Intelligent Data Analysis 18 (2014) 1027–1047
DOI 10.3233/IDA-140683
IOS Press

Discovering high utility itemsets with multiple minimum supports

Heungmo Ryang[a], Unil Yun[a,*] and Keun Ho Ryu[b]
[a]*Department of Computer Engineering, Sejong University, Seoul, Korea*
[b]*Department of Computer Science, Chungbuk National University, Cheongju, Korea*

Abstract. Generally, association rule mining uses only a single minimum support threshold for the whole database. This model implicitly assumes that all items in the database have the same nature. In real applications, however, each item can have different nature such as medical datasets which contain information of both diseases and symptoms or status related to the diseases. Therefore, association rule mining needs to consider multiple minimum supports. Association rule mining with multiple minimum supports discovers all item rules by reflecting their characteristics. Although this model can identify meaningful association rules including rare item rules, not only the importance of items such as fatality rate of diseases but also attribute of items such as duration of symptoms are not considered since it treats each item with equal importance and represents the occurrences of items in transactions as binary values. In this paper, we propose a novel tree structure, called MHU-Tree (Multiple item supports with High Utility Tree), which is constructed with a single scan. Moreover, we propose an algorithm, named MHU-Growth (Multiple item supports with High Utility Growth), for mining high utility itemsets with multiple minimum supports. Experimental results show that MHU-Growth outperforms the previous algorithm on both real and synthetic datasets, and can discover useful rules from a medical dataset.

Keywords: Frequent itemsets, multiple minimum supports, rare frequent itemsets, utility mining

1. Introduction

Data mining finds important and meaningful knowledge hidden in the huge database, and it is motivated from the explosive increase of data. Association rule mining is one of the data mining techniques and discovers important rules. Association rule [1] is denoted as $X \rightarrow Y$, where X is a set of items and Y is an item, and it means that transactions containing all items of X in a database have a high possibility to also include Y. More specifically, let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of m items and a transaction database $D = \{T_1, T_2, \ldots, T_n\}$ be a set of n transactions, where $T_i \in D$ and $T_i \subseteq I$, then support of $X \rightarrow Y$ is defined as s when the number of transactions containing $X \cup Y$ is $s\%$ in D, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \phi$. Besides, the rule $X \rightarrow Y$ holds in the database D with confidence c if $c\%$ of transactions in D that include X also include Y. Mining association rules is to find all important rules that have no less support and confidence than user-specified minimum support and confidence, respectively, and many researches [1,4,5,10,20,26] have been studied. In general, association rule mining uses only a single minimum support threshold for the whole database and regards all items as the

*Corresponding author: Unil Yun, Department of Computer Engineering, Sejong University, Seoul, Korea. E-mail: yunei@sejong.ac.kr.

same importance. It is to implicitly assume that all items in the database have the same nature. However, in real applications, each item can have different nature, frequency and importance, and thus there is a need to consider the nature. To address this issue, association rule mining with multiple minimum supports [6–8,13,14,17,21,22] and the importance of items [2,16,18,27–30,32] were proposed.

Association rule mining with multiple minimum supports discovers all meaningful rules, containing rarely occurred but important rules, by applying different minimum supports in respect to each item. Meanwhile, in association rule mining model using only a single minimum support, it is required to set a minimum support very low for finding rare association rules. However, it may cause too many rules as well as a lot of meaningless rules. On the contrary, these rules cannot be found when a high threshold is given. This dilemma is called *rare item problem* [24]. Consider an example of mining association rules on medical datasets. Flu occurs much more frequently than SARS (Serve Acute Respiratory Syndrome), and they lead to the same symptoms of fever and persistent cough. If a minimum support is set a high value, $SARS \rightarrow \{fever, cough\}$ cannot be discovered though $flu \rightarrow \{fever, cough\}$ can be extracted. To find the $SARS \rightarrow \{fever, cough\}$, it is necessary to exploit a very low minimum support, and as a result a large number of useless rules are also mined under the threshold. To solve this problem, previous studies [9,15] split data into some blocks according the frequency of items or group items together which are related to each other. However, those methods are ad hoc and approximate approaches. Mining association rules based on multiple minimum supports can solve the problem by using different thresholds with respect to both flu and SARS. Nevertheless, it still treats all items with equal importance and represents the occurrences of items in transactions as binary values. Consider the above example continuously. It is a common symptom of a slight headache regardless of disease, and it can appear frequently for a short term. That is, $flu \rightarrow \{fever, cough, headache\}$ or $SARS \rightarrow \{fever, cough, headache\}$ can be found. In this case, the item, *headache*, should be pruned. However, if it occurs with a certain disease consistently, then it can be a significant sign even if a slight headache. In other words, $flu \rightarrow \{fever, cough, headache\}$ and $SARS \rightarrow \{fever, cough, headache\}$ can be meaningful rules according to relevance of the symptom or duration time. To solve this problem, the importance of items as well as the occurrence of items in transactions has to be reflected. Association rule mining with item importance have been proposed to address this issue, and this model can be divided into two types, weighted [2,16,27,30,32] and utility [18,28,29] mining. Especially, the latter also considers non-binary item quantity, and thus this model can deal with the example. However, the model does not reflect nature of items such as frequency. In other words, multiple minimum supports are not applied in this model. In this paper, motivated from the above, we propose a framework to consider the characteristics of real world databases, the importance of each item and non-binary values of the items in transactions, in the multiple minimum supports model. For this purpose, we propose a novel algorithm as well as a tree structure for efficiently discovering high utility itemsets with multiple minimum supports. By applying the proposed algorithm, we can discover important itemsets with high utilities from identified ones by the MIS approach. That is, we can mine significant itemsets efficiently with considering multiple minimum supports and utilities. In addition, we can conduct analysis to important sales and disease patterns in market and medical databases using the proposed algorithm. In web path analysis, moreover, we can discover more meaningful information by considering not only frequently visited web paths but also the importance of each web page and staying time of users in the pages. Major contributions of this paper are summarized as follow: A novel tree structure, called *MHU-Tree* (*Multiple item supports with High Utility Tree*), is proposed for maintaining information related to high utility itemsets with multiple minimum supports. Moreover, a restructuring method is also developed, and MHU-Tree can be constructed with a single scan by applying the method.

We propose a novel algorithm, named *MHU-Growth* (*Multiple item supports with High Utility Growth*), in order to reduce search space and the number of candidates in the mining process. By adopting the proposed algorithm, the number of candidates and runtime can be decreased, and thereby high utility itemsets with multiple minimum supports can be generated from MHU-Tree efficiently. Various experiments for performance evaluation and comparison are conducted between the proposed algorithm and a state-of-the-art algorithm [14] based on multiple minimum supports on both real and synthetic datasets. Experimental results show that MHU-Growth outperforms other algorithm substantially in terms of execution time and the number of generated itemsets. The rest of this paper is organized as follows. In Section 2, we introduce the related work and explain preliminaries. In Section 3, we illustrate the proposed tree structure and mining algorithm for pruning candidates and reducing search space in detail. In addition, restructuring method is also described. In Section 4, we show and analyze experimental results for performance evaluation. Finally, conclusions are given in Section 5.

2. Background

2.1. Related work

For association rule mining, extensive studies [4,5,20] have been proposed since Apriori [1] was proposed. Apriori is the initial solution based on a candidate set generation-and-test method. It has revealed many drawbacks, multiple database scans and generation of a large number of candidates. FP-Growth [10] based on pattern growth was afterward proposed to achieve a better performance than Apriori-based methods with a prefix tree structure, called FP-Tree, and it is performed with twice database scans. In the framework, only a single minimum support threshold is used for the whole database. As a result, it is hard to discover rare association rules in this model because a low threshold leads to generation of a large number of rules including many useless rules. Furthermore, the rules cannot be discovered under a high threshold. It is called rare item problem [24], and the initial solutions [9,15] use ad hoc and approximate approaches. The former splits data into a few blocks depending on the frequency of items and then finds rules in each block with a different threshold. However, it is difficult to find rules that contain items across different blocks. On the other hand, the latter groups a large number of rare items which are related to each other. It is also unable to find rare rules that involve each rare item and the more frequent items.

To address this issue, multiple minimum supports model has been studied [6–8,11,12,17,21]. In this model, each item has a different support threshold value according to their frequency. MSApriori [22] is an algorithm based on Apriori and multiple minimum supports, and it allows users to set a threshold value in respect to each item for reflecting characteristics of items. MSApriori has defined *MIS* (*Minimum Item Support*) for mining rare association rules without generating a large number of meaningless rules, and each item has an MIS value. For example, consider a database with four items {*fever*, *cough*, *nausea*, *pain*} and MIS values of the items, *MIS* (*fever*) = 1%, *MIS* (*cough*) = 0.8%, *MIS* (*nausea*) = 0.7%, and *MIS* (*pain*) = 0.1%. If support of {*fever*, *pain*} is 0.05%, then {*fever*, *pain*} is infrequent since a minimum support threshold of {*fever*, *pain*} is a minimum MIS value, *min*[*MIS* (*fever*), *MIS* (*pain*)] = 0.1%. On the contrary, if support of {*fever*, *cough*} is 0.9%, then {*fever*, *cough*} is frequent because *min* [*MIS* (*fever*), *MIS* (*cough*)] = 0.8%. However, MSApriori consumes a huge amount of runtime, especially when databases contain many long transactions. The reason is that it performs several database scans and uses a candidate generation-and-test method. To solve this problem, CFP-Growth [13] based

