Hindawi
Complexity
Volume 2018, Article ID 8487641, 9 pages
https://doi.org/10.1155/2018/8487641

Research Article

An Efficient Method for Mining Erasable Itemsets Using Multicore Processor Platform

Bao Huynh[1,2] **and Bay Vo**[3]

[1]*Division of Data Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam*
[2]*Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam*
[3]*Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Vietnam*

Correspondence should be addressed to Bay Vo; vd.bay@hutech.edu.vn

Received 19 April 2018; Revised 10 July 2018; Accepted 30 July 2018; Published 22 October 2018

Academic Editor: Ireneusz Czarnowski

Mining erasable itemset (EI) is an attracting field in frequent pattern mining, a wide tool used in decision support systems, which was proposed to analyze and resolve economic problem. Many approaches have been proposed recently, but the complexity of the problem is high which leads to time-consuming and requires large system resources. Therefore, this study proposes an effective method for mining EIs based on multicore processors (pMEI) to improve the performance of system in aspect of execution time to achieve the better user experiences. This method also solves some limitations of parallel computing approaches in communication, data transfers, and synchronization. A dynamic mechanism is also used to resolve the load balancing issue among processors. We compared the execution time and memory usage of pMEI to other methods for mining EIs to prove the effectiveness of the proposed algorithm. The experiments show that pMEI is better than MEI in the execution time while the memory usage of both methods is the same.

1. Introduction

Data mining is an interesting field that has attracted many experts because of the huge amounts of data that were collected every day and the need to transfer such data into useful information to use in intelligence systems such as recommendation systems, decision making, and expert systems. Data mining has been widely used in market basket analysis, manufacturing engineering, financial banking, bioinformatics and future healthcare, and so on. The mining frequent pattern (FP) has a vital position in many data mining fields including association rule mining [1], clustering [2], and text mining [3]. Mining FP is to find all patterns that have the frequency satisfying the user-given threshold. There are many methods [4, 5] for mining FPs in recent years. In addition, some issues related to FP mining has been proposed such as maximal frequent patterns [6], top-k cooccurrence items with sequential pattern [7], weighted-based patterns [8], periodic-frequent patterns [9], and their applications [10, 11].

In 2009, the erasable itemset mining was first introduced [12], which comes from predicting merchandises of the production scheming as an exciting alteration of pattern mining. For an example, a factory needs to produce several new products, each of which requires some amount of raw materials to produce. However, the factory does not have enough budget to purchase all materials. Therefore, the factory managers need to determine what essential materials are needed to produce while the profit is not affected. The main problem is how to efficiently find these materials, without which the loss of the profit is less than the given threshold. These elements are also called as erasable itemsets. Based on these erasable itemsets, the consulting team can give the managers several suggestions about production plans in the near future. It has attracted a lot of research and become an ideal topic in recent years. There are many approaches, which were summarized in [13], to mine such patterns including META [12], MERIT [14], MEI [15], and EIFDD [16]. Several related problems of mining erasable closed itemsets [17], top-rank-k

erasable itemsets [18], erasable itemsets with constraints [19], and weighted erasable itemsets [20, 21] have also been developed. Erasable closed itemset [17], a condensed representation of EIs without information loss, was proposed to reduce the computational cost. Top-rank-k erasable itemset [18] is to merge the mining and ranking phases into one phase that only returns a small number of EIs to use in intelligent systems. Erasable itemset with constraints [19] is another approach only producing a small number of EIs, which delight a special requirement. In addition, mining weighted erasable itemset [20, 21] is a framework for mining erasable itemsets with the weight conditions for each item.

The existing algorithms for mining EIs have high computational complexity. It leads to very long execution time, especially on huge datasets and inefficiently intelligent systems. Therefore, this study proposes a parallel approach named parallel mining erasable itemsets (pMEI) using a multicore processor platform to improve the execution time for mining EIs. The major contributions of this paper are as follows:

(i) A parallel method, namely, pMEI for mining EIs, splits the jobs into several duties to lessen the operating cost.

(ii) Applying the difference pidset (dPidset) structure for quickly determining EIs information.

(iii) A dynamic mechanism is used for load balancing the workload among cores when some processes are free.

The experiment results show that pMEI algorithm is better than MEI in execution time for the most of experimental datasets.

The remaining parts of the paper are arranged as follows: the preliminaries and related works, comprising the erasable itemset mining problem, several methods for mining EIs, as well as multicore processor architecture are presented in Section 2. Section 3 presents the pMEI algorithm proposed. The experiments on runtime and memory usage of pMEI and MEI methods for mining EIs are shown in Section 4. Finally, Section 5 reviews the outcomes and proposes some potential study topics.

2. Related Works

2.1. Preliminaries. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n distinct items, which are the conceptual descriptions of elements of products created in a manufactory. For example, assuming that $I = \{\text{sugar, milk, cheese, cafe, snack, and wine}\}$, then the included items are sugar, milk, cheese, cafe, snack, and wine. A product dataset D consists of m products, $\{P_1, P_2, \ldots, P_m\}$, where $P_k (1 \le k \le m)$ is a product shown in the pair of (items, value). In this pair, items are the items that compose product P_k, and value is the profit value of the product P_k. Table 1 shows the example datasets that have five items including $\{a, b, c, d, e, f\}$ and the set of products $\{P_1, P_2, \ldots, P_{10}\}$.

Table 1: An example of product dataset D.

