More Efficient Algorithm for Mining Frequent Patterns with Multiple Minimum Supports

Wensheng Gan[1], Jerry Chun-Wei Lin[1(✉)], Philippe Fournier-Viger[2], and Han-Chieh Chao[1,3]

[1] School of Computer Science and Technology,
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
wsgan001@gmail.com, jerrylin@ieee.org
[2] School of Natural Sciences and Humanities,
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
philfv@hitsz.edu.cn
[3] Department of Computer Science and Information Engineering,
National Dong Hwa University, Hualien County, Taiwan
hcc@ndhu.edu.tw

Abstract. Frequent pattern mining (FPM) is an important data mining task, having numerous applications. However, an important limitation of traditional FPM algorithms, is that they rely on a single minimum support threshold to identify frequent patterns (FPs). As a solution, several algorithms have been proposed to mine FPs using multiple minimum supports. Nevertheless, a crucial problem is that these algorithms generally consume a large amount of memory and have long execution times. In this paper, we address this issue by introducing a novel algorithm named efficient discovery of Frequent Patterns with Multiple minimum supports from the Enumeration-tree (FP-ME). The proposed algorithm discovers FPs using a novel Set-Enumeration-tree structure with Multiple minimum supports (ME-tree), and employs a novel *sorted downward closure (SDC) property* of FPs with multiple minimum supports. The proposed algorithm directly discovers FPs from the ME-tree without generating candidates. Furthermore, an improved algorithms, named FP-ME$_{\text{DiffSet}}$, is also proposed based on the *DiffSet* concept, to further increase mining performance. Substantial experiments on real-life datasets show that the proposed approaches not only avoid the "rare item problem", but also efficiently and effectively discover the complete set of FPs in transactional databases.

Keywords: Frequent patterns · Multiple minimum supports · Sorted downward closure property · Set-enumeration-tree · DiffSet

1 Introduction

In the process of knowledge discovery in database (KDD) [2,3], many approaches have been proposed to discover more useful and invaluable information from

B. Cui et al. (Eds.): WAIM 2016, Part I, LNCS 9658, pp. 3–16, 2016.
DOI: 10.1007/978-3-319-39937-9_1

huge databases. Among them, frequent pattern mining (FPM) and association rule mining (ARM) [2–4] have been extensively studied. Most studies in FPM or ARM focus on developing efficient algorithms to mine frequent patterns (FPs) in a transactional database, in which the occurrence frequency of each itemset is no less than a specified number of customer transactions w.r.t. the user-specified minimum support threshold (called *minsup*). However, they suffer from an important limitation, which is to utilize a single minimum support threshold as the measure to discover the complete set of FPs. Using a single threshold to assess the occurrence frequencies of all items in a database is inadequate since each item is different and should not all be treated the same. Hence, it is hard to carry out a fair measurement of the frequencies of itemsets using a single minimum support when mining FPs.

For market basket analysis, a traditional FPM algorithm may discover many itemsets that are frequent but generate a low profit and fail to discover itemsets that are rare but generate a high profit. For example, clothes i.e., $\{shirt, tie, trousers, suits\}$ occur much more frequent than $\{diamond\}$ in a supermarket, and both have positive contribution to increase the profit amount. If the value of *minsup* is set too high, though the rule $\{shirt, tie \Rightarrow trousers\}$ can be found, we would never find the rule $\{shirt, tie \Rightarrow diamond\}$. To find the second rule, we need to set the *minsup* very low. However, this will cause lots of meaningless rules to be found at the same time. This is the co-called "rare item problem" [7]. To address this issue, the problem of frequent pattern mining with multiple minimum supports (FP-MMS) has been studied. Liu et al. [7] introduced the problem of FP-MMS and proposed the MSApriori algorithm by extending the level-wise Apriori algorithm. The goal of FP-MMS is to discover the useful set of itemsets that are "frequent" for the users. It allows the users to free set multiple minimum support thresholds instead of an uniform minimum support threshold to reflect different natures and frequencies of all items. Some approaches have been designed for the mining task of FP-MMS, such as MSApriori [7], CFP-growth [11], CFP-growth++ [10], etc. The state-of-the-art CFP-growth++ was proposed by extending the FP-growth [4] approach to mine FPs from a condensed CFP-tree structure. However, the mining efficiency of them is still a major problem. Since the previous studies of FP-MMS still suffer the time-consuming and memory usage problems, it is thus quite challenging and critically important to design an efficient algorithm to solve this problem. In this paper, we propose a novel mining framework named mining frequent patterns from the Set-enumeration-tree with multiple minimum supports to address this important research gap. Major contributions are summarized as follows:

– Being different from the Apriori-like and FP-growth-based approaches, we propose a novel algorithm for directly extracting FPs with multiple minimum supports from the Set-enumeration-tree (abbreviated as FP-ME). It allows the user to specify multiple minimum support thresholds to reflect different natures and frequencies of items. This increases the applicability of FPM to real-life situations.