on FP-Tree was proposed. It constructs MIS-Tree with a single scan, and nodes are sorted by MIS descending order. CFP-Growth has defined MIN which is the minimum MIS value of items in the database. When a global MIS-Tree is constructed, the tree is restructured by pruning items having less supports than MIN from the tree. Consider the above example continuously. Minimum MIS value is *MIS* (*pain*) $= 0.1\%$, that is MIN $= 0.1\%$. If *sup* (*fever*) $= 1\%$, *sup* (*cough*) $= 0.9\%$, *sup* (*nausea*) $= 0.1\%$, and *sup* (*pain*) $= 0.05\%$, then *pain* is pruned from a global MIS-Tree due to the less *sup* (*pain*) than MIN, and then the tree is restructured. The reason is that supports of rules containing an item which has less MIS value than MIN are also not greater than or equal to MIN by anti-monotone property [1]. Rare association rules are discovered from a restructured MIS-Tree, called compact MIS-Tree. Mining process is performed by selecting each item from header table of the compact MIS-Tree and creating a conditional pattern base for the each selected item. Here, CFP-Growth repeats growth process until each conditional pattern base becomes empty. Therefore, it spends too much time for finding complete set of frequent association rules. To solve the problem, CFP-Growth++ [14] was proposed, and it employs *LMS* (*Least Minimum Support*) instead of MIN. LMS refers to the least MIS value among MIS values of frequent items. For example, CFP-Growth uses a minimum MIS value, *MIS* (*pain*) $= 0.1\%$ in the previous example, and *pain* is an infrequent item. In contrast, LMS is *MIS* (*cough*) $= 0.8\%$ since *nausea* and *pain* are infrequent items, and *cough* has the least MIS value between *fever* and *cough* which are frequent items. To satisfy downward closure property [1], CFP-Growth++ has defined *conditional closure property*, and this algorithm extracts all association rules based on the property without performing mining process until a conditional pattern base becomes empty. Hence, search space and runtime can be reduced. On the other hand, with various types of pattern mining approaches, mining algorithms based on multiple minimum supports were also proposed, such as fuzzy [7,11,17], periodic patterns [8], sequential patterns [11,12], and so on. For example, FQSP-MMS (Fuzzy Quantitative Sequential Pattern with Multiple Minimum Supports) was proposed [11], which discovers FQSPs with both multiple minimum supports and adjustable membership functions. Although the multiple minimum supports model can reflect item frequency, the importance of items and item quantities in transactions are not considered in contrast to real world databases.

To reflect the importance of items and item quantity, *utility mining* [3,18,19,23,28,29,31] has emerged as a significant research topic in the data mining area. In the framework of utility mining, items have two types of utility: (1) external utility and (2) non-binary item quantity in transactions, internal utility. Utility of an itemset is defined as the sum of the product of external and internal utilities of items in the itemset. If external utilities of items {*fever*, *cough*, *nausea*, *pain*} are {1, 3, 2, 1}, utility of {*fever*, *cough*, *pain*} is $(1 \times 1) + (2 \times 2) + (1 \times 2) = 6$. Although utility mining model employs both item importance and quantity, it does not apply multiple minimum supports model. For these reasons, this study aims to reflect the characteristics of items, importance, frequency, and non-binary transactions. By applying the proposed algorithm, we can efficiently discover all high utility itemsets with multiple minimum supports.

2.2. Preliminaries

An itemset X is a set of k distinct items $\{i_1, i_2, \ldots, i_k\}$, where $X \subseteq I, 1 \leqslant k \leqslant m$. Each transaction T_i, where $T_i \in D$ and $T_i \subseteq I$, has a unique identifier, called *TID*, and each item in T_i is associated with non-binary item quantity. Table 1 is an example of a transaction database.

Definition 1. *Minimum Item Support* of an item i_p is a minimum support threshold of i_p and denoted as *MIS*(i_p). For example, *MIS*$(B) = 5$ in Table 2.

Table 1
An example of transaction database

TID	Transaction
T_1	A (1), B (1), E (3)
T_2	B (2), E (1)
T_3	A (1), C (2), D (1)
T_4	D (1), F (2)
T_5	C (1), D (2), G (1)
T_6	A (2), B (1), C (1)
T_7	A (1), H (2)
T_8	A (1), B (2), F (1)
T_9	B (1), C (1), E (1)

Table 2
MIS values for every item

Item	A	B	C	D	E	F	G	H
MIS	6	5	4	3	3	3	2	2

Table 3
External utility table

Item	A	B	C	D	E	F	G	H
Utility	2	1	1	3	1	2	1	1

Definition 2. *External utility* of an item i_p indicates the importance of the item such as fatality rate of diseases in medical databases, or price or profit in market databases, and it is denoted as $eu(i_p)$. For example, $eu(D) = 3$ in Table 3.

Definition 3. *Internal utility of* an item i_p in a transaction T_i refers to a non-binary value of i_p in T_i such as duration time of symptom in medical databases or the number of sold copies in market databases, and it is denoted as $iu(i_p, T_i)$. For example, $iu(A, T_3) = 1$ in Table 1.

Definition 4. *Utility* of an item i_p in a transaction T_i is denoted as $u(i_p, T_i)$ and defined as the product of external and internal utilities, $eu(i_p) \times iu(i_p, T_i)$. For example, $u(A, T_3) = eu(A) \times ie(A, T_3) = 2 \times 1 = 2$ in Table 1.

Definition 5. *Utility* of an itemset X in a transaction T_i is denoted as $u(X, T_i)$ and defined as $\sum u(i_p, T_i)$, where $X \subseteq T_i$ and $i_p \in X$. For example, $u(AC, T_3) = u(A, T_3) + u(C, T_3) = 2 + 2 = 4$.

Definition 6. *Utility* of an itemset X in D is denoted as $u(X)$ and defined as $\sum u(X, T_i)$, where $T_i \in D, X \subseteq T_i$, and $i_p \in X$. For example, $u(AB) = u(AB, T_1)+u(AB, T_6)+u(AB, T_8) = 3+5+4 = 12$.

Definition 7. *Transaction Utility* (*TU*) of a transaction T_i is denoted as $TU(T_i)$ and defined as $\sum u(i_p, T_i)$, where $i_p \in T_i$. For example, $TU(T_2) = u(B, T_2) + u(E, T_2) = 2 + 1 = 3$.

Definition 8. *Minimum support* of an itemset $X = \{i_1, i_2, \ldots, i_k\}$ refers to the least MIS value of items in X, and it is defined as $\min[MIS(i_1), MIS(i_2), \ldots, MIS(i_k)]$, where $i_p \in X$ and $1 \geqslant p \geqslant k$. For example, a minimum support of $\{AB\}$ is $\min[MIS(A), MIS(B)] = 5$.

Definition 9. If utility of an itemset X is no less than a user-specified minimum utility threshold *minutil* and support of X is greater than or equal to the least value among MIS values of items in X, then X is called HUF (High Utility and Frequent) itemset.

Example 1. In Table 1, for example, MIS of an itemset $\{B\}$ is 5 and support of the itemset is 5. Thus, $\{B\}$ is a frequent itemset due to no less $sup(B)$ than the MIS value of $\{B\}$. In addition, utility of $\{B\}$ is $u(B) = u(B, T_1) + u(B, T_2) + u(B, T_6) + u(B, T_8) + u(B, T_9) = 1 + 2 + 1 + 2 + 1 = 7$. If a user-specified minimum utility threshold *minutil* is 5, then $\{B\}$ is an HUF itemset because $\{B\}$ is a frequent itemset and utility of $\{B\}$ is greater than *minutil*. On the contrary, $\{AC\}$ is not an HUF itemset. The reason is that the least MIS is $\min[MIS(A), MIS(C)] = 4$ and $sup(AC) = 1$.

To maintain the downward closure property [1] is a significant issue in high utility itemset mining. Following definitions are related to *TWDC* (*Transaction Weighted Downward Closure*) [23] for satisfying the property.

Definition 10. *Transaction Weighted Utility* (*TWU*) of an itemset X is the sum of utilities of transactions containing X. It is denoted as $TWU(X)$ and defined as $\sum TU(T_i)$, where $X \subseteq T_i$. For example, $TWU(AB) = TU(T_1) + TU(T_6) + TU(T_8) = 6 + 5 + 6 = 17$.

Definition 11. If TWU value of an itemset is no less than *minutil*, the itemset is called a high transaction weighted utility pattern. Moreover, a low utility itemset is not pruned if TWU value of the itemset is no less than *minutil*. For example, utility of {*AB*} is 12 and its TWU value is 17, and therefore the itemset is not pruned if *minutil* is 15.

3. The proposed tree structure and method

In this Section, we first describe the proposed data structure, MHU-Tree. Then, we illustrate the proposed algorithm, MHU-Growth, in detail. The framework of the proposed method consists of three steps. In the first step, a global tree is constructed with a single scan. In the second step, the constructed global tree is restructured by pruning infrequent or unimportant items, and rearranging nodes. In the last step, HUF itemsets are extracted from the restructured tree.

3.1. The proposed tree structure: MHU-tree

The proposed tree structure, named MHU-Tree (Multiple item supports with High Utility Tree), is used to maintain information of transactions and HUFIs. The initial tree is constructed by a single scan with multiple minimum supports, item utilities, and non-binary transactions. In following subsections, elements of MHU-Tree are first defined. Then, how to construct the initial MHU-Tree and to restructure the initial tree with three pruning conditions are illustrated in detail.

3.1.1. Elements in MHU-tree

In MHU-Tree, each node N consists of *N.name*, *N.count*, *N.parent*, *N.nodelink*, and a set of child nodes. Furthermore, in contrast to previous tree structures [13,14], *N.nu* is also an element of the tree to store information of node utility. *N.name* is an item name of N. *N.count* is a support count of N. *N.parent* and *N.nodelink* point to the parent node of N and a node which has the same item name as *N.name*, respectively.

