Information Sciences 180 (2010) 4263–4289

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Toward boosting distributed association rule mining by data de-clustering

Frank S.C. Tseng [a], Yen-Hung Kuo [b], Yueh-Min Huang [c,*]

[a] Department of Information Management, National Kaohsiung First University of Science and Technology, No. 1, University Road, YenChao, 824 Kaohsiung County, Taiwan, ROC

[b] Innovative DigiTech-Enabled Applications and Services Institute, Institute for Information Industry, 8F., No. 133, Sec. 4, Minsheng East Road, Taipei City 105, Taiwan, ROC

[c] Department of Engineering Science, National Cheng Kung University, No. 1, University Road, Tainan 701, Taiwan, ROC

ARTICLE INFO

Article history:
Received 11 July 2008
Received in revised form 14 June 2010
Accepted 22 July 2010

Keywords:
Association rule mining
De-clustering
Distributed association rule mining
Gray code
Sampling
Shortest spanning path

ABSTRACT

Existing parallel algorithms for association rule mining have a large inter-site communication cost or require a large amount of space to maintain the local support counts of a large number of candidate sets. This study proposes a de-clustering approach for distributed architectures, which eliminates the inter-site communication cost, for most of the influential association rule mining algorithms. To de-cluster the database into similar partitions, an efficient algorithm is developed to approximate the shortest spanning path (SSP) to link transaction data together. The SSP obtained is then used to evenly de-cluster the transaction data into subgroups. The proposed approach guarantees that all subgroups are similar to each other and to the original group. Experiment results show that data size and the number of items are the only two factors that determine the performance of de-clustering. Additionally, based on the approach, most of the influential association rule mining algorithms can be implemented in a distributed architecture to obtain a drastic increase in speed without losing any frequent itemsets. Furthermore, the data distribution in each de-clustered participant is almost the same as that of a single site, which implies that the proposed approach can be regarded as a sampling method for distributed association rule mining. Finally, the experiment results prove that the original inadequate mining results can be improved to an almost perfect level.

1. Introduction

Most enterprises collect huge amounts of business data from daily transactions and store them in a database; the volume of such data is expected to grow considerably in the future. To effectively utilize this data, it is necessary to derive valuable patterns that can guide marketing strategies and management policies. Data mining technologies have thus been widely adopted to explore previously untapped knowledge to support decision making. Generally speaking, the kinds of knowledge that can be mined in a database can be categorized as follows [23]: (1) characterization and discrimination of concepts and classes; (2) frequent patterns, associations, and correlations; (3) predication rules obtained using classification analysis; and (4) data clustering and outlier detection obtained using cluster analysis among the various data mining categories, association rule discovery from vast amounts of business transaction data has been extensively studied in the last decade [2–4,8,18,22,23,25–27,30,32,42,43,48,49,52], and has gained considerable attention in the database community [1,24,38].

In this study, the mining problem of discovering frequent patterns and association rules from extremely large databases is considered. Since there are many ways of obtaining data, the amount of data and number of dimensions are continuously

* Corresponding author. Tel.: +886 6 2757575x63336.
E-mail addresses: imfrank@ccms.nkfust.edu.tw (F.S.C. Tseng), keh@iii.org.tw (Y.-H. Kuo), huang@mail.ncku.edu.tw (Y.-M. Huang).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.07.020

increasing. Moreover, due to the enormous target data volume, most of the related algorithms suffer from in finding all the candidates that fit the subjective conditions, which is a time-consuming process. Therefore, it is believed that the most effective way to improve the performance of mining frequent patterns is to develop distributed algorithms by parallelizing database scanning and the required disk I/O operations. To address this problem, several notable distributed architectures and algorithms have been proposed [5,14–17,28,35,39,40,50]. These approaches overcome the data mining issues related to database scanning, candidate itemset generation, and disk I/O operations. However, they still suffer from the inter-site communication overhead during the itemset generation process. When the data is skewed, the performance may deteriorate significantly. This motivated us to develop a distributed algorithm that de-clusters the problem into subtasks and dispatches them to all participating sites. The purpose is to eliminate inter-site communications after distributing all subtasks. That is, communications occur only at the very beginning and at the end of the process. As a result, there is no communication cost during the itemset generation process, which makes the proposed framework very efficient and effective.

The proposed method is based on the concept of data de-clustering adopted from [19]. Based on this approach, all the transaction data is de-clustered into subgroups, such that all subgroups are similar to each other and to the original group. The subgroups are then assigned to the participating sites using a round-robin approach. Finally, the complete frequent itemsets can be obtained by directly collecting all the locally obtained frequent itemsets.

Extensive experiments were conducted to evaluate the effectiveness of the proposed approach. The objective was to determine the performance and stability of the method when utilizing various data sizes, support thresholds, data complexities, and data dimensions. The distributed mining results show that the proposed approach improves speed without losing any frequent itemsets. Moreover, the data distribution in each de-clustered participant is almost the same as that of a single site, which implies that the proposed approach can be considered as a sampling method for distributed association rule mining. Based on comparisons, the proposed approach outperforms the random sampling based approach for various experimental parameter settings. Finally, two refining processes were conducted to show the potential of the improvement in effectiveness.