- Based on the proposed Set-Enumeration-tree with Multiple minimum supports (ME-tree), a new *sorted downward closure (SDC) property* of FPs in ME-tree holds. Therefore, the baseline algorithm FP-ME can directly discover FPs by spanning the ME-tree with the *SDC property*, without the candidate generate-and-test, thus greatly reduce the running time and memory consumption.
- The *DiffSet* concept is further extend in the improved FP-ME$_{\text{DiffSet}}$ algorithm to greatly speed up the process of mining FPs.
- Extensive experiments show that the proposed FP-ME algorithm is more efficient than the state-of-the-art CFP-growth++ algorithm for mining FPs in terms of runtime, effectiveness of prune strategies and scalability, especially the improved algorithm considerably outperforms the baseline algorithm.

2 Related Work

Up to now, many approaches have been extensively developed to mine FPs. Since only a single *minsup* value is used for the whole database, the model of ARM implicitly assumes that all the items in the database have similar occurrence frequencies. However, as most of the real-world databases are non-uniform in nature, mining FPs with a single *minsup* (or *mincof*) constraint leads to the following problems: (i) If *minsup* is set too high, we will not find the patterns involving rare items. (ii) In order to find the patterns that involve both frequent and rare items, we have to set *minsup* very low. However, this may cause combinatorial explosion, producing many meaningless patterns.

The problem of frequent pattern mining with multiple minimum support thresholds has been extensively studied, and algorithms such as MSApriori [7], CFP-growth [11], CFP-growth++ [10], REMMAR [8] and FQSP-MMS [6] have been proposed, among others [9]. MSApriori extends the well-known Apriori algorithm to mine FPs or ARs by considering multiple minimum support thresholds [7]. The major idea of MSApriori is that by assigning specific minimum item support (*MIS*) values to each item, rare ARs can be discovered without generating a large number of meaningless rules. MSApriori mines ARs in a level-wise manner but suffers from the problem of "pattern explosion" since it relies on a generate-and-test approach. Lee et al. [9] then proposed a fuzzy mining algorithm for discovering useful fuzzy association rules with multiple minimum supports by using maximum constraints. An improved tree-based algorithm named CFP-growth [11] was then proposed to directly mine frequent itemsets with multiple thresholds using the pattern growth method based on a new MIS-Tree structure. An enhanced version of CFP-growth named CFP-growth++ [10] was also proposed, it employs *LMS* (least minimum support) instead of *MIN* to reduce the search space and improve performance. *LMS* is the least *MIS* value amongst all *MIS* values of frequent items. Moreover, three improved strategies were also presented to reduce the search space and runtime. However, it is still too time-consuming and memory cost.

Table 1. An example database

TID	Transaction
T_1	a, c, d
T_2	a, d, e
T_3	b, c
T_4	a, c, e
T_5	a, b, c, d, e
T_6	b, d
T_7	a, b, c, e
T_8	b, c, d
T_9	c, d, e
T_{10}	a, c, d

Table 2. Derived FPs

Itemset	$MIS \times \|D\|$	*sup*	Itemset	$MIS \times \|D\|$	*sup*
(a)	4	6	(ae)	4	4
(b)	5	5	(bc)	3	4
(c)	3	8	(cd)	3	5
(d)	6	7	(ce)	3	4
(ac)	3	5	(acd)	3	3
(ad)	4	4	(ace)	3	3

3 Preliminaries and Problem Statement

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of m distinct items. A transactional database $D = \{T_1, T_2, \ldots, T_n\}$, where each transaction $T_q \in D$ is a subset of I, contains several items with their purchase quantities $q(i_j, T_q)$, and has an unique identifier, *TID*. A corresponding multiple minimum supports table, $MMS\text{-}table = \{ms_1, ms_2, \ldots, ms_m\}$, indicates the user-specified minimum support value ms_j of each item i_j. A set of k distinct items $X = \{i_1, i_2, \ldots, i_k\}$ such that $X \subseteq I$ is said to be a k-itemset. An itemset X is said to be contained in a transaction T_q if $X \subseteq T_q$. An example database is shown in Table 1. It consists of 10 transactions and 5 items, denoted from (a) to (e), respectively. For example, transaction T_1 contains items a, c and d. The minimum support value of each item, denoted as ms, is defined and as shows the $MMS\text{-}table = \{ms(a)$: 40 %; $ms(b)$: 50 %; $ms(c)$: 30 %; $ms(d)$: 60 %; $ms(e)$: 100 %$\}$.