MHU-Tree also includes a header table to facilitate tree traversal in restructuring and mining processes. Each entry in the header table is composed of an item name, a TWU value, a support count, and a link. The link points to the last occurred node which has the same item name. By using the links in the header table and node links, the nodes with the same item name can be traversed efficiently. In addition, entries are ordered by TWU descending order.

3.1.2. Construction of MHU-tree

In the first database scan, each transaction is inserted as a branch inside the initial tree by rearranging items according to MIS descending order, and its TU is computed; at the same time, TWU of each item in the transaction is also accumulated. To insert transaction $T_i = \{i_1, i_2, \ldots, i_k\}$, where $i_k \in I$ and $1 \leqslant k \leqslant m$, a function *Insert_Transaction* (*Tree*, T) is called. This subroutine is shown in Fig. 1, where *Tree* is the initial tree and T is a transaction. Items in T_i are reordered by MIS descending order $\{i_{1'}, i_{2'}, \ldots, i_{k'}\}$, and then each item $i_{k'}$ in T_i' with $q_{k'}$ which is a quantity value associated with the item is added to the tree as a node, $N_{ik'}$, such that $N_{ik'}.name = i_{k'}$. First, $i_{1'}$ is inserted as a child node, $N_{i1'}$,

```
Insert_Transaction(Tree, T)
1.  N ← the root node of Tree
2.  Reorder items in T by MIS descending order
3.  Calculate transaction utility of T, TU(T)
4.  For each item i in transaction T
5.      If N has not a child N_i such that N_i.name = i
6.          Create a new child N_i under N, where N_i.name = i
7.      Increase N_i.count and N_i.nu by 1 and TU(T)
8.      If Header has not an entry for i
9.          Create a new entry E_i, where E_i.TWU = 0 and E_i.support = 0
10.     Increase E_i.TWU and E_i.support by TU(T) and 1
11.     N ← N_i
```

Fig. 1. Algorithm for construction of a global MHU-tree.

of N_R, where N_R is the root node. If N_R has not a child node $N_{i1'}$, then $N_{i1'}$ is created under N_R, and $N_{i1'}$.*count* and $N_{i1'}$.*nu* are initialized as zero. Then, $N_{i1'}$.*count* and $N_{1i'}$.*nu* are increased by 1 and TU of T_i, respectively. After that, rest of items $\{i_{2'}, i_{3'}, \ldots, i_{k'}\}$ are also added as child nodes. If there is a node that has the same name with an item in a transaction when the item is inserted, support count and node utility of the node are just raised by 1 and TU without creating a child node.

Example 2. Consider the database in Table 1 with both MIS and external utility tables of Tables 2 and 3. The construction process of the initial MHU-Tree is performed by inserting a transaction $T_1 = \{A, B, E\}$, and the root node N_R is empty at this stage. First, items in T_1 are sorted and utilities of the items are computed. The calculated values are $u(A, T_1) = eu(A) \times iu(A, T_1) = 2 \times 1 = 2, u(B, T_1) = eu(B) \times iu(B, T_1) = 1 \times 1 = 1$, and $u(E, T_1) = eu(E) \times iu(E, T_1) = 1 \times 3 = 3$. Hence, transaction utility of T_1 is $TU(T_1) = u(A, T_1) + u(B, T_1) + u(E, T_1) = 2 + 1 + 3 = 6$. The first item A is added to the tree as a child node of N_R, and N_A is created under N_R since the initial tree is empty at the very beginning. Then, N_A.*count* and N_A.*nu* are increased by 1 and 6, respectively. After that, the rest of items, B and E, are also inserted. Besides, each entry for the inserted items is added in a header table, and support and TWU are assigned as 1 and TU of T_1. Next, a transaction $T_2 = \{B, E\}$ is inserted into the tree in the same steps, and entries for the items are not created at this stage because they are already added in the header. Instead, support and TWU of each item in $\{B, E\}$ are raised by 1 and $TU(T_2)$. When an item B in transaction T_3 is inserted, N_R already has a child node N_B, and thus a child node N_B under N_R is not created. In this case, N_B.*count* and N_B.*nu* are just increased by 1 and $TU(T_3)$, respectively. Through the same way, the others are also inserted.

Figure 2 shows the constructed initial MHU-Tree by inserting all transactions in Table 1. After construction process, the initial tree is restructured by pruning unimportant or infrequent items. Following definitions, lemmas, and property are for reducing search space and satisfying anti-monotone property [1] in multiple minimum supports model with utilities and non-binary transactions. If TWU of a prefix itemset is less than a user-specified minimum utility threshold, *minutil*, all super itemsets which are generated from the prefix itemset have less utility values than the threshold. TWU of an itemset X refers to the sum of $TU(T_i)$, where $T_i \in D$ and $X \subseteq T_i$, and therefore $TWU(X) \geqslant u(X)$. Moreover, the length of a super itemset which is a maximum expanded itemset from X is no longer than that of the longest transaction of T_i. That is, all super itemsets of X must be included whole or part of transactions containing X. Therefore, utility of a super itemset is always no greater than *TWU*(X).

Property 1. Transaction Weighted Downward Closure. For any itemset X, if $TWU(X)$ is less than a user-specified utility threshold *minutil*, any super itemset of X is not a high utility itemset. That is, any super itemset of a low utility itemset has less utility value than *minutil*.

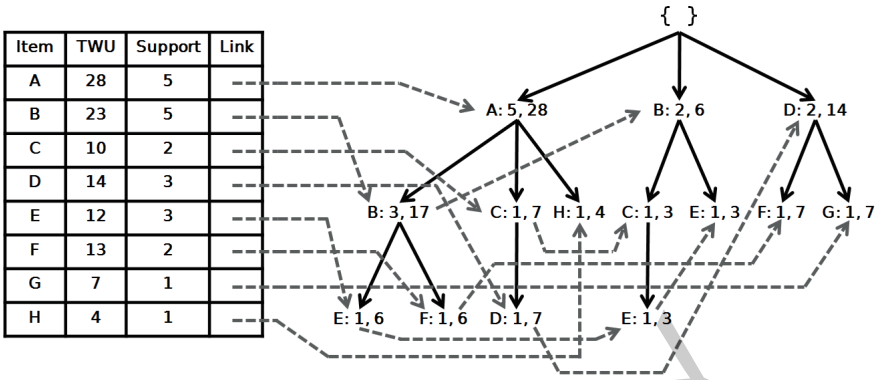


Fig. 2. Constructed global MHU-tree.

Pruning condition 1. ($TWU(i_p) < minutil$) TWU of an item i_p in a header table is less than *minutil*.

Example 3. Assume that *minutil* is set to 10. In Fig. 2, TWU values of items, G and H, are 7 and 4, and thereby utility of supersets which are generated from the items is no greater than or equal to *minutil* according to Property 1. It means that any high utility itemset is not extracted from the unimportant items, and thus G and H should be pruned.

Definition 12. *Least Minimum Support* indicates a minimum support threshold for the whole database D, and it is defined as the least MIS value of frequent items. It is denoted as *LMS* and its item is represented as i_{LMS}.

Example 4. Consider MIS values in Table 2 and the header table in Fig. 2. MIS values of items, G, H, and F, are $MIS(G) = 2, MIS(H) = 2$, and $MIS(F) = 3$. Thus, they are infrequent items since $sup(G), sup(H)$, and $sup(F)$ are less than their MIS values. On the other hand, MIS of E is 3 and support of the item is the same with the MIS value. Hence, E is a frequent item, and the rest of items $\{A, B, C, D\}$ in the header have no less MIS values than $MIS(E)$. The reason is that items are sorted by MIS descending order. Therefore, i_{LMS} is E and *LMS* is $MIS(E) = 3$.

Lemma 1. In a global MHU-Tree, items under i_{LMS} cannot generate any frequent itemset.

Proof Let $\{i_1, i_2, \ldots, i_{LMS}, \ldots, i_k\}$ be items in a header table, where i_1 and i_k are the topmost and bottommost items, respectively. Minimum support of an itemset X refers to the least MIS value of items in X. Hence, minimum support of itemsets extracted from each item in $\{i_{LMS+1}, i_{LMS+2}, \ldots, i_k\}$ is the same with MIS of the item selected from a global tree in mining process. The reason is that items in the tree are sorted by MIS descending order and mining process is performed by bottom-up traversal. For example, if i_{LMS+1} is selected from a global tree, items in a conditional pattern base for i_{LMS+1} are $\{i_1, i_2, \ldots, i_{LMS}\}$ and $MIS(i_{LMS+1})$ is the least MIS value. Moreover, i_{LMS} is an item which has the least MIS value among frequent items. Therefore, the items under i_{LMS} are infrequent and cannot generate any frequent itemset by anti-monotone property.

Example 5. In the initial MHU-Tree of Fig. 2, i_{LMS} is E and items under i_{LMS} are $\{F, G, H\}$. In Table 2, MIS values of the items are $MIS(F) = 3, MIS(G) = 2$, and $MIS(H) = 2$. Besides, $sup(F) = 2, sup(G) = 1$, and $sup(H) = 1$, and thus all of them are infrequent items. In addition, minimum supports of itemsets generated from the each item are 3, 2, and 2. Therefore, they are all infrequent itemsets and any frequent itemset cannot be extracted from the items.

Pruning condition 2. ($MIS(i_p) \leqslant MIS(i_{LMS})$ and $sup(i_p) < MIS(i_p)$) MIS of an item i_p in a header table of a global tree is no greater than that of i_{LMS}, LMS, and support of i_p is less than its MIS value.

Example 6. Consider the minimum utility, *minutil* $= 10$, used in Example 3. Although F can be an important item due to its greater TWU value than *minutil*, it is under i_{LMS}, E, and support of F is less than its MIS value. Hence, F has to be pruned by pruning condition 2. On the contrary, a frequent item E should not be eliminated because its TWU value is greater than *minutil*.

