ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20230719000815
Preliminary communication

A Parallel Mining Algorithm for Maximum Erasable Itemset Based on Multi-core Processor

Qunli ZHAO*, Hesheng CHENG, Chen SHEN

Abstract: Mining the erasable itemset is an interesting research domain, which has been applied to solve the problem of how to efficiently use limited funds to optimise production in economic crisis. After the problem of mining the erasable itemset was posed, researchers have proposed many algorithms to solve it, among which mining the maximum erasable itemset is a significant direction for research. Since all subsets of the maximum erasable itemset are erasable itemsets, all erasable itemsets can be obtained by mining the maximum erasable itemset, which reduces both the quantity of candidate and resultant itemsets generated during the mining process. However, computing many itemset values still takes a lot of CPU time when mining huge amounts of data. And it is difficult to solve the problem quickly with sequential algorithms. Therefore, this proposed study presents a parallel algorithm for the mining of maximum erasable itemsets, called PAMMEI, based on a multi-core processor platform. The algorithm divides the entire mining task into multiple subtasks and assigns them to multiple processor cores for parallel execution, while using an efficient pruning strategy to downsize the space to be searched and increase the mining speed. To verify the efficiency of the PAMMEI algorithm, the paper compares it with most advanced algorithms. The experimental results show that PAMMEI is superior to the comparable algorithms with respect to runtime, memory usage and scalability.

Keywords: data mining; erasable itemset; maximum erasable itemset; multi-core platform; product sets

1 INTRODUCTION

With the fast growth of the speed and size of data generation, how to quickly mine useful information from database is the main goal of researchers. Over the past decades, researchers have proposed many data mining methods and theories, among which mining erasable itemset is an interesting area of research. In 2009, Deng proposed a theory related to erasable itemsets. The theory is used to extract itemsets from databases of products whose value is less than a given profit threshold. A typical application scenario is how to efficiently schedule the production of a product in the event of an economic crisis or shortage of funds. An enterprise cannot produce all of its products because it does not have enough capital, which means that some of the products need to be discontinued. But how does the enterprise select the products to be discontinued? The product to be discontinued should be the one that causes the least loss of profit for the enterprise. If a product is considered to be a set of components, each of components can be regarded as an item, then a product is an itemset and the profit of the product can be regarded as the value of the itemset. Enterprises choose to go to discontinued products, which is equivalent to searching for erasable itemsets in a product database. However, this is a very time-consuming work in massive databases. So, we need to research efficient algorithms to solve the problem. After Deng first proposed the META [1] method to mine erasable itemsets in 2009, researchers have proposed several improved methods like VME [2], MERIT [3], MERIT+ [4], dMERIT+ [4], MEI [5], sMEI [6], EIFDD [7], MEIC [8], pMEI [9], and MSPPC [10]. These algorithms can be highly efficient in mining erasable itemsets from product databases. However, when the threshold is large, the mining process will generate a substantial amount of erasable itemsets. This requires not only a lot of storage space, but also a lot of CPU time to compute the gain value of each itemset. In order to decrease the amount of erasable itemsets and the occupied storage capacity generated during mining process, researchers have proposed many other algorithms like mining closed , top-rank-k, constraints and maximum erasable itemsets. The published algorithms are VM [11], dVM [12], TEP [13], TEPUS [13], MECP [14], dNC-ECMM [15], WEPS [16], IWEI [17], IWEL [18], GenMax-EI [19], Flag-GenMax-EI [19] and PE- GenMax-EI [19]. These algorithms can efficiently complete the corresponding mining tasks and produce fewer itemsets during the mining process, thus saving storage space and CPU time. It has been shown in the literature [7, 14, 19] that when these types of algorithms are used to mine huge amounts of data, only the algorithm that mines the maximum erasable itemsets produces the smallest number of resultant itemsets. Since the maximum erasable itemset contains erasable itemsets, the task of mining erasable itemsets may translate into mining maximum erasable itemsets. However, the existing algorithms for mining maximum erasable itemsets still perform sub-optimally when mining massive data. Therefore,a strong need exists to develop new algorithms to improve the performance of mining maximum erasable itemset. This paper proposes an efficient algorithm for mining maximum erasable itemsets in a multi-core based platform. For the mining work, the proposed method decomposes the mining task into multiple subtasks and assigns them to multiple processor cores for parallel execution, while adopting a global pruning strategy to decrease the quantity of candidate itemsets, thus improving the mining efficiency. The contribution of this study consists of three main items.

(1) A parallel approach is applied to the algorithm for mining the set of maximal erasable terms. By assigning multiple subtasks to multiple processor cores for parallel execution, the execution efficiency is improved and the total execution time is reduced in the proposed method.

(2) The concept of inclusion relationship between product sets is proposed. The computation of itemsets values can be reduced by utilizing this concept, thus increasing the speed of mining maximum erasable itemset.

(3) The study designed a global pruning strategy that minimizes the available space for searching and thus reduces the amount of candidate itemsets produced within the process of mining.

The later part of the paper is arranged below. Section 2 describes the related work, Section 3 provides an introduction to the related concepts, Section 4 presents a description of the algorithm and gives examples to illustrate the algorithm, Section 5 shows some experiments with the algorithm, Section 6 presents a conclusion of the paper and an outlook for later works.

2 RELATED WORKS
2.1 Erasable Itemset Mining

Numerous methods for the mining of erasable itemsets have been put forward, some of which have been summarized in the literature [18, 19]. Deng first proposed the algorithm META in 2009, which is similar to the Apriori [20] algorithm and uses a width-first approach to search for the erasable itemset. The algorithm requires several scans on databases and produces a substantial amount of candidate itemsets during the mining process, thus consuming a large amount of CPU time when scanning databases and calculating the gain values of the candidate itemsets. In 2010, Deng put forward the algorithm VME that uses the Pid_List to speed up the computation of the candidate itemsets and only requires two database scans when constructing the structure, and no further database scans are required during subsequent mining. Although this algorithm is superior to the META algorithm, calculating the profit of the candidate itemsets still needs extensive CPU time. In 2012, Deng suggested the algorithm MERIT, which uses WPPC-Tree to store the database and employs a depth-first strategy to mine erasable itemsets, which also outperforms META on performance. But the algorithm generates new candidate itemsets by merging the nodes in the tree, which demands substantial storage space and CPU time. In 2013, Le and Vo proposed the MERIT+ and dMERIT+ algorithms, which improve on the MERIT algorithm. Among them, the dMERIT+ algorithm uses the dNC-Sets structure, which greatly reduces the quantity of candidate itemsets within the process of mining, and thus saves CPU time and memory occupation, and improves the efficiency of the algorithm. dMERIT+ algorithm outperforms MERIT+ and MERIT. In 2014, Le and Vo proposed the MEI algorithm, which uses a partitioning approach and the dPidset theory to decrease CPU occupancy time and memory usage. It outperforms the MERIT+ and dMERIT+ algorithms, but is not suitable for mining dense datasets. In 2015, Nguyen and Le proposed the EIFDD algorithm, which solves the problem that the MEI algorithm is not well-suited for mining densely populated datasets. In 2017, Vo and Le proposed the pMEIC algorithm, which uses subset and superset constraints to decrease the quantity of candidate itemsets. In 2018, Huynh and Vo proposed the pMEI algorithm, which improves mining performance by using a multi-core processor platform to mine erasable itemsets in parallel. Hong et al. proposed a new algorithm in the literature [21, 22] based on the FUP [23] algorithm, these two algorithms address the concern of mining the erasable itemsets in dynamic datasets. In 2019, Le and Vo introduced the MSPPC method, which uses the SPPC-Tree structure to store itemsets for fast computation of the values of candidate itemsets and mining of the erasable itemsets. It outperforms MERIT, dMERIT+, MEI and EIFDD algorithms. In 2020, Baek and Yun presented the MED [24] method for the mining of erasable itemsets in dynamic data streams, in which a MED-Tree with a damping window and a new pruning technique are utilized to lower the quantity of candidate itemsets and enhance the performance of the method. Baek, Yun, Lin introduced the IUEP [25] to mine uncertain erasable itemsets from dynamic incremental databases. It uses IUEP-tree and IN-List to save itemsets information. In 2022, Hong and Chang presented a new approach that used to solve the problem of mining erasable itemsets in different temporal contexts in literature [26].

