International Journal of Innovative
Computing, Information and Control
Volume **15**, Number **2**, April **2019**

ICIC International ©2019 ISSN 1349-4198
pp. 479–488

A PARALLEL METHOD FOR MINING FREQUENT PATTERNS WITH MULTIPLE MINIMUM SUPPORT THRESHOLDS

Bao Huynh[1], Cuong Trinh[2], Vu Dang[3] and Bay Vo[4,*]

[1]NLP-KD Lab, Faculty of Information Technology, Ton Duc Thang University, Vietnam
[2]Department of Computing and Computer Services, Ton Duc Thang University, Vietnam
[3]Center for Applied Information Technology, Ton Duc Thang University, Vietnam
No. 19, Nguyen Huu Tho Street, Tan Phong Ward, District 7, Ho Chi Minh City 700000, Vietnam
{ huynhquocbao; trinhphicuong; dangngocvu }@tdtu.edu.vn

[4]Faculty of Information Technology
Ho Chi Minh City University of Technology (HUTECH)
475A Dien Bien Phu Street, Ward 25, Binh Thanh District, Ho Chi Minh City 700000, Vietnam
*Corresponding author: vd.bay@hutech.edu.vn

Received May 2018; revised September 2018

Abstract. *Frequent pattern mining plays an important role in data mining. It has been widely applied in many fields such as trade, medicine and customer services. Many effective methods were developed to find out the complete set of frequent patterns by using a single minimum support threshold. However, using a single minimum support threshold is not practical for many real-work applications since it does not indicate the characteristics of each item. If the minimum support threshold is set too high, many interesting patterns (called rare itemsets) are possibly missed. Conversely, if it is set too low, the number of patterns will explode. On the other hand, the main problem of these methods is due to the huge consumption of memory and long execution time. This study proposes a method named MMS-FPM (Multiple Minimum Support - Frequent Pattern Mining) for fast mining frequent patterns with multiple minimum support thresholds based on multi-core processor platforms. Experimental results show that the proposed method is more effective than MSApriori (Multiple Support Apriori) and scalable on various types of databases.*

Keywords: Pattern mining, Multiple minimum support thresholds, Rare itemset mining, Multi-core platform

1. **Introduction.** Frequent Pattern Mining (FPM) [1] plays an important role in data mining and has attracted much attention from researchers because it has been essential in many areas such as decision support systems [2], networks detection [3], medical treatment [4], and trade strategy [5]. Frequent Patterns (FPs) refer to itemsets or subsequences, the appearance frequency of which in the database is not less than a user-specified threshold value. FPM has substantially affected the data analysis field, and it has a deep impact on data mining methods and applications. Although numerous efficient methods have been proposed such as Apriori [6], FP-growth (Frequent Pattern-growth) [7], Eclat [8], PrePost [9], PrePost+ [10], these methods only used a single minimum support (*minsup*) threshold. However, in many real-work applications, using a single minimum support threshold is not suitable since it does not indicate the characteristics of each item. If *minsup* is set too high, many interesting patterns, called rare items will be missed. Conversely, if *minsup* is set too low, many patterns will explode. The rare items contain highly valuable

DOI: 10.24507/ijicic.15.02.479

information, which is essential for many areas such as classification or periodic pattern mining, particularly in medicine or biology.

Many methods for FPM with *Multiple Minimum Support* (MMS) thresholds have been proposed, and they can be categorized into two groups: Apriori-like methods [11,12] and FP-growth-like methods [13,14].

In Apriori-like methods, the databases have to be scanned many times; hence, they take a long time to execute. To overcome this weakness, FP-growth-like methods were proposed. These methods just scan database twice in the mining process. However, these methods still require a huge amount of memory and high execution time for post prune and re-construct phase, especially for long patterns. It is, thus, quite challenging and very important to design an efficient algorithm to solve this problem.

In this study, we propose an effective method named MMS-FPM (*Multiple Minimum Support - Frequent Pattern Mining*) for fast mining frequent patterns with multiple minimum support thresholds based on multi-core processor platforms. The major contributions of this paper are as follows.

1) Develop lemmas to early pruning nodes in the tree.
2) Propose a parallel approach for fast mining frequent itemsets with multiple minimum support thresholds using multi-core processor platforms.

The experiments show that the proposed algorithm is more efficient than MSApriori algorithm in terms of runtime, due to the effectiveness of pruned strategies and scalability, especially the improved algorithm considerably outperforms the baseline algorithm.

