IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 23, 2021, accepted July 21, 2021, date of publication July 27, 2021, date of current version August 4, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3100584

A Bitmap Approach for Mining Erasable Itemsets

TZUNG-PEI HONG[1,4], (Senior Member, IEEE), WEI-MING HUANG[2], GUO-CHENG LAN[3], MING-CHAO CHIANG[4], (Associate Member, IEEE), AND JERRY CHUN-WEI LIN[5], (Senior Member, IEEE)

[1]Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan
[2]Department of Electrical and Control, China Steel, Inc., Kaohsiung 806, Taiwan
[3]Data Analytics Team, D8AI, Inc., Taipei 114, Taiwan
[4]Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan
[5]Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Science, 5020 Bergen, Norway

Corresponding author: Wei-Ming Huang (granthill168@gmail.com)

ABSTRACT Erasable-itemset mining is a valuable method of pattern extraction for helping the manager of a factory analyze production planning. The erasable itemsets derived can be considered important production information regarding how to plan the production of a factory during an economic depression or financial shortage for the manager. After the erasable-itemset mining was proposed in 2009, several efficient mining approaches for finding erasable itemsets have been developed. However, these methods require a considerable amount of execution time when the amount of product data is large. Especially, manufacturing small amounts of versatile products has been a trend today, and it will generate a large product database. This paper adopts a bitmap representation for itemsets in an erasable-itemset mining algorithm to speed up the execution. Unlike the traditional bitmap meaning for frequent itemsets, a bitmap for erasable-itemset mining here denotes the relationship that a product includes at least one material (item) in a specified itemset. Using the bitmap representation can easily find the desired products to check, thus decreasing the scans of a database. Experimental evaluation on synthesized and real datasets was used to compare the proposed approach with the other two under different parameter values. The experimental results show that the proposed approach can make a good trade-off between execution time and memory usage.

INDEX TERMS Data mining, erasable itemset mining, bitmap representation, product database.

I. INTRODUCTION

Along with the increase of huge amounts of data, using effective methods to store and manage them is becoming remarkably critical. Powerful technology is thus developed to facilitate data processing and demands in large databases. Consequently, finding out useful implicit information to provide for decision-makers has been one of the most important tasks. Recently, the study of data mining is very popular because it can discover useful knowledge from large data sets in particular applications. For example, the relationship among product associations in a retail market can help managers make good promotion strategies. Many mining approaches based on the principle of downward-closure have thus been proposed for finding frequent itemsets and association rules [1], [2]. Besides, Han *et al.* proposed a tree-structure mining approach called FP-tree [12] to reduce database scans. Several extended mining problems, such as classification association rule [26], sequential patterns [15], utility mining [5], [24], [25], [29], temporal mining [3], fuzzy utility mining [6], [18], [28], erasable-itemset mining [8]–[10], [16], [22], [27], [30] among others were also proposed.

Erasable-itemset mining, first presented by Deng *et al.* [8] in 2009, aims to handle and analyze the production planning in a factory. Assume a variety of goods are produced by a factory, each of which uses some kinds of materials to make it. The factory thus needs to purchase all sorts of materials to satisfy the production goal, which will bring in the revenue by products sold. While the factory encounters a financial shortage, not all the materials could be bought, and therefore some products cannot be manufactured. It follows that the revenue of the factory may drop, or the factory may be forced to shut down.

Although the erasable-itemset mining problem was initially designed for factory manufacture, it may be easily extended to solving similar problems in other domains. For example, we may convert the erasable-itemset mining problem to a personal job allocation problem. Each job needs some persons with different kinds of expertise to finish and

The associate editor coordinating the review of this manuscript and approving it for publication was Michael Lyu.

has its profit. We can analyze the acceptable personnel combinations without affecting the gain much. Similarly, we can also use the erasable-itemset mining problem to decide the provision of some systems or services composed of some modules. Therefore, the idea of erasable mining can be used to determine the keeping or removal of objects with some components. Its applications are thus wide. Several methods were developed to solve the erasable-itemset mining problem [9], [10], [19]–[21].

In frequent-itemset mining, the *Apriori*-like methods find frequent itemsets level by level. The anti-monotone property is the primary basis in the *Apriori*-like algorithms. Therefore, frequent 1-itemsets are first found, and then the candidate 2-itemsets relying on the frequent 1-itemsets are produced by combining each pair of frequent items. These candidate 2-itemsets are then checked for whether they satisfy the given threshold or not. After the frequent 2-itemsets are mined, a similar procedure is done for itemsets with more than two items. According to the anti-monotonic property, any superset of an infrequent itemset is also infrequent. Pruning infrequent itemsets can reduce the search space. The process can be applied to erasable-itemset mining as well. The processing, however, needs a considerable amount of time.

Some tree structures were then proposed to reduce the execution time. They are more complex and need much memory to store the whole tree for mining. In this paper, we develop another approach based on the bitmap representation to reduce the processing time but not consume much memory space for erasable-itemset mining. The primary contributions of this paper include the following three points.

1) Bitmap representation was adopted to reduce the level-wise steps of the original erasable-itemset mining.
2) An algorithm is presented based on the bitmap representation as a fast algorithm for erasable-itemsets mining.
3) Several experiments were conducted on real and synthesized datasets to show the difference between the proposed and the other two methods. Experimental results show that the proposed algorithm performed well in execution time for real and synthetic datasets and can achieve a good trade-off between execution time and memory usage.

The remaining of the paper is organized as follows. The related works about erasable-itemset mining are reviewed in Section 2. The problem to be solved and some definitions are stated in Section 3. The execution details of the proposed approach are explained in Section 4. An example of how to perform the mining procedure is described in Section 5. The results of the performance evaluation of the proposed approach are shown in Section 6. Eventually, conclusions and future work are given in Section 7.

II. RELATED WORKS FOR ERASABLE ITEMSET MINING

Mining interesting rules and patterns is one of the significant knowledge discovery methods [1], [2]. In the past, the association-rule mining techniques were commonly used to achieve this goal since they could find high frequencies of purchased product combinations from a set of transactions. Agrawal and Srikant proposed a famous data-mining algorithm named the *Apriori* algorithm [1], which was first offered to cope with this problem. The algorithm makes use of the downward-closure property that can mine short-length frequent itemsets first and then find long-length ones. The *Apriori* algorithm then derives association rules from the frequent itemsets by conditional probability.

Different from frequent-itemset mining, erasable-itemset mining [8], proposed by Deng *et al.*, is helpful in production planning. Each product is made up of some kinds of material. Reducing stocks of raw material may cause the stoppage of the manufacture of some products, resulting in a reduction in the income of a factory. Erasable-itemset mining aims to find the kinds of material that could be removed with affecting the revenue only slightly.