Definition 1. The number of transactions that contains an itemset is known as the occurrence frequency of that itemset. This is also called the support count of the itemset. The support of an itemset X, denoted by $sup(X)$, is the number of transactions containing X w.r.t. $X \subseteq T_q$.

Definition 2. The minimum support threshold of an item i_j in a database D, which is related to *minsup*, is redenoted as $ms(i_j)$ in this paper. A structure called *MMS-table* indicates the minimum support threshold of each item in D and is defined as: $MMS\text{-}table = \{ms_1, ms_2, \ldots, ms_m\}$.

Definition 3. The minimum item support value of a k-itemset $X = \{i_1, i_2, \ldots, i_k\}$ in D is denoted as $MIS(X)$, and defined as the smallest ms value for items in X, that is: $MIS(X) = min\{ms(i_j)|i_j \in X\}$.

For example, $MIS(a) = min\{ms(a)\} = 40\,\%$, $MIS(ae) = min\{ms(a), ms(e)\} = min\{40\,\%, 100\,\%\} = 40\,\%$, and $MIS(ace) = min\{ms(a), ms(c), ms(e)\} = min\{40\,\%, 30\,\%, 100\,\%\} = 30\,\%$. The extended model enables the

user to simultaneously specify high *minsup* for a pattern containing only frequent items and low *minsup* for a pattern containing rare items. Thus, efficiently addressing the "rare item problem".

Definition 4. An itemset X in a database D is called a frequent pattern (FP) iff its support count is no less than the minimum itemset support value of X, such that $sup(X) \geq MIS(X) \times |D|$.

Definition 5. Let be a transactional database D ($|D| = n$) and a *MMS-table*, which defines the minimum support thresholds $\{ms_1, ms_2, \ldots, ms_m\}$ of each item in D. The problem of mining FPs from D with multiple minimum supports (FP-MMS) is to find all itemsets X having a support no less than $MIS(X) \times |D|$.

For the running example, the derived complete set of FPs is shown in Table 2.

4 Proposed FP-ME Algorithm for FP-MMS

4.1 Proposed ME-Tree

Based on the previous studies, the search space of mining frequent patterns with multiple minimum supports can be represented as a lattice structure [12] or a Set-enumeration tree [12], both of them for the running example are respectively shown in Fig. 1. Note that the well-known downward closure property in association rule mining does not hold for FP-MMS. For example, the item (e) is not a FP but its supersets (ae) and (ace) are FPs in the running example. To solve this problem, Liu et al. [7] proposed a concept called *sorted closure property*, which assumes that all items within an itemset are sorted in increasing order of their minimum supports.

Property 1. If a sorted k-itemset $\{i_1, i_2, \ldots, i_k\}$, for $k \geq 2$ and $MIS(i_1) \leq MIS(i_2) \leq \ldots \leq MIS(i_k)$, is frequent, then all of its sorted subsets with $k-1$ items are frequent, except for the subset $\{i_2, i_3, \ldots, i_k\}$ [6].

Definition 6 (Total order $\prec$ on items). Without loss of generality, assume that items in each transaction of a database are sorted according to the lexicographic order. Furthermore, assume that the total order $\prec$ on items in the designed ME-tree is the ascending order of items *MIS* values.

Definition 7 (Set-enumeration-tree with multiple minimum supports). A ME-tree is a sorted Set-enumeration tree using the defined total order $\prec$ on items.

Definition 8 (Extension nodes in the ME-tree). In the designed ME-tree with the total order $\prec$, all child nodes of any tree node are called its extension nodes.

Since $MIS(c) < MIS(a) < MIS(b) < MIS(d) < MIS(e)$, the total order $\prec$ on items in the ME-tree is $c \prec a \prec b \prec d \prec e$. The ME-tree for the running example is illustrated in Fig. 2, and the following lemmas are obtained.

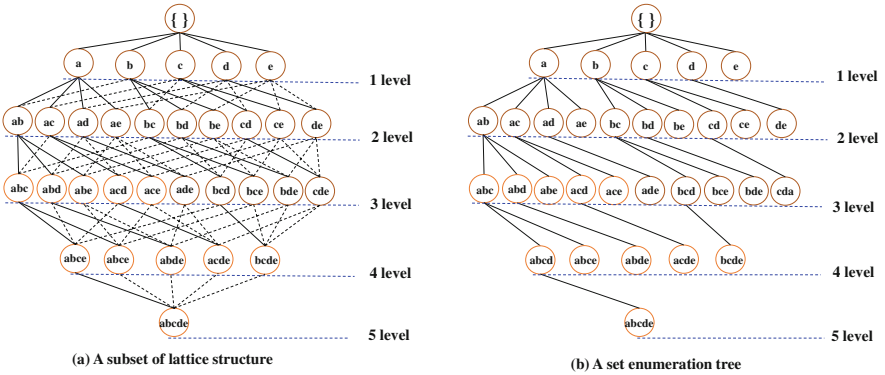

Fig. 1. The search space presentation of FP-MMS.