The rest of this paper is organized as follows. Review related works in Section 2. Section 3 summarizes the basic concept. The parallel computing, two lemmas and proposed algorithm developed for fast pruning candidates are presented in Section 4. Section 5 shows the experimental results. Finally, the conclusions and future works are given in Section 6.

2. **Related Works.** Frequent Patterns (FPs) mining has attracted a lot of research in many decades; however, the interest in this problem still persists [15]. In addition, various methods of FP have also interests such as association rules [16], sequence mining [17,18], colossal mining [19], and high utility itemsets [20,21], especially on huge database volume. If *minsup* is set too high, we will lose rare item patterns. Otherwise, a huge number of redundant patterns will explode. To overcome the rare item problem [22], many solutions have been proposed such as MSApriori algorithm [11], which is an extension of Apriori algorithm [6] so it has the same strategy with Apriori approach. ARIMA (A Rare Itemset Miner Algorithm) [23] used a level-wise bottom-up approach that can find rare itemset without generating zero itemsets, and the itemset support was computed by scanning the database at each level. Apriori inverse algorithm [24] can find the sporadic itemsets much more quickly than Apriori; however, it cannot generate all the rare itemsets. Conditional Frequent Pattern-growth (CFP-growth) [13], which is an extension of the FP-growth [7] approach is developed for mining all frequent itemsets using MIS-tree structure. Since *apriori property* no longer holds in multiple *minsup* contexts, the CFP-growth algorithm has to carry out a complete search in the constructed tree, thus increasing the storage requirements of MIS-tree. CFP-growth++ [14] is an improvement of the CFP-growth algorithm, this method introduced four pruning techniques to reduce the search space in mining frequent patterns with multiple minimum supports, including Least Minimum Support (*LMS*), conditional minimum support, conditional closure property and infrequent leaf node pruning. *LMS* is the least *MIS* value amongst all *MIS* values of frequent items is used to reduce the search space and improve performance. However, it is still too time-consuming and memory costly.

3. **Basic Concepts.** Let $I = \{i_1, i_2, \ldots, i_n\}$ be set of n distinct *items* and a transaction database $D = \{T_1, T_2, \ldots, T_m\}$, where $T_i \subseteq I$ $(1 \leq i \leq m)$ is a transaction. A transaction $T_i = (TID, X)$, which is a tuple including transaction identifier *TID* and an itemset X. The itemset $X = \{x_1, x_2, \ldots, x_k\}$ is a set of k items in I, $x_j \in I$ $(1 \leq j \leq k)$. A pattern containing k items is called k-pattern. In this paper, we use the terms "itemset" and "pattern" interchangeably. For example, Table 1 contains 6 transactions and 8 items $\{a, b, c, d, e, f, g, h\}$ with an assumption that items are sorted by the lexicographical order in a transaction.

Table 1. Transaction database D

TID	**Item**
1	$a,\ c,\ d,\ f$
2	$a,\ c,\ e,\ f,\ g$
3	$a,\ b,\ c,\ f,\ h$
4	b, f, g
5	b, c
6	f, g

Definition 3.1. *(Support of a pattern). The support of a pattern X in the transaction database D, denoted as $\sigma(X)$, is the number of transactions containing X in D, that is $\sigma(X) = |\{T_k \in D | X \subseteq T_k\}|$. The support of a pattern can also be represented in percentage of $|D|$.*

Definition 3.2. *(Frequent pattern with a single minimum support threshold). The pattern X is frequent if its support is no less than a user-defined minimum support threshold value (denoted by minsup), i.e., $\sigma(X) \geq minsup$.*

Definition 3.3. *(Multiple Item Support – MIS). Given an itemset $X = \{x_1, x_2, \ldots, x_k\}$, the minimum item support value of X is defined by the smallest MIS value of the items in X, that is $MIS(X) = \min(MIS(x_1), MIS(x_2), \ldots, MIS(x_k))$.*

For example, given an itemset $X = \{c, d, e\}$ and the *MIS* value of each item in Table 2, then the $MIS(X) = \min(MIS(c), MIS(d), MIS(e)) = \min(2, 1, 1) = 1$.