In general, Deng's erasable-itemset mining algorithm [8] consists of two stages, generating candidate erasable itemsets and finding erasable itemsets that satisfy the loss ratio (represented as a threshold) of the factory's income. Different from the frequent-itemset mining, the condition to be satisfied in erasable-itemset mining is that the total revenue after an erasable itemset is removed must be equal to or less than the original revenue multiplied by the given threshold. The processing steps of finding erasable itemsets are still based on the concept of level-wise mining in the *Apriori* algorithm. All the single materials are defined as the set of candidate erasable 1-itemsets (CE_1), and then their individual gain values are found. The gain value of a material is to calculate the sum of the revenue of the products which apply the material in the production process. The erasable materials, called erasable 1-itemsets (E_1), whose gain values divided by the total revenue are equal to or less than the user-specified threshold, are found from the set of candidate erasable 1-itemsets (CE_1). Next, the candidate erasable 2-itemsets (CE_2) are generated from the erasable 1-itemsets (E_1). Thus we calculate the gain value of each candidate erasable 2-itemset in CE_2 from the database. The same as before, the erasable 2-itemsets (E_2), whose gain values divided by the original total revenue are equal to or less than the threshold, are found. The same process is done for itemsets with more than two materials until no candidate erasable itemsets are produced. Finally, all the erasable itemsets found are outputted to users as the auxiliary information in planning. Therefore, the key to the problem is finding inessential materials without losing the factory's income more than the factory managers wish.

In recent years, several algorithms were proposed to solve the erasable-itemset mining problem. For example, VME uses the PID_list structure to store the profits of all products for efficiently pruning irrelevant data and improving the processing time [10]. However, from experimental evaluation, the performance of VME is two times faster than that of the META [8] algorithm. MERIT uses a new data representation, called NC_set, to keep overall information for pruning unrelated data [9]. However, it does not perform well in getting erasable itemsets with long lengths.

dMERIT+, proposed by Le *et al.*, is an improved version of MERIT, which speeds up the processing by using the weight index, hash table, and dNC-Sets [20]. MEI uses a divide-and-conquer strategy and the dPidset data structure to reduce execution time and memory consumption [19]. It is capable of mining erasable-itemsets with larger thresholds. WEP (Weighted Erasable Patterns) [21] and WEPS (Weighted Erasable Pattern mining algorithm on Sliding window-based data streams) [30] consider different weights of materials in erasable-itemset mining. Some methods for handling incremental data in erasable pattern mining were also proposed [13], [14], [22].

There were several bitmap algorithms for frequent itemsets [4], [7], [23]. This paper, however, adopts a different processing strategy to reduce the execution time of erasable-itemset mining.

III. PROBLEM STATEMENT AND DEFINITIONS

Here, the erasable-itemset mining problem to be solved by the proposed bitmap algorithm is formally defined. Assume there is a product database (*PD*) with eleven products and eight materials, as shown in Table 1. Each product comprises three features: product ID (*PID*), materials, and profit. The eight distinct elements are expressed as *A* to *H,* respectively.

A set of terms is defined as follows.

Definition 1: A product database *PD* is composed of a set of products. That is, $PD = \{P_1, P_2, \ldots, P_i, \ldots, P_z\}$, where P_i is the i-th product in *PD* and z is the number of products.

Definition 2: A material set $M = \{m_1, m_2, \ldots, m_i, \ldots, m_n\}$ is the set of distinct materials used to manufacture products *PD*, where m_i represents the i-th material in M and n expresses the number of materials. Any material m_i can only occur once in a product but can appear in different products.

For example, in Table 1, M includes all the materials *A* to *H*.

Definition 3: The profit value, $profit_i$, is the profit of a product P_i in *PD*.

For instance, the profit value $profit_4$ of the product P_4 in Table 1 is 150. P_4 is made of three items, *B*, *C* and *E*, and the factory earns 150 dollars from the production of the product.

Definition 4: An itemset X is a collection of one or more materials. Namely, $X \subseteq M$. If $|X| = r$, the itemset X is called an r-itemset.

For instance, in Table 1, the itemset $\{BCE\}$ in P_4 contains 3 materials and can be treated as a *3*-itemset.

The goal of the erasable-itemset mining [8] is to find unimportant itemsets which allow the tolerable loss to a factory if the materials in any one of the itemsets are lacking. Some definitions about obtaining erasable itemsets are described as follows.

Definition 5: The gain value of an itemset X is represented as:

$$gain\,(X) = \sum_{x \in X \& x \in P_i} profit_i,$$

TABLE 1. An example of a product database.

PID	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*Profit*
P_1	✓	✓	✓						2100
P_2	✓	✓							1000
P_3	✓		✓						1000
P_4		✓	✓		✓				150
P_5		✓			✓				50
P_6			✓		✓				100
P_7			✓	✓	✓	✓	✓		200
P_8				✓	✓	✓		✓	100
P_9				✓		✓			*50*
P_{10}		✓				✓		✓	*150*
P_{11}			✓			✓			*100*

where x represents an item (material) in both X and product P_i.

For instance, assume $X = \{A, B\}$. From Table 1, the six products P_1, P_2, P_3, P_4, P_5 and P_{10} include an item *A*, *B*, or both. Therefore, $gain(X) = profit_1 + profit_2 + profit_3 + profit_4 + profit_5 + profit_{10}$, which is 4450.

Definition 6: The total profit T of a product database (*PD*) is the sum of the profits of all the products in *PD*. That is, T is calculated as follows:

$$T = \sum_{P_i \in PD} profit_i.$$

For example, the total profit T of the product database in Table 1 is 5000.

Definition 7: Given a threshold ratio r and a product dataset *PD*, an itemset X, which satisfies $gain(X) \leq T \times r$ is called an erasable itemset.

For instance, in Table 1, let $r = 35\%$. Take the item (1-itemset) F to explain it. Item F appears in P_7, P_8, P_9, P_{10}, and P_{11}. Its gain is $200 + 100 + 50 + 150 + 100$, which is 600. Item F is an erasable itemset because its gain (600) is smaller than $5000 \times 35\%$ (= 1750). It means that item F is ignorable since its absence allows a tolerable loss to the factory. In this case, the factory will not manufacture products P_7, P_8, P_9, P_{10}, and P_{11} if item F is missed. On the contrary, take item *A* as another example. Its gain is 3100 and is not an erasable itemset since its gain is larger than $5000 \times 35\%$. For the 2-itemset $\{A, F\}$, the products with at least one of them include P_1, P_2, P_3, P_7, P_8, P_9, P_{10}, and P_{11}. Its gain is thus $2100 + 1000 + 1000 + 200 + 100 + 50 + 150 + 100$, which is 3700.

Definition 8: Given a product dataset *PD* and a threshold ratio r, erasable-itemset mining is to find all the erasable itemsets from *PD*.

The original erasable-itemset mining approach [8] needs multiple database scans to find erasable itemsets. In this paper, we handle the problem by using the bitmap approach for mining erasable itemsets. The formal definitions for the proposed method are described below.

A bitmap, which is a one-dimensional binary bit string, represents the relationship of a material used to manufacture products. The i-th bit value is 1 if the material exists in the i-th product and is 0 otherwise. Take material *A* and *B*

in Table 1 as an example. Material A is included in P_1, P_2, and P_3, and material B in P_1, P_2, P_4, P_5, and P_6. In the example, the bitmap length of each material is eleven because there are eleven products. The bitmap representation of A is (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0). Similarly, B is (1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0).