2.2 Erasable Closed Itemset Mining

The erasable closed itemset (named CloseEI) contains the information of the erasable itemset, but its number is less than that of the erasable itemset. Therefore, mining erasable closed itemsets can reduce the memory usage during mining. In 2015, Nguyen and Le proposed the MECP algorithm, which utilizes a divide-and-conquer strategy and a hash table to find erasable closed itemsets efficiently. In 2017, Vo and Le proposed two algorithms, ECPat and dNC-ECPM. The ECPat algorithm is based on the dPidset structure and achieves fast mining of CloseEIs. The dNC-ECPM algorithm uses the dNC_Set instead of the dPidset and improves the mining method in the ECPat algorithm. So, the dNC-ECMM algorithm outperforms the ECPat algorithm.

2.3 Weighted Erasable Itemset Mining

In 2015, Lee and Yun introduced WEPS, a weighted erasable itemset mining algorithm. WEPS uses a WEP tree structure to build a structure of the data through a single scan of the database. The subsequent mining process is performed on the tree without scanning the database again. In 2016, Lee, Yun, Ryan and Kim proposed the algorithm IWEI for mining weighted erasable itemsets from streaming databases. The algorithm constructs a data structure called the IWEI-tree to store information about the itemsets and updates the structure during the reconstruction of the tree. In 2020, Nam, Yun and Yoon proposed IWEL, an algorithm for mining weighted erasable itemsets from dynamically growing databases, which uses a list structure to store candidate erasable itemsets and efficiently performs the mining operation with weighted conditions. Its performance is better than that of the IWEI algorithm.

2.4 Top-rank-k Erasable Itemset Mining

In 2013, Deng first presented VM, a top-rank-k erasable itemset mining algorithm using the Pid_List structure. In 2014, Nguyen and Le proposed an improved algorithm, dVM, which uses the dPid_List structure. The dVM has smaller runtime and memory usage than the VM. In 2018, Le and Vo proposed TEP and TEPUS algorithms. TEP uses dPidset structure and dynamic threshold pruning strategy to improve the mining speed. TEPUS is an improved algorithm of TEP which uses the concept of subume and indexing strategy to decrease the execution time and memory usage. In 2022, Nguyen and Le proposed the TRK-ECP algorithm [27], which uses a virtual threshold-based pruning method and dPidset structure to speed up the mining process. It can efficiently mine top-row-k erasable closed itemset.

2.5 Maximum Erasable Itemset Mining

Since a maximum erasable itemset contains many erasable itemsets, mining the maximum erasable itemset can greatly reduce the number of mining result sets as well as CPU time and memory space usage. Nguyen and Le proposed three algorithms in the literature [19] in 2019, namely GenMax-EI, Flag-GenMax-EI and PE-GenMax-EI. GenMax-EI is based on the algorithm GenMax [28]. The Flag-GenMax-EI adds constraints on subset and superset relations to GenMax-EI and uses the isMax tag to accelerate the determination of maximum erasable itemset. The PE-GenMax-EI algorithm builds on the Flag-GenMax-EI algorithm and improves on it. It uses the dPidset structure and an early pruning strategy, and takes up less runtime and memory space than GenMax-EI and Flag-GenMax-EI. Although these algorithms can mine maximum erasable itemsets effectively, it is difficult to get the mining results quickly when they are used to mine huge amount of sparse data. We can see from the experimental results in Reference [19] that the PE-GenMax-EI algorithm is currently the most effective algorithm for mining maximum erasable itemsets, but it still requires a large amount of CPU time when mining Retail datasets. To quickly mine maximum erasable itemsets in massive data, we propose a parallel mining algorithm based on a multi-core processor platform called PAMMEI (Parallel Algorithm for Mining Maximum Erasable Item Sets). It divides the mining process into multiple tasks, each assigned to a single processor core, so that multiple tasks can be executed in parallel. At the end of this paper we have implemented the PAMMEI algorithm and compared it with the PE-GenMax-EI algorithm.

3 RELATED CONCEPTS

3.1 Erasable Itemset and Maximum Erasable Itemset

Set the product database *PDB* as $\{P_1, P_2, \ldots, P_m\}$, P_i is a product, and the items contained in the product are expressed as P_i*items*, the value of the product is expressed as P_i*value*, the size of *PDB* is represented as |*PDB*|. Let *I* be the set of all items in *PDB*, $I = \{i_1, i_2, \ldots, i_n\}$, and P_i can be expressed as $P_i = \{i_{k1}, i_{k2}, \ldots, i_{km} | k_j \in [1, \ldots, n]\}$. We present an example product database, as shown in Tab. 1. For an itemset *S* and the items in *S* are contained in *I*, the value of *S* is expressed as *Gain*(*S*), which can be computed by the value of products in *PDB*. The equation is as follows.

$$Gain(S) = \sum_{\{P_i | S \cap P_i items \neq 0\}} P_i value \tag{1}$$

Definition 1 (Erasable Itemset). Let the sum of all products value in *PDB* be expressed by *Sum*(*PDB*), and δ is a threshold. If *S* meets the following conditions: $Gain(S) \le Sum(PDB) \cdot \delta$, then *S* is called an erasable itemset, which is represented as EI.

Definition 2 (Maximum Erasable Itemset). Let *T* be the set of all *EI* in *PDB*, $S \in T$. If there are no supersets of *S* in *T*, then *S* is called a maximum erasable itemset, which is expressed as MaxEI, and the set of all MaxEI is expressed as MESet (Maximum Erasable itemset Set).

Example 1. According to Definitions 1 and 2, when δ is 60%, the erasable itemsets and the maximum erasable itemsets mined from the product database in Tab. 1 are shown in Tab. 2.

Table 1 A product database (PDB)

Product ID (Pid)	Items	Value
P_1	i_1, i_2, i_3	230
P_2	i_1, i_2	120
P_3	i_1, i_3	110
P_4	i_2, i_3, i_5	240
P_5	i_2, i_5	120
P_6	i_3, i_5	130
P_7	i_3, i_4, i_5, i_6, i_7	210
P_8	i_4, i_5, i_6, i_8	140
P_9	i_4, i_6	105
P_{10}	i_2, i_6, i_8	201
P_{11}	i_3, i_6	115

Property 1. If $Y \in$ MESet and $S \subseteq Y$, then $Gain(S) \le Gain(Y) \le Sum(PDB) \times \delta$.

Proof. Suppose that the items in *S* are included in product sequence P_S, $P_S = \{P_k | k \in [1, n]\}$, and the items in *Y* are included in product sequence P_Y, $P_Y = \{P_j | j \in [1, n]\}$. Because $S \subseteq Y$, so $P_S \subseteq P_Y$. According to Eq. (1), we can get $Gain(S) \le Gain(Y)$. And because $Y \in$ MESet, so $Gain(Y) \le Sum(PDB) \times \delta$. Finally, we can deduce that $Gain(S) \le Gain(Y) \le Sum(PDB) \times \delta$.