Lemma 2. Even if support of an item i_p in a global MHU-Tree is less than its MIS value (or infrequent item), i_p can be contained a frequent itemset if $sup(i_p)$ is no less than LMS.

Proof Support of a frequent itemset is no less than LMS and items in a global MHU-Tree are sorted by MIS descending order. Here, infrequent items under i_{LMS} are pruned according to pruning condition 2. Hence, items in the tree except i_{LMS} are located in the above i_{LMS} and some of the items can be infrequent items. Therefore, an infrequent item i_p can be included a frequent itemset X when $sup(i_p)$ is greater than or equal to a minimum support of X in contrast to the pruned items under i_{LMS}. Moreover, $sup(i_p)$ should be no less than LMS since all frequent itemsets have a minimum support at least LMS.

Pruning condition 3. ($sup(i_p) <$ LMS) support of an item i_p is less than LMS.

Example 7. In Fig. 2, C is an infrequent item due to less support than $MIS(C)$, and it has to be pruned because $sup(C)$ is less than LMS. On the contrary, A is also an infrequent item, its support is greater than LMS, and thus it is not pruned.

After construction of the initial MHU-Tree, pruning process is conducted for reducing search space. The three pruning conditions are employed for this purpose. To facilitate the process, the initial tree is restructured by eliminating nodes for items that are unimportant or infrequent from the tree. First, the bottommost item of a header table is checked whether the item is unimportant or infrequent based on the pruning conditions. However, at the beginning, we cannot know which item is i_{LMS}. Thus, we use only the first and second pruning conditions, $TWU(i_p) <$ *minutil* and $sup(i_p) < MIS(i_p)$, until i_{LMS} is identified. Once i_{LMS} is determined, all of the pruning conditions are applied for the checking. If an item is satisfied at least one of the conditions, it is eliminated from the header and tree. To remove all nodes which have the same item name with the pruned item, link and node links are used. When an item node N_{ip}, where $N_{ip}.name = i_p$, is removed from the tree, if N_{ip} has any child node N_{Child}, it is necessary to change $N_{Child.parent}$ to a parent node N_{Parent} of the pruned node N_{ip}. Here, if N_{Parent} already has a child node N_{PChild} such that $N_{PChild}.name = N_{Child}.name$, then they need to be merged. In this case, support and node utility of N_{PChild} are just increased by corresponding values of N_{Child}. Through the method, the initial MHU-Tree is restructured, and thus we can construct a global tree with a single database scan.

Example 8. Consider the initial MHU-Tree in Fig. 2 and a minimum utility *minutil* $= 10$. First, it starts with the bottommost item H in the header for restructuring. TWU of H is 4 and it is less than *minutil*, and thereby H is not an important item. Hence, it is pruned according to the pruning condition 1. To remove all of nodes for H, the tree is traversed by following a link in the header. Since there is only one node for H and the node has not any child node, it is just eliminated. The next item G is also not an important item, $TWU(G) <$ *minutil*, and only one node for G has not any child node, and thus G is pruned. Although F is an important item, $TWU(F) >$ *minutil*, $sup(F)$ is less than $MIS(F)$, that is the pruning condition 2 is satisfied. As a result, F is eliminated. Figure 3(a) shows the result of pruning items, H, G, and F. Meanwhile, TWU of E is greater than *minutil* and $sup(E) \geqslant MIS(E)$, and therefore E is an important and frequent item. Items in the tree are sorted by MIS descending order, and thus E is a frequent item having the least MIS value. Hence, E is i_{LMS} and LMS is $MIS(E) = 3$, and the whole

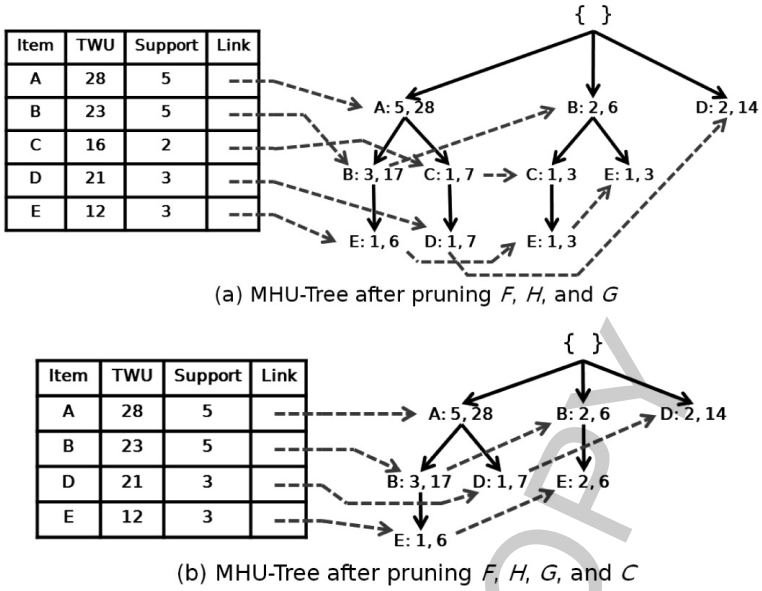


Fig. 3. Restructured MHU-tree by pruning items.

pruning conditions can be used. On the other hand, TWU of D is no less than *minutil* and its support is also greater than LMS. Hence, D is not pruned. On the contrary, support of C is less than LMS, and thus C is removed from the tree according to pruning condition 3. In contrast to items H, G, and F, nodes for C have child nodes. Therefore, it needs to process the child nodes. The process starts by following a link in the header. The first node for C is removed. Here, the node has a child node for D, and thus the child node is added to a parent node for A of the node as a child node. Then, we move to the next node through a node link. The last node for C also has a child node for E, N_{Child_E}. Moreover, a parent node for B already has a child node for E, N_{PChild_E}. Therefore, N_{PChild_E}.*count* and N_{PChild_E}.*nu* are raised by corresponding values of N_{Child_E}, N_{Child_E}.*count* and N_{Child_E}.*nu*, respectively. As the result, the support and node utility of N_{PChild_E} become 2 and 6. In addition, N_{Child_E} is eliminated from the tree. Figure 3(b) is the restructured tree by the pruning process.

Figure 4 is an algorithm for restructuring the initial MHU-Tree. A function, *Restructure_Tree*(*Tree*), where *Tree* is the initial tree, is called to perform the restructuring process based on the pruning conditions. When any unimportant or infrequent item i_p is found from header, *Prune*(*Tree*, i_p) is used for pruning the item and nodes. If a removed node has any child node, *Merge*(*Parent*, *Children*) function, where *Parent* is a parent node of i_p and *Children* is a set of child nodes of i_p, is called for reorganizing nodes in the tree.

3.2. The proposed mining method: MHU-growth

In this Section, we illustrate the proposed method, MHU-Growth, for discovering HUFIs from the MHU-Tree in detail. Following definitions and property are for efficient mining HUFIs.

Definition 13. Let $\{i_1, i_2, \ldots, i_j\}$ be a prefix itemset, where i_1 and i_j are selected items from a global tree and a local tree for $\{i_1, i_2, \ldots, i_{j-1}\}$, respectively. In local MHU-Trees, *Conditional Minimum Support* is defined as *MIS*(i_1) and denoted as *CMS*$(\{i_1, i_2, \ldots, i_j\})$. In Fig. 3(b), for example, if a

```
Restructure_Tree(Tree)
1. LMS ← φ
2. For each entry of an item i in header of Tree /* Bottom-up */
     /* Pruning Condition 1 */
3. If TWU(i) > minutil then Call Prune(Tree, i)
     /* Pruning Condition 2 */
4. Else if sup(i) < MIS(i) and LMS = 0 then Call Prune(Tree, i)
5. Else if sup(i) ≥ MIS(i) and LMS = 0 then LMS ← MIS(i)
     /* Pruning Condition 3 */
6. Else if sup(i) < LMS then Call Prune(Tree, i)

Prune(Tree, i)
1. For each node N_i in Tree such that N_i.name = i
2.       If N_i has children then Call Merge(N_i.parent, N_i.children)
3. Remove N_i from Tree

Merge(Parent, Children)
1. For each child node C in Children
2.       If Parent has a child N_Child such that N_Child.name = C.name
3.             Increase N_Child.count and N_Child.nu by C.count and C.nu
4.             Call Merge(N_Child, C.children)
5.       Else C.parent ← Parent
```

Fig. 4. Algorithm for restructuring of a global MHU-tree.

conditional pattern base for E is created, a prefix is $\{E\}$ and conditional minimum support of $\{E\}$ is $CMS(\{E\}) = 3$ that is MIS of E in Table 2.

Conditional minimum support is used as a threshold in local trees (or conditional pattern trees). In other words, if there is a local tree for $\{i_1, i_2, \ldots, i_j\}$, a minimum support value of all itemsets generated from the tree is $CMS(\{i_1, i_2, \ldots, i_j\})$, $MIS(i_1)$.

Property 2. Conditional Closure. If support of an itemset that is generated from a conditional pattern tree for $\{i_1, i_2, \ldots, i_j\}$ is less than $CMS(\{i_1, i_2, \ldots, i_j\})$, then all super itemsets of the itemset are infrequent.

Proof In MHU-Trees, items are sorted by MIS descending order and a minimum support of an itemset X indicates the least MIS value of items in X. In addition, mining process is performed based on bottom-up tree traversal. That is, items in a conditional pattern base for $\{i_1, i_2, \ldots, i_j\}$ have no less MIS values than MIS of an item i_1 in the prefix selected from a global tree. Thus, MIS values of the items are greater than or equal to $CMS(\{i_1, i_2, \ldots, i_j\})$. Moreover, all itemsets extracted from the conditional pattern base are super itemsets of the prefix, $\{i_1, i_2, \ldots, i_j\}$. Therefore, if an itemset has a less support value than CMS, then all super itemsets of the itemset are infrequent according to the anti-monotone property.