The original processing steps of mining erasable itemsets rely on the concept of the *Apriori*-like algorithm. The executing steps need to use erasable short-length itemsets to generate candidate long-length ones based on the monotonic property and then find out erasable long-length itemsets according to the revenue loss ratio. The bitmap representation can efficiently handle this situation. For the above example, assume $\{A\}$ and $\{B\}$ are two erasable 1-itemsets. According to the level-wise processing, the candidate erasable 2-itemset is found from them. In this example, one candidate 2-itemset is $\{A, B\}$. Along with the bitmap representation, the products including a candidate 2-itemset can easily be obtained by using a logical inclusive *OR* operation on the bitmaps of the components in the 2-itemset. For example, the bitmap of the candidate 2-itemset $\{A, B\}$ can be found by performing a logical inclusive *OR* operation on A (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0) and B (1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0). The result is (1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0). It represents $\{A, B\}$ will affect the manufacture of products P_1, P_2, P_3, P_4, P_5, and P_{10}. Based on the above concept, some terms about the bitmap representation are formally defined as follows.

Definition 9: A bitmap of a *1*-itemset X is a one-dimensional binary bit string, denoted $bitmap_X = (b_1^X, b_2^X, \ldots, b_i^X, \ldots, b_z^X)$, where z represents the number of products in PD. The value of each b_i^X is 1 if the item in X exists in product P_i and is 0 otherwise.

For instance, in Table 1, item A is included in P_1, P_2 and P_3, but not in the other products. Therefore, $bitmap_A$ of the 1-itemset $\{A\}$ is represented as (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0).

Definition 10: Let X be an itemset with $|X| >= 2$. The bitmap of X is defined as:

$$bitmap_X = (b_1^X, b_2^X, \ldots, b_i^X, \ldots, b_z^X) = (\overset{|X|}{\underset{j=1}{OR}} b_1^{X_j}, \overset{|X|}{\underset{j=1}{OR}} b_2^{X_j}, \ldots, \overset{|X|}{\underset{j=1}{OR}} b_i^{X_j}, \ldots, \overset{|X|}{\underset{j=1}{OR}} b_z^{X_j}),$$

where $|X|$ is the number of the items in X, $b_i^{X_j}$ is the i-th bit value of the j-th item in X, and $\overset{|X|}{\underset{j=1}{OR}} b_1^{X_j}$ performs the inclusive *OR* operation on the $b_i^{X_j}$ values of all the items in X.

For example, in Table 1, the bitmaps of the two items A and B are (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0) and (1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0), respectively. By the inclusive *OR* operation on the two bitmaps, the result of the 2-itemset $\{A, B\}$ can be found as (1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0).

Definition 11: The gain of a bitmap representation $bitmap_X$ is the sum of the profits of the products with the corresponding bit values being 1 in $bitmap_X$. That is,

$$gain(X) = \sum_{i=1}^{z} (b_i^X * profit_i),$$

where z is the number of the products in PD. The value of each b_i^X is 1 if the item in X exists in product P_i and $profit_i$ represents the profit of product P_i.

In the above example, the bitmap of $\{A, B\}$ is (1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0). Thus, $gain(\{A, B\}) = profit_1 + profit_2 + profit_3 + profit_4 + profit_5 + profit_{10} = 2100 + 1000 + 1000 + 150 + 50 + 150 = 4450$.

IV. THE PROPOSED BITMAP ALGORITHM FOR ERASABLE-ITEMSET MINING

Based on the above definitions, we propose the Bitmap-Representation Erasable Mining (*BREM*) algorithm to find erasable itemsets efficiently. Its goal is to shorten multiple database scans in erasable mining compared with the *META* method in level-wise processing [8]. The bitmap representation, as mentioned in Section III, is used to calculate the gain value of itemset rapidly without rescanning a database. The proposed approach can get the products for an itemset quickly, thus saving much execution time. Besides, it will derive the same erasable itemsets as the original erasable-itemset mining algorithm [8].

The proposed approach proceeds as follows. It first sets up the threshold and then finds the erasable 1-itemsets with their gains satisfying the threshold condition. The erasable 1-itemsets are used to build their bitmaps according to the number of products in the database. The candidate erasable itemsets in the next level are then generated and checked with the bitmaps of their sub-itemsets according to the downward-closure principle. The same procedure is repeated until no new candidate itemsets are formed. The proposed algorithm is described as follows.

The BREM algorithm for mining erasable itemsets:

INPUT: (1) A product database PD with n products, each of which consists of its product identification, material list and profit, and (2) A predefined minimum erasable-itemset mining threshold called *MinEIthreshold*.

OUTPUT: All the erasable itemsets (*EIs*) satisfying *MinEIthreshold*.

STEP 1. Calculate the total profit T of the products in PD as:

$$T = \sum_{P_i \in PD} profit_i,$$

where P_i is the i-th product and $profit_i$ is the profit of P_i.

STEP 2. Initially, set the erasable 1-itemset (E_1) table as empty, where each tuple comprises three fields: itemset, gain of the itemset, and bitmap of the itemset.

STEP 3. Carry out the following substeps for each item (material) m_j:

(a) Find the gain value of m_j in PD as:

$$gain(m_j) = \sum_{m_j \in P_i} profit_i;$$

(b) Build the bitmap of m_j as:

$$bitmap_j = (b_1^{m_j}, b_2^{m_j}, \ldots, b_k^{m_j}, \ldots, b_z^{m_j}),$$

where z represents the number of the products in PD. The value of each $b_k^{m_j}$ is 1 if m_j exists in P_k and is 0 otherwise.

(c) Check whether $gain(m_j)$ is equal to or smaller than $T * MinEIthreshold$. If it is, insert the item m_j, $gain(m_j)$, and $bitmap_j$ as a tuple into the E_1 table.

STEP 4. Set $r = 1$, where r represents the number of items in the current set of erasable itemsets to be processed.

STEP 5. Initially set the erasable $(r + 1)$-itemset $(E_{(r+1)})$ table as empty, with each tuple consisting of three fields: itemset, gain of the itemset, and bitmap of the itemset.

STEP 6. Generate the candidate erasable set $CE_{(r+1)}$ from the set E_r in a way similar to the *Apriori* algorithm. All the r-sub-itemsets of each candidate erasable $(r+1)$-itemset in $CE_{(r+1)}$ must exist in E_r.

STEP 7. Carry out the following substeps for each $(r+1)$-itemset X in $CE_{(r+1)}$:

(a) Build the bitmap of X by applying the union operator on the bitmaps of its any two r-sub-itemsets S_1 and S_2 from E_r as follows:

$$\begin{aligned} &bitmap_X \\ &= (b_1^X, b_2^X, \ldots, b_k^X, \ldots, b_z^X) \\ &= (b_1^{S_1} \text{ or } b_1^{S_2}, b_2^{S_1} \text{ or } b_2^{S_2}, \ldots, b_k^{S_1} \text{ or } b_k^{S_2}, \\ &\quad \ldots, b_z^{S_1} \text{ or } b_z^{S_2}), \end{aligned}$$

where $b_k^{S_1}$ and $b_k^{S_2}$ are the k-th bits in S_1 and S_2, respectively, and $b_k^{S_1}$ *or* $b_k^{S_2}$ performs the inclusive *OR* operation on $b_k^{S_1}$ and $b_k^{S_2}$.