Table 2. *MIS* and support of each item in D

Item	a	b	c	d	e	f	g	h
MIS	1	2	3	3	2	3	2	2
$\boldsymbol{\sigma}$	3	3	4	1	1	5	3	1

Definition 3.4. *(Frequent pattern with multiple minimum support thresholds). An itemset X is a frequent pattern if and only if its support is no less than MIS(X), i.e., $\sigma(X) \geq MIS(X)$.*

Definition 3.5. *(MIS value of an item). The MIS value of each item is calculated by the formula: $MIS(x_k) = \max(\beta \times \sigma(x_k), LS)$, where: $\sigma(x_k)$ is the support of item x_k, $\beta \in [0, 1]$ and LS (Least Support) is a parameter determined by the user.*

4. Mining Frequent Patterns with Multiple Minimum Support Thresholds.

4.1. Theorems and properties.

Theorem 4.1. [14] *Let $X = \{x_1, x_2, \ldots, x_k\}$ be a frequent pattern, then $\sigma(x_i) \geq MIS(x_i)$, $\forall x_i \in X$, i.e., item x_i is a frequent item.*

Property 4.1. [14] *Let $X = \{x_1, x_2, \ldots, x_k\}$ be a pattern and a subset $Y = \{y_1, y_2, \ldots, y_h\} \subseteq X$, then $\sigma(X) \leq \sigma(Y)$.*

Property 4.2. [14] *Let $X = \{x_1, x_2, \ldots, x_k\}$, $Y = \{y_1, y_2, \ldots, y_h\} \subset X$ be frequent patterns, if $\sigma(X) < MIS(X)$, then $\sigma(Y) < MIS(X)$.*

Theorem 4.2. [14] *Let $X = \{x_1, x_2, \ldots, x_k\}$ be a frequent pattern and a subset $Y = \{y_1, y_2, \ldots, y_h\} \subseteq X$. If $\exists x_i \in Y$, $Y \subseteq X$ then $\sigma(Y) \geq MIS(Y)$, where $x_i \in X$ is an item and $MIS(x_i) = \min(MIS(x_1), MIS(x_2), \ldots, MIS(x_k))$.*

Lemma 4.1. *Let $X = \{x_1, x_2, \ldots, x_k\}$ be a pattern. If $\sigma(X) < MIS(X)$ then x_i is not a frequent pattern, where $x_i \in X$ is an item and $MIS(x_i) = \min(MIS(x_1), MIS(x_2), \ldots, MIS(x_k))$.*

Proof: Assume that pattern $X = \{x_1, x_2, \ldots, x_k\}$ has the support $\sigma(X) < MIS(X) = \min(MIS(x_1), MIS(x_2), \ldots, MIS(x_k)) = MIS(x_i)$, because $\sigma(X) < MIS(x_i)$ so X is not a frequent pattern; therefore, $x_i \in X$ is not a frequent pattern.

For example, pattern $X = \{e, f\}$ in Table 1 is an infrequent pattern because $MIS(X) = \min(MIS(e), MIS(f)) = \min(2, 3) = 2 > \sigma(X) = 1$. Item $e \in X$ is also infrequent, since $\sigma(e) = 1 < MIS(e)$.

Lemma 4.2. *Let $X = \{x_1, x_2, \ldots, x_k\}$ be a pattern and a subset $Y = \{y_1, y_2, \ldots, y_h\} \subseteq X$. If $\sigma(X) \geq MIS(X)$ then Y may not be a frequent pattern.*

For example, an itemset $X = \{a, c, e\}$ in Table 1 is frequent pattern because $MIS(X) = \min(MIS(a), MIS(c), MIS(e)) = \min(1, 3, 2) = 1 \geq \sigma(X) = 1$, in which the support threshold of each item is $MIS(a) = 1$, $MIS(c) = 3$ and $MIS(e) = 2$. A subset $Y = \{c, e\} \subseteq X$ is not a frequent pattern since the $MIS(Y) = \min(MIS(c), MIS(e)) = \min(3, 2) = 2 \geq \sigma(Y) = 1$, so Y is not a frequent pattern.

Because the well-known *apriori property* is no longer true with multiple minimum support thresholds problem, a concept called *sorted closure property* was proposed in [11], which assumes that all items within an itemset are sorted in increasing order of their minimum supports.

Property 4.3. *(Sorted closure property).* [11] *If a sorted k-itemset $X = \{x_1, x_2, \ldots, x_k\}$, where $k \geq 2$ and $MIS(x_1) \leq MIS(x_2) \leq \cdots \leq MIS(x_k)$, is a frequent pattern, then all of its sorted subsets with $(k-1)$ items are frequent.*

The *sorted closure property* can reduce the number of candidate patterns because a different order of items will generate a different number of candidate patterns. Hence, the items in I are sorted in ascending order according to their *MIS* value. The scanning times of the database will be lessened when the number of candidate patterns is small; therefore, it can save much time.