Product	Items	Value (USD)
P_1	a, b, c	3100
P_2	a, b	1500
P_3	a, c	1200
P_4	b, c, e	250
P_5	b, e	150
P_6	c, e	110
P_7	c, d, e, f	250
P_8	d, e, f	100
P_9	d, f	350
P_{10}	b, f	50

Definition 1 (itemset). An itemset $Y = (i_1, i_2, \ldots, i_u)$, such that $i_k \in I (1 \le k \le u)$, is called a k-itemset.

Definition 2 (profit of itemset). Given an itemset $Y \subseteq I$, the profit of itemset Y (denoted $\text{pro}(Y)$) is calculated by

$$\text{pro}(Y) = \sum_{\{P_k \in D | Y \cap P_k \text{ items} \neq \emptyset\}} P_k \text{ value}. \quad (1)$$

For example, let $Y = \{ad\}$ the set of products that contains $\{a\}$, $\{d\}$, or $\{ad\}$ from Table 1 which are $\{P_1, P_2, P_3, P_7, P_8, P_9\}$. Therefore, $\text{pro}(Y) = P_1$ value + P_2 value + P_3 value + P_7 value + P_8 value + P_9 value = 6500 USD.

An itemset Y in product dataset D is said to be erasable iff $\text{pro}(Y) \le \theta \times \delta$, where δ is the minimum threshold defined by user; θ is the total value of D determined by this formula $\theta = \sum_{P_k \in D} P_k$ value.

Definition 3 (EI mining problem). The EI mining problem is to discover all itemsets (Y) that have $\text{pro}(Y)$ not greater than $\theta \times \delta$.

For the example dataset in Table 1 and $\delta = 20\%$, we have $S = 7060$ USD by summing the value of all products. The itemset d is an erasable itemset due to $\text{pro}(d) = 700 \le 7060 \times 20\% = 1412$.

2.2. Erasable Itemset Mining. The erasable itemset (EI) mining problem was first introduced in 2009 [12], which comes from predicting merchandises of the production industry. It supports managers to decide their manufacturing strategy to guarantee the development of the company. The managers can decide which new merchandises are suitable for the factory without affecting the company's profit. For example, a company that makes many types of goods, each good that is created will have a profit value. To create all products, the factory has to buy all essential materials. Currently, the company has no enough budget to purchase all materials. Hence, the managers of this company should deliberate their manufacturing strategies to make sure the

steadiness of the company. The challenge is to obtain the itemsets that can be excluded but do not significantly change the company's profit.

2.3. Methods for Erasable Itemset Mining. Many methods have been suggested to mine EIs such as META [12], MERIT [14], MEI [15], and EIFDD [16]. Firstly, META, an Apriori-based algorithm, generates candidate itemsets using level-wise approach. Let S be the set of erasable $k-1$-itemsets. An itemset $Y \in S$ is verified with $Z \in S \wedge Y \neq Z$ for coherency to produce applicant erasable k-itemsets. However, only a small number of $Z \in S \wedge Z \neq Y$ that have the same prefix as Y are joined. Later, MERIT based on the NC_sets is structured to decrease memory manipulation, which is its main improvement. The performance of MERIT is better than that of META significantly. However, storing NC_sets structure leads to high computational cost including memory usage and runtime. Therefore, MEI uses a divide-and-conquer approach associated with the difference of pidsets (dPidset) for mining EIs to improve the memory usage and runtime. It only scans the dataset one time to determine the total profit (S), the index of gain (pro), and the erasable 1-itemsets with their pidsets. Although the runtime and memory consumption are enhancing than those of META and MERIT, however, the MEI's performance from several dense datasets is quite weak. To resolve this drawback of MEI for dense datasets, EIFDD is proposed by using the subsume theory. This concept helps to quickly determine the information of EIs, without the generation cost. In brief, EIFDD is regularly applied to mine EIs for dense datasets, while MEI is applied to mine EIs for the other kinds of datasets.

Although existing methods improved the computation of mining EIs, however, these methods still consume more time in large datasets or large thresholds. Hence, in this paper, we develop a new technique to improve the computational cost for mining EIs.

2.4. dPidset Structure. The dPidset structure was suggested by Le and Vo [15] to lessen memory consumption by using an index of profit for effectively mining EIs. This structure is summarized as follows.

Definition 4. Given an itemset Y and an item $X \subseteq Y$, the pidset of itemset Y is denoted as

$$p(Y) = \bigcup_{X \in Y} p(X), \tag{2}$$

where $p(X)$ is the pidset of item X, that is, the set of product identifiers (IDs) which contain item X.

Definition 5. Let $\text{pro}(Y)$ be the profit of itemset Y that is computed as follows:

$$\text{pro}(Y) = \sum_{P_k \in p(Y)} \text{value}(P_k). \tag{3}$$

Theorem 1 [15]. *Two given itemsets YA and YB have the same prefix which is Y. $p(YA)$ and $p(YB)$ are pidsets of YA and YB, correspondingly. $p(YAB)$ is determined as follows:*

$$p(YAB) = p(YB) \cup p(YA). \tag{4}$$

Example 1. Consider the illustration datasets in Table 1, $p(de) = \{4, 5, 6, 7, 8\}$ and $p(df) = \{7, 8, 9, 10\}$. We have $p(def) = p(dfe) = p(de) \cup p(df) = \{4, 5, 6, 7, 8, 9, 10\}$.