(b) Calculate the gain value of X as:

$$gain\,(X) = \sum_{k=1}^{z} (b_k^X * profit_k),$$

where z is the number of products in PD.

(c) Check whether $gain\,(X)$ is equal to or smaller than $T * MinEIthreshold$. If it is, insert the itemset X, its gain value, and its bitmap as a tuple into the $E_{(r+1)}$ table.

STEP 8. If the set $E_{(r+1)}$ is null, then do STEP 9; Otherwise, set $r = r + 1$ and do STEPs 4 to 8.

STEP 9. Output the erasable itemsets in all the erasable tables as the desired results *EIs*.

V. AN EXAMPLE FOR THE PROPOSED BREM APPROACH

An example to describe how the proposed *BREM* approach can efficiently mine erasable itemsets from a product database is given in this section. Table 1 in Section 3 is used as an example. The erasable-itemset mining threshold is set at 10%. The mining procedure of the proposed *BREM* algorithm is stated as follows.

STEP 1: The total profit T of the product database is first calculated. According to Table 1, $T = \sum_{P_i \in PD} profit_i = 2100 + 1000 + 1000 + 150 + 50 + 100 + 200 + 100 + 50 + 150 + 100 = 5000$.

STEP 2: The erasable 1-itemset (E_1) table is initialized as empty. Each tuple in the table consists of three fields: itemset, gain of the itemset, and bitmap of the itemset.

STEP 3: Each erasable 1-itemset is found in this step and then inserted with its gain and bitmap into the E_1 table. Take material D as an example. Material D is included in the three products, P_7, P_8 and P_9. Thus, the bitmap of D is (0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0), and the gain of D is: $gain(D) = 200 + 100 + 50 = 350$. Since its value is smaller than or equal to T * $MinEIthreshold$ $(= 5000 * 10\% = 500)$, the tuple ($\{D\}$, 350, (0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)) is then added into the E_1 table. The other 1-itemsets are checked in the same way. The results are shown in Table 2.

TABLE 2. The E_1 table after STEP3.

1-itemset	*Gain of the itemset*	*Bitmap of the itemset*
D	350	(0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)
G	200	(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)
H	250	(0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0)

STEP 4: The variable r is initially set at 1, which is used to represent the number of items to be processed in the current set of erasable itemsets.

STEP 5: The erasable 2-itemset table, E_2, is set as empty with three fields: 2-itemset, gain of the itemset, and bitmap of the itemset.

STEP6: The candidate erasable 2-itemsets are generated in a way similar to the *Apriori* algorithm in association-rule mining. Three candidate erasable 2-itemsets are generated from E_1 as follows: $\{D, G\}$, $\{D, H\}$ and $\{G, H\}$, which are kept in the candidate erasable 2-itemsets table CE_2.

STEP 7: The erasable 2-itemsets are found in this step. Take the 2-itemset $\{D, G\}$ in the CE_2 table as an example. According to Table 3, the bitmaps of D and G are operated by the inclusive *OR* operation as follows:

(0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0) *or* (0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0) = (0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0).

Thus, the bitmap of the 2-itemset $\{D, G\}$ is (0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0), while $b_7(= 1)$, $b_8(= 1)$ and $b_9(= 1)$ represent each of the three products P_7, P_8 and P_9 includes at least one of the materials D and G. The gain of the 2-itemset $\{D, G\}$ is then calculated as: $profit_7 + profit_8 + profit_9 = 200 + 100 + 50 = 350$. Since the value is smaller than T * $MinEIthreshold$, the tuple ($\{D, G\}$, 350, (0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)) is inserted into the E_2 table. All the other itemsets can be obtained similarly, and Table 3 shows the results for all the erasable 2-itemsets.

STEP 8: Because the set E_2 is not empty, the value of r is then increased to 2. STEPs 4 to 8 are executed again to find the set of E_3. The results are shown in Table 4.

TABLE 3. The E_2 table after STEP 7.

2-itemset	Gain of the itemset	Bitmap of the itemset
{D, G}	350	(0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)
{D, H}	500	(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0}
{G, H}	450	(0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0)

TABLE 4. The E_3 after STEPs 4 to 8.

3-itemset	Gain of the itemset	Bitmap of the itemset
{D, G, H}	500	(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0)

Then, the variable r becomes 3, and STEPs 4 to 8 are executed again. No candidate erasable 4-itemset is generated, and the final E_4 table is empty. Therefore, Step 9 will be executed.

STEP 9: All the E_1, E_2 and E_3 tables are output as shown in Table 5.

TABLE 5. The final results of all the erasable itemsets, *EIs*.

Itemset	Gain of the itemset	Bitmap of the itemset
{D}	350	(0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)
{G}	200	(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)
{H}	250	(0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0)
{D, G}	350	(0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0)
{D, H}	500	(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0)
{G, H}	450	(0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0)
{D, G, H}	500	(0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0)

VI. PERFORMANCE EVALUATION

A. EXPERIMENTAL SCENARIO

To evaluate the performance of *BREM*, experiments were performed on a computer with an Intel Core i5-4590 CPU 3.3GHz and 16GB of RAM running Windows 7 OS environment. The original erasable-itemset mining (*META*) approach [8] and the vertical-formal-based algorithm [10] for mining erasable itemsets (*VME*) were also conducted for comparison. The proposed *BREM*, the original *META*, and the *VME* methods were implemented in J2SDK 1.8. Four real datasets used in the experiments were adapted as product datasets in a way similar to that in [19]. They included the datasets of Connect, Chess, Mushroom, and Foodmart, which were downloaded from [11], [31]. Each tuple was thought of as a product, and its profit was randomly generated according to a uniform distribution within 100 and 500. The attributes of all the datasets are shown in Table 6.

TABLE 6. The real datasets used in the experiments.

Dataset	Number of Products	Number of Materials
Connect	67557	130
Mushroom	8124	120
Chess	3196	76
Foodmart	4141	1559

The same simulation environments and experimental scripts were built for the *BREM*, the *META*, and the *VME* methods. The erasable itemsets derived by the three methods were always the same. Therefore, we compared efficiency on the execution time and peak memory consumption for the different conditions in Tables 6. We run the designed programs with the total mining time measured from executing the program to the termination.

B. EVALUATION OF EXECUTION TIME

We compared the runtime of *BREM*, *META*, and *VME* on the real databases in Table 6, and the results were observed to assess the efficiency of the three methods. The four real databases of Connect, Mushroom, Chess, and Foodmart in Table 6 were tested, with the threshold values varying. Figures 1 to 4 show the execution results of the three methods on the datasets for the different thresholds.

The execution time of *BREM* was less than that of the other methods on the real datasets. Besides, the execution time also increased along with the increase of the threshold value because more itemsets satisfied the larger threshold, and thus the processing time grew. For the highest thresholds in these figures, the difference of execution time for the three methods was obvious for the databases of Connect, Mushroom and Chess because there were more erasable itemsets found in each level.

