenance of association rules using pre-large itemsets, Intell. Databases: Technol. Appl. (2007) 44–60.
[13] T.P. Hong, C.Y. Wang, An efficient and effective association-rule maintenance algorithm for record modification, Expert Syst. Appl. 37 (2010) 618–626.
[14] C.W. Lin, T.P. Hong, G.C. Lan, J.W. Wong, W.Y. Lin, Efficient updating of discovered high-utility itemsets for transaction deletion in dynamic databases, Adv. Eng. Inform. (2015) 16–27.
[15] C.W. Lin, T.P. Hong, W.H. Lu, An effective tree structure for mining high utility itemsets, Expert Syst. Appl. 38 (2011) 7419–7424.
[16] C.W. Lin, G.C. Lan, T.P. Hong, An incremental mining algorithm for high utility itemsets, Expert Syst. Appl. 39 (2012) 7173–7180.
[17] C.W. Lin, T.P. Hong, G.C. Lan, J.W. Wong, W.Y. Lin, Incrementally mining high utility patterns based on pre-large concept, Appl. Intell. 40 (2014) 343–357.
[18] M. Liu, J. Qu, Mining high utility itemsets without candidate generation, in: ACM International Conference on Information and Knowledge Management, 2012, pp. 55–64.
[19] Y. Liu, W.K. Liao, A. Choudhary, A two-phase algorithm for fast discovery of high utility itemsets, Adv. Knowl. Discov. Data Min. (2005) 689–695.
[20] Y. Liu, W.K. Liao, A. Choudhary, A fast high utility itemsets mining algorithm, in: International Workshop on Utility-based Data Mining, 2005, pp. 90–99.
[21] Microsoft. Example Database Foodmart of Microsoft Analysis Services. <http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx>.
[22] B. Nath, D.K. Bhattacharyya, A. Ghosh, Incremental association rule mining: a survey, WIREs Data Min. Knowl. Discov. 3 (2013) 157–169.
[23] V.S. Tseng, B.E. Shie, C.W. Wu, P.S. Yu, Efficient algorithms for mining high utility itemsets from transactional databases, IEEE Trans. Knowl. Data Eng. 25 (2013) 1772–1786.
[24] C.W. Wu, Philippe Fournier-Viger, Philip S. Yu, Vincent S. Tseng, Efficient algorithms for mining the concise and lossless representation of closed+ high utility itemsets, in: IEEE International Conference on Data Mining, 2011, pp. 824–833.
[25] H. Yao, H.J. Hamilton, C.J. Butz, A foundational approach to mining itemset utilities from databases, in: SIAM International Conference on Data Mining, 2004, pp. 482–486.
[26] H. Yao, H.J. Hamilton, Mining itemset utilities from transaction databases, Data Knowl. Eng. 59 (2006) 603–626.