Definition 6. The dPidset of itemset YAB signified by $dP(YAB)$ is defined as follows:

$$dP(YAB) = p(YB) - p(YA), \tag{5}$$

where $p(YB) - p(YA)$ is the list of itemset IDs that only exist in $p(YB)$.

Example 2. We have $p(de) = \{4, 5, 6, 7, 8\}$ and $p(df) = \{7, 8, 9, 10\}$, so $dP(def) = p(df) - p(de) = \{9, 10\}$.

Theorem 2 [15]. *Two given itemsets YA and YB have the dPidsets which are $dP(YA)$ and $dP(YB)$. The $dP(YAB)$ is determined as follows:*

$$dP(YAB) = dP(YB) - dP(YA). \tag{6}$$

Example 3. We have $p(d) = \{7, 8, 9\}$, $p(e) = \{4, 5, 6, 7\}$, and $p(f) = \{7, 8, 9, 10\}$. As in Definition 6, $dP(de) = p(e) - p(d) = \{4, 5, 6\}$ and $dP(df) = p(f) - p(d) = \{10\}$. As in Theorem 2, $dP(def) = dP(df) - dP(de) = \{10\}$.

Theorem 3 [15]. *The profit of YAB, denoted by* pro *(YAB), is calculated by YA as follows:*

$$\text{pro}(YAB) = \text{pro}(YA) + \sum_{P_k \in dP(YAB)} \text{value}(P_k), \tag{7}$$

where $\text{pro}(YA)$ *is the profit of Y and* $\text{value}(P_k)$ *is the profit of P_k.*

Example 4. We have $p(d) = \{7, 8, 9\}$ and $p(e) = \{4, 5, 6, 7\}$, and thus, $\text{pro}(e) = 700$, $\text{pro}(e) = 860$, and $\text{pro}(f) = 750$. According to Example 3, $dP(de) = \{4, 5, 6\}$ and $dP(df) = \{10\}$, and thus, $\text{pro}(d) + \sum_{P_k \in dP(de)} \text{value}(P_k) = 1050$ and $\text{pro}(df) = \text{pro}(d) + \sum_{P_k \in dP(df)} \text{value}(P_k) = 1400$. $dP(def) = \{10\}$, so $\text{pro}(def) = \text{pro}(de) + \sum_{P_k \in dP(def)} \text{value}(P_k) = 1400$.

2.5. Multicore Processor Platform. A multicore processor (MCP) is a physical chip including many separate cores in the same circuit [22]. MCPs enable executed multiple missions concurrently to increase the performance of applications. In a multicore processor, each core has a distinct L1 cache and execution module and uses a public L2 cache for the whole processor. This makes the greatest use of the resources and optimizes the communication between inter-cores. If several tasks carry on separate cores of the same circuit and if they portion data that match in the cache, then the public last-level cache between cores will reduce the data

Input: product database D and a minimum given threshold δ
Output: E_{result}, the list of EIs
1 root=NULL
2 Scan D to calculate the whole profit of $D(S)$, the index of profit (pro), and erasable 1-itemsets with their pidsets (E_1)
3 Sort E_1 by the size of their pidsets in descending order
4 $root$ = root $\cup$ E_1, where E_i is a child node
5 For each (k in $root$) do
6 Start new task t_k
7 **pMEI_Ext** (k, t_k)
8 End for

Algorithm 1: pMEI method.

replication. Hence, it is further effective in interaction. The major improvement of multicore processors is decreasing the temperature occurring off CPU and to extensively growing the speedup of processors while it is low cost than distributed systems, so it broadly applied in many fields such as computer vision, social network, image processing, and embedded system.

In data mining, there are many studies using this architecture to enhance the performance. Nguyen et al. [23] implement a technique for parallel mining class association rules on a computer that has the multicore processor platform which uses SCR-tree (single constraint rule-tree) structure and task parallel mechanism on .NET framework. Huynh et al. in [24] utilize multicore processors for mining frequent sequential patterns and frequent closed sequential patterns which use DBV-tree (dynamic bit vector-tree) structure and data parallel strategy based on TPL (task parallel library). Laurent et al. [25] proposed PGP-mc, which uses parallel gradual pattern mining used by the POSIX thread library. Flouri et al. [26] implement GapMis-OMP, a tool for pairwise short read alignment used by the OpenMP application programming interface, and Sánchez et al. [27] propose SW (Smith-Waterman), a method comparing sequence lengths based on the CellBE hardware. Recently, Kan et al. [28] proposed the parallelization and acceleration of the shuffled complex evolution utilizing the multicore CPU and many-core GPU. In 2018, Le et al. [29] suggested a parallel approach for mining intersequence patterns with constraints, which used DBV-PatternList structure and task parallel approach on multicore architecture.