From Figures 1 to 4, it could be observed that the proposed *BREM* method outperformed *META* and *VME* for nearly

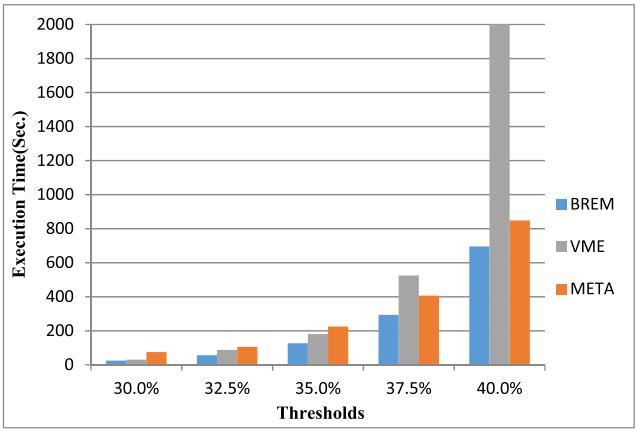

FIGURE 1. Execution time of the Chess dataset for the different thresholds.

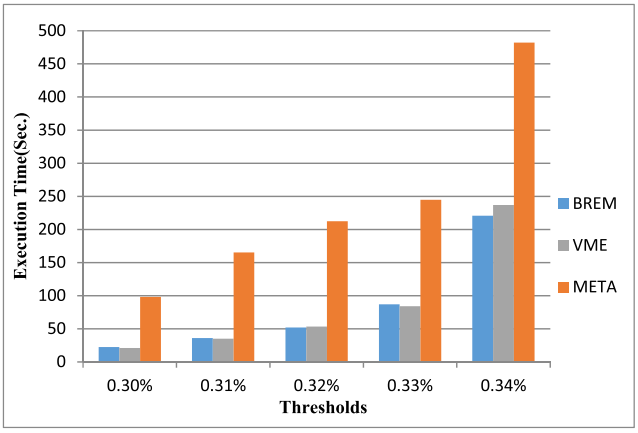

FIGURE 2. Execution time of the Foodmart dataset for the different thresholds.

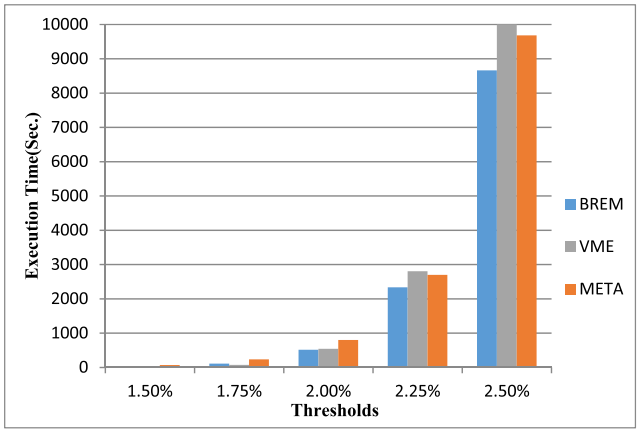

FIGURE 3. Execution time of the Mushroom dataset for the different thresholds.

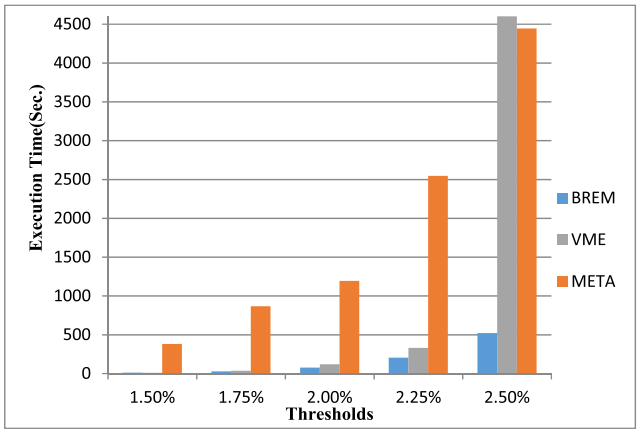

FIGURE 4. Execution time of the Connect dataset for the different thresholds.

all the thresholds in terms of the execution time for the four real databases. That is, the advantage of the proposed bitmap representation strategy, when compared with the other two methods, could simplify the mining process and thus shortened the mining time.

C. EVALUATION OF MEMORY USAGE

In this subsection, we show the memory usage of the three methods. First, the four databases of Connect, Mushroom, Chess and Foodmart were tested, with the threshold values varying. The peak of consumed memory was measured by finding erasable itemsets in each level. Figures 5 to 8 show the memory usage of the three methods for different thresholds.

From Figures 5 to 8, it was observed that the memory usage of *BREM* was more than that of *META* because the former needed to build and store bitmaps. However, in the memory usage of the three methods, the *VME* is the highest. Moreover, the memory usage increased along with the variety of the threshold values since more erasable itemsets resulted in more processing and memory requirements.

In summary, the memory usage of *BREM* was more than that of *META* for additionally keeping the bitmaps in different real datasets, but less than that of *VME* while the threshold value increased in different real datasets.

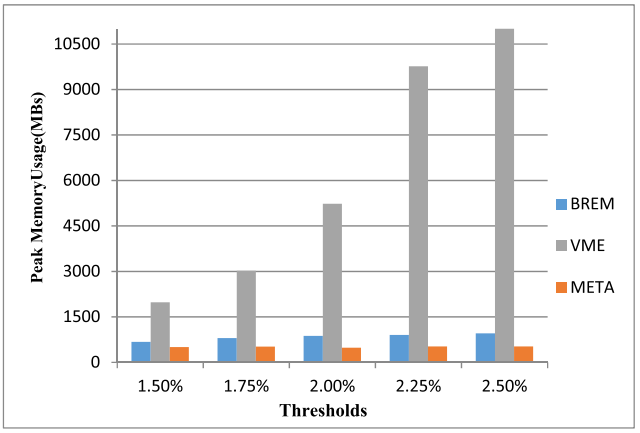

FIGURE 5. Memory usage of the Connect dataset for the different thresholds.

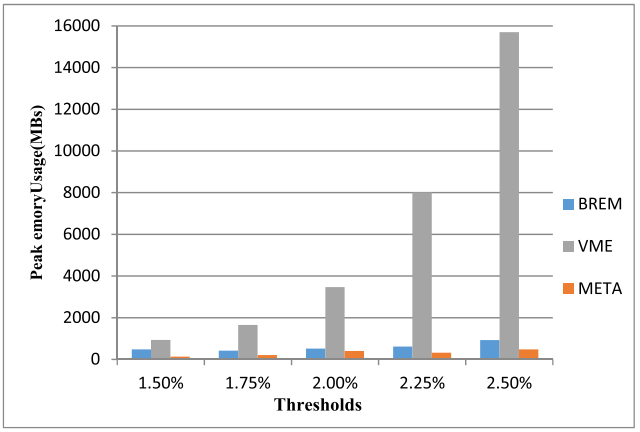

FIGURE 6. Memory usage of the Mushroom dataset for the different thresholds.