4.2. Parallel computing.

Parallel computing is an information processing process that emphasizes the fact that multiple units of data are processed simultaneously by one or more processors. Two types of parallel are data parallel and task parallel. Data parallel is a mechanism partitioning data into multiple parts and using many processing units to

perform the same operation. Task parallel is the simultaneous execution of many different tasks running on the same data.

There are many parallel programming models, such as shared memory models (OpenMP), Message Passing Interface and MapReduce. These models provide powerful tools and are widely used in parallel processing. Currently, multi-core processor architectures allow multiple tasks performing at the same time. Programmers just need to focus on parallel programming models to speed up the processing and optimization of system resources to reduce the memory cost.

In addition, task parallel is an advantage of multi-core architecture over multi-threads because (i) tasks require less memory than threads; (ii) a thread runs on only one core, whereas a task can run on multiple cores; (iii) threads require more processing time than tasks because the operating system needs to allocate data structures for threads, such as initialization and destruction, and must perform context switching between threads.

Multi-core has been applied in many data mining fields such as mining frequent sequences [26], mining frequent closed sequences [27], subgraph mining [28], PGP-mc (Parallel Gradual Pattern-multiple core): an approach for parallel gradual pattern extraction [29], GapMis-OMP: a tool for pairwise short-read alignment [30], SW (Smith-Waterman): a method based on comparing sequence lengths [31].

4.3. **The proposed algorithm.** In this section, we describe the proposed method, named MMS-FPM (Multiple Minimum Support - Frequent Pattern Mining) for fast mining the complete set of frequent patterns with multiple minimum support thresholds using MIS-tree. All items are sorted in ascending order by their support value instead of the *MIS* value, shown in Table 3(a). Their information is stored in the *Sup-list*, which is a list with three fields: item name, support and the *MIS* value of the item. In addition, to decrease the search space, we perform early pruning on the itemsets that cannot generate any frequent patterns. Starting from the first item in the *Sup-list*, the items that have support value less than *MIS* value will be pruned. Next, we compare the support value of the remaining items in the *Sup-list* with the least support *LS* value, the items having support value less than *LS* value are also pruned. For example: in Table 3(a), item e has $\sigma(e) = 1 < MIS(e)$ and the same applies to items h and d, these items cannot generate any frequent patterns at higher-order; hence, it will be pruned as shown in Table 3(b).

TABLE 3. Items' supports are sorted in ascending order (a), pruned the items have support less than *MIS* value (b)

Item	e	h	d	b	a	g	c	f
MIS	2	2	3	2	1	2	3	3
$\boldsymbol{\sigma}$	1	1	1	3	3	3	4	5

(a)

Item	b	a	g	c	f
MIS	2	1	2	3	3
$\boldsymbol{\sigma}$	3	3	3	4	5

(b)

The MMS-FPM pseudo code can be described as Figure 1.

First, the MMS-FPM algorithm determines the support and *MIS* value for each item (line 3) and identifies the set of frequent 1-patterns F_1 (line 4), which is an initial collection of items that can be extended. To reduce the search space and more effectively balance the workload in the system, the F_1 is sorted in ascending order according to their support value instead of the *MIS* value (line 5) and each item in F_1 is added to MIS-tree as a child node (line 6). Next, the MMS-FPM creates new tasks corresponding to each pattern in F_1 (line 9). Each task is assigned to a processor core p_i to execute the procedure FPM-Extension (line 10) to extend all the patterns. If all processor cores are busy, new tasks

```
Input: Database D, β and LS parameters
Output: The full set of frequent patterns
1. Begin
2.     Initial root = ∅
3.     Compute(D, β, LS)
4.     F1 = {f | f ∈ F1, σ(f) ≥ MIS(f)}
5.     Sort all items in F1 in ascending order by their support value
6.     root.add-child(f ∈ F1)
7.     for each pattern x in F1
8.           create new task ti
9.           FPM-Extension(x, β, LS)
10.    end for
11. End
Procedure FPM-Extension(F, β, LS)
12. Begin
13.    for each pattern Si in F
14.       Ri = ∅
15.       for each pattern Sj in F, where i > j
16.          if (Si is not pruned)
17.            Sij = Join-pattern(Si, Sj)
18.            if (Sij is not pruned && σ(Sij) ≥ MIS(Sij))
19.              Ri = Ri∪ pat[Sij]
20.            end if
21.          end if
22.       end for
23.       FPM-Extension(Ri, β, LS)
24.    end for
25. End
```

FIGURE 1. The MMS-FPM strategy

will be put into a task queue. In contrast, when a processor core p_i is idle and the queue is not empty, it will push a task from task queue and handle the task, this process is repeated until the task queue is empty. Tasks run in parallel to generate a partial set of FPMs and the final set of FPMs is the union of all the partial results.