Table 2 The EIs and MaxEIs in PDB (δ = 60%)

Size	EIs	MaxEIs
1	$\{i_2\}, \{i_5\}, \{i_6\}, \{i_1\}, \{i_4\}, \{i_8\}, \{i_7\}$	
2	$\{i_2, i_1\}, \{i_5, i_4\}, \{i_5, i_7\}, \{i_6, i_4\}, \{i_6, i_8\}, \{i_6, i_7\}, \{i_1, i_4\}, \{i_1, i_4\}, \{i_1, i_8\}, \{i_1, i_7\}, \{i_4, i_8\}, \{i_4, i_7\}, \{i_8, i_7\}$	$\{i_2, i_1\}$
3	$\{i_5, i_4, i_7\}, \{i_6, i_4, i_8\}, \{i_6, i_4, i_7\}, \{i_6, i_8, i_7\}, \{i_1, i_4, i_7\}, \{i_1, i_8, i_7\}, \{i_4, i_8, i_7\}$	$\{i_5, i_4, i_7\}, \{i_1, i_4, i_7\}, \{i_1, i_8, i_7\}$
4	$\{i_6, i_4, i_8, i_7\}$	$\{i_6, i_4, i_8, i_7\}$

3.2 Set of Product Identifiers

Definition 3 (Pidset). Given an item i_k, the products containing i_k in *PDB* are $\{P_{k1}, P_{k2}, \ldots, P_{km} | 1 \le k_j \le |PDB|\}$, which is a set of product identifiers (shorted for Pidset), denoted as $P(i_k)$. For example, in Tab. 1, $P(i_4) = \{7, 8, 9\}$. Let *S* be an itemset, $S = \{i_{k1}, i_{k2}, \ldots, i_{km} | k_j \in [1, n]\}$, so that the Pidset of *S* is $P(S) = P(i_{k1}) \cup P(i_{k2}) \cup, \ldots, \cup P(i_{km})$. The value of *S* can be computed from *P*(*S*) by the following equation.

$$Gain(S) = \sum_{P_i \in P(S)} P_i value \tag{2}$$

Example 2. Given an itemset $Y = \{i_4, i_5\}$ in Tab. 1, the Pidset of *Y* is *P*(*Y*), and $P(Y) = P(i_4) \cup P(i_5) = \{7, 8, 9\} \cup \{4, 5, 6, 7, 8\} = \{4, 5, 6, 7, 8, 9\}$, $Gain(Y) = P_4value + P_5value + P_6value + P_7value + P_8value + P_9value = 240 + 120 + 130 + 210 + 140 + 105 = 945$.

3.3 Inclusion Relationship Between Pidsets

Definition 4. Let X and Y be two different itemsets, and if $P(X) \subseteq P(Y)$, the Pidset of X is included in the Pidset of Y. We denote the inclusion relationship between Pidsets as follows: $C(Y) = \{X \mid X \in I, P(X) \subseteq P(Y)\}$. According to Eq. (2), we can get $Gain(X \cup Y) = Gain(Y)$.

Example 3. From Tab. 1, we can get $P(i_4) = \{7, 8, 9\}$, $P(i_6) = \{7, 8, 9, 10, 11\}$, $P(i_7) = \{7\}$, $P(i_8) = \{8, 10\}$. Since $P(i_4) \subseteq P(i_6)$, $P(i_7) \subseteq P(i_4)$, $P(i_7) \subseteq P(i_6)$, $P(i_8) \subseteq P(i_6)$, we can get $C(i_6) = \{i_4, i_7, i_8\}$, $C(i_4)=\{i_7\}$, $C(i_4i_6) = \{i_7, i_8\}$.

Property 2. Given three different itemsets X, Y and Z, if $X \in C(Y)$, $Y \in C(Z)$, then $X \in C(Z)$.

Proof. Since $X \in C(Y)$, $Y \in C(Z)$, we can know $P(X) \subseteq P(Y)$, $P(Y) \subseteq P(Z)$, so, $P(X) \subseteq P(Z)$, we can get that X is an element of $C(Z)$, that is $X \in C(Z)$. The property is proved.

3.4 Parallel Mining Process

In a multi-core processor platform, each processor contains multiple processor cores. A computational task can be decomposed into multiple independent subtasks, and then each subtask is assigned to a processor core for independent execution, which enables parallel execution of multiple subtasks. For a processor with m cores, its performance is about m times higher than that of a single core processor. When mining the maximum erasable itemsets with multi-core processors, it is necessary to decompose the mining task into multiple independent subtasks in the mining algorithm, and then assign those subtasks to multiple processor cores for parallel execution in order, and finally merge the mining results of each subtask. The process is shown in Fig. 1. In Fig. 1, the input mining task is first divided into multiple subtasks. Since the number of subtasks is more than the number of processor cores, the subtasks are first placed in the task queue waiting for execution in order to balance the load on the cores. When one core in the processor is idle, the queue head subtask is taken out of the task queue and assigned to the idle core for execution. Multiple cores are relatively independent in the process of executing subtasks, and there is no cross-interference between them. When the execution of a subtask is completed, the output itemsets it produces are placed in the set of result itemsets. When all the subtasks in the queue are executed, the whole mining process is over.

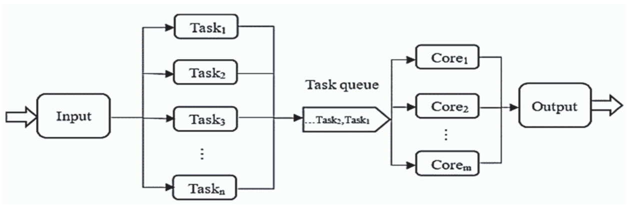

Figure 1 Multi-core processor mining process

4 PAMMEI ALGORITHM

Let L_1 be the set of all erasable items, represented by one-dimensional array. The ith element in L_1 is expressed as $L_1(i)$, and the erasable items of $L_1(i)$ are expressed as $L_1(i)items$. The *Pidset* corresponding to $L_1(i)items$ is represented by $L_1(i)pids$, and the value of $L_1(i)items$ is represented by $L_1(i)value$. All items in L_1 are arranged in the descending order by the size of $L_1(i)pids$. The length of L_1 is represented by $|L_1|$.

Property 3. Let X and Y be the two different erasable itemsets. If $X \in C(Y)$, neither X nor Y is the maximum erasable itemset.

Proof. Because $X \in C(Y)$, $P(X) \subseteq P(Y)$ and $P(X \cup Y) = P(Y)$. According to Eq. (2), we get $Gain(X \cup Y) = Gain(Y)$, so $X \cup Y$ is also an erasable itemset. Since $X \subseteq X \cup Y$ and $Y \subseteq X \cup Y$, X and Y can't be the maximum erasable itemset.

Property 4. Let X be an erasable itemset, $X \cap (i_k \cup i_{k+1} \cup, \ldots, \cup i_{|L1|}) = \emptyset$, $i_k, i_{k+1}, \ldots, i_{|L1|}$ are the last $|L_1|-k+1$ items in L_1, $Y = X \cup i_k \cup i_{k+1} \cup, \ldots, \cup i_{|L1|}$, if $(P(i_k) \cup P(i_{k+1}) \cup, \ldots, \cup P(i_{|L1|})) \subseteq P(X)$, and no itemsets longer than Y in MESet, then Y must be the maximum erasable itemset.

Proof. Because $(P(i_k) \cup P(i_{k+1}) \cup, \ldots, \cup P(i_{|L1|})) \subseteq P(X)$, so $Gain(Y) = Gain(X \cup i_k \cup i_{k+1} \cup, \ldots, \cup i_{|L1|}) = Gain(X)$, Y is also an erasable itemset. Suppose Y is not the maximum erasable itemset and Z is a superset of Y. Then Z must contain more items than Y contains, which contradicts the condition in the property. Therefore, the property is true and the proof is complete.