MHU-Growth is conducted for mining HUFIs by bottom-up traversal of a global tree following each link of a header table. In this process, from the bottommost item in the header, a conditional pattern base for each item is created by extracting all paths starting from the item; at the same time, support count and TWU of each item in the paths are calculated.

Example 9. In Fig. 3(b), mining process begins with the bottommost item E. All paths composed of items above E are extracted by following link and node links. The first path from a node $\{E : 1, 6\}$ is $\langle A, B : 1, 6\rangle$. Then, follow a node link of the node and extract the next path $\langle B : 2, 6\rangle$ from a node

Table 4
Conditional pattern base for E

Path	Support	Utility
$\langle A, B\rangle$	1	6
$\langle B\rangle$	2	6

Table 5
Mining results

Item	HUF model	MMS model
A	ϕ	ϕ
B	ϕ	$\{B\}$
D	$\{D\}$	$\{D\}$
E	$\{E\}$	$\{E\}, \{EB\}$

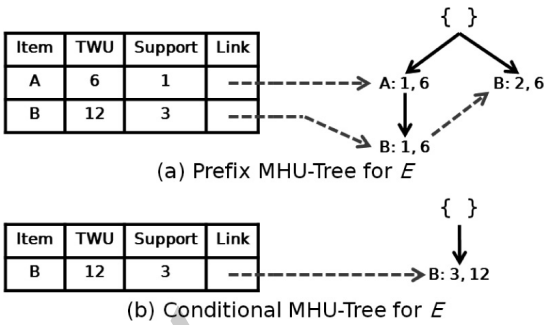


Fig. 5. Prefix and conditional trees for E.

$\{E : 2, 6\}$. Table 4 is a conditional pattern base for E, and it consists of the two paths. There are A and B in the pattern base, and $sup(A) = 1, sup(B) = 3, TWU(A) = 6$, and $TWU(B) = 12$.

Minimum support of itemsets generated from a prefix itemset $\{i_1, i_2, \ldots, i_j\}$ is $CMS(\{i_1, i_2, \ldots, i_j\})$ based on the conditional closure property. Hence, it needs to prune items having less support values than $CMS(\{i_1, i_2, \ldots, i_j\})$.

Pruning condition 4. $(sup(i_p) < CMS(\{i_1, i_2, \ldots, i_j\}))$ support of an item i_p in a conditional pattern base for $\{i_1, i_2, \ldots, i_j\}$ is less than $CMS(\{i_1, i_2, \ldots, i_j\})$.

Example 10. Consider the conditional pattern base for E in Table 4. A prefix itemset is $\{E\}$ and $CMS(\{E\}) = 3$, and thereby A is pruned due to a less support value of A than $CMS(\{E\})$ according to pruning condition 4. In addition, it also satisfies the pruning condition 1, $TWU(A) < minutil$. On the contrary, an item B is not eliminated because $sup(B) \geqslant CMS(\{E\})$ and $TWU(B) \geqslant minutil$. Figures 5(a) and (b) are prefix and conditional trees for $\{E\}$.

After construction of a local tree, HUFIs itemsets are generated from the tree based on the bottom-up traversal method. $\{E : 3, 12\}$ is extracted as a candidate itemset since $TWU(\{E\}) \geqslant minutil$ and $sup(\{E\}) \geqslant CMS(\{E\})$. Moreover, a candidate itemset $\{EB : 3, 12\}$ is generated from the conditional tree for E of Fig. 5(b). In the same way, $\{D : 3, 21\}$ and $\{B : 5, 23\}$ are extracted from conditional pattern trees for $\{D\}$ and $\{B\}$, respectively. Then, complete HUFIs are discovered finally by checking actual utilities of the candidate itemsets. For example, utility of a candidate itmeset $\{EB\}$ is $u(\{EB\}) = 9$, and it is less than *minutil*. Therefore, $\{EB\}$ is not an HUF itemset. Complete HUFIs mined in this way from the database of Table 1 are presented in Table 5. In addition, Table 5 also shows mining result based on multiple minimum supports (MMS) model without considering utility and non-binary transactions. We can observe that the reduced number of itemsets is generated in multiple minimum supports model with high utility. As a result, we can more efficiently discover significant itemsets reflected the characteristics of real world items. Figure 6 is the algorithm of MHU-Growth.

4. Performance evaluation

4.1. Experimental environment and datasets

In this section, various experiments are conducted in order to evaluate performance of the proposed algorithm, MHU-Growth. To the best of our knowledge, there is no framework of multiple minimum

Table 6
Characteristics of real datasets

Dataset	$\|D\|$	T_{avg}	$\|I\|$	R
Mushroom	8,124	23	119	19.328
Retail	88,162	10.3	16,470	0.625
Heart disease	303	14	136	0.103

Table 7
Characteristics of synthetic datasets

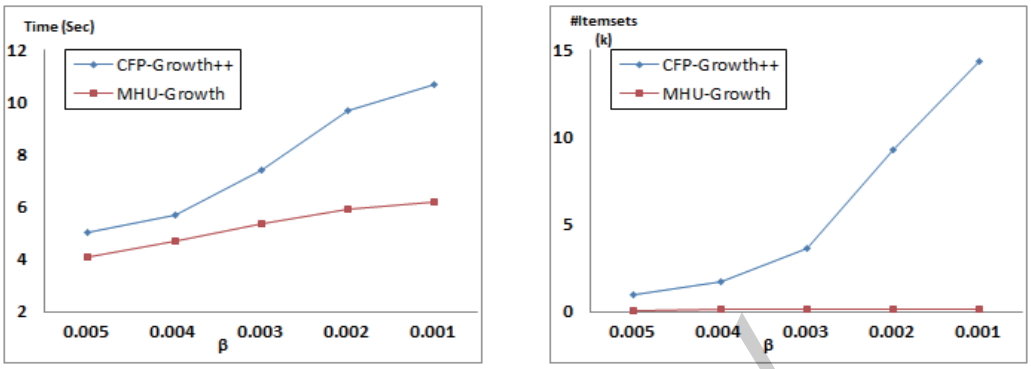
Dataset	$\|D\|$	T_{avg}	$\|I\|$	R	Size
T10I4D200K	200,000	10	1,000	0.010	11.7 MB
T10I4D400K	400,000	10	1,000	0.010	23.4 MB
T10I4D600K	600,000	10	1,000	0.010	35.1 MB
T10I4D800K	800,000	10	1,000	0.010	46.8 MB
T10I4D1000K	1,000,000	10	1,000	0.010	58.5 MB

MHU-Growth(*Tree*)
1. **For each** entry of an item *i* in header of *Tree* /* Bottom-up */
2. *CMS* ← *MIS*(*i*)
3. **For each** node N_i such that $N_i.name = i$ in *Tree*
4. **Extract** a path *P* consists of nodes such that *MIS* ≥ *CMS*
5. **For each** item node N_{ik} in the extracted path *P*
6. **Accumulate** $N_{ik}.count$ to support and $N_{ik}.nu$ to TWU
7. **If** *TWU*(*i*) ≥ *minutil* then **Generate** {*i*}
8. **Create** Prefix Tree for *i*, T_i
   /* Pruning Condition 1 and 4 */
9. **Prune** items from T_i such that *TWU* < *minutil* and *sup* < *CMS*
10. **Call** *MHU-Growth*(T_i, {*i*}, *CMS*)

MHU-Growth(T_x, *X, CMS*)
1. **For each** entry of an item *i* in header of T_X /* Bottom-up */
2. **For each** node N_i such that $N_i.name = i$ in T_X
3. **Extract** a path *P* consists of nodes such that *MIS* ≥ *MIS*(*i*)
4. **For each** item node N_{ik} in the extracted path *P*
5. **Accumulate** $N_{ik}.count$ to support and $N_{ik}.nu$ to TWU
6. **If** *sup*(*i*) ≥ *CMS* and *TWU*(*i*) ≥ *minutil* then **Generate** $X \cup \{i\}$
7. $X \leftarrow X \cup \{i\}$
8. **Create** Prefix Tree for {*X*}, T_X
   /* Pruning Condition 1 and 4 */
9. **Prune** items from T_X such that *TWU* < *minutil* and *sup* < *CMS*
10. **Call** *MHU-Growth*(T_i, {*i*}, *CMS*)

Fig. 6. MHU-growth algorithm.

supports with item utilities. Therefore, we compare the performance of our algorithm with a state-of-the-art one, CFP-Growth++ [14], based on the multiple minimum supports. Moreover, CFP-Growth++ showed better performance than the existing algorithm, CFP-Growth [13] in their experiments [14], and thus we only compare the proposed algorithm with CFP-Growth++. Experiments were performed on a 3.3 GHz Intel Processor with 8 GB memory. Moreover, all algorithms used in the experiments were written in C++ language and ran with the Windows 7 operating system. Common settings for the experiments are as follows. First, the initial MHU-Tree and MIS-Tree are constructed with a single database scan. In this stage, before inserting each transaction, items in the transaction are sorted by MIS descending order. Second, the initial trees are restructured by pruning unimportant or infrequent items. Finally, frequent itemsets including rare frequent ones with or without high utilities are discovered. Both real and synthetic datasets as well as a medical dataset were used in the experiments. Furthermore, the experiments are performed with respect to two aspects, traditional and medical association rule mining. Table 6 shows characteristics of the real datasets, where $|D|, T_{avg}, |I|$, and R refer to the number of transactions in a dataset, the average length of the transactions, the number of items in the dataset, and the dense/sparse characteristics ratio [2], $(T_{avg}/|I|) \times 100$. Mushroom and Retail datasets are obtained from FIMI Repository (http://fimi.cs.helsinki.fi); Heart Disease dataset is obtained from UCI Machine



(a) Runtime on T10.|I|1000.|D|100K (b) Number of itemsets on T10.|I|1000.|D|100K

Fig. 7. Performance comparison on synthetic dataset. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140683)

Learning Repository (http://archive.ics.uci.edu/ml/). Mushroom dataset contains characteristics of various species of mushrooms. Retail dataset is sparse and about product sales data in retail stores. It has lots of items and the average length is short. Heart Disease dataset is profile containing personal information such as age and heart rate of patient. In addition, synthetic datasets are described in Table 7 and used to evaluate scalability, and they are generated from the data generator in [1]. In the datasets from T10I4D100K to T10I4D1000K, the number of transactions is gradually increases, but items are constant. In all the datasets, likewise the previous studies [28], external utilities are generated between 0.01 and 10 by using a log-normal distribution, and the item quantities are generated randomly between 1 and 10.