3. A Parallel Method for Erasable Itemset Mining

Our proposed approach (Algorithm 1) in this study first determines the total profit of dataset, the erasable 1-itemsets (E_1) with their pidsets (line 2). Then, pMEI will sort E_1 by the size of their pidsets in not ascending order (line 3) and add them to child node of the root node (line 4). Finally, for each node in root, pMEI will start a new task (line 6) and call pMEI_Ext procedure to process this node with the created task in parallel that is a light-weight object in .NET framework for handling a parallel

Input: node p, the task T
1 **for** i $\leftarrow$ 0 to$|E_v|$
2 $E_{next} \leftarrow \varnothing$
3 **for** j $\leftarrow$ (i + 1) to$|E_v|$ **do**
4 E.Items=E_v [i].Items $\cup$ E_v [j].Items
5 (E.pidset, pro) $\leftarrow$ E_v [i].pidset\E_v [j].pidset
6 E.pro=E_v [i].pro+pro
7 **if** E.pro<$T\times\delta$ **then**
8 $E_{next} \leftarrow E$
9 $E_{result} \leftarrow E$
10 **if** $|E_{next}| > 1$ **then**
11 **pMEI_Ext**(E_{next})

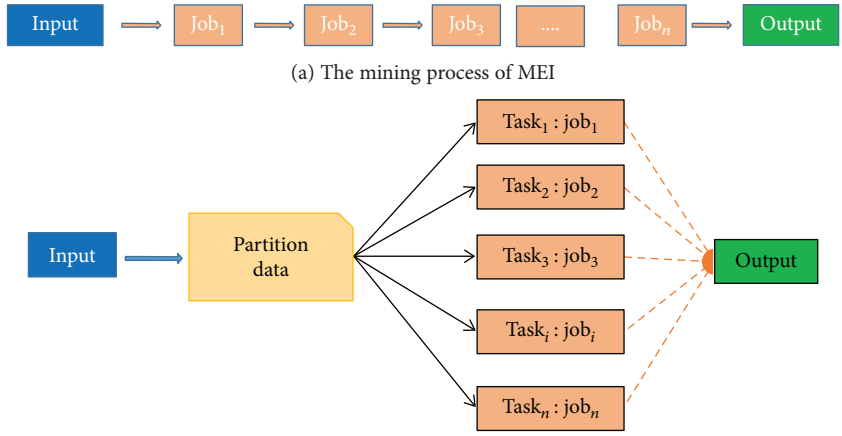
Algorithm 2: Procedure pMEI_Ext.

element of work. It can be used when we would like to perform something in parallel. The works (or jobs) are stretched across multiple processors to maximize performance of computer. Tasks are adjusted for leveraging multicore processors to improve performance.

For procedure pMEI_Ext, the algorithm will combine each node in root together and create the next level of candidates. This strategy will be used until there is no new EI to create. The entire procedure pMEI_Ext task will be performed in parallel to achieve the good performance.

An outstanding point of pMEI algorithm is that it uses a dynamic load balancing mechanism. pMEI uses a queue to store jobs (a list of work to perform), and if the queue is not empty and there exists a task that is idle, then task is assigned a job to execute. In contrast, if the queue is full, the task will perform a job until completion. After a task completes a job or is idle, then it will be immediately assigned a new job and this job will be removed from the queue, and the process continues until the queue is empty. This mechanism is effective and can help avoid both task idling and achieve balanced workloads.

In addition, one of the differences between the pMEI and parallel methods in [24, 29] is in the sorting strategy to balance the search space. For mining frequent patterns, we can sort patterns according to their supports. In mining EIs, we do not need to compute the support of itemsets, so



(a) The mining process of MEI

(b) The mining process of pMEI

Figure 1: Illustration of the mining process of MEI and pMEI.

Table 2: Dataset descriptions.

Dataset	Number of products	Number of items	Average length	Density
Accidents	340183	468	33.8	0.072
Chess	3196	76	35	0.460
Connect	67557	130	43	0.333
Mushroom	8124	120	23	0.192
Pumsb	49046	2113	74	0.035
T10I4D100K	100000	870	10	0.011

pMEI sorts itemsets according to the size of their pidsets to balance the search space.

The illustration of the mining process of MEI and pMEI is shown in Figure 1.

pMEI executed in depth-first search; the result (EIs) is writing to global memory. Thus, it does not need to merge or synthesize process. In addition, pMEI uses a global queue for parent task and local queue for child task; both are accessed in LIFO order. The synchronization between tasks is not necessary because of the task using local queue does not involve any shared data.

4. Experimental Results

This section presents the results of the experiments that were executed on a PC with Intel Core I5-6200U (2.30 GHz, 4 threads) with 4 GB RAM and implemented in C# in Visual Studio 2015.

The experiments have been performed on Accidents, Chess, Connect, Mushroom, Pumsb, and T10I4D100K datasets which were downloaded from UCI datasets (http://fimi.ua.ac.be/data/). We have added a new column to hold the value associated to each product because the value of products has not existed in these datasets. The value was calculated based on the normal distribution $N(100, 50)$. The characteristics of these datasets are exhibited in Table 2 and are accessible at http://sdrv.ms/14eshVm.

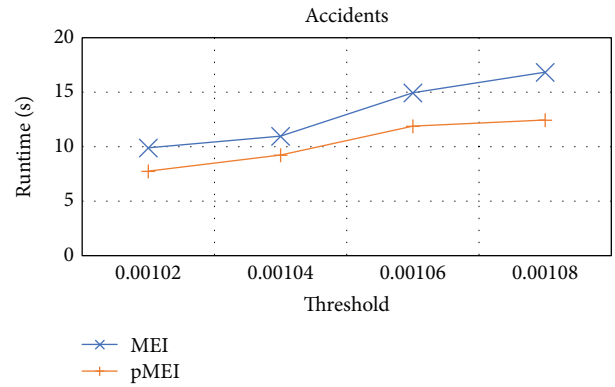


Figure 2: Runtimes of the MEI and pMEI methods on the Accidents.

In this part, we will evaluate the memory manipulation and running time of the suggested algorithm with MEI algorithm [15] to show the effectiveness of pMEI algorithm.