Based on a literature survey, the work proposed in this paper is the first distributed approach that utilizes data de-clustering [19] to parallelize the association mining task. The merit of this method is that there is no communication cost during frequent itemset generation. The method guarantees that all distributed partitions contain no skewed or imbalanced cases, and that any partition is similar to the others and the original database. This enables any conventional association mining algorithm to derive high quality approximate patterns from a single partition. Furthermore, all frequent itemsets can be obtained by simply collecting all the generations of the participating sites.

The rest of this article is organized as follows. Section 2 surveys works related to distributed data mining and data de-clustering. Section 3 presents the proposed approach in detail. Section 4 shows the experimental results, which are discussed in Section 5. Finally, conclusions and suggestions for future research are given in Section 6.

2. Related works

2.1. Distributed association rule mining

In order to alleviate the time-consuming process of finding frequent itemsets that satisfy a given support threshold, the distributed association mining problem has been widely studied [5,14–17,28,35,39,40,50]. Two survey articles have discussed these studies from an overall viewpoint [37,51]. In this subsection, some influential works which have contributed to the field of distributed association mining are introduced.

To parallelize the Apriori algorithm [6], Agrawal and Shafer proposed the CD (count distribution) method to solve the problem of association mining in a parallel manner [5]. Since the disk searching space in each participating site of a distributed approach has already been reduced, the focus of CD is on minimizing the number of inter-site communications. To achieve this goal, each site is responsible for computing the local support counts of all the candidate itemsets. If the local support count of a candidate itemset satisfies the predefined support threshold, then it is called a local frequent itemset. By exchanging the local support counts, all sites compute the global support counts to derive the target global frequent itemsets. CD parallelizes the disk I/O of the Apriori algorithm. The communication complexity of CD is $O(|C|n)$, where C is the group of all candidate itemsets generated by Apriori, and n is the number of sites. However, this approach requires a large amount of space to maintain the local support counts of candidate sets.

Based on the CD approach, Cheung et al. proposed the FDM (fast distributed mining) [14] algorithm and a FDM's expanded work named DMA (distributed mining of association rules) [15]. The underlying idea of FDM is that every globally large itemset must be locally large at some site(s). By this principle, FDM adopts two effective techniques, namely local pruning and global pruning, to reduce the number of candidates produced in each iteration. Furthermore, FDM also utilizes count polling to optimize the inter-site communication cost. Through these techniques, FDM further improves the communication complexity to $O(Pr_{\text{potential}}|C|n)$, where $Pr_{\text{potential}}$ is the probability that an itemset is locally frequent in any site. This means that $Pr_{\text{potential}}$ depends on n; FDM is thus neither scalable nor efficient when n is large. FDM was further enhanced into FPM (fast parallel mining) by Cheung and Xiao [16]. FPM requires fewer rounds of message exchanging than does FDM, and provides the scalability to handle any size of candidate set while maintaining the benefit of effective pruning. Nevertheless, FPM is sensitive to low balance and skewness, for which it produces results that are only slightly better than those obtained using CD.

To handle the skewness of datasets and to improve the communication complexity of FDM, Schuster and Wolff proposed the DDM (distributed decision miner) [39] algorithm. DDM utilizes the private hypothesis P on each site and the common hypothesis H on all sites to verify whether an itemset is frequent before broadcasting it. If the two hypotheses disagree on whether an unexpressed itemset is frequent in one site, then the site broadcasts the frequency of the itemset. The sites listen to the broadcast messages instead of exchanging their local frequency information. Meanwhile, they continuously maintain their own local frequencies using incoming messages until the two hypotheses of all sites agree on all local verifications. Finally, all itemsets judged by both H and P as frequent in each site are broadcasted to generate the global frequent itemsets. DDM is efficient even when the database is skewed or the partition sizes are imbalanced. Furthermore, DDM reduces the communication complexity to $O(Pr_{\text{above}}|C|n)$, where Pr_{above} is the probability that an itemset is locally frequent in a certain partition. Pr_{above} is smaller than or equal to $Pr_{\text{potential}}$ of FDM. Unlike FDM, DDM is still efficient even when n is large, as Pr_{above} does not depend on n.

Schuster et al. further developed a hybrid distributed association mining algorithm, namely D-Sampling [40], which combines a sampling-based association mining algorithm called Sampling [44] and the DDM algorithm to mine associations in parallel. D-Sampling adopts a modified DDM to mine the sampled dataset of a partition that uses a lower support threshold in each site until the partition is consumed. It subsequently collects all locally frequent itemsets generated in each iteration as the negative border. Finally, the actual threshold is utilized to generate the global frequent itemsets. In addition to inheriting all the advantages of DDM, D-Sampling is the first distributed algorithm that only scans the database once to perform the association mining task. It also has a lower communication cost than that of DDM. The only drawback of D-Sampling is that it has the potential of falling into failure mode. Nevertheless, such situations are rare, and can be overcome with further checking.