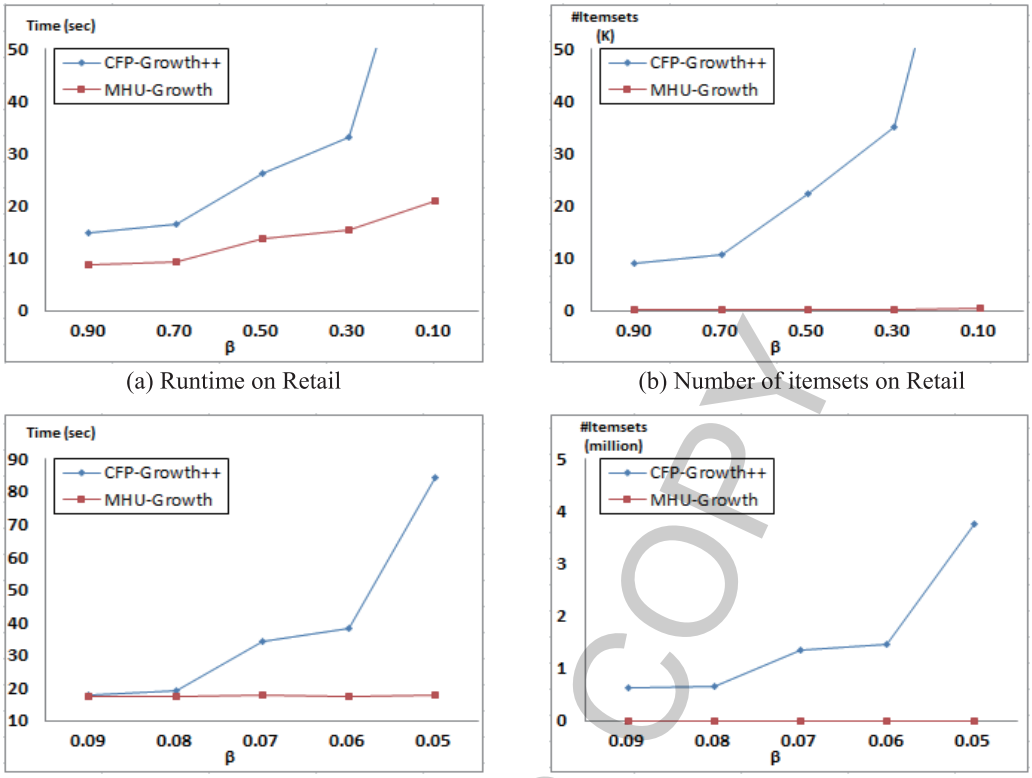
$$MIS(i) = \max[\beta \times f(i), LS] \tag{1}$$

Furthermore, we compute MIS value of each item based on the discussed method in [22] using the above Eq. (1). In the equation, β is a parameter to control how the MIS values for items should be related to their frequencies, where $0 \leqslant \beta \leqslant 1$, *LS* is the user-specified least minimum support value, and $f(i)$ is the frequency of an item i. If β is assigned to zero, we have only one minimum support, *LS*, which is the same with the traditional association rule mining. In all experiments, parameters are set for avoiding that a single minimum support is used.

4.2. Performance comparison on different datasets

In this part, we compare performance in terms of runtime and the number of generated itemsets with synthetic dataset, T10.|I|1000.|D|100 K, real datasets, Retail and Mushroom. Figure 7 shows the experimental results on the synthetic dataset, and *LS* and *minutil* are set to 0.2 and 0.004, respectively. In the figure, runtime of the proposed algorithm outperforms CFP-Growth++, and the substantially larger number of itemsets is discovered by CFP-Growth++. In addition, we can observe that the runtimes increase with decreasing value of parameter β. Especially, CFP-Growth++ generates the more number of itemsets according to decreasing value of β. It means that the proposed algorithm, MHU-Growth, can discover meaningful itemsets including rare itemsets faster than the other algorithm with multiple minimum supports model.

Next, we show experimental results on real datasets in Fig. 8. The performance on Retail is shown in Figs 8(a) and (b). In this experiment, *LS* is 0.0001 and *minutil* is 0.001, and MHU-Growth outperforms



(a) Runtime on Retail (b) Number of itemsets on Retail

(c) Runtime on Mushroom (d) Number of itemsets on Mushroom

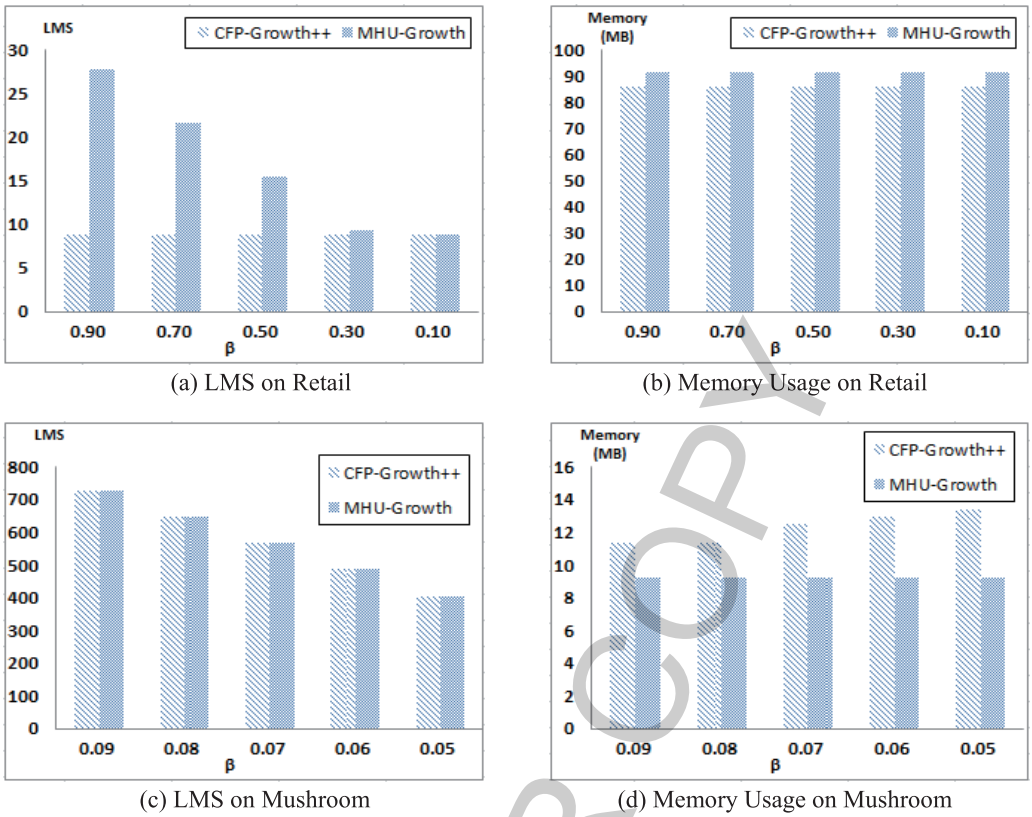
Fig. 8. Performance comparison on real datasets. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140683)

CFP-Growth++ in terms of both runtime and the number of itemsets with respect to every β. We can see that the runtime is proportional to the number of generated itemsets. In other words, the more number of itemsets consumes the greater execution time.

Figures 8(c) and (d) are the performance on real dense dataset, Mushroom, and *LS* and *minutil* are set to 0.0001 and 0.16. In Fig. 8(c), we can observe that the proposed algorithm outperforms CFP-Growth++ in almost every β. Furthermore, when β is no less than 0.08, the runtimes are almost the same. Especially, the more number of itemsets is generated on dense datasets than sparse datasets even though dataset size is much smaller such as between Retail and Mushroom datasets.

Meanwhile, Fig. 9 shows LMS values used in each experiment and memory. MHU-Growth uses the same LMS values with CFP-Growth++ on Mushroom dataset. On the contrary, MHU-Growth employs much larger LMS values in almost every β. That is, the proposed method effectively reduces search space because LMS refers to the least minimum support in the database. Besides, algorithms use constant memory regardless of β and memory usage is almost the same. The main reason is that the initial trees are constructed without pruning unimportant or infrequent items. Thus, MHU-Growth can find complete meaningful frequent itemsets including rare itemsets faster under similar memory usage.