4.1. Running Time. We evaluate the execution times between MEI and pMEI algorithms on six experimental datasets (Figures 2–7). Note that the running times are averaged across five runs.

For dense datasets, such as Chess, Connect, and Mushroom, the execution time of pMEI is much better than MEI (Figures 3–5). In detail, for Chess dataset at $\delta = 30\%$, pMEI

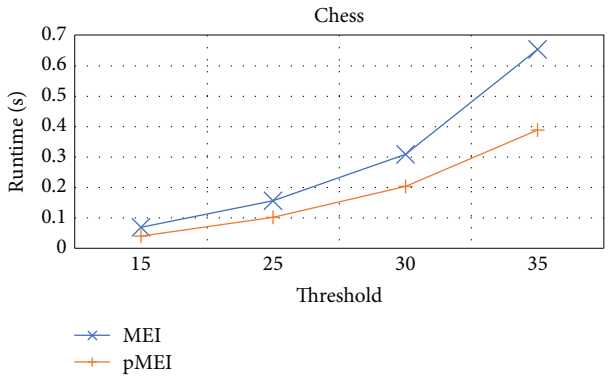


Figure 3: Runtimes of the MEI and pMEI methods on the Chess.

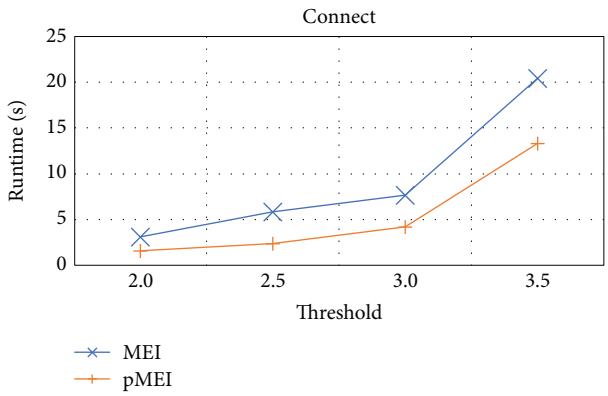


Figure 4: Runtimes of the MEI and pMEI methods on the Connect.

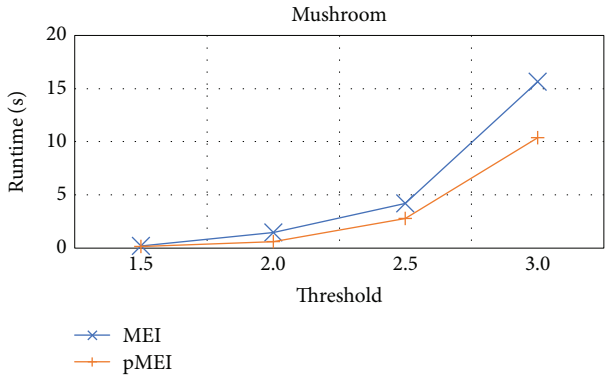


Figure 5: Runtimes of the MEI and pMEI methods on the Mushroom.

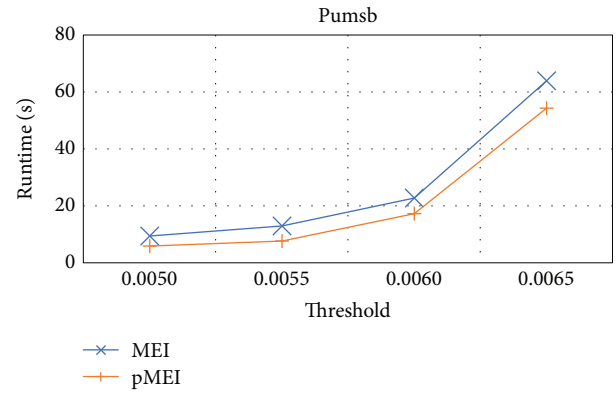


Figure 6: Runtimes of the MEI and pMEI methods on the Pumsb.

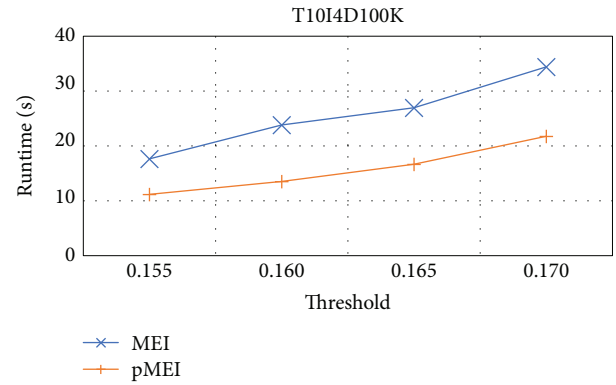


Figure 7: Runtimes of the MEI and pMEI methods on the T10I4D100K.