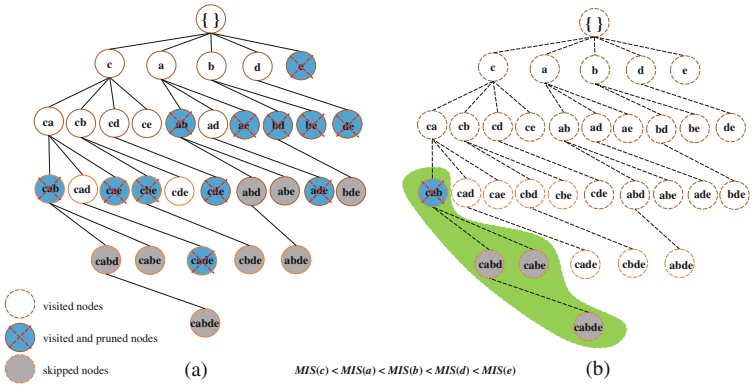

Fig. 2. Applied pruning strategies in the ME-tree.

Lemma 1. *The complete search space of the proposed FP-MMS algorithm can be represented by a ME-tree where items are sorted according to the ascending order of the MIS values on items.*

Lemma 2. *The support of a node in the ME-tree is no less than the support of any of its child nodes (extension nodes).*

Proof. Let X^k be a node in the ME-tree containing X items, and let X^{k-1} be any parent node of X^k containing (k-1) items. It is straightforward from the well-known Apriori property that $sup(X^k) \leq sup(X^{k-1})$, this lemma can be proven.

Lemma 3. *The MIS of a node in the ME-tree equals to the MIS of any of its child nodes (extension nodes).*

4.2 Proposed Pruning Strategies

It is important to note that the *sorted downward closure (SDC) property* of FPs in the ME-tree can only guarantee partial anti-monotonicity for FPs, but not general anti-monotonicity. In other words, the *SDC property* holds for any extensions (child nodes) of a given node, but it may not hold for any supersets of that node. Thus, if the *SDC property* of FPs is used to determine if all supersets of an itemset should be explored, some FPs may not be found. For instance, in the running example database, the item (e) is not a FP since $sup(e) = 5$ (<10), while its supersets (ae), (ce) and (ace) are FPs, as shown in Table 2. It is thus incorrect to directly determine the FPs based only on the proposed *SDC property*. In MSApriori and CFP-growth++, it was shown that the MIN/LMS concept can guarantee the global anti-monotonicity of frequent patterns with multiple minimum supports and ensure the completeness of the set of derived FPs. To address this problem, we further adopt the MIN/LMS concept in the proposed FP-ME algorithm.

Definition 9 (Least minimum support, *LMS*). The least minimum support (LMS) refers to the lowest $minsup$ of all frequent patterns. Therefore, the LMS in a database is always equal to the lowest MIS value among all frequent items. Thus, the LMS is equal to the lowest value in the *MMS-table* and is defined as $min\{ms(i_1), ms(i_2), \ldots, ms(i_m)\}$, where m is the total number of items in a database.

For example, the LMS of Table 2 is calculated as $LMS = min\{ms(a), ms(b), ms(c), ms(d), ms(e)\} = min\{40\,\%, 50\,\%, 30\,\%, 60\,\%, 100\,\%\} = 30\,\%$.

Property 2. If $X = \{i_1, i_2, \ldots, i_k\} \subseteq I$, where $1 \leq k \leq n$, is a pattern such that $sup(X) < LMS$, then $sup(X) < min\{MIS(i_1), MIS(i_2), \ldots, MIS(i_k)\}$, it never could be a frequent pattern.

Property 3. If X and Y are two patterns such that $X \subset Y$ and $sup(X) < LMS$, then $sup(Y) < LMS$. It indicates that LMS guarantees the global anti-monotonicity of frequent patterns with multiple minimum supports.

Proof. Let be an itemset X such that X is a subset of Y. Thus, $sup(Y) \leq sup(X)$. The relationship $sup(Y) \leq sup(X) < LMS$ holds.

Note that the set of 1-items which having $sup(X) \geq LMS$ is denoted as $LMS\text{-}FP^1$, the following theorem can be obtained.