Finally, we present the experimental results in terms of the number of tree nodes in mining process through Fig. 10. Figures 10(a) and (b), and (c) are the results on the synthetic dataset, T10.|I|1000.|D|100K, and the real datasets, Retail and Mushroom. Overall, the performance of the proposed method, MHU-Growth, is better than that of the previous algorithm, CFP-Growth++. In the



(a) LMS on Retail

(b) Memory Usage on Retail

(c) LMS on Mushroom

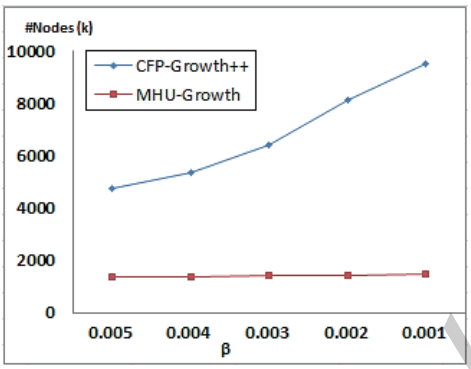
(d) Memory Usage on Mushroom

Fig. 9. Experimental results of LMS and memory usages. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140683)

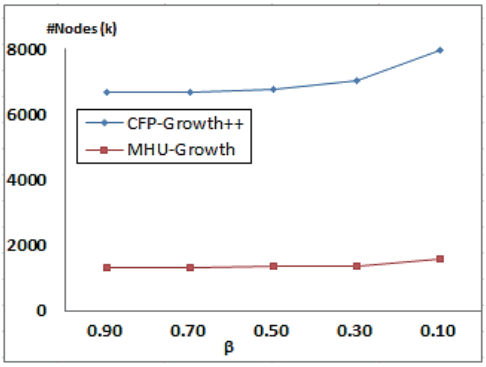
results, the compared algorithms create more nodes with decreasing β, and the increase rate by CFP-Growth++ is substantially larger than that by MHU-Growth. The reason is that the amount of eliminated search space by the compared method is highly reduced when β becomes smaller. In contrast, our algorithm decreases more tree nodes than CFP-Growth++ by removing search space effectively in mining process. In the experiment on the dense dataset, Mushroom, especially, they make more tree nodes than in the other experiments since average 19.328 items appear in each transaction of the dataset, and thereby the volume of information to be stored into trees becomes larger. The reason why overall performance of the proposed method outperforms that of CFP-Growth++ is that MHU-Growth reduces search space effectively by considering not only multiple minimum supports but also a minimum utility threshold. Through all of the experimental results, we can learn that the proposed algorithm effectively decreases both search space and the number of itemsets and make the performance much better than the CFP-Growth++ algorithm.

4.3. Scalability of the proposed method

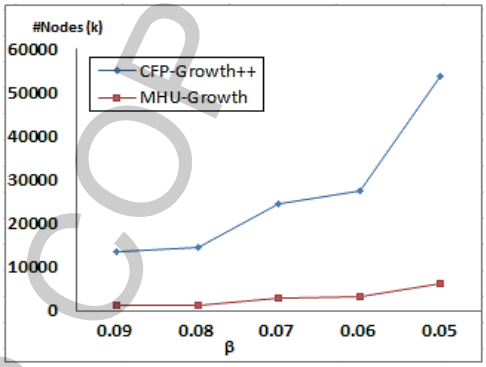
We perform scalability evaluation for the compared algorithms on synthetic datasets in Table 7. The experiments are conducted in terms of runtime and LMS, where $\beta = 0.01$, $LS = 0.0001$, and *minutil* $= 0.2$. First, experiment under varied database size on T10.|I|1000.|D|xK is performed. Figures 11(a) and (b) show results of runtime and LMS, respectively. In Fig. 11(a), we can observe that all algorithms



(a) Number of nodes on T10.|I|1000.|D|100K



(b) Number of nodes on Retail



(c) Number of nodes on Mushroom

Fig. 10. Experimental results of the number of nodes. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140683)

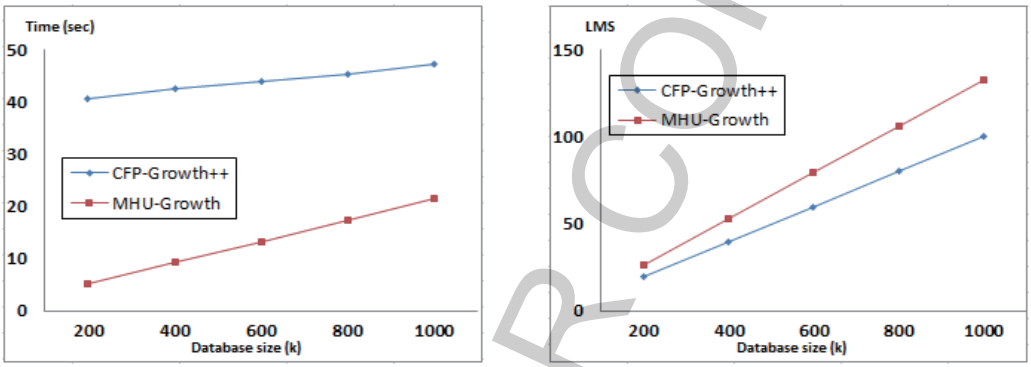
have good scalability on runtime. Meanwhile, LMS gets higher with increasing database size as shown in Fig. 11(b). The reason is that frequency of items increases with constant β, and thus MIS values of the items become higher based on the Eq. (1). Nevertheless, algorithms consume more runtime when database is larger.

4.4. Experimental results on real medical dataset

In this part, we perform the experiment on real medical data, Heart Disease dataset, for discovering meaningful patterns. The real medical dataset consists of 14 attributes, and Table 8 represents meaning of each attribute. It is necessary to convert each attribute value in the dataset according to the characteristics of heart disease because values have wide range. Thus, we split values of each attribute into several ranges based on risk level for heart attack, which is provided in [25], and Table 9 shows the information. That is, we employ weight values for ranges of each attribute, which are set by medical expert, as internal utilities. By using the table, we can convert the dataset into a transaction database, where each attribute value is represented as a non-binary risk level value. For example, if there is a patient who is 55 years old in the dataset, he or she is in high risk to get heart attack in accordance with [25]. In the study, moreover, weight of age between 50 and 70 is 0.7. In this experiment, we assign a quantity value of the range to a converted integer value 7. In a similar way, we assign non-binary quantity values for the rest ranges. Especially, the dataset includes an attribute, concept class which is used to classify disease status

Table 8
Attributes of heart disease dataset

Attribute	Description
#1 Age	Age in years
#2 Sex	Male, female
#3 CP	Chest pain type
#4 Blood pressure	Resting blood pressure upon hospital admission
#5 Cholesterol	Serum cholesterol in mg/dl
#6 fasting blood sugar	Fasting blood sugar is greater than 120 mg/dl or not
#7 Resting ECG	Resting electrocardiographic results
#8 Thalach	Maximum heart rate
#9 Induced angina	Patient experience angina as a result of exercise
#10 Old peak	ST depression induced by exercise relative to rest
#11 Slope	Slope of the peak exercise ST segment
#12 CA	Number of major vessels colored by fluoroscopy
#13 Thal	Normal, fixed defect, reversible defect
#14 Concept class	Angiographic disease status



(a) Runtime on T10.|I|1000.|D|xK (b) LMS on T10.|I|1000.|D|xK

Fig. 11. Experimental results of scalability evaluation. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-140683)

of each patient, where a zero value indicates normal status, and vice versa. It is also possible to employ the other factor such as duration time for item quantity. Moreover, item names are given to each range sequentially. In Table 9, the first and second ranges of the age attribute, for instance, are represented as A and B, respectively. We also use cost data provided with the dataset as external utilities, and it is also presented in Table 9. In this experiment, we conduct mining HUF patterns through four steps. In the first step, each data with respect to a patient in the medical dataset is represented as a non-binary transaction based on Table 9. In the second step, external utility of each item is assigned using the cost data. In the third step, not only the user-specified minimum utility threshold *minutil* but also β and *LS* are set, and then MIS value of each item is computed. In the last step, a complete set of HUF patterns is discovered.

Since the compared algorithm only considers multiple minimum supports, it cannot apply the information in Table 9. That is, itemsets related to the heart disease are extracted simply on a basis of their frequency in the framework with multiple minimum supports. On the contrary, our algorithm can identify significant itemsets by considering external and internal utilities of features that influence on the disease in the table. Therefore, we perform this experiment only with the MHU-Growth algorithm. In the proposed framework, each extracted pattern has its utility. That is, we can identify more important patterns from the result. Accordingly, we can find top k significant patterns. Parameters, *minutil*, β, and

Table 9
Utility of each item

Attribute	Utility	Quantity	
Age	1.00	1∼29	1
		30∼49	3
		50∼69	7
		70∼	8
Sex	1.00	Male (1)	1
		Female (0)	1
CP	1.00	Typical angina (1)	1
		Atypical angina (2)	1
		Non-angina pain (3)	1
		Asymptomatic (4)	1
Blood pressure	1.00	Low (∼119/79)	8
		Normal (130/80)	1
		High (200∼/160)	9
Cholesterol	7.27	Normal (∼159)	1
		High (160∼200)	8
		Very high (201∼)	9
Fasting blood sugar	5.20	True (121∼)	1
		False (∼120)	1
Resting ECG	15.50	Normal (1)	1
		ST-T wave abnormality (1)	1
		Left ventricular hypertrophy (2)	1
Thalach	102.90	Low (∼59)	9
		Normal (60∼100)	1
		High (101∼)	9
Induced angina	87.30	Yes (1)	1
		No (0)	1
Old peak	87.30		1
Slope	87.30	Up-sloping (1)	1
		Flat (2)	1
		Down-sloping (3)	1
CA	100.90	Number of major vessels (0∼3)	1
Thal	102.90	Normal (0)	1
		Fixed defect (1)	1
		Reversible defect (2)	1
Concept class	1.00	Normal (0)	1
		Abnormal (1∼4)	1