Otey et al. [35] presented a ZigZag-based algorithm to mine frequent itemsets. They proposed parallel and distributed methods, and offered an incremental scheme to find the frequent itemsets. The ZigZag-based approach aims to find and continuously maintain all local maximal frequent itemsets (a maximal frequent itemset is not a subset of any other frequent itemset) to reduce the communication cost. Since the local maximal frequent itemsets can be used to generate all possible local frequent itemsets, all global frequent itemsets can be derived from the local maximal frequent itemsets. After exchanging the local maximal frequent itemsets, each site obtains the potential global frequent itemsets, and then calculates the counts of itemsets from the local dataset. Finally, all sites exchange their support counts in a straightforward fashion to produce the global frequent itemsets. The ZigZag-based approach only requires two rounds of communication to generate global frequent itemsets when a query is made.

2.2. Data de-clustering

The concept of de-clustering [7,11–13,19,20,34,45–47] is opposite to that of clustering. From the definition in [19], for a set of data S, clustering produces n groups $S_1, S_2, \ldots, S_n$, such that all groups $S_1, S_2, \ldots, S_n$ are different from each other. In contrast, for a set of data S, de-clustering divides S into n groups $G_1, G_2, \ldots, G_n$, such that all groups $G_1, G_2, \ldots, G_n$ are similar to the original set S, and that $G_1, G_2, \ldots, G_n$ are similar to each other.

In [19], the shortest spanning path (SSP) method was proposed to de-cluster a set of data, in which the original group is assumed to be divided into two subgroups, which are similar to each other. For a set of transactions, the shortest spanning path is constructed with a unique ordering of nodes [29]. The nodes are then labeled according to their positions on the path. Finally, the nodes with odd numbers are put into one subgroup and nodes with even numbers are put into another. The SSP approach divides the number of nodes in the original group evenly, and guarantees that the difference between the number of nodes in each subgroup is one at the most [19]. Although finding an SSP is generally NP-hard [21,36], in the present paper, a polynomial approximation algorithm is presented for constructing a spanning path for a set of transactions $D = \{T_1, T_2, \ldots, T_n\}$ with the summation of the weights of the affinity of all the consecutive neighbors being nearly minimum. In Section 4, experiment results show the de-clustering performance of the approximate SSP algorithm.

3. Data de-clustering

The architecture of the proposed approach is depicted in Fig. 1. The architecture assumes a shared-nothing distributed environment, where each participant has a private memory and its own disk. The de-clustering process first loads the source dataset into the data de-clustering system, and then sends the de-clustered datasets to all distribution sites. The process has two main stages:

(1) *Data transformation.* In the first stage, the set of transactions $D = \{T_1, T_2, \ldots, T_n\}$ is transformed and represented as a set of patterns $G = \{p_1, p_2, \ldots, p_n\}$, where each pattern p_i is represented in binary format.
(2) *Data de-clustering.* In the second stage, suppose that there are m participants; then, SSP is used to de-cluster the set of transaction patterns $G = \{p_1, p_2, \ldots, p_n\}$ into m groups $G_1, G_2, \ldots, G_m$, which satisfy the following conditions:
   (i) All groups $G_1, G_2, \ldots, G_m$ are similar to the original set G.
   (ii) $G_1, G_2, \ldots, G_m$ are similar to each other.

After the original set of transactions is de-clustered into subgroups, they are copied from the original database to their corresponding participants.

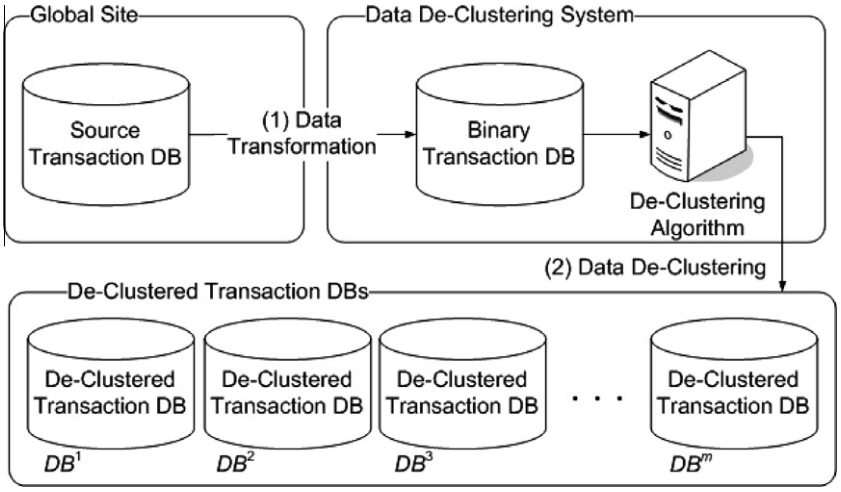


Fig. 1. Proposed system architecture. Note that the de-clustered transaction dataset $DB^i = (1/m)$ of the source transaction DB, and that the workload = $(1/m)$ of the original workload.