Theorem 1. *Assume that 1-itemsets which having a MIS lower than LMS are discarded and that the sorted downward closure (SDC) property is applied. We have that if an itemset is not a $LMS\text{-}FP^1$, then it is not a FP as well as all its supersets.*

Proof. Let X^{k-1} be a (k-1)-itemset and its superset k-itemset is denoted as X^k. Since $X^{k-1} \subseteq X^k$, (1) For a $LMS\text{-}FP^1$, $sup(X) \geq LMS$; (2) Since items are sorted by ascending order of MIS values, $sup(X^{k-1}) \geq sup(X^k)$

and $MIS(X^{k-1}) = MIS(X^k) = min\{MIS(i_1), MIS(i_2), \ldots, MIS(i_m)\} = MIS(i_1)$. Thus, if X^{k-1} is not a $LMS\text{-}FP^1$ ($MIS(X^{k-1}) < LMS$), none of its supersets are FPs.

Based on the designed ME-tree, the above lemmas and theorems ensure that all FPs are included in the extensions of the set of $LMS\text{-}FP^1$ and thus that we can safely discard the other itemsets. Thus, the designed *sorted downward closure (SDC) property* in ME-tree can guarantee the completeness and correctness of the proposed FP-MMS framework. These properties facilitate to use LMS as a constraint to reduce the search space. Based on the above analysis, it can be shown that the LMS plays a significant role in the FP-ME algorithm, it not only ensures the completeness of the set of derived FPs, but also can be used to prune the search space.

Strategy 1. *In the designed ME-tree using the total order $\prec$, if a node X has a support value less than the LMS, then any nodes which contains X w.r.t. all supersets of X can be directly pruned, X would not be used to explored in the ME-tree.*

Lemma 4. *Given a transactional database D and multiple minimum support threshold $MIS(i_k)$ of each item i_k, the constructed ME-tree contains the complete information about frequent patterns in D.*

Proof. In the construction process of ME-tree, each transaction in D can be mapped to one path in the ME-tree whenever necessary. And according to Theorems 1 and 2, all promising itemsets, the $LMS\text{-}FP^1$, their information in each transaction is completely stored in the ME-tree based on the total order $\prec$. Notice that we retained those infrequent items with supports no less than LMS w.r.t. the $LMS\text{-}FP^1$ in the ME-tree because these items their supersets may be frequent.

Strategy 2. *Let be the designed ME-tree using the MIS-ascending order of items. If a node X has a support value less than its MIS w.r.t. the MIS of its prefix level-2 node, then any extension of X w.r.t. all child nodes of X can be directly pruned.*

For example, the effect of **Pruning Strategy 1** and **Pruning Strategy 2** in the running example are respectively shown in Fig. 2(a) and (b). The itemset (cab) is not considered to be a FP since $sup(cab)$ ($= 2 < MIS(cab) \times |D|$). By applying the **Pruning Strategy 2**, all the child nodes of itemset (cab) are not considered to be FPs since their support values are always no greater than those of (cab). Hence, the child nodes ($cabd$), ($cabe$) and ($cabde$) (the shaded nodes in Fig. 2) are guaranteed to be uninteresting and can be pruned safely.

4.3 Proposed FP-ME Algorithm

Note that the top-down traversal strategy is adopted in the ME-tree. As shown in Algorithm 1, the FP-ME algorithm first scans the database to calculate the MIS

value of each single item, as well as the LMS, then discover the set of I^* w.r.t. $LMS\text{-}FP^1$ (Lines 1 to 2). It is important to notice that here the derived patterns are the set of $LMS\text{-}FP^1$ but not the set of FPs, the reason had been mentioned before. The discovered $LMS\text{-}FP^1$ are then sorted in MIS-ascending in order to construct their TID-Sets by scanning database again, thus forming the final set of TID-Sets of all item $i \in I^*$ (Lines 3 to 4). Afterwards each 1-itemset X in the set of I^* is processed in designed order $\prec$ to find FPs from the ME-tree, using the constructed TID-Sets without multiple rescanning the database (Line 5, FP-Spanning procedure). As mentioned before, the spanning miner mechanism in the FP-ME is different form the generate-and-test algorithm and the pattern-growth approaches. The main idea of the FP-Spanning procedure (c.f. Algorithm 2) is that for each 1-itemset X_a, extensions of X_a are recursively explored using a depth-first search. Moreover, each itemset encountered during that search is evaluated to determine if it is a defined FP (Lines 2 to 4). Note that the depth search is only performed for an itemset X_a if the support of X_a (which is directly obtained from the relevant TID-Sets by calculating its TIDs) is larger than or equal to $MIS(X_a) \times |D|$ (Lines 3 to 4). Simultaneously, the TID-Sets of the