In the FPM-Extension procedure, all pattern extensions of an equivalence class F is an empty set (line 14), denoted by R_i. Before extending new patterns, the algorithm performs the check function and eliminates prefixes that cannot extend frequent patterns based on Theorem 4.2 and Lemma 4.1 (line 16). Each node is extended by calling Join-Patterns (line 17) to create a new pattern S_{ij}. If S_{ij} patterns do not satisfy the *MIS* value, it can be immediately discarded. Otherwise, they are then put into equivalence class R_i (line 19) and this process is repeated for the new equivalence class R_i (line 23).

For example, the set of frequent patterns for database D in Table 1, with $\beta = 0.2$ and $LS = 0.2$, is shown in Figure 2.

5. **Experimental Results.** This section compares the mining time of MSApriori and MMS-FPM, to show the effectiveness of the proposed method. The experiments were performed on a computer with an Intel Core i7 - 4770HQ (6M Cache, 2.2GHz, 8 cores), 16GB main memory and using .NET Framework 4.5 for implementation.

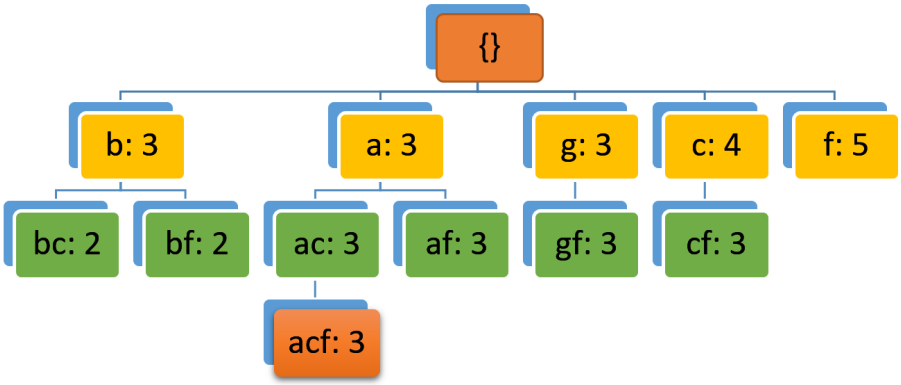
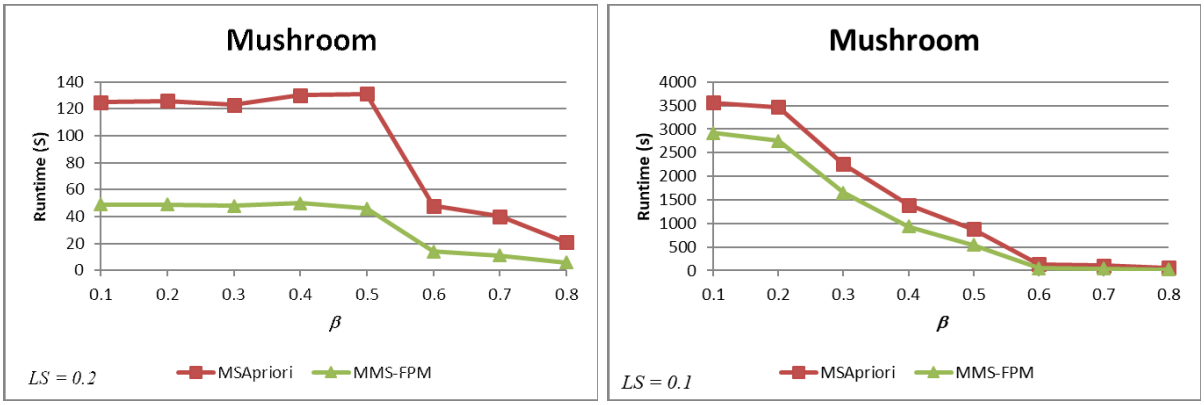