Property 5. Let X be an erasable itemset, i_k is an erasable item in L_1. If $X \cup i_k$ is not an erasable itemset, all itemsets prefixed with $X \cup i_k$ are not erasable itemsets.

Proof. Because $X \cup i_k$ is not an erasable itemset, according to Property 1, all supersets of $X \cup i_k$ are not erasable itemsets. Therefore, the property is true.

4.1 Pruning Strategy

According to property 2 and property 3, in mining the maximum erasable itemset prefixed by the itemset X, if $Y \in C(X)$, X and Y are merged directly without searching for subsets of Y, which reduces some computational cost. According to property 4, if $Y \in C(X)$ and the items contained in Y are the last k items in L_1, there is no need to search for these k items in all tasks. Since X is erasable itemset, neither X nor Y is maximum erasable itemset according to property 3, while $X \cup Y$ may be a maximum erasable itemset. Therefore, in the algorithm, we set a flag called *noMaxflag* for each item in Y. When *noMaxflag* = 1, it means that the item and its subsequent items do not need to be searched anymore, and the mining process of the maximum erasable itemsets prefixed by X can be stopped. Meanwhile, in all subtasks, itemset Y does not need to be searched. Therefore, this can achieve global pruning in multiple subtasks, and also reduce the CPU time and memory usage. According to property 5, if $X \cup i_k$ is not an erasable itemset, there is no need to search for branches after i_k during the mining process. This can save a lot of CPU time and memory usage.

4.2 The Algorithm

According to the above properties and pruning strategy, we give a description of PAMMEI algorithm as follows (see Fig. 2 to Fig. 4). A *havesuperset* flag is set in the algorithm. When a superset of X is an erasable itemset, *havesuperset* = 1. When X has no superset or its superset is not an erasable itemset, *havesuperset* = 0. At this time, X must be a maximum erasable itemset.

```
Input: PDB, δ, MESet=∅
Output: MESet
(1) Scan PDB, find all erasable items and their pids,
    and store them in L1;
(2) Sort L1 in descending order according to the
    length of L1[i].pids, i∈[1, |L1|];
(3) for each k∈[1, |L1|] do
(4)    S = L1[k];
(5)    Create Taski (1≦i≦|L1|) with three parameters
           S, L1, k and put the task into the TaskQueue;
(6) end for
(7) while TaskQueue≠∅ do
(8)     When Corej (j∈[1, m]) is idle, take a task
        from the TaskQueue to Corej for execution;
(9)     call StartTask(S, L1,k);
(10) end while
```

Figure 2 The PMMEI algorithm

```
Procedure StartTask (S, L1,k) //start a task
(1) result = SearchMaxEI(S, L1,k+1);
(2) if result =1 then
(3)    delete all tasks created afer this task;
(4) end if
End Procedure
```

Figure 3 StartTask executes a task

```
Procedure SearchMaxEI(S, L1,pos)
(1) if S.noMaxFlag = 1 then
(2)         return 1;
(3) end if
(4) haveSuperset = 0;
(5) for j←pos to |L1| do
(6)     if S.pids ⊠ L1[j].pids then
(7)         S.items = S.items⊠ L1[j].items;
(8)         if j=|L1| then
(9)            MESet= MESet⊠ S;
(10)           if { i_pos,i_k+1,...,i_|L1|]}∈C(S) then
(11)              for all k∈[pos, |L1|] do L1[k].noMaxFlag=1;
(12)           end if
(13)        end if
(14)    else
(15)        T.items = S.items⊠ L1[j].items;
(16)        T.value = Gain(S⊠ L1[j]);
(17)        if T.value ≤ Sum(PDB) ×δ   then
(18)           haveSuperset = 1; //S has a superset
(19)           T.pids = S.pids⊠ L1[j].pids;
(20)           if j<|L1| then
(21)              if SearchMaxEI(T, L1, j+1) = 1 then
(22)                 return 1;
(23)              end if
(24)           else if j=|L1| then
(25)              MESet= MESet⊠ T;
(26)              if { i_pos,i_k+1,...,i_|L1|]}∈C(T) then
(27)                for all k∈[pos, |L1|] do L1[k].noMaxFlag=1;
(28)              end if
(29)           end if
(30)        else if j=|L1| and haveSuperset = 0 then
(31)           MESet= MESet⊠ S;
(32)        end if
(33)    end if
(34) end for
(35) if pos >|L1| then MESet= MESet⊠ S; end if
End Procedure
```

Figure 4 The procedure SearchMaxEI

4.3 An Illustrated Example

From the description of PAMMEI algorithm, we can see that the task of mining the maximum erasable itemsets is mainly completed in the procedure SearchMaxEI. In this part, we use the product database in Tab.1 to illustrate the execution process of SearchMaxEI. In Tab. 1, *SUM(PDB)* = 1721, when δ = 60%, the corresponding $L_1 = \{i_2{:}911,\ i_5{:}40,\ i_6{:}771,\ i_1{:}460,\ i_4{:}455,\ i_8{:}341,\ i_7{:}210\}$. This means that the erasable items are i_2, i_5, i_6, i_1, i_4, i_8, i_7 in the PDB. The pidsets of each item in L_1 are $P(i_2) = \{1, 2, 4, 5, 10\}$, $P(i_5) = \{4, 5, 6, 7, 8\}$, $P(i_6) = \{7, 8, 9, 10, 11\}$, $P(i_1) = \{1, 2, 3\}$, $P(i_4) = \{7, 8, 9\}$, $P(i_8) = \{8, 10\}$, $P(i_7) = \{7\}$. Assuming that the processor has three cores, Core$_1$, Core$_2$ and Core$_3$, the algorithm divides the mining task into seven subtasks, *Task*$_1$, *Task*$_2$, *Task*$_3$, *Task*$_4$, *Task*$_5$, *Task*$_6$ and *Task*$_7$, for mining the maximum erasable itemsets prefixed with i_2, i_5, i_6, i_1, i_4, i_8 and i_7, and puts them into TaskQueue. Since the three cores are idle at the beginning, the algorithm takes *Task*$_1$, *Task*$_2$ and *Task*$_3$ out of TaskQueue in turn and puts them into *Core*$_1$, *Core*$_2$ and *Core*$_3$ for execution. The mining process is shown in Fig. 5 to Fig. 7. In Fig. 5, since all supersets of $\{i_2i_1\}$ are not erasable itemset, according to definition 2, $\{i_2i_1\}$ is a maximum erasable itemset. In Fig. 6, since $P(\{i_7\}) \subset P(\{i_5i_4\})$, we merge $\{i_7\}$ and $\{i_5i_4\}$ and take the value of $\{i_5i_4\}$ as the value of $\{i_5i_4i_7\}$. Because i_7 is the last item in L_1, according to property 4, $\{i_5i_4i_7\}$ is the maximum erasable itemset, and set $L_1(i_7)$. noMaxFlag = 1. In Fig.7, since $P(\{i_4i_8i_7\}) \subseteq P(\{i_6\})$, we merge $\{i_4i_8i_7\}$ and $\{i_6\}$ directly. Because i_4, i_8 and i_7 are the last three items in L_1, according to property 4, $\{i_6i_4i_8i_7\}$ is the maximum erasable itemset, and set $L_1(i_4)$.noMaxFlag = 1 and $L_1(i_8)$. noMaxFlag = 1, so i_4 and its later items i_8, i_7 are no longer searched. When a core is idle, Task$_4$ is removed from the TaskQueue and executed in the Core, as shown in Fig. 8. Since $L_1(i_4)$. noMaxFlag = 1, *Task*$_5$, *Task*$_6$ and *Task*$_7$ do not need to be executed again according to the first line of SearchMaxEI and the third line of StartTask. The maximum erasable itemsets obtained at the end of the algorithm execution are shown in Tab. 2.