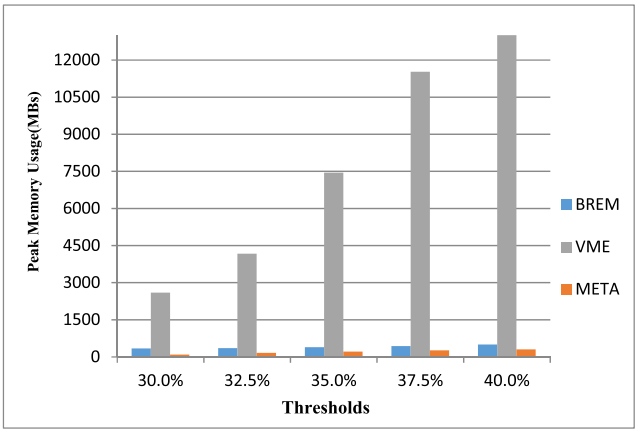

FIGURE 7. Memory usage of the Chess dataset for the different thresholds.

D. EVALUATION OF LARGE DATASETS

We then created three synthetic datasets (T10I4N4KD100K, T10I4N4KD300K, T10I4N4KD500K) based on the IBM data generator [17] and modified them to fit the problem of erasable-itemset mining with the corresponding parameters T (the average number of materials in each product), I (the size of a maximal potentially erasable itemset), N (the number of materials) and D (the number of products in a dataset). Each

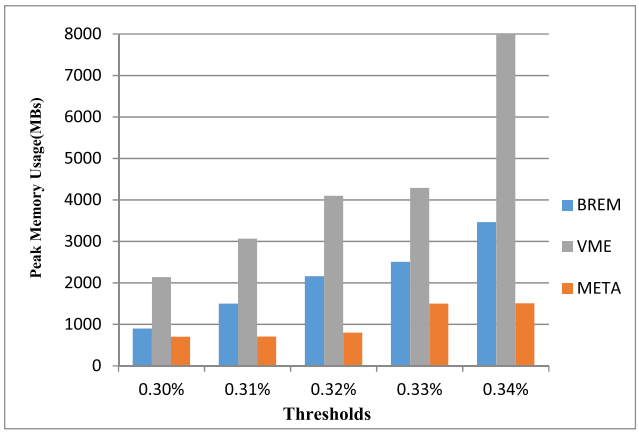

FIGURE 8. Memory usage of the Foodmart dataset for the different thresholds.

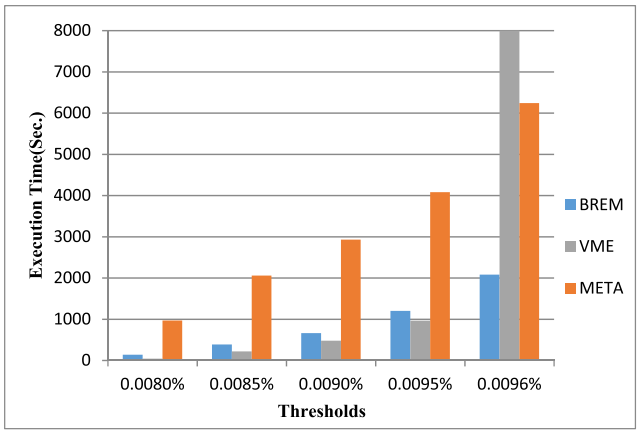

FIGURE 9. Execution time of the T10I4N4KD100K dataset for the different thresholds.

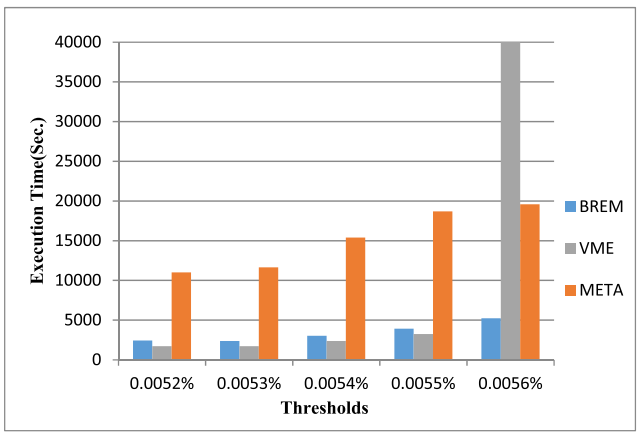

FIGURE 10. Execution time of the T10I4N4KD300K dataset for the different thresholds.

tuple was thought of as a product, and its profit was randomly generated according to a uniform distribution within 100 and 500.

The three synthetic databases were tested with the five different threshold values. Figures 9 to 11 show the execution results of the three methods for the different threshold values on three different datasets.

The execution time of the three methods grew along with the increase in the number of products. Especially, the

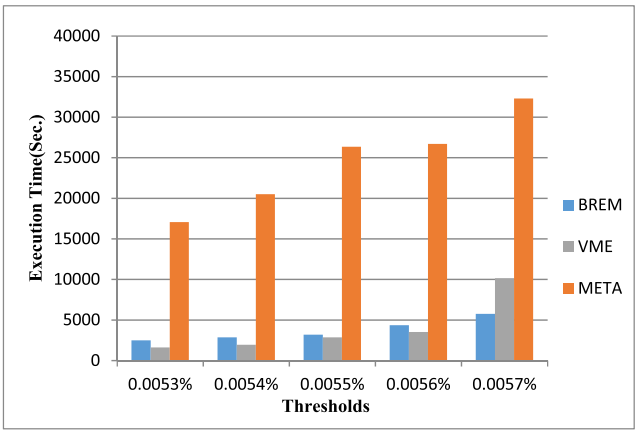

FIGURE 11. Execution time of the T10I4N4KD500K dataset for the different thresholds.

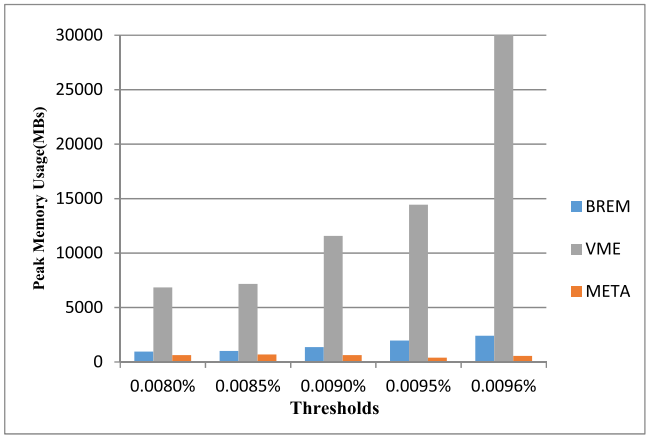

FIGURE 12. Memory usage of the T10I4N4KD100K dataset for the different thresholds.

execution time of the *META* method grew more when the database size increased since the increased number of products in the product database caused more processing time.

It could be observed that the execution time of *BREM* was less than that of *META* for five different threshold values on the three different datasets. However, *BREM* spent more than *VME* for the first four threshold values on the three different datasets. But for the fifth (largest) threshold value for the first two datasets, the processing time of *VME* was larger than *BERM* and *META*. This is because the method of *VME* needs to record the relationship of materials and products in the database.

Figures 12 to 15 show the memory usage of the three methods for the different threshold values on the three synthesized datasets.

The same as before, the memory usage of *BREM* was more than that of *META* for additionally keeping the bitmaps in the three datasets, but the memory usage of *BREM* was much less than that of *VME*.