Input: D $(n = |D|)$, *MMS-table*.
Output: The set of frequent patterns (FPs).
1 scan *MMS-table* to calculate the MIS value of each single item and put into the set of $MISArrary$, as well as the LMS among them ;
2 scan D to find $I^* \leftarrow \{i \in I | sup(i) \geq LMS\}$, w.r.t. the $LMS\text{-}FP^1$;
3 sort the set I^* in MIS ascending order $\prec$;
4 scan D once again to construct the TID-Set of each $i \in I^*$ such as $i.TidSet$;
5 **call FP-Spanning**$(\phi, I^*, MISArray)$;
6 **return** *FPs*;

Algorithm 1: FP-ME algorithm

Input: X: an itemset, $extensionsOfX$: a set of all extensions of X, $MISArray$: an array containing the minimum item support thresholds of all items.
Input: The set of frequent patterns (FPs).
1 **for** *each itemset* $X_a \in extensionsOfX$ **do**
2 calculate the $sup(X_a)$ and $MIS(X_a)$ from the built structure of X_a;
3 **if** $sup(X_a) \geq MIS(X_a) \times |D|$ **then**
4 $FPs \leftarrow FPs \cup X_a$;
5 $extensionsOfX_a \leftarrow \emptyset$;
6 **for** *each itemset* $X_b \in extensionsOfX_a$ *such that b after a* **do**
7 $X_{ab} \leftarrow X_a \cup X_b$;
8 calculate the $X_{ab}.TidSet$ by merging $X_a.TidSet$ and $X_b.TidSet$;
9 $extensionsOfX_a \leftarrow extensionsOfX_a \cup X_{ab}$;
10 **call FP-Spanning**$(X_a, extensionsOfX_a, MIS(X_a))$;
11 **return** FPs;

Algorithm 2: FP-Spanning Procedure

extensions of X_a is built (Lines 6 to 9). The above process is recursively executed until all items in $LMS\text{-}FP^1$ have been processed (Line 10).

4.4 Improved Algorithm with the *DiffSet* Strategy

In [13], a novel vertical data representation called *DiffSet* was presented, that only keeps track of differences in the TIDs of a candidate pattern from its generating frequent patterns. It had been shown that *DiffSet* drastically cut down the size of memory required to store intermediate results. Furthermore, we incorporated the *DiffSet* into previous vertical mining method FP-ME, significantly increasing the mining performance, and denoted this improved algorithm as FP-ME$_{\text{DiffSet}}$, for fast mining FPs. Details of the concept *DiffSet* and its construction in the FP-ME$_{\text{DiffSet}}$ algorithm were skipped here due to the space limitation.

5 Experimental Results

In this section, substantial experiments were conducted to verify the effectiveness and efficiency of the proposed baseline FP-ME algorithm (FP-ME$_{\text{baseline}}$) and the improved FP-ME$_{\text{DiffSet}}$ algorithm. Note that there are several studies have been done previously on the topic of mining FPs with multiple minimum supports, and it had been shown that the CFP-growth++ [10] significantly outperforms the MSApriori [7] and CFP-growth [11]; the state-of-the-art CFP-growth++ was thus executed to derive FPs, which can provide a benchmark to verify the efficiency of the proposed two algorithms. All algorithms are implemented in Java language and performed on a personal computer with an Intel Core i5-3460 dual-core processor and 4 GB of RAM and running the 32-bit Microsoft Windows 7 operating system. And experiments are conducted on two datasets named kosarak [1] and BMSPOS [1]. The source code of the MSApriori and CFP-growth++ algorithms can be download from the SPMF data mining library [5]. Note that the front 100 K transactions in kosarak are selected in the experiments.

Furthermore, the discussed method in [7] is adopt in the proposed FP-ME algorithm to automatically assign multiple item supports to items. The methodology is as follows: $MIS(i_j) = max[\beta \times f(i_j), LMS]$, where β is a constant used to set the *MIS* values of items as a function of their frequency (or support). To ensure the randomness and equipment diversity, β was set in the [0.0, 1.0] interval for the datasets. The parameter *LMS* is the user-specified least minimum itemset support allowed, and $f(i_j)$ is the frequency (or support) of an item i_j. Note that if β is set to zero, then a single *LMS* value will be used for all items, and this will be equivalent to traditional FPM. If $\beta = 1$ and $f(i_j) \geq LMS$, then $MIS(i_j) = f(i_j)$.