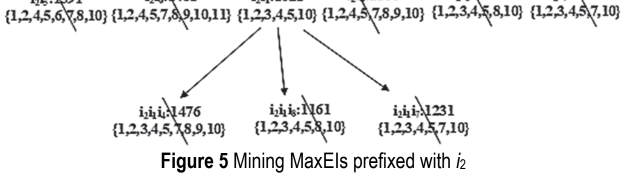

Figure 5 Mining MaxEIs prefixed with i_2

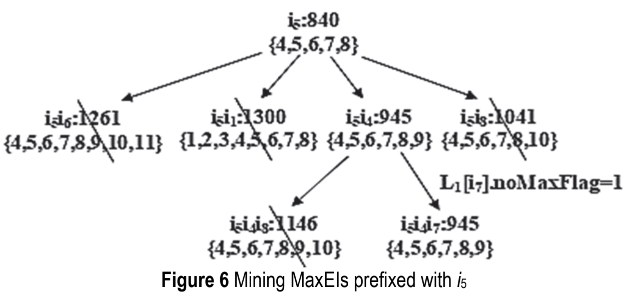

Figure 6 Mining MaxEIs prefixed with i_5

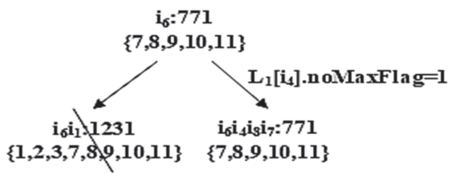

Figure 7 Mining MaxEIs prefixed with *i*6

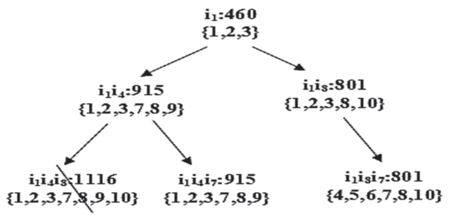

Figure 8 Mining MaxEIs prefixed with *i*1

5 EXPERIMENTS

In this part, we chose PE-GenMax-EI, the state-of-the-art algorithm for mining maximum erasable itemsets, to compare with PAMMEI. We implemented PAMMEI and PE-GenMax-EI respectively in C++ Builder 10 using the C++ language. When implementing the PAMMEI algorithm, we designed five threads, corresponding to five cores of the processor. When implementing the PE-GenMax-EI algorithm, we designed only one thread, corresponding to a single core of the processor. The computer used in the experiments was installed with an Intel Core i7 11700K CPU (with 8 cores), 16 GB RAM and Windows 10 operating system. The datasets used in the experiments are the same as in the literature [19]. These datasets can be downloaded from http://fimi.cs. helsinki.fi/data/. During the experiments, we ran the PAMMEI algorithm and the PE-GenMax-EI algorithm separately and recorded the running time used and the maximum memory used for each algorithm, as shown in Fig. 9 to Fig. 26. We have also conducted algorithm scalability experiments (in Fig. 27 and Fig. 28) and multi-core processor platform validation experiments (in Fig. 29 to Fig. 32).

5.1 The Running Time

The experimental results show that the performance of PAMMEI algorithm based on multi-core platform is better than that of PE- GenMax-EI algorithm. From these results, we can see that for dense datasets Chess and Connect, the gap between PAMMEI and PE-GenMax-EI is small, while for sparse datasets Mushroom, T10I4D100K, BMP-Pos, Pumb, Retail, Kosarak and Pumsb_Star, the gap between the two algorithms is large.

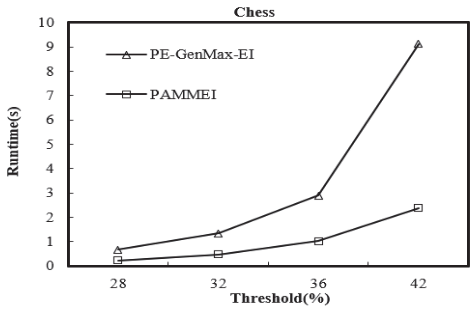

Figure 9 Runtime on chess

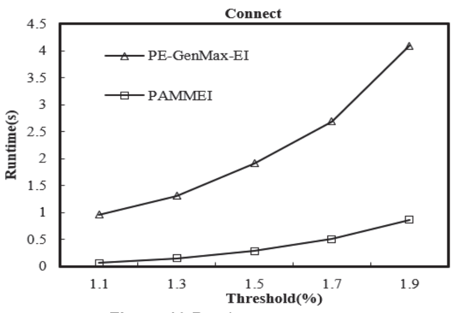

Figure 10 Runtime on connect

On the Chess dataset, when the thresholds are 28%, 32%, 36% and 42%, the runtime of PE-GenMax-EI algorithm is 721 ms, 1345 ms, 2897 ms and 9123 ms respectively, and the runtime of PAMMEI algorithm is 198 ms, 478 ms, 1023 ms and 2387 ms respectively. The ratios of the runtime of the two algorithms are 2.807, 2.814, 2.832 and 3.822 respectively, with an average of 3.069. On the Connect dataset, the average value of runtime ratio is 7.877. On the sparse datasets of Mushroom, T10I4D100K, BMP-POS, Pumb, Retail, Kosarak and Pumsb_Star, the average values of runtime ratio are 29.483, 14.865, 49.278, 13.474, 22.148, 18.572 and 13.1 respectively. From these ratios, we can see that there is a large gap between the two algorithms in running time on sparse datasets and a small gap on dense datasets. Thus the PAMMEI algorithm is better than PE-GenMax-EI in terms of runtime and can efficiently mine both dense and sparse datasets.