Overall, the *BREM* method required more space than the *META* method on consuming memory, but the processing time of *BREM* was several times faster than that of *META*. However, the *VME* method consumed much memory space to store information, making the processing speed slower.

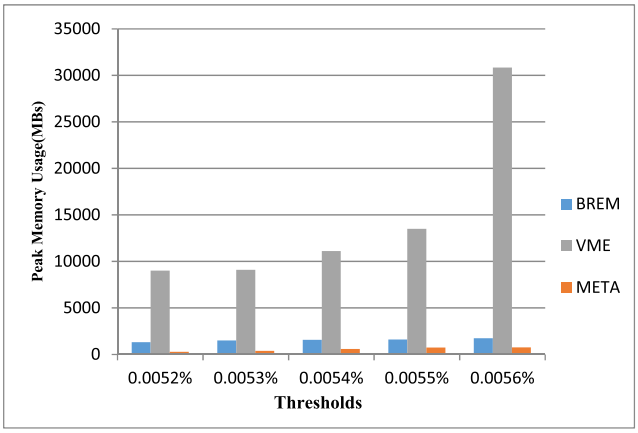

FIGURE 13. Memory usage of the T10I4N4KD300K dataset for the different thresholds.

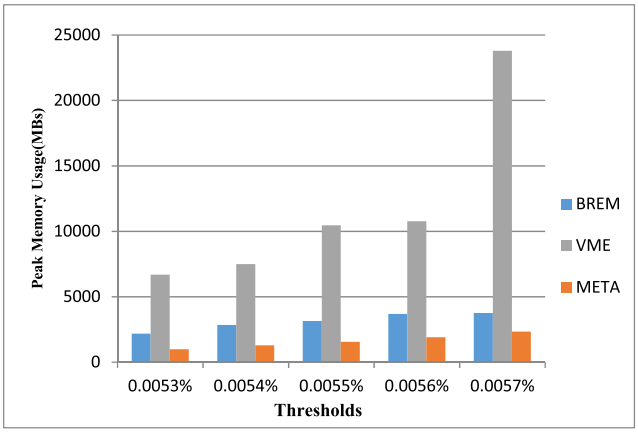

FIGURE 14. Memory usage of the T10I4N4KD500K dataset for the different thresholds.

VII. CONCLUSION

In this paper, the bitmap representation is used in the mining algorithm to find erasable itemsets. Especially, the process of determining erasable k-itemsets based on $(k\text{-}1)$-itemsets could be conducted quickly with the aid of the bitmap representation of itemsets. Some experiments, which pointed out the efficiency of *BREM*, were conducted regarding execution time and peak memory usage for real and simulated datasets. The proposed algorithm was also compared with the *META* and the *VME* algorithms. The execution time of *BREM* was less than that of *META* because the bitmap representation strategy could simplify the mining process and thus shorten the mining time. Besides, although *VME* may spend only a little less time than *BREM* for small thresholds, the former will increase dramatically for large thresholds. However, the memory usage of BREM was more than that of *META* because the former needed to build and store bitmaps for itemsets. The memory usage of *BREM* was much less than that of *VME*, especially in the highest thresholds, for the first three real datasets. In contrast to the simulated datasets, the memory usage of *BREM* was also much less than that of *VME,* but a little more than that of *META,* due to some extra memory for keeping the bitmaps. The *BREM* method could thus make a good trade-off between processing time and memory usage.

Although the proposed method can get good performance, some work needs to be developed in the future. For example, we may further solve the memory-consuming problem of *BREM*. Additionally, we will extend the proposed approach to handle the maintenance problem of erasable-itemset mining. We can also use the proposed approach to solve similar problems in other domains, such as the personnel or service combinations, to determine the keeping or removal of the targets.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases*, 1994, pp. 487–499.

[2] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large database," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1993, pp. 207–216.

[3] J. M. Ale and G. H. Rossi, "An approach to discovering temporal association rules," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2000, pp. 294–300.

[4] B. De Alwis, S. Malinga, K. Pradeeban, D. Weerasiri, and S. Perera, "Horizontal format data mining with extended bitmaps," in *Proc. Int. Conf. Soft Comput. Pattern Recognit.*, Dec. 2010, pp. 220–223.

[5] R. Chan, Q. Yang, and Y.-D. Shen, "Mining high utility itemsets," in *Proc. 3rd IEEE Int. Conf. Data Mining*, Nov. 2003, pp. 19–26.

[6] C.-H. Chen, A.-F. Li, and Y.-C. Lee, "Actionable high-coherent-utility fuzzy itemset mining," *Soft Comput.*, vol. 18, no. 12, pp. 2413–2424, Dec. 2014.

[7] J. Chen and K. Xiao, "BISC: A bitmap itemset support counting approach for efficient frequent itemset mining," *ACM Trans. Knowl. Discovery From Data*, vol. 4, no. 3, pp. 1–39, 2010.

[8] Z.-H. Deng, G.-D. Fang, Z.-H. Wang, and X.-R. Xu, "Mining erasable itemsets," in *Proc. 8th Int. Conf. Mach. Learn. Cybern.*, Jul. 2009, pp. 67–73.

[9] Z.-H. Deng and X.-R. Xu, "Fast mining erasable itemsets using NC_sets," *Expert Syst. Appl.*, vol. 39, no. 4, pp. 4453–4463, Mar. 2012.

[10] Z. H. Deng and X. R. Xu, "An efficient algorithm for mining erasable itemsets," in *Proc. 6th Int. Conf. Adv. Data Mining Appl.*, 2010, pp. 214–225.

[11] *Frequent Itemsets Mining Dataset Repository*. Accessed: Feb. 2017. [Online]. Available: http://fimi.cs.helsinki.fi/data/

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

[13] T.-P. Hong, L.-H. Chen, S.-L. Wang, C.-W. Lin, and B. Vo, "Quasi-erasable itemset mining," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 1816–1820.

[14] T.-P. Hong, K.-Y. Lin, C.-W. Lin, and B. Vo, "An incremental mining algorithm for erasable itemsets," in *Proc. IEEE Int. Conf. Innov. Intell. Syst. Appl. (INISTA)*, Jul. 2017, pp. 286–289.

[15] B. Huynh, B. Vo, and V. Snasel, "An efficient parallel method for mining frequent closed sequential patterns," *IEEE Access*, vol. 5, pp. 17392–17402, 2017.

[16] B. Huynh and B. Vo, "An efficient method for mining erasable itemsets using multicore processor platform," *Complexity*, vol. 2018, pp. 1–9, Oct. 2018.

[17] (1996). *IBM Quest Data Mining Projection, Quest Synthetic Data Generation Code*. [Online]. Available: http://www.almaden.ibm.com/cs/quest/syndata.htm

[18] G.-C. Lan, T.-P. Hong, Y.-H. Lin, and S.-L. Wang, "Fuzzy utility mining with upper-bound measure," *Appl. Soft Comput.*, vol. 30, pp. 767–777, May 2015.

[19] T. Le and B. Vo, "MEI: An efficient algorithm for mining erasable itemsets," *Eng. Appl. Artif. Intell.*, vol. 27, pp. 155–166, Jan. 2014.