5.1 Execution Time

For the conducted experiments, the parameter β was randomly set to a fixed number of each item. Fig. 3 shows the runtime of the algorithms under various

LMS with a fixed β and under various β with a fixed *LMS* for different datasets. From Fig. 3, it can be observed that both the FP-ME$_{\text{baseline}}$ and the improved FP-ME$_{\text{DiffSet}}$ algorithms outperform the CFP-growth++ algorithm on the two datasets under various LMS with a fixed β or under a fixed LMS with various β. Moreover, we can see that FP-ME$_{\text{DiffSet}}$ has in general the best performance among them. It is reasonable since CFP-growth++ consists of three phases. It first scans the database once to construct a global MIS-Tree. All nodes in the Header-table of a MIS-Tree are sorted by the order of descending MIS values. Then, the MIS-Tree is restructured to reduce the search space by four pruning techniques. At last, CFP-growth++ recursively mines the tree by creating projected trees to generate all desired itemsets. This process is too time-consuming, thus it performs worse than the two proposed FP-ME algorithms. In contrast, the proposed algorithms utilize the "structuring when mining" property, and apply two pruning strategies to early prune unpromising itemsets and search space in ME-tree, which can avoid the costly join operations of a huge number of unpromising patterns. Moreover, the FP-ME$_{\text{DiffSet}}$ algorithm applies the *DiffSet* to quickly calculate the TID-Set, thus greatly reducing the computations than the baseline algorithm.

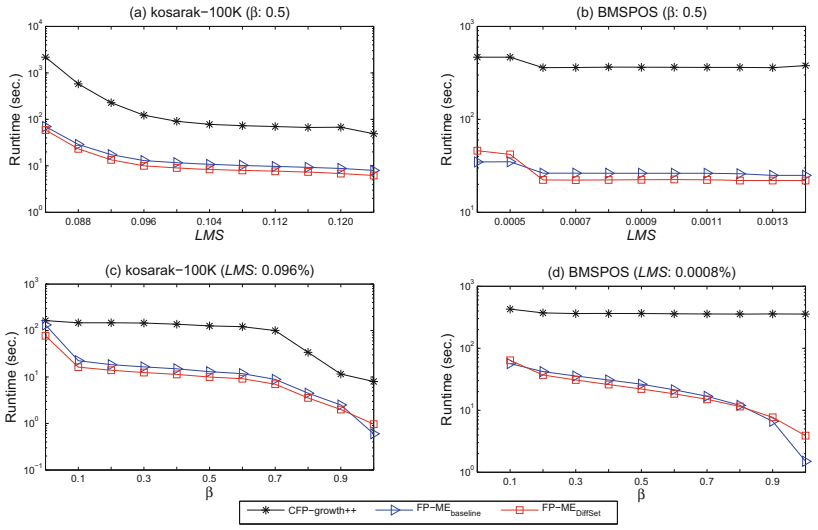

Fig. 3. Execution time.

5.2 Memory Usage

With the same test parameters, we also assessed the memory consumption of the compared algorithms. Memory measurements were done using the Java API. Note that the peak memory consumption of each algorithm was recorded for all datasets. Results are shown in Fig. 4. It can be clearly seen that the proposed algorithms, both FP-ME$_{\text{baseline}}$ and the improved FP-ME$_{\text{DiffSet}}$, require

less memory compared to the state-of-the-art CFP-growth++ algorithm under various parameters on the two datasets, and even up to 8 times as shown in Fig. 4(a). Specifically, the two ME-tree-based approaches require nearly constant memory under various parameter values on the all datasets. The memory usage of the generation-and-test CFP-growth++ algorithm dramatically increases as LMS or β decreases, while the memory usage of the proposed algorithms remain stable. Besides, the improved FP-ME$_{\text{DiffSet}}$ always consumes little more memory than that of FP-ME$_{\text{baseline}}$. This result is reasonable since the vertical data structure *DiffSet* is adopted to keep track of differences in the TIDs of a prefix pattern from its generating FPs.

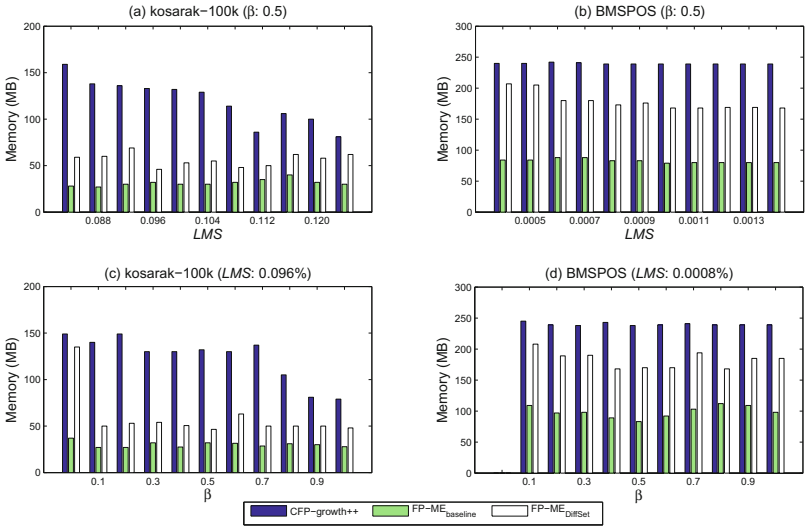

Fig. 4. Memory usage. (Color figure online)