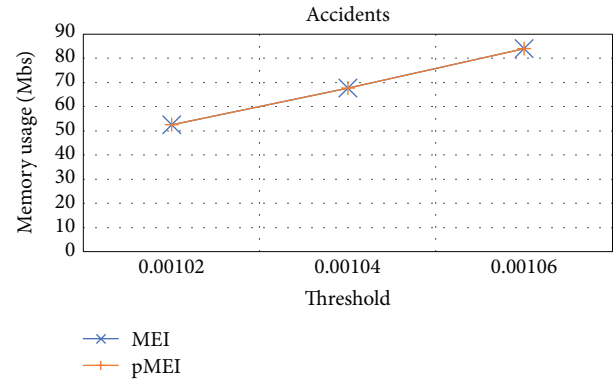


Figure 8: Memory usage of the MEI and pMEI methods on the Accidents.

only requires 0.203 s while MEI requires 0.31 s; and at $\delta =$ 35%, pMEI only requires 0.389 s while MEI requires 0.654 s, respectively. Specifically, this gap will be increased as the threshold increases. Like Connect dataset, at $\delta = 3\%$, the time gap (Δs) between the execution time of pMEI and that of MEI is 3.49 s while at $\delta = 3.5\%$, the time gap is 7.07 s.

For sparse datasets, such as Accidents, Pumsb, and T10I4D100K, the time gaps between pMEI and MEI are small (Figures 2, 6, and 7). Therefore, pMEI outperforms MEI in terms of the execution times for all experimental datasets especially for dense datasets.

4.2. Memory Usage. For all experimental datasets, pMEI and MEI have the same memory usage (see Figures 8–13). In summary, pMEI improves the execution times for mining

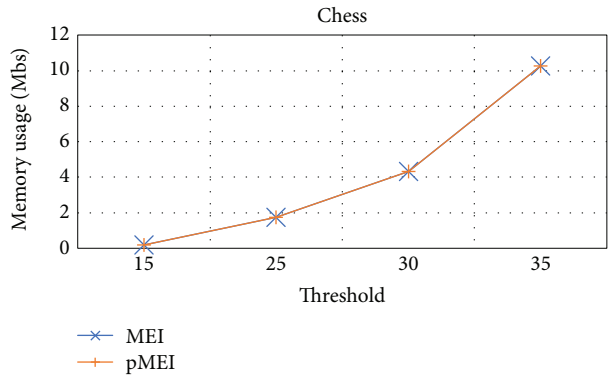


Figure 9: Memory usage of the MEI and pMEI methods on the Chess.

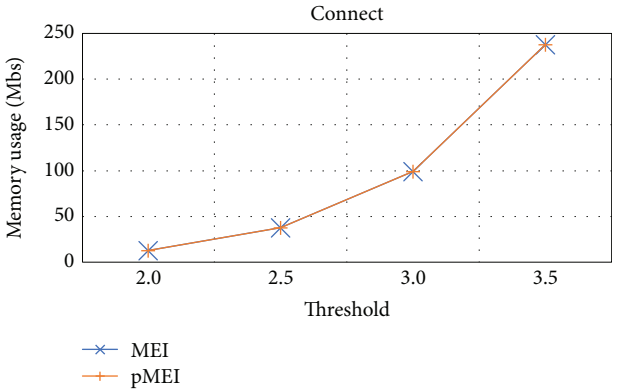


Figure 10: Memory usage of the MEI and pMEI methods on the Connect.

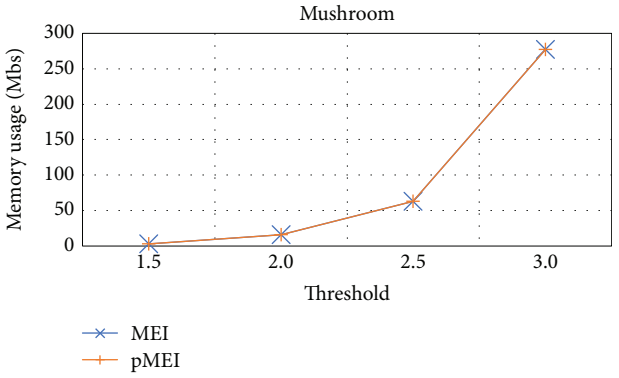


Figure 11: Memory usage of the MEI and pMEI methods on the Mushroom.

EIs on all experimental datasets while keeping the memory usage compared with MEI algorithm.

To evaluate the effectiveness of multicore systems, we executed the proposed method with the various numbers of cores (Figure 14). The speedup increases nearly two times on 2 cores and increases nearly four times on 4 cores. The

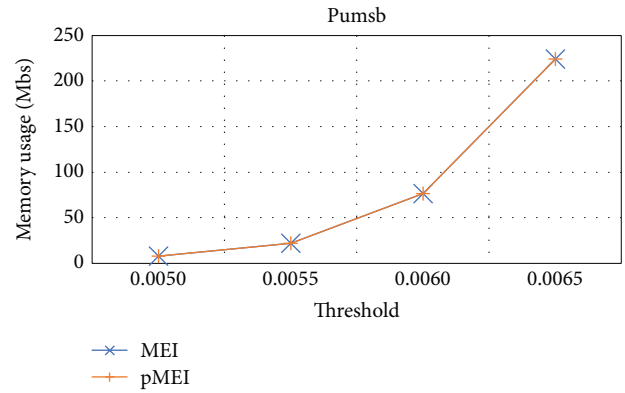


Figure 12: Memory usage of the MEI and pMEI methods on the Pumsb.