The two stages are described in detail in the following subsections.

3.1. Data transformation process

To reduce the data size, all transactions are transformed into their corresponding patterns, which are represented as bit strings. For example, let $\tau = \{A_{n-1}, A_{n-2}, \ldots, A_0\}$ be a set of n items, and suppose that there are k transactions in $D = \{T_1, T_2, \ldots, T_k\}$, where $T_1 = \{A_0, A_1\}$, $T_2 = \{A_1, A_2, A_3, \ldots, A_{n-1}\}$, …, and $T_k = \{A_0, A_3, A_{n-1}\}$. The data transformation process transforms them into a binary sparse matrix, as shown in Fig. 2.

3.2. Data de-clustering process

To de-cluster data records, according to Liu and Setiono [31], the nominal attribute values should be ordered and assigned with different weights to differentiate them. The weight of an item is defined in Definition 1.

Definition 1. For any item $A_i \in \tau = \{A_{n-1}, A_{n-2}, \ldots, A_0\}$, $0 \leqslant i \leqslant n-1$, the weight of A_i, $w(A_i)$, is defined as 2^i.

The weight of a transaction is defined in Definition 2.

Definition 2. For a transaction T with pattern $p = a_{n-1}a_{n-2}\cdots a_0$, $T \subseteq \tau$, the weight of T, denoted as $w(T)$, is defined as $\sum_{i=0}^{n-1} a_i \cdot w(A_i) = \sum_{i=0}^{n-1} a_i \cdot 2^i$.

To compute the difference between two transactions, the affinity between them and the corresponding weight are defined in Definitions 3 and 4, respectively.

Definition 3. For two transactions, T_i and $T_j \subseteq \tau$, the affinity between T_i and T_j, denoted as $aff(T_i, T_j)$, is defined as a bit string $p_i \oplus p_j = b_{n-1}b_{n-2}\cdots b_0$, where $\oplus$ represents the bit-wise exclusive-or operation.

Definition 4. For the affinity between T_i and T_j, $aff(T_i, T_j) = b_{n-1}\ b_{n-2}\cdots b_0$, the weight of $aff(T_i, T_j)$, denoted as $w(aff(T_i, T_j))$, is defined as $\sum_{i=0}^{n-1} b_i \cdot 2^i$.

ORIGINAL TRANSACTIONS

T_{id}	Items
T_1	$\{A_0, A_1\}$
T_2	$\{A_1, A_2, A_3, \cdots, A_{n-1}\}$
$\cdots$	$\cdots$
T_k	$\{A_0, A_3, A_{n-1}\}$

BINARY SPARSE MATRIX

T_{id}	A_0	A_1	A_2	A_3	$\cdots$	A_{n-1}
T_1	1	1	0	0	$\cdots$	0
T_2	0	1	1	1	$\cdots$	1
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
T_k	1	0	0	1	$\cdots$	1

Fig. 2. Example of data transformation process.

Based on the work presented in [31] and the definitions presented above, the following can be assumed:

(1) *Uniqueness of any transaction weight*. For any two transactions T_i and T_j containing different items, it is guaranteed that $w(T_i) \neq w(T_j)$.
(2) *Uniqueness of any affinity*. For any two transactions T_i and T_j containing different items, the weight of the affinity between T_i and T_j, i.e., $w(aff(T_i, T_j))$, is unique.

3.3. De-clustering using the approximate shortest spanning path approach

A shortest spanning path is defined as the shortest path connecting all of the data points. A divide-and-conquer algorithm is here developed to find an approximation of a shortest spanning path, a nearly shortest spanning path. The algorithm recursively divides the set of transaction patterns into two subsets, tries to find a local shortest spanning path for each subset, and finally merges them into a complete spanning path. The proposed algorithm regards the transaction patterns as data points in a graph, and uses the exclusive-or operation to define their distances (i.e., the affinity between two transactions). That is, the shortest spanning path can be obtained directly from the transaction patterns themselves. A shortest spanning path of a set of transactions is defined as follows.

Definition 5. A shortest spanning path $S = [T_{\pi(1)}, T_{\pi(2)}, \ldots, T_{\pi(i)}, \ldots, T_{\pi(n)}]$ of a set of transactions $D = \{T_1, T_2, \ldots, T_n\}$ is defined as a path connecting all transactions, such that $\sum_{i=1}^{n-1} w(aff(T_{\pi(i)}, T_{\pi(i+1)}))$ is the minimum, where π is a permutation of $\{1, 2, \ldots, n\}$.

From Definition 5, a shortest spanning path $S = [T_{\pi(1)}, T_{\pi(2)}, \ldots, T_{\pi(i)}, \ldots, T_{\pi(n)}]$ and its reverse path, denoted as $S^{-1} = [T_{\pi(n)}, T_{\pi(n-1)}, \ldots, T_{\pi(i)}, \ldots, T_{\pi(1)}]$, have the same cost, since $\sum_{i=1}^{n-1} w(aff(T_{\pi(i)}, T_{\pi(i+1)})) = \sum_{i=1}^{n-1} w(aff(T_{\pi(i+1)}, T_{\pi(i)}))$.