5.3 Scalability Analysis

The scalability of the proposed methods are further evaluated by performing experiments when the kosarak dataset size was varied from 100 K to 500 K, by increments of 100 K each time. Figure 5 shows the runtime and memory usage for the three compared algorithms when the LMS and β were set to 0.001 and 0.5, respectively. It can be observed that the runtime of all compared algorithms is linear increased along with the increasing of dataset size $|X|$. The performance of P-ME$_{\text{baseline}}$ and the improved FP-ME$_{\text{DiffSet}}$ significantly scale better than that of CFP-growth++. With the increasing of the size of dataset, the runtime of FP-ME$_{\text{baseline}}$ is close to that of FP-ME$_{\text{DiffSet}}$. Specially, the gap of runtime among them grows wider with the increasing of dataset size. It also can be clearly seen that the proposed two algorithms require less memory compared to the state-of-the-art CFP-growth++ algorithm in a wide range of dataset size. From the observed results of the scalability test, it can be concluded that the proposed algorithms are more scalable than the previous algorithms.

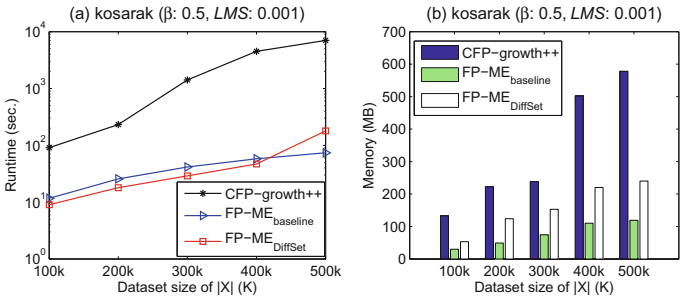

Fig. 5. Scalability of compared algorithms. (Color figure online)

6 Conclusion

In this paper, we proposed a novel FP-MMS framework namely FP-ME for mining frequent patterns with multiple minimum supports. Based on the designed Set-enumeration-tree with multiple minimum supports (ME-tree), a novel *sorted downward closure (SDC) property* was proposed. In order to guarantee the completeness of derived results, the *LMS* concept was extend in FP-ME to mine the FPs. Different from the generate-and-test approach, FP-ME can directly discover FPs by spanning the ME-tree with two pruning strategies. In addition, an improved algorithm by adopting the *DiffSet* concept is further developed to speed up the mining process by reducing the cost of database scans and pruning search space. From the experiments, it can be found that the proposed two algorithms significantly outperform the state-of-the-art CFP-growth++ algorithm in terms of execution time, memory usage and scalability. Specifically, the improved algorithm outperforms the baseline algorithm.

Acknowledgment. This research was partially supported by the National Natural Science Foundation of China (NSFC) under grant No. 61503092 and by the Tencent Project under grant CCF-TencentRAGR20140114.

References

1. Frequent itemset mining dataset repository. http://fimi.ua.ac.be/data/
2. Agrawal, R., Imielinski, T., Swami, A.: Database mining: A performance perspective. IEEE Trans. Knowl. Data Eng. **5**, 914–925 (1993)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: The International Conference on Very Large Data Bases, pp. 487–499 (1994)
4. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Disc. **8**(1), 53–87 (2004)
5. Fournier-Viger, P., Gomariz, A., Soltani, A., Gueniche, T., Wu, C.W., Tseng, V.S.: SPMF: A java open-source pattern mining library. J. Mach. Learn. Res. **15**, 3389–3393 (2014)

6. Huang, T.C.K.: Discovery of fuzzy quantitative sequential patterns with multiple minimum supports and adjustable membership functions. Inf. Sci. **222**, 126–146 (2013)
7. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 337–341 (1999)
8. Liu, Y.C., Cheng, C.P., Tseng, V.S.: Discovering relational-based association rules with multiple minimum supports on microarray datasets. Bioinformatics **27**(22), 3142–3148 (2011)
9. Lee, Y.C., Hong, T.P., Wang, T.C.: Mining fuzzy multiple-level association rules under multiple minimum supports. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 4112–4117 (2006)
10. Kiran, R.U., Reddy, P.K.: Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. In: ACM International Conference on Extending Database Technology, pp. 11–20 (2011)
11. Hu, Y.H., Chen, Y.L.: Mining association rules with multiple minimum supports: A new mining algorithm and a support tuning mechanism. Decis. Support Syst. **42**(1), 1–24 (2006)
12. Rymon, R.: Search through systematic set enumeration. Technical reports (CIS), vol. 297 (1992)
13. Zaki, M.J., Gouda, K.: Fast vertical mining using diffsets. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 326–335 (2003)