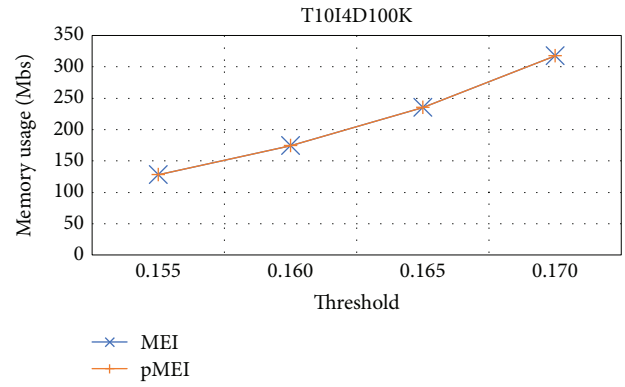


Figure 13: Memory usage of the MEI and pMEI methods on the T10I4D100K.

average speedup rate is 1.94, 1.95, 1.95, 1.98, 1.99, and 2.06 on 2 cores and 3.82, 3.87, 3.82, 3.80, 3.82, and 3.93 on 4 cores with Accidents, Chess, Connect, Mushroom, Pumsb, and T10I4D100K datasets, respectively. Generally, the speedup is always proportional to the number of cores of the computer.

5. Conclusions

This study proposed a proficient technique for mining EIs, namely, pMEI based on multicore computers to enhance the performance. This method overcomes these drawbacks of parallel computing approaches including the interactive expense cost, synchronization, and data duplication. A dynamic mechanism for load balancing the processor workloads was also used. Experiments show that pMEI is better than MERIT for mining EIs in execution time.

In the future, we will study mining the top-rank-k erasable closed itemsets and maximal erasable itemsets. In addition, we will expand the study of EI mining associated with some kinds of item constraints. Besides, approaches for mining such patterns on distributed computing systems will be developed.

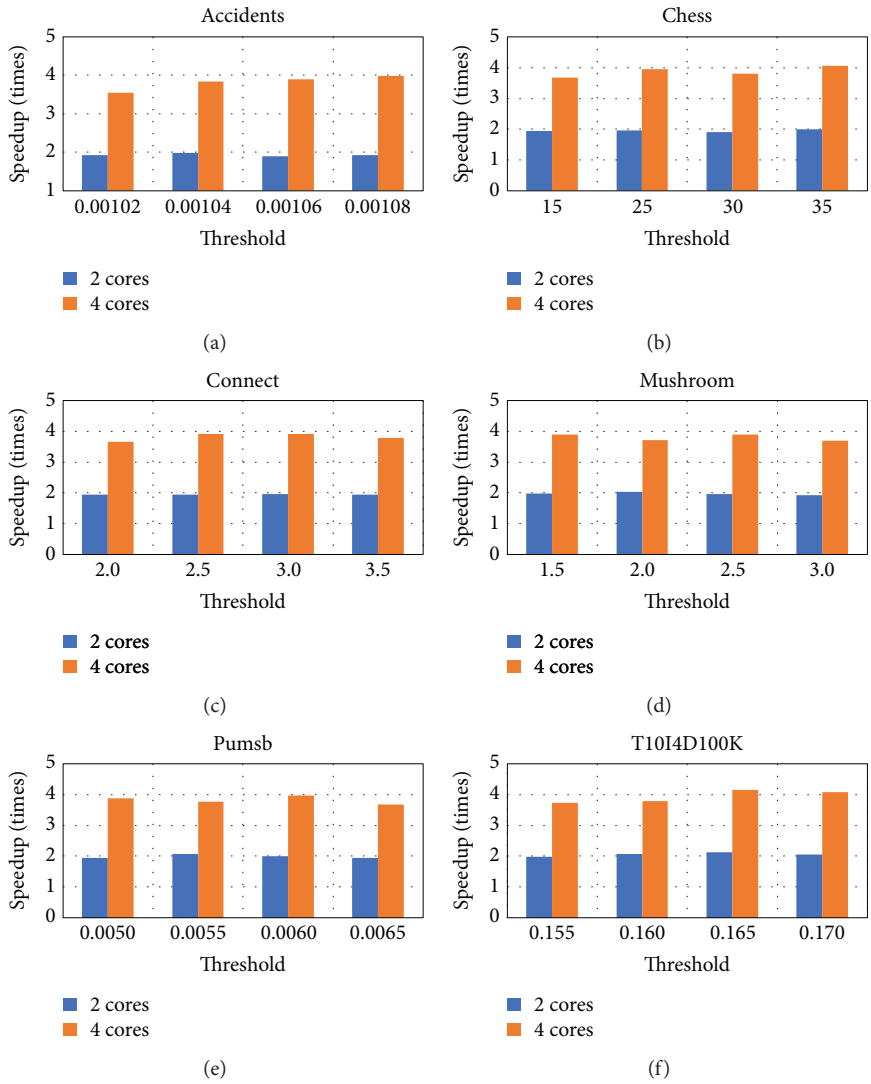


Figure 14: Speedup of the pMEI with the different numbers of cores on various datasets.

Data Availability

The product data used to support the findings of this study have been deposited in the Frequent Itemset Mining Dataset Repository (http://fimi.ua.ac.be/data/).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors wish to thank Tuong Le for his valuable comments and suggestions.

References

[1] T. Mai, B. Vo, and L. T. T. Nguyen, "A lattice-based approach for mining high utility association rules," *Information Sciences*, vol. 399, pp. 81–97, 2017.

[2] D. T. Hai, L. H. Son, and T. L. Vinh, "Novel fuzzy clustering scheme for 3D wireless sensor networks," *Applied Soft Computing*, vol. 54, pp. 141–149, 2017.

[3] N. Indurkhya, "Emerging directions in predictive text mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 4, pp. 155–164, 2015.