Additionally, from Definitions 3 and 4, the cost of a shortest spanning path of a set of transactions can be directly evaluated from the corresponding transaction patterns. That is, $\sum_{i=1}^{n-1} w(aff(T_{\pi(i)}, T_{\pi(i+1)})) = \sum_{i=1}^{n-1} p_{\pi(i)} \oplus p_{\pi(i+1)}$, where p_l is the pattern of transaction T_l.

To evaluate the cost of an SSP directly from the corresponding transaction patterns, it has been shown that the most efficient way is to arrange the transaction patterns into a hypercube according to the sequence conforming to Gray code. Gray code [33,41] is a binary numeral system where two successive values differ in only one bit. A binary Gray code with n digits corresponds to a Hamiltonian path [41] on an n-dimensional hypercube.

Fig. 3 shows a cube with a path (marked with solid arrows) representing the sequence conforming to Gray code. The number labeled between two patterns is the cost (the affinity or distance) between the two corresponding transactions. In such an n-dimensional hypercube, data points with the same leading k bits, $k > 0$, are considered to be on the same $(n - k)$-dimension. For example, 110, 111, 101, and 100 are all considered to be on the same two-dimension (the top plane in Fig. 3) and 000, 001, 011, and 010 are all considered to be on the same two-dimension (the bottom plane in Fig. 3).

The distance between any two points T_i and T_j in the hypercube can be obtained by directly accumulating the distances of points for any shortest path connecting T_i and T_j. For instance, in Fig. 3, the distance between 000 and 111, i.e. 7, can be evaluated directly by finding the shortest path (say, 000, 001, 101, and 111) connecting both points (i.e., 7 = 1 + 4 + 2).

In such n-dimensional hypercubes, the distance between two points located in the same dimension is always less than the distance between any two points located in different dimensions. For an n-dimensional hypercube, the distance between two points located in the same dimension is always less than $2^{n-1} - 1$ and the distance between two points located in different dimensions is always greater than or equal to 2^{n-1}.

This observation is the basic rationale of the proposed divide-and-conquer algorithm, which recursively partitions an n-dimensional hypercube into two $(n - 1)$-dimensional hypercubes, then tries to find the corresponding shortest spanning paths, and finally merges the results obtained into a complete path.

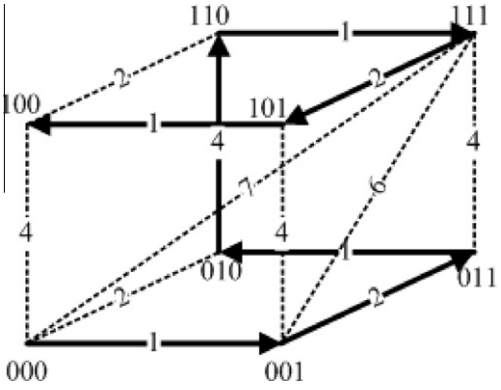


Fig. 3. SSP conforming to the sequence of Gray code.

In general, a divide-and-conquer algorithm usually consists of the following three phases:

(1) *When the problem size is small enough, directly solve the problem; otherwise, split the original problem into two sub-problems.* When there are only two transaction patterns, the SSP can be obtained by connecting them together and the cost can be evaluated using a bit-wise exclusive-or operation. Otherwise, the algorithm splits the transaction patterns based on the leftmost bit value. That is, transaction patterns whose leftmost bit value is zero form one group, and transaction patterns whose leftmost bit value is one form the other group.
(2) *Recursively solve the two sub-problems by applying the divide-and-conquer strategy.* Then, the recursive algorithm, called DIVIDE_AND_CONQUER, rapidly determines the split point by sorting the original input transaction patterns into an array named $T[n]$ and passing them into DIVIDE_AND_CONQUER as an input parameter.
(3) *Merge the solutions of the two sub-problems into a solution of the original problem.* Suppose that there are two paths to be merged, namely $[T_i,\ldots,T_j]$ and $[T_m,\ldots,T_n]$, and that both are already spanning paths with respect to their corresponding transaction patterns. In the merge stage, the distances of the pairs (T_i,T_m), (T_i,T_n), (T_j,T_m), and (T_j,T_n) are evaluated using bit-wise exclusive-or operations and the cases are grouped into the following four categories:
   (i) The distance between (T_i,T_m), i.e., $p_i \oplus p_m$, is the minimum. The path $[T_i,\ldots,T_j]$ is reversed into $[T_j,\ldots,T_i]$ to connect to the unchanged path $[T_m,\ldots,T_n]$ to form a new path $[T_j,\ldots,T_i,T_m,\ldots,T_n]$.
   (ii) The distance between (T_i,T_n), i.e., $p_i \oplus p_n$, is the minimum. Both paths $[T_i,\ldots,T_j]$ and $[T_m,\ldots,T_n]$ are reversed into $[T_j,\ldots,T_i]$ and $[T_n,\ldots,T_m]$, respectively, and then connected together to form a new path $[T_j,\ldots,T_i,T_n,\ldots,T_m]$.
   (iii) The distance of (T_j,T_m), i.e., $p_j \oplus p_m$, is the minimum. Both paths $[T_i,\ldots,T_j]$ and $[T_m,\ldots,T_n]$ are unchanged and connected to form a new path $[T_i,\ldots,T_j,T_m,\ldots,T_n]$.
   (iv) The distance of (T_j,T_n), i.e., $p_j \oplus p_n$, is the minimum. The path $[T_m,\ldots,T_n]$ is reversed into $[T_n,\ldots,T_m]$ and appended to the unchanged path $[T_i,\ldots,T_j]$ to form a new path $[T_i,\ldots,T_j,T_n,\ldots,T_m]$.