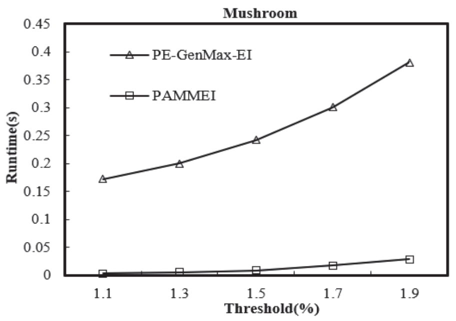

Figure 11 Runtime on mushroom

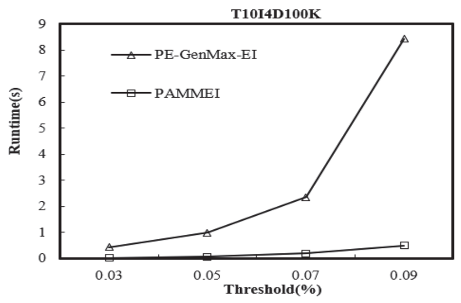

Figure 12 Runtime on T10I4100K

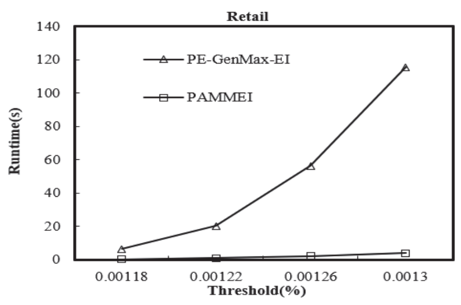

Figure 13 Runtime on retail

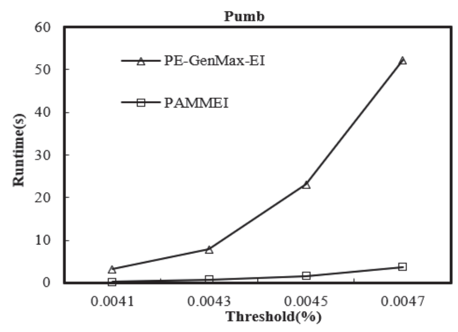

Figure 14 Runtime on Pumb

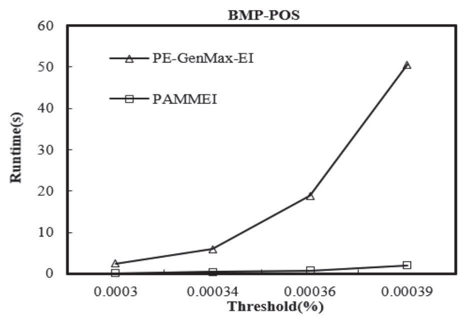

Figure 15 Runtime on BMP-POS

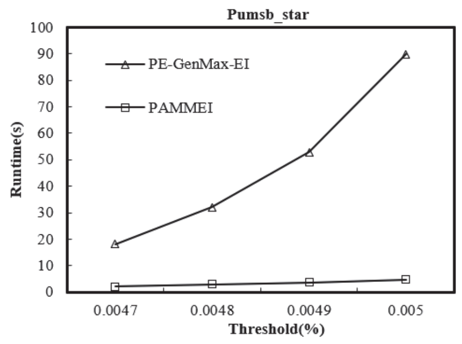

Figure 16 Runtime on Pumb_star

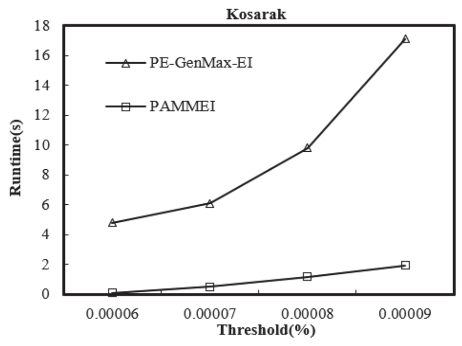

Figure 17 Runtime on kosarak

5.2 The Memory Usage

In the memory usage test, we recorded the maximum amount of memory used by the algorithm during its execution. By comparing the memory usage of the PAMMEI and PE-GenMax-EI algorithms it can be seen that the PAMMEI algorithm uses less memory than the PE-GenMax-EI algorithm. This result is due to the fact that the pruning strategy used in the PAMMEI algorithm results in a relatively small number of the candidate itemsets being generated during the mining process, which not only reduces the time spent by the CPU to calculate the itemset values, but also reduces the amount of memory usage.

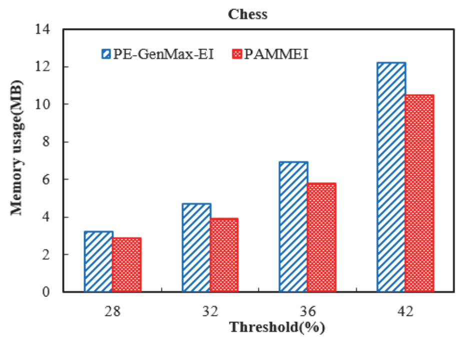

Figure 18 Memory usage on chess

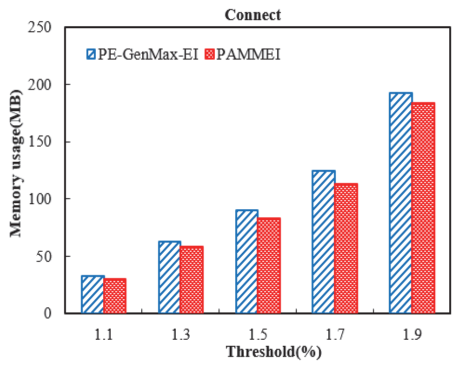

Figure 19 Memory usage on connect

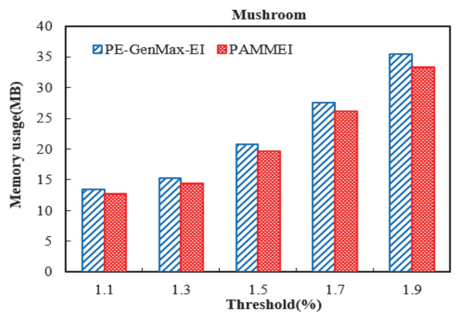

Figure 20 Memory usage on mushroom

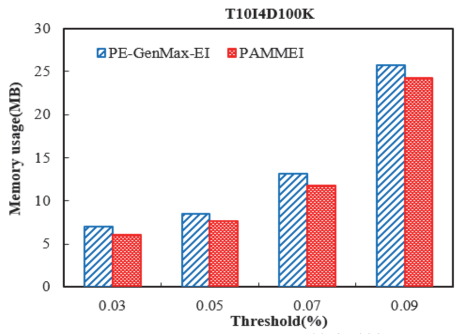

Figure 21 Memory usage on T10I4D100K

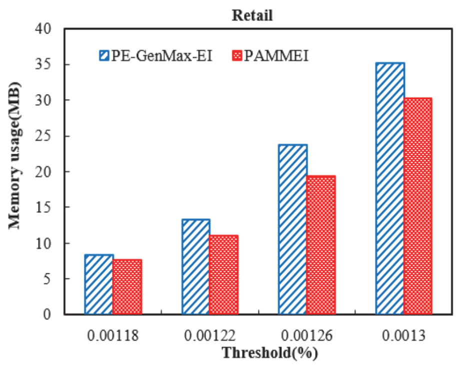

Figure 22 Memory usage on retail

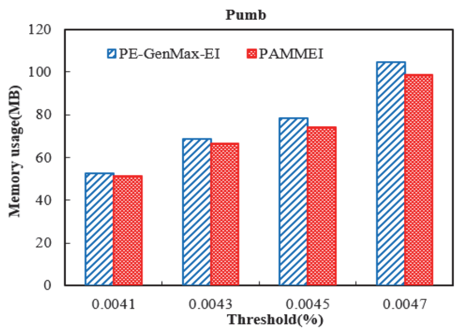

Figure 23 Memory usage on Pumsb

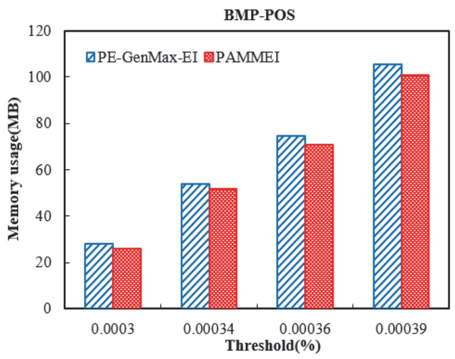

Figure 24 Memory usage on BMP-POS

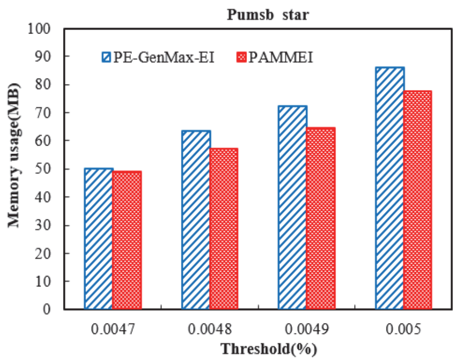

Figure 25 Memory usage on Pumsb_star

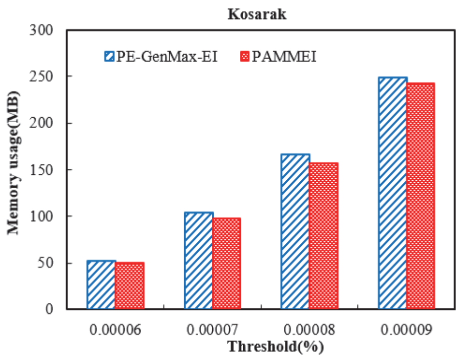

Figure 26 Memory usage on kosarak

5.3 Scalability Tests

To verify the scalability of algorithms, we ran the two algorithms on datasets of different sizes and tested their runtimes to check the effect of dataset size on the performance of the two algorithms. To make the tests more representative, we chose two datasets, BMP-POS and Connect, which are sparse and dense datasets respectively. The results of the tests are shown in Fig. 27 and Fig. 28. In Fig. 27, we set the threshold to 0.018 and in Fig. 28, we set the threshold to 0.0000038.