FIGURE 2. The set of frequent patterns for database D in Table 1 with $\beta = 0.2$ and $LS = 0.2$

TABLE 4. Databases used in the experiments

Database	#transaction	#items	#Avg. Items	Source
Mushroom	8,142	119	23	UCI repository
C20D10K	10,000	385	20	Synthetic database
BMS-POS	515,597	1657	6.5	KDD Cup 2000

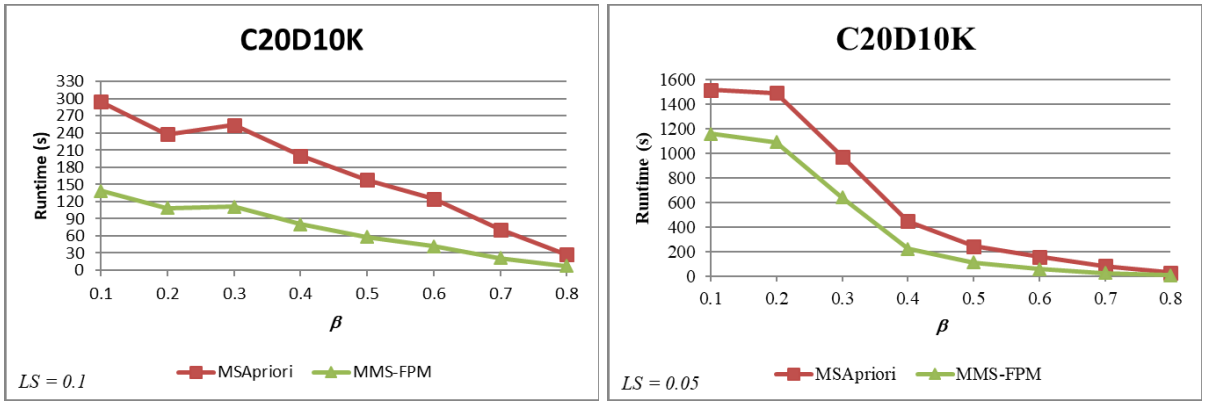


(a) (b)

FIGURE 3. Running time of MMS-FPM and MSApriori for Mushroom database with varying β and (a) $LS = 0.2$; (b) $LS = 0.1$

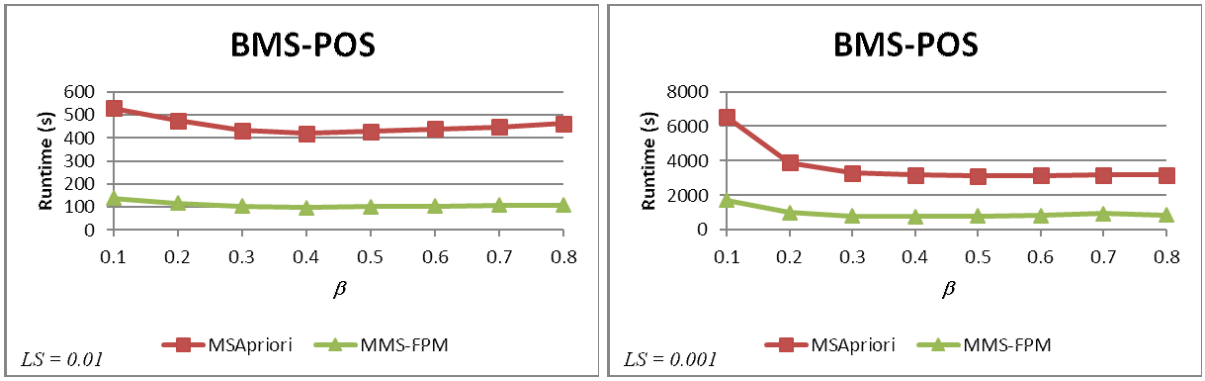
Experiments were carried out on two synthetic databases and a real-life database having varied characteristics and representing different types of data. Information about the databases is shown in Table 4. The definitions of the parameters used to generate the synthetic databases are as follows: C was the average number of item per transaction, and D was the number of transactions. The experiments of both algorithms run on each database while the LS parameter is varied and β is changed from 0.1 to 0.8.

In Figure 3, we compare the runtime of MMS-FPM to MSApriori for the Mushroom database with various settings. Figure 3(a) fixed $LS = 0.2$ and changes β from 0.1 to 0.8, and we easily found that the runtime of MMS-FPM is less than that of MSApriori. In a similar way, Figure 3(b) fixed $LS = 0.1$, and MMS-FPM is much better than MSApriori for this database.