[4] Z.-H. Deng, "Diffnodesets: an efficient structure for fast mining frequent itemsets," *Applied Soft Computing*, vol. 41, pp. 214–223, 2016.

[5] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, and J. Zhan, "Mining of frequent patterns with multiple minimum supports," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 83–96, 2017.

[6] B. Vo, S. Pham, T. Le, and Z.-H. Deng, "A novel approach for mining maximal frequent patterns," *Expert Systems with Applications*, vol. 73, pp. 178–186, 2017.

[7] T. Kieu, B. Vo, T. Le, Z.-H. Deng, and B. Le, "Mining top-k co-occurrence items with sequential pattern," *Expert Systems with Applications*, vol. 85, pp. 123–133, 2017.

[8] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, J. M.-T. Wu, and J. Zhan, "Extracting recent weighted-based patterns

from uncertain temporal databases," *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 161–172, 2017.

[9] R. U. Kiran, J. N. Venkatesh, M. Toyoda, M. Kitsuregawa, and P. K. Reddy, "Discovering partial periodic-frequent patterns in a transactional database," *Journal of Systems and Software*, vol. 125, pp. 170–182, 2017.

[10] A. Hellal and L. Ben Romdhane, "Minimal contrast frequent pattern mining for malware detection," *Computers & Security*, vol. 62, pp. 19–32, 2016.

[11] J. Wen, M. Zhong, and Z. Wang, "Activity recognition with weighted frequent patterns mining in smart environments," *Expert Systems with Applications*, vol. 42, no. 17-18, pp. 6423–6432, 2015.

[12] Z.-H. Deng, G.-D. Fang, Z.-H. Wang, and X.-R. Xu, "Mining erasable itemsets," in *2009 International Conference on Machine Learning and Cybernetics*, pp. 67–73, Hebei, China, 2009.

[13] T. Le and B. Vo, "The lattice-based approaches for mining association rules: a review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 6, no. 4, pp. 140–151, 2016.

[14] Z.-H. Deng and X.-R. Xu, "Fast mining erasable itemsets using NC_sets," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4453–4463, 2012.

[15] T. Le and B. Vo, "MEI: an efficient algorithm for mining erasable itemsets," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 155–166, 2014.

[16] G. Nguyen, T. Le, B. Vo, and B. Le, "EIFDD: an efficient approach for erasable itemset mining of very dense datasets," *Applied Intelligence*, vol. 43, no. 1, pp. 85–94, 2015.

[17] B. Vo, T. le, G. Nguyen, and T.-P. Hong, "Efficient algorithms for mining erasable closed patterns from product datasets," *IEEE Access*, vol. 5, no. 1, pp. 3111–3120, 2017.

[18] T. Le, B. Vo, and S. W. Baik, "Efficient algorithms for mining top-rank-*k* erasable patterns using pruning strategies and the subsume concept," *Engineering Applications of Artificial Intelligence*, vol. 68, pp. 1–9, 2018.

[19] B. Vo, T. le, W. Pedrycz, G. Nguyen, and S. W. Baik, "Mining erasable itemsets with subset and superset itemset constraints," *Expert Systems with Applications*, vol. 69, pp. 50–61, 2017.

[20] G. Lee, U. Yun, H. Ryang, and D. Kim, "Erasable itemset mining over incremental databases with weight conditions," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 213–234, 2016.

[21] U. Yun and G. Lee, "Sliding window based weighted erasable stream pattern mining for stream data applications," *Future Generation Computer Systems*, vol. 59, pp. 1–20, 2016.

[22] A. Vajda, "Multi-core and many-core processor architectures," in *Programming Many-Core Chips*, pp. 9–43, Springer, Boston, MA, USA, 2011.

[23] D. Nguyen, B. Vo, and B. Le, "Efficient strategies for parallel mining class association rules," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4716–4729, 2014.

[24] B. Huynh, B. Vo, and V. Snasel, "An efficient method for mining frequent sequential patterns using multi-core processors," *Applied Intelligence*, vol. 46, no. 3, pp. 703–716, 2017.

[25] A. Laurent, B. Négrevergne, N. Sicard, and A. Termier, "Efficient parallel mining of gradual patterns on multicore processors," in *Advances in Knowledge Discovery and Management*, vol. 398 of Studies in Computational Intelligence, pp. 137–151, Springer, Berlin, Heidelberg, 2012.

[26] T. Flouri, C. S. Iliopoulos, K. Park, and S. P. Pissis, "GapMis-OMP: pairwise short-read alignment on multi-core architectures," in *Artificial Intelligence Applications and Innovations*, vol. 382 of IFIP Advances in Information and Communication Technology, pp. 593–601, Springer, Berlin, Heidelberg, 2012.

[27] F. Sánchez, F. Cabarcas, A. Ramirez, and M. Valero, "Long DNA sequence comparison on multicore architectures," in *Euro-Par 2010 - Parallel Processing*, vol. 6272 of Lecture Notes in Computer Science, pp. 247–259, Springer, Berlin, Heidelberg, 2010.

[28] G. Kan, T. Lei, K. Liang et al., "A multi-core CPU and many-core GPU based fast parallel shuffled complex evolution global optimization approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 332–344, 2016.

[29] T. Le, A. Nguyen, B. Huynh, B. Vo, and W. Pedrycz, "Mining constrained inter-sequence patterns: a novel approach to cope with item constraints," *Applied Intelligence*, vol. 48, no. 5, pp. 1327–1343, 2018.