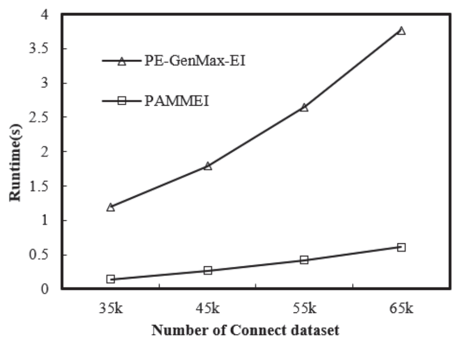

Figure 27 Scalability test on runtime of the two algorithms for Connect dataset with different sizes

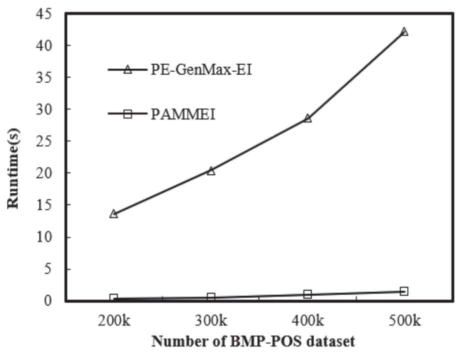

Figure 28 Scalability test on runtime of the two algorithms for BMP-POS dataset with different sizes

From the two Figures, it can be seen that the PAMMEI algorithm has better scalability compared to the PE-GenMax -EI algorithm.

5.4 Muti-Core Processor Test

We ran the PAMMEI algorithm on different numbers of processor cores to verify the effectiveness of the multi-core processors, as shown in Fig. 29 to Fig. 32.

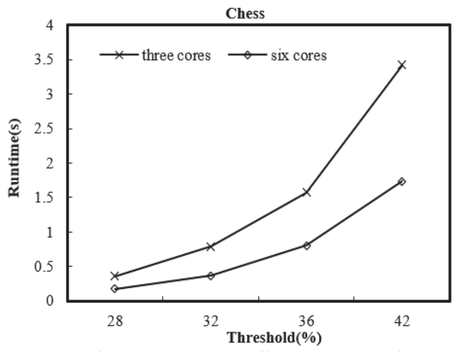

Figure 29 Runtime of PAMMEI with the different numbers of cores on chess

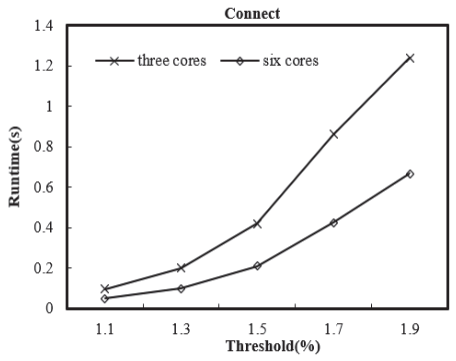

Figure 30 Runtime of PAMMEI with the different numbers of cores on connect

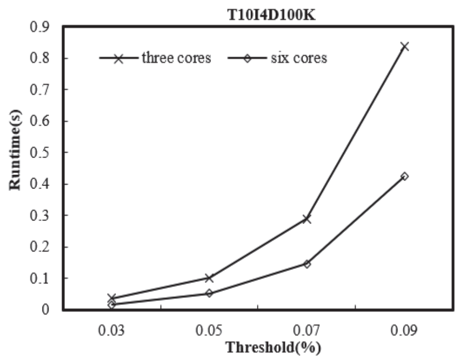

Figure 31 Runtime of PAMMEI with the different numbers of cores on T10I4D100K

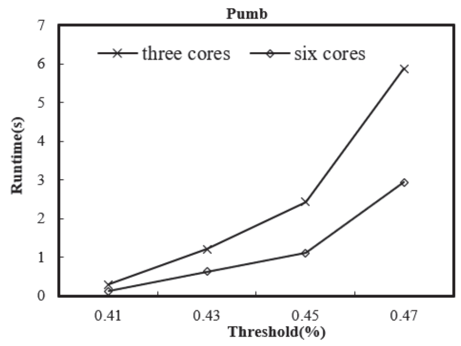

Figure 32 Runtime of PAMMEI with the different numbers of cores on Pumb

We chose two dense datasets, Chess and Connect, and two sparse datasets, T10I4D100K and Pumb. From the experimental data, it can be concluded that on the Chess, Connect, T10I4D100K and Pumb datasets, the ratio of the average execution time of the algorithm on three and six processor cores is 2.016, 1.972, 2.056 and 2.095 respectively. The execution time of the algorithm on three processor cores is approximately equal to twice that of six processor cores.

6 CONCLUSIONS

PAMMEI is an efficient algorithm for finding maximum erasable itemsets developed on a multi-core platform. The algorithm integrates a parallel approach to maximum erasable itemset mining for the first time. In the executing process of PAMMEI , the whole mining task is first divided into multiple sub-tasks, and then the multiple sub-tasks are sequentially arranged to be executed in parallel on multiple processor cores, and finally the mining results of each sub-task are merged to get the desired maximum erasable itemset. This parallel mining method used in the algorithm makes it better in runtime, memory usage and scalability. Furthermore, a novel pruning technique is designed, which utilizes an inclusion relationship between the product sets and cuts off the branches in the search space that do not need to be searched, which decreases volume of candidate sets generated by the process of mining, and thus improving mining efficiency. The experimental data shows that PAMMEI performs much superior to comparison algorithm in mining the maximum erasable itemsets. Therefore, PAMMEI can better accomplish the task of mining maximum erasable itemsets. In our future work, a method for mining maximum erasable itemsets in ultra-large-scale streaming data will be the next step in our research. This method mainly addresses the application of efficiently finding useful information in ever-increasing amounts of massive data.

Acknowledgements

This study was supported by the Natural Science Foundation of Anhui Province in China (No. KJ2020A0113), the Natural Science Key Program of Hefei Normal University (No. 2021KJZD19), and the Anhui Province Key Laboratory of Simulation and Design for Electronic Information System (No. 2020ZDSYSYB08).