[20] T. Le, B. Vo, and F. Coenen, "An efficient algorithm for mining erasable itemsets using the difference of NC-sets," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 2270–2274.

[21] G. Lee, U. Yun, and H. Ryang, "Mining weighted erasable patterns by using underestimated constraint-based pruning technique," *J. Intell. Fuzzy Syst.*, vol. 28, no. 3, pp. 1145–1157, 2015.

[22] G. Lee and U. Yun, "Single-pass based efficient erasable pattern mining using list data structure on dynamic incremental databases," *Future Gener. Comput. Syst.*, vol. 80, pp. 12–28, Mar. 2018.
[23] H. Li and H. Chen, "Improve frequent closed itemsets mining over data stream with bitmap," in *Proc. 9th ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw., Parallel/Distrib. Comput.*, 2008, pp. 399–404.
[24] Y. Liu, W. K. Liao, and A. Choudhary, "A fast high utility itemsets mining algorithm," in *Proc. 1st Int. Workshop Utility-Based Data Mining*, 2005, pp. 90–99.
[25] Y. Liu, J. Li, W. K. Liao, A. Choudhary, and Y. Shi, "High utility itemsets mining," *Int. J. Inf. Technol. Decis. Making*, vol. 9, no. 6, pp. 905–934, 2010.
[26] L. T. T. Nguyen, B. Vo, H. S. Nguyen, and S. H. Nguyen, "Mining class association rules with synthesis constraints," in *Proc. Asian Conf. Intell. Inf. Database Syst.*, 2017, pp. 556–565.
[27] B. Vo, T. Le, G. Nguyen, and T.-P. Hong, "Efficient algorithms for mining erasable closed patterns from product datasets," *IEEE Access*, vol. 5, pp. 3111–3120, 2017.
[28] C.-M. Wang, S.-H. Chen, and Y.-F. Huang, "A fuzzy approach for mining high utility quantitative itemsets," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Aug. 2009, pp. 1909–1913.
[29] S. J. Yen and Y. S. Lee, "Mining high utility quantitative association rules," in *Proc. 9th Int. Conf. Data Warehousing Knowl. Discovery*, 2007, pp. 283–292.
[30] U. Yun and G. Lee, "Sliding window based weighted erasable stream pattern mining for stream data applications," *Future Gener. Comput. Syst.*, vol. 59, pp. 1–20, Jun. 2016.
[31] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, "SPMF: A Java open-source pattern mining library," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3389–3393, 2014.

TZUNG-PEI HONG (Senior Member, IEEE) received the B.S. degree in chemical engineering from the National Taiwan University, in 1985, and the Ph.D. degree in computer science and information engineering from the National Chiao-Tung University, in 1992. He worked with the Department of Computer Science, Chung-Hua Polytechnic Institute, from 1992 to 1994, and with the Department of Information Management, I-Shou University, from 1994 to 2001. He was in charge of the whole computerization and library planning with the National University of Kaohsiung, Taiwan, in preparation, from 1997 to 2000, where he served as the First Director of the Library and Computer Center, from 2000 to 2001, as the Dean of Academic Affairs, from 2003 to 2006, as the Administrative Vice President, from 2007 to 2008, and as the Academic Vice President, in 2010. He is currently a Distinguished and Chair Professor with the Department of Computer Science and Information Engineering and with the Department of Electrical Engineering, and the Director of AI Research Center, National University of Kaohsiung. He is also a Joint Professor with the Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan. He received the first national flexible wage award from the Ministry of Education, Taiwan. He has published more than 600 research articles in international/national journals and conferences and has planned more than 50 information systems. His current research interests include knowledge engineering, data mining, soft computing, management information systems, and WWW applications. He is also a Board Member of more than 40 journals and the program committee member of more than one thousand conferences.

WEI-MING HUANG received the B.S. degree from the Department of Computer Science and Information Engineering, Da-Yen University, Yuanlin, Taiwan, in 2001, the M.S. degree from the Department of Computer and Communication Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, in 2003, and the Ph.D. degree from the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, in 2019. He has been working with the Department of Electrical and Control, China Steel, Taiwan, since 2005. His research interest includes data mining.

GUO-CHENG LAN received the B.S. and M.S. degrees from the Department of Information Management, Southern Taiwan University, Tainan, Taiwan, in 2004 and 2006, respectively, and the Ph.D. degree in computer science and information engineering from the National Cheng Kung University, Taiwan, in 2012. He is a member of Taiwanese Association for Artificial Intelligence (TAAI). He is currently a Senior Data Scientist and a Research and Development Manager with the Data Analytics Team, D8AI, Inc., Taiwan. His current research interests include data mining, machine learning, medical data mining, soft computing, ontology knowledge, fuzzy theory, and WWW applications.

MING-CHAO CHIANG (Associate Member, IEEE) received the B.S. degree in management science from the National Chiao Tung University, Hsinchu, Taiwan, in 1978, and the M.S., M.Phil., and Ph.D. degrees in computer science from Columbia University, New York, NY, USA, in 1991, 1998, and 1998, respectively. He has over 12 years of experience in the software industry encompassing a wide variety of roles and responsibilities in large and start-up companies, before joining the Faculty of the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, in 2003, where he is currently a Professor. His research interests include image processing, evolutionary computation, and system software.

JERRY CHUN-WEI LIN (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, in 2010. He is currently a Full Professor with the Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway. He has published more than 400 research articles in refereed journals [IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), IEEE TRANSACTIONS ON CYBERNETICS (TCYB), IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS (TII), IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS (TITS), IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING (TNSE), IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE (TETCI), IEEE SYSTEMS JOURNAL, IEEE SENSORS JOURNAL, IEEE INTERNET OF THING JOURNAL (IOTJ), ACM TRANSACTIONS ON KNOWLEDGE DISCOVERY FROM DATA (TKDD), ACM TRANSACTIONS ON DATA SCIENCE (TDS), ACM TRANSACTIONS ON MANAGEMENT INFORMATION SYSTEMS (TMIS), ACM TRANSACTIONS ON INTERNET TECHNOLOGY (TOIT), and ACM TRANSACTIONS ON INTELLIGENT SYSTEMS AND TECHNOLOGY (TIST)], and international conferences (IEEE ICDE, IEEE ICDM, PKDD, and PAKDD), 12 edited books, as well as 33 patents (held and filed, 3 U.S. patents). His research interests include data mining, soft computing, artificial intelligence/machine learning, and privacy preserving and security technologies. He is the Editor-in-Chief of the *International Journal of Data Science and Pattern Recognition* and the Guest Editor/Associate Editor for several IEEE/ACM journals such as IEEE TRANSACTIONS ON FUZZY SYSTEMS (TFS), IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS (TII), ACM TRANSACTIONS ON MANAGEMENT INFORMATION SYSTEMS (TMIS), ACM TRANSACTIONS ON INTERNET TECHNOLOGY (TOIT), and IEEE ACCESS. He has recognized as the most cited Chinese Researcher in 2018, 2019, and 2020 by Scopus/Elsevier. He is the Fellow of IET (FIET) and a Senior Member of the ACM.