(a) (b)

FIGURE 4. Running time of MMS-FPM and MSApriori for C20D10K database with varying β and (a) $LS = 0.1$; (b) $LS = 0.05$



(a) (b)

FIGURE 5. Running time of MMS-FPM and MSApriori for BMS-POS database with varying β and (a) $LS = 0.01$; (b) $LS = 0.001$

Figure 4 performs the same for the C20D10K database, Figure 4(a) fixed $LS = 0.1$ and Figure 4(b) fixed $LS = 0.05$, and we easily found that the runtime of MMS-FPM is much better than MSApriori.

In Figure 5, we compare the runtime of MMS-FPM and MSApriori for the BMS-POS database with various settings. Figure 5(a) fixed $LS = 0.01$ and Figure 5(b) fixed $LS = 0.001$, MMS-FPM is always better for BMS-POS database, and the runtime of MMS-FPM is less than nearly three as that of MSApriori. Especially, when we decrease β, and the runtime of MSApriori has significantly increased while that of MMS-FPM has slightly increased. In general, MMS-FPM outperforms MSApriori for this database.

The gap between the memory usage of MMS-FPM and MSApriori is insignificant, the memory usages of both algorithms are nearly the same. Sometimes, the memory usage of MMS-FPM is slightly larger than that of MSApriori (although it is quite small), and this can be explained as follows: MMS-FPM requires more memory usage because parallel processing divides the tasks to be processed into independent branches, and thus needs more memory to store the results. When LS and β parameters were decreased, more patterns were obtained, and thus the runtime and memory usage increased. Besides the execution time of MMS-FPM method is better than MSApriori, the proposed method is

also improved memory usage, and the memory size is not explosive in mining processing, especially on large databases.

6. **Conclusions.** Mining frequent patterns with multiple minimum support thresholds is an important problem because the items in the database do not often have the same characteristics. In this paper, we proposed the MMS-FPM algorithm using multiple core technique to solve the rare item problem. We have compared MMS-FPM with MSApriori in runtime details to evaluate the effectiveness of the proposed method. Additionally, MMS-FPM applied an early pruning mechanism to eliminating infrequent candidates generated to reduce the search space. The experimental results have shown that MMS-FPM was much more efficient than MSApriori.

As a part of future work, we will extend this work by conducting extensive experiments by considering different types of databases. Due to the effectiveness of N-list structure, we are also planning to re-implement CFP-growth++ algorithm based on this structure to mine FPM with multiple minimum support thresholds and to compare it with MMS-FPM and CFP-Growth++.

REFERENCES

[1] R. Agrawal, T. Imielinski and A. N. Swami, Mining association rules between sets of items in large databases, *SIGMOD Conference*, pp.207-216, 1993.

[2] F. O. Isinkaye, Y. O. Folajimi and B. A. Ojokoh, Recommendation systems: Principles, methods and evaluation, *Egyptian Informatics Journal*, vol.16, no.3, pp.261-273, 2015.

[3] M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita, Towards generating real-life datasets for network intrusion detection, *I. J. Network Security*, vol.17, no.6, pp.683-701, 2015.

[4] C. Schönbach, C. Verma, P. J. Bond and S. Ranganathan, Bioinformatics and systems biology research update from the 15th International Conference on Bioinformatics (InCoB2016), *BMC Bioinformatics*, vol.17, no.S-19, pp.87-90, 2016.

[5] F. Höppner, Association rules, *Data Mining and Knowledge Discovery Handbook*, pp.299-319, 2010.

[6] R. Agrawal and R. Srikant, Fast algorithms for mining association rules in large databases, *Proc. of the 20th International Conference on Very Large Data Bases*, pp.487-499, 1994.

[7] J. Han, J. Pei, Y. Yin and R. Mao, Mining frequent patterns without candidate generation: A frequent-pattern tree approach, *Data Min. Knowl. Discov.*, vol.8, no.1, pp.53-87, 2004.

[8] M. Zaki, Scalable algorithms for association mining, *IEEE Trans. Knowl. Data Eng.*, vol.12, no.3, pp.372-390, 2000.

[9] Z. H. Deng, Z. Wang and J. J. Jiang, A new algorithm for fast mining frequent itemsets using N-lists, *Science China Information Sciences*, vol.55, no.9, pp.2008-2030, 2012.