Since an SSP and its reverse have the same distance, the two shortest spanning paths ($[T_i,\ldots,T_j]$ and $[T_m,\ldots,T_n]$) can be merged by minimizing the distance of the two paths' contact point to obtain a new path with a distance smaller than or equal to the distance of the path $[T_i,\ldots,T_j,T_m,\ldots,T_n]$ generated by directly merging the two spanning paths without interlacing.

The aforementioned first two phases and the last phase are conducted using the algorithms DIVIDE_AND_CONQUER and MERGE, as shown in Figs. 4 and 5, respectively. In Fig. 4, the DIVIDE_AND_CONQUER algorithm recursively seeks the split point in an array

Algorithm: Finding an SSP by Divide-and-Conquer Strategy.
Input:
1. $T[n]$ – An array of sorted transactions by individual weight.
   We use $T[n][m]$ to denote the m-th bit of $T[n]$.
2. *head* – Start index point of array $T[n]$.
3. *tail* – End index point of array $T[n]$.
4. m – The leftmost bit index. That is, $T[n][m]$ is the leftmost bit.

Output:
1. A shortest spanning path in array $T[n]$.

```
Function Divide_And_Conquer(T[n], head, tail, m)
1)  if head ≥ tail then
2)        return T[n]
3)  if tail = head + 1 then
4)        return T[n]
5)  if m = 0 then   //boundary condition
6)        return T[n]
7)  if there exists a T[k] such that T[k][m] ≠ T[k + 1][m] then
8)        T[n] ← Divide_And_Conquer(T[n], head, k, m – 1)
9)        T[n] ← Divide_And_Conquer(T[n], k + 1, tail, m – 1)
10)       T[n] ← Merge(T[n], head, tail, k)
11) else
12)       /*
13)       The leftmost bit values of current elements in T[n] are all the same.
14)       Shift the bit index right to set the leftmost bit to the next bit.
15)       */
16)       T[n] ← Divide_And_Conquer(T[n], head, tail, m – 1)
17) return T[n]
```

Fig. 4. Algorithm for finding an SSP with the DIVIDE_AND_CONQUER strategy.

of sorted transactions ($T[n]$) according to the leftmost bit index (m) of transactions in $T[n]$ (step 16). If a split point index (k) can be found, $T[n]$ is split into two groups: transactions whose leftmost bit is zero form one group, and transactions whose

Algorithm: Merging the two shortest spanning paths divided by k.
Input:
1. $T[n]$ – An array of shorted transactions by individual weight.
2. i – The start index point in $T[n]$.
3. j – The end index point in $T[n]$.
4. k – The separate position in $T[n]$. That is, one of the spanning path is represented by $T[i]$ to $T[k]$, and the other spanning path is denoted by $T[k+1]$ to $T[j]$.

Output:
1. A shortest spanning path in array $T[n]$.

```
Function Merge(T[n], i, j, k)
1)  if i ≥ j then
2)        return T[n]
3)  //choose the minimum XOR result and execute the procedure.
4)  select case: (T[i] ⊕ T[k + 1]), (T[i] ⊕ T[j]), (T[k] ⊕ T[k + 1]), (T[k] ⊕ T[j])
5)        case 1: (T[i] ⊕ T[k + 1]) is the minimum
6)                T[n] ← Reverse(T[n], i, k)
7)        case 2: (T[i] ⊕ T[j]) is the minimum
8)                T[n] ← Reverse(T[n], i, k)
9)                T[n] ← Reverse(T[n], k + 1, j)
10)       case 3: (T[k] ⊕ T[k + 1]) is the minimum
11)               break
12)       case 4: (T[k] ⊕ T[j]) is the minimum
13)               T[n] ← Reverse(T[n], k + 1, j)
14) return T[n]
```

Fig. 5. Algorithm for merging the two shortest spanning paths divided by k.