Table 10
The number of important rules

Concept class	Number of rules
0	465
1	82
2	125
3	117
4	348

Table 11
Identified significant patterns

Concept class	Patterns
1	#1(50∼70)-#4(Normal)-#5(Very High)-#6(0)-#8(High)
	#2(Male)-#4(Normal)-#5(Very High)-#6(0)-#8(High)
	#5(Normal)-#8(High)
	#6(0)-#8(High)
	#8(High)
2	#1(50∼70)-#4(Normal)-#5(Very High)-#11(2)
	#4(Normal)-#5(Very High)-#8(High)-#11(2)
	#1(50∼70)-#8(High)
	#5(Normal)-#8(High)
	#8(High)
3	#3(4)-#4(Normal)-#8(High)-#13(7)
	#4(Normal)-#5(Very High)-#8(High)-#13(7)
	#4(Normal)-#8(High)
	#8(High)-#13(7)
	#8(High)
4	#1(50∼70)-#3(4)-#4(Normal)-#5(Very High)-#7(2)-#8(High)-#11(2)
	#1(50∼70)-#3(4)-#4(Normal)-#6(0)-#7(2)-#8(High)-#11(2)
	#5(Very High)-#8(High)
	#6(0)-#8(High)
	#8(High)

LS, are assigned as 0.5, 0.5, and 0.001. It is also set to avoid applying only a single minimum support threshold for the whole items.

Table 10 shows the number of generated patterns with respect to each attribute of the concept class. Most discovered patterns are about normal patients with zero of the class, followed by 4, 2, 3, and 1. In this experiment, we analyze significant patterns related to abnormal patients. The part of the most significant or longest rule of each abnormal concept class is shown in Table 11. We can learn these facts from the result: 1) there is the high possibility of a heart attack in age between 50 and 70 or flat slope, 2) patients have very high cholesterol, 3) high maximum heart rate, 4) reversible defect thal, and so on. Above this, we can also obtain other significant information about heart attack from other mined

patterns. On the other hand, we used risk level of each range as item quantity but it is possible to employ other factors related to disease such as duration time. In addition, we can also divide each attribute into more ranges.

5. Conclusions

For reflecting item characteristics of real applications, we proposed an algorithm, MHU-Growth, with multiple minimum supports, item utilities, and non-binary transactions. Moreover, we also suggested a novel tree structure, called MHU-Tree, which is constructed with a single database scan. To effectively reduce search space and the number of generated itemsets, four pruning conditions were used. The experimental results on both real and synthetic datasets show that proposed framework improves performance and outperforms a state-of-the-art algorithm. Moreover, the experiment on real medical dataset, Heart Disease, shows that it is possible to discover and analyze significant patterns related to the characteristics of medical dataset.

Acknowledgment

This research was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF No. 2013005682 and 20080062611), the MSIP (Ministry of Science, ICT & Future Planning), Korea, under ICT/SW Creative research program supervised by the NIPA (National ICT Industry Promotion Agency) (NIPA-2014-H0502-14-3008), and the Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2015 (Grants No. C0232102).

References

[1] R. Agrawal and R. Srikant, Fast algorithms for mining association rules, in: *Proc of the 20th Int'l Conf, on Very Large Data Bases (VLDB 1994)* (1994), 487–499.
[2] C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong, Y.-K. Lee and H.-J. Choi, Single-pass incremental and interactive mining for weighted frequent patterns, *Expert Systems with Applications* **39**(9) (2012), 7976–7994.
[3] C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong and Y.-K. Lee, Efficient tree structures for high utility pattern mining in incremental databases, *IEEE Transactions on Knowledge and Data Engineering* **21**(12) (2009), 1708–1721.
[4] V.P. Álvarez and J.M. Vázquez, An evolutionary algorithm to discover quantitative association rules from huge databases without the need for an apriori discretization, *Expert Systems with Applications* **39**(1) (2012), 2012.
[5] R. Chaves, J. Ramírez, J.M. Górriz and C.G. Puntonet, Association rule-based feature selection method for alzheimer's disease diagnosis, *Expert Systems with Applications* **39**(14) (2012), 11766–11774.
[6] C.-H. Chen, T.-P. Hong and V.S. Tseng, An improved approach to find membership dunctions and multiple minimum supports in fuzzy data mining, *Expert Systems with Applications* **36**(6) (2009), 10016–10024.
[7] C.-H. Chen, T.-P. Hong and V.S. Tseng, Genetic-fuzzy mining with multiple minimum supports based on fuzzy clustering, *Soft Computing* **15**(12) (2011), 2319–2333.
[8] S.-S. Chen, T.C.-K. Huang and Z.-M. Lin, New and efficient knowledge discovery of partial periodic patterns with multiple minimum supports, *Journal of Systems and Software* **84**(10) 1638–1651.
[9] J. Han and Y. Fu, Discovery of multiple-level association rules from large databases, in: *Proc of the 21th Int'l Conf on Very Large Database (VLDB 1995)* (1995), 420–431.
[10] J. Han, J. Pei and Y. Yin, Mining frequent patterns without candidate generation, in: *Proc of the 2000 ACM SIGMOD Int'l Conf on Management of Data* (2000), 1–12.
[11] T.C.-K. Huang, Discovery of fuzzy quantitative sequential patterns with multiple minimum supports and adjustable membership functions, *Information Sciences* **222** (2013), 126–146.

[12] Y.-H. Hu, F. Wu and Y.-J. Liao, An efficient tree-based algorithm for mining sequential patterns with multiple minimum supports, *Journal of Systems and Software* **86**(5) (2013), 1224–1238.
[13] Y.-H. Hu and Y.-L. Chen, Mining association rules with multiple minimum supports: A new mining algorithm and a support tuning mechanism, *Decision Support Systems* **42**(1) (2006), 1–24.
[14] R.U. Kiran and P.K. Reddy, Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms, *The 14th Int'l Conf on Extending Database Technology (EDBT 2011)* (2001), 11–20.
[15] W. Lee, S.J. Stolfo and K.W. Mok, Mining audit data to build intrusion detection models, in: *Proc the 4th Int'l Conf on Knowledge Discovery and Data Mining (KDD 1998)* (1998), 66–72.
[16] G. Lee, U. Yun and K. Ryu, Sliding window based weighted maximal frequent pattern mining over data streams, *Expert Systems with Applications* **41**(2) (Feb 2014), 694–708.
[17] Y.-C. Lee, T.-P. Hong and T.-C. Wang, Multi-level fuzzy mining with multiple minimum supports, *Expert Systems with Applications* **34**(1) (2008), 459–468.
[18] C.-W. Lin, G.-C. Lan and T.-P. Hong, An incremental mining algorithm for high utility itemsets, *Expert Systems with Applications* **39**(8) (2012), 7173–7180.
[19] C.-W. Lin, T.-P. Hong and W.-H. Lu, An effective tree structure for mining high utility itemsets, *Expert Systems with Applications* **38**(6) (2011), 7419–7424.
[20] X.-B. Liu, K. Zhai and W. Pedrycz, An improved association rules mining method, *Expert Systems with Applications* **39**(1) (2012), 1362–1374.
[21] Y.-C. Liu, C.-P. Cheng and V.S. Tseng, Discovering relational-based association rules with multiple minimum supports on microarray datasets, *Bioinformatics* **27**(22) (2011), 3142–3148.
[22] B. Liu, W. Hsu and Y. Ma, Mining association rules with multiple minimum supports, in: *Proc of the Fifth ACM SIGKDD Int'l Conf on Knowledge Discovery and Data Mining (KDD 1999)* (1999), 337–341.
[23] Y. Liu, W.-K. Liao and A.N. Choudhary, A two-phase algorithm for fast discovery of high utility itemsets, *Advances in Knowledge Discovery and Data Mining (PAKDD 2005)* (2005), 689–695.
[24] H. Mannila, Database methods for data mining, in: *ACM SIGKDD Conf on Knowledge Discovery and Data Mining (KDD 1998) Tutorial*, 1998.
[25] S.B. Patil and V.S. Kumaraswamy, Intelligent and effective heart attack prediction system using data mining and artificial neural network, *European Journal of Scientific Research* **31**(4) (2009), 642–656.
[26] G. Pyun, U. Yun and K. Ryu, Efficient frequent pattern mining based on linear prefix tree, *Knowledge-Based Systems* **55** (Jan 2014), 125–139.
[27] M. Shinoda, T. Ozaki and T. Ohkawa, Weighted frequent subgraph mining in weighted graph databases, *IEEE Int'l Conf on Data Mining Workshops (ICDM 2009)* (2009), 58–63.
[28] V.S. Tseng, B.-E. Shie, C.-W. Wu and P.S. Yu, Efficient algorithms for mining high utility itemsets from transactional databases, *IEEE Transactions on Knowledge and Data Engineering* **25**(8) (2013), 1772–1786.
[29] C.-W. Wu, P. Fournier-Viger, P.S. Yu and V.S. Tseng, Efficient mining of a concise and lossless representation of high utility itemsets, *The 11th IEEE Int'l Conf on Data Mining (ICDM 2011)* (2011), 824–833.
[30] U. Yun, G. Lee and K. Ryu, Mining maximal frequent patterns by considering weight conditions over data streams, *Knowledge-Based Systems* **55** (Jan 2014), 49–65.
[31] U. Yun, H. Ryang and K. Ryu, High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates, *Expert Systems with Applications* **41**(8) (June 2014), 3861–3878.
[32] U. Yun and K. Ryu, Efficient mining of maximal correlated weight frequent patterns, *Intelligent Data Analysis* **17**(5) (Sep 2013), 917–939.