[10] Z.-H. Deng and S.-L. Lv, PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets via Children-Parent Equivalence pruning, *Expert Syst. Appl.*, vol.42, no.13, pp.5424-5432, 2015.

[11] B. Liu, W. Hsu and Y. Ma, Mining association rules with multiple minimum supports, *Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.337-341, 1999.

[12] T. Xu and X. Dong, Mining frequent patterns with multiple minimum supports using basic Apriori, *The 9th International Conference on Natural Computation (ICNC)*, pp.957-961, 2013.

[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*, vol.42, no.1, pp.1-24, 2006.

[14] R. U. Kiran and P. K. Reddy, Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms, *Proc. of the 14th International Conference on Extending Database Technology*, pp.11-20, 2011.

[15] L. T. T. Nguyen, B. Vo, L. T. T. Nguyen, P. Fournier-Viger and A. Selamat, ETARM: An efficient top-k association rule mining algorithm, *Appl. Intell.*, vol.48, no.5, pp.1148-1160, 2018.

[16] I. G. Czibula, G. Czibula and D. L. Miholca, Enhancing relational association rules with gradualness, *International Journal of Innovative Computing, Information and Control*, vol.13, no.1, pp.289-305, 2017.

[17] P. Fournier-Viger, A. Gomariz, M. Campos and R. Thomas, Fast vertical mining of sequential patterns using co-occurrence information, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, no.1, pp.40-52, 2014.
[18] H.-F. Li, C.-C. Ho, H.-S. Chen and S.-Y. Lee, A single-scan algorithm for mining sequential patterns from data streams, *International Journal of Innovative Computing, Information and Control*, vol.8, no.3(A), pp.1799-1820, 2012.
[19] T.-L. Nguyen, B. Vo, B. Huynh, V. Snásel and L. T. T. Nguyen, Constraint-based method for mining colossal patterns in high dimensional databases, *International Conference on Information Systems Architecture and Technology*, no.1, pp.195-204, 2017.
[20] T. Mai, B. Vo and L. T. T. Nguyen, A lattice-based approach for mining high utility association rules, *Inf. Sci.*, vol.339, pp.81-97, 2017.
[21] J. Ren, J. Yan, R. Gao, J. Wang and H. He, Mining concise representations of high utility itemsets with negative utilities from software executing traces, *International Journal of Innovative Computing, Information and Control*, vol.12, no.4, pp.1115-1128, 2016.
[22] M. Adda, L. Wu and Y. Feng, Rare itemset mining, *The 6th International Conference on Machine Learning and Applications*, pp.73-80, 2007.
[23] L. Szathmary, A. Napoli and P. Valtchev, Towards rare itemset mining, *Proc. of the 19th IEEE International Conference on Tools with Artificial Intelligence*, vol.1, pp.305-312, 2007.
[24] Y. S. Koh and N. Rountree, Finding sporadic rules using apriori-inverse, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp.97-106, 2005.
[25] T. Huang, Discovery of fuzzy quantitative sequential patterns with multiple minimum supports and adjustable membership functions, *Inf. Sci.*, vol.222, pp.126-146, 2013.
[26] B. Huynh, B. Vo and V. Snásel, An efficient method for mining frequent sequential patterns using multi-core processors, *Appl. Intell.*, vol.46, no.3, pp.703-716, 2017.
[27] B. Huynh, B. Vo and V. Snásel, An efficient parallel method for mining frequent closed sequential patterns, *IEEE Access*, vol.5, pp.17392-17402, 2017.
[28] B. Huynh, D. Nguyen and B. Vo, Parallel frequent subgraph mining on multi-core processor systems, *ICIC Express Letters*, vol.10, no.9, pp.2105-2113, 2016.
[29] A. Laurent, B. Négrevergne, N. Sicard and A. Termier, Efficient parallel mining of gradual patterns on multicore processors, *Advances in Knowledge Discovery and Management*, pp.137-151, 2010.
[30] T. Flouri, C. S. Iliopoulos, K. Park and S. P. Pissis, GapMis-OMP: Pairwise short-read alignment on multi-core architectures, *IFIP International Conference on Artificial Intelligence Applications and Innovations*, no.2, pp.593-601, 2012.
[31] F. Sánchez, F. Cabarcas, A. Ramírez and M. Valero, Long DNA sequence comparison on multicore architectures, *European Conference on Parallel Processing*, no.2, pp.247-259, 2010.