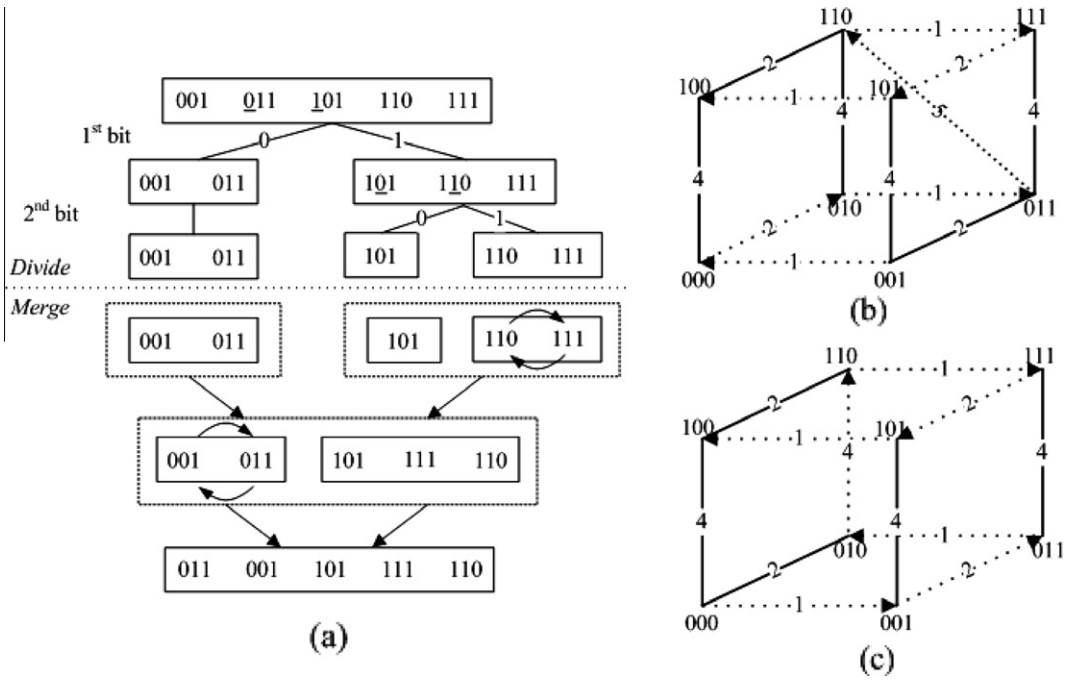


Fig. 6. Example illustrating proposed Divide_and_Conquer method. (a) Example illustrating the Divide_and_Conquer process. (b) Non-optimum spanning path (dotted arrows) may be constructed by the proposed Divide_and_Conquer algorithm. (c) Optimum spanning path (dotted arrows).

leftmost bit is one form the other group (step 7). The two groups are then treated as the input parameters of the DIVIDE_AND_Conquer algorithm (steps 8–9). Steps 7–9 are recursively executed until there exists only one or two transactions in a group (steps 1–4). A bottom-up merging process is then invoked to merge all subgroups using the merging strategy (as shown in phase 3) of the MERGE algorithm (step 10) to produce an approximate SSP in $T[n]$.

For the MERGE algorithm (see Fig. 5), if there are two transaction groups in $T[n]$, then they are merged into a new group using the merging strategy as discussed in phase 3 (steps 4–13). Otherwise, the group is returned immediately (steps 1–2). Note that the Reverse $(T[n],i,j)$ function, which is invoked by the MERGE algorithm, is used to reverse the transaction order between $T[i]$ and $T[j]$ in $T[n]$. For instance, suppose that $T[n] = [001,011,110]$; when a function Reverse $(T[n],2,3)$ is invoked, the returned result is $T[n] = [001,110,011]$.

If there are five transaction patterns in $T[n] = [001,011,101,110,111]$, then the above functions return spanning path $T[n] = [011,001,101,111,110]$ with a total cost of nine. Fig. 6a depicts the whole process. If there are two sites participating in the distributed environment, a round-robin approach is applied to assign the transactions; the two sites alternatively receive nodes with odd labels, i.e., $\{011,101,110\}$, and nodes with even labels, i.e., $\{001,111\}$, respectively, as their databases. In general, if there are m participating sites in the distributed environment, a round-robin approach is applied to assign the transactions; the sites circularly receive nodes as their databases. For example, if the SSP contains $\{n_1,n_2,n_3,\ldots,n_k\}$, then the first site contains $\{n_1,n_{m+1},n_{2m+1},\ldots\}$, the second site contains $\{n_2,n_{m+2},n_{2m+2},\ldots\}$, and the ith site contains $\{n_i,n_{m+i},n_{2m+i},\ldots\}$.

Note that since the solution derived by the proposed algorithm is just an approximation, the DIVIDE_AND_CONQUER algorithm may fail to find a shortest spanning path in the k-dimension when the paths to be merged in the $(k-1)$-dimensions are "twisted" in their directions, as shown in Fig. 6b (the dotted arrows). The expected optimum spanning path is shown by the dotted arrows in Fig. 6c. Nevertheless, the experiment results given in Section 4 show that the performance is good enough even for an extremely large set of transactions.

Table 1
Datasets.