7 REFERENCES

[1] Deng, Z. H., Fang, G. D., Wang, Z. H., & Xu, X. R. (2009). Mining erasable itemsets. *2009 International Conference on Machine Learning and Cybernetics*, Baoding, China, *1*, 67-73. https://doi.org/10.1109/ICMLC.2009.5212520

[2] Deng, Z. & Xu, X. (2010). An efficient algorithm for mining erasable itemsets. *Advanced Data Mining and Applications: 6th International Conference*, ADMA 2010, Chongqing, China, November 19-21, Proceedings, Part I 6, Springer Berlin Heidelberg, 214-225. https://doi.org/10.1007/978-3-642-17316-5_20

[3] Deng, Z. H. & Xu, X. R. (2012). Fast mining erasable itemsets using NC_sets. *Expert Systems with Applications*, *39*(4), 4453-4463. https://doi.org/10.1016/j.eswa.2011.09.143

[4] Le, T., Vo, B., & Coenen, F. (2013). An efficient algorithm for mining erasable itemsets using the difference of NC-Sets. *2013 IEEE International Conference on Systems, Man, and Cybernetics*, Manchester, UK, 2270-2274. https://doi.org/10.1109/SMC.2013.388

[5] Le, T. & Vo, B. (2014). MEI: an efficient algorithm for mining erasable itemsets. *Engineering Applications of Artificial Intelligence*, *27*, 155-166. https://doi.org/10.1016/j.engappai.2013.09.002

[6] Nguyen, G., Le, T., Vo, B., Le, B., & Trinh, P. C. (2014). Subsume Concept in Erasable Itemset Mining. *Computational Collective Intelligence. Technologies and Applications: 6th International Conference*, *ICCCI 2014,* Seoul, Korea, September 24-26, Proceedings 6, Springer International Publishing, 515-523. https://doi.org/10.1007/978-3-319-11289-3_52

[7] Nguyen, G., Le, T., Vo, B., & Le, B. (2015). EIFDD: An efficient approach for erasable itemset mining of very dense datasets. *Applied Intelligence*, *43*, 85-94. https://doi.org/10.1007/s10489-014-0644-8

[8] Vo, B., Le, T., Pedrycz, W., Nguyen, G., & Baik, S.W. (2017). Mining erasable itemsets with subset and superset itemset constraints. *Expert Systems with Applications*, *69*, 50-61. https://doi.org/10.1016/j.eswa.2016.10.028

[9] Huynh, B. & Vo, B. (2018). An efficient method for mining erasable itemsets using multi-core processor platform. *Complexity*, 8487641. https://doi.org/10.1155/2018/8487641

[10] Le, T., Vo, B., Fournier-Viger, P., Lee, M. Y., & Baik, S. W. (2019). SPPC: A new tree structure for mining erasable patterns in data streams. *Applied Intelligence*, *49*, 478-495. https://doi.org/10.1007/s10489-018-1280-5

[11] Deng, Z. (2013). Mining top-rank-k erasable itemsets by PID_lists. *International Journal of Intelligent Systems*, *28*(4), 366-379. https://doi.org/10.1002/int.21580

[12] Nguyen, G., Le, T., Vo, B., & Le, B. (2014). A new approach for mining top-rank-k erasable itemsets. *Intelligent Information and Database Systems: 6th Asian Conference*, ACIIDS 2014, Bangkok, Thailand, April 7-9, Proceedings, Part I 6, Springer International Publishing, 73-82.

https://doi.org/10.1007/978-3-319-05476-6_8

[13] Le, T., Vo, B., & Baik, S. W. (2018). Efficient algorithms for mining top-rank-k erasable patterns using pruning strategies and the subsume concept. *Engineering Applications of Artificial Intelligence*, *68*, 1-9. https://doi.org/10.1016/j.engappai.2017.09.010

[14] Nguyen, G., Le, T., Vo, B., & Le, B. (2015). Discovering erasable closed patterns. *Intelligent Information and Database Systems: 7th Asian Conference*, ACIIDS 2015, Bali, Indonesia, March 23-25, Proceedings, Part I 7, Springer International Publishing, 368-376. https://doi.org/10.1007/978-3-319-15702-3_36

[15] Vo, B., Le, T., Nguyen, G., & Hong, T. P. (2017). Efficient algorithms for mining erasable closed patterns from product datasets. *IEEE Access*, *5*, 3111-3120. https://doi.org/10.1109/ACCESS.2017.2676803

[16] Lee, G., Yun, U., & Ryang, H. (2015). Mining weighted erasable patterns by using underestimated constraint-based pruning technique. *Journal of Intelligent & Fuzzy Systems*, *28*(3), 1145-1157. https://doi.org/10.3233/IFS-141398

[17] Lee, G., Yun, U., Ryang, H., Kim, D. (2016). Erasable itemset mining over incremental databases with weight conditions. *Engineering Applications of Artificial Intelligence*, *52*, 213-234. https://doi.org/10.1016/j.engappai.2016.03.003

[18] Nam, H., Yun, U., Yoon, E., & Lin, J. C. W. (2020). Efficient approach for incremental weighted erasable pattern mining with list structure. *Expert Systems with Applications*, *143*, 113087. https://doi.org/10.1016/j.eswa.2019.113087

[19] Nguyen, L., Nguyen, G., & Le, B. (2019). Fast algorithms for mining maximal erasable patterns. *Expert Systems with Applications*, *124*, 50-66. https://doi.org/10.1016/j.eswa.2019.01.034

[20] Agrawal, R. & Srikant, R. (1994). Fast algorithms for mining association rules. *Proc. 20th int. conf. very large data bases, VLDB*, *1215*, 487-499.

[21] Hong, T. P., Li, C. C., Wang, S. L., & Lin, C. W. (2018). Maintenance of Erasable Itemsets for Product Deletion. *Proceedings of the 5th Multidisciplinary International Social Networks Conference*, New York, NY, 1-4. https://doi.org/10.1145/3227696.3227718

[22] Hong, T. P., Li, C. C., Wang, S. L., & Lin, J. C. W. (2018). Reducing Database Scan in Maintaining Erasable Itemsets from Product Deletion. *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2627-2632. https://doi.org/10.1109/BigData.2018.8621965

[23] Cheung, D. W., Han, J., Ng, V. T., & Wong, C. Y. (1996). Maintenance of discovered association rules in large databases: An incremental updating technique. *Proceedings of the twelfth international conference on data engineering*, New Orleans, LA, US, 106-114. https://doi.org/10.1109/ICDE.1996.492094

[24] Baek, Y., Yun, U., Kim, H., Nam, H., Lee, G., Yoon, E., Vo. B., & Lin, J. C. W. (2020). Erasable pattern mining based on tree structures with damped window over data streams. *Engineering Applications of Artificial Intelligence*, *94*, 103735. https://doi.org/10.1016/j.engappai.2020.103735

[25] Baek, Y., Yun, U., Lin, J. C. W., Yoon, E., & Fujita, H. (2020). Efficiently mining erasable stream patterns for intelligent systems over uncertain data. *International Journal of Intelligent Systems*, *35*(11), 1699-1734. https://doi.org/10.1002/int.22269

[26] Hong, T. P., Chang, H., Li, S. M., & Tsai, Y.C. (2021). A dedicated temporal erasable-itemset mining algorithm. *International Conference on Intelligent Systems Design and Applications*, Cham: Springer International Publishing, 977-985. https://doi.org/10.1007/978-3-030-96308-8_91

[27] Nguyen, H. & Le, T. (2022). A fast algorithm for mining top-rank-k erasable closed patterns. *Computers, Materials & Continua*, *72*(2), 3571-3583, https://doi.org/10.32604/cmc.2022.024765

[28] Gouda, K. & Zaki, M. J. (2005). Genmax: An efficient algorithm for mining maximal frequent itemsets. *Data Mining and Knowledge Discovery*, *11*, 223-242. https://doi.org/10.1007/s10618-005-0002-x

Contact information:

Qunli ZHAO, Master, Lecturer
(Corresponding author)
School of Computer and Artificial Intelligence,
Hefei Normal University,
Hefei, China 230061
E-mail: zql@hfnu.edu.cn

Hesheng CHENG, Master, Lecturer
School of Computer and Artificial Intelligence,
Hefei Normal University,
Hefei, China 230061
E-mail: hesheng@hfnu.edu.cn

Chen SHEN, Doctor, Lecturer
School of Computer and Artificial Intelligence,
Hefei Normal University,
Hefei, China 230061
E-mail: sc@hfnu.edu.cn