Name	D	T	I	Size (MB)
D200K.T10.I4	200K	10	4	11.3
D200K.T10.I8	200K	10	8	13.1
D200K.T20.I4	200K	20	4	20.5
D200K.T20.I8	200K	20	8	22.4
D400K.T10.I4	400K	10	4	22.6
D600K.T10.I4	600K	10	4	33.9
D600K.T10.I8	600K	10	8	39.4
D600K.T20.I4	600K	20	4	61.8
D600K.T20.I8	600K	20	8	67.3
D800K.T10.I4	800K	10	4	45.3
D1000K.T10.I4	1000K	10	4	56.6
D1200K.T10.I4	1200K	10	4	67.9

D, number of transactions.
T, average transaction length.
I, average size of maximal frequent item.
N, number of different items (=1K).

Table 2
Time cost of data de-clustering.

Dataset name	Data de-clustering	Number of sites							
		2	4	6	8	10	12	14	16
D200K.T10.I4	Without I/O	7.406	7.438	7.438	7.422	7.421	7.422	7.437	7.421
	With I/O	16.266	16.047	16.031	16.062	16.109	15.984	16.063	16.000
D400K.T10.I4	Without I/O	16.422	16.141	16.109	16.141	16.187	16.140	16.141	16.156
	With I/O	34.954	33.641	33.484	33.610	35.437	34.140	33.828	33.860
D600K.T10.I4	Without I/O	24.500	24.484	24.453	24.468	24.985	24.531	24.469	24.469
	With 170	52.015	52.281	51.000	51.000	51.266	51.031	51.219	50.890
D800K.T10.I4	Without I/O	32.875	32.875	32.922	33.203	32.937	32.907	33.203	33.281
	With I/O	68.110	67.766	68.828	68.609	68.563	68.328	70.328	68.891
D1000K.T10.I4	Without I/O	40.875	41.063	40.906	40.906	40.875	40.891	40.891	41.031
	With I/O	86.344	86.625	86.250	87.110	86.547	87.156	87.328	87.157
D1200K.T10.I4	Without I/O	50.515	50.219	50.156	50.453	50.125	50.468	50.406	50.109
	With I/O	108.203	106.422	108.343	108.656	108.547	108.922	108.031	107.953

Units: seconds.

4. Experimental results

To verify and evaluate the effectiveness of the proposed approach, the proposed algorithms were implemented and experiments were conducted on various datasets. Microsoft Visual C# was used to develop the testing program, which was executed in Microsoft Windows XP with the.NET Framework 2.0. In addition, Bodon's data mining tool [10] was adopted as the mining algorithm to mine the de-clustered datasets. All experiments were performed on a shared-nothing distributed environment, which consisted of 16 PCs, each with a 1.73 GHz CPU and 2 GB of main memory.

Table 1 shows the synthetic datasets used in the experiments. These datasets were generated by the program developed by IBM Almaden Research Center [6], which has been extensively adopted for benchmarking distributed association mining algorithms. Five experiments were designed for evaluating the performance of the proposed de-clustering algorithm.

In the first experiment, the datasets were de-clustered into several subgroups. The aim was to analyze which database attributes affect the performance of de-clustering. The second experiment adopted Ferenc Bodon's mining tool to mine

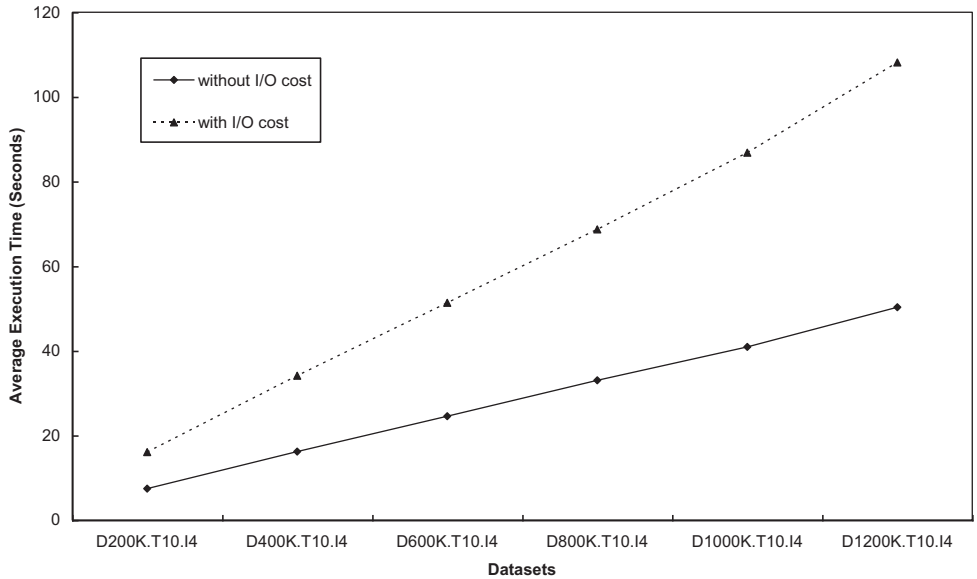


Fig. 7. De-clustering performance evaluation data size ranging from 200K to 1200K. Each measured point corresponds to the average value of its corresponding row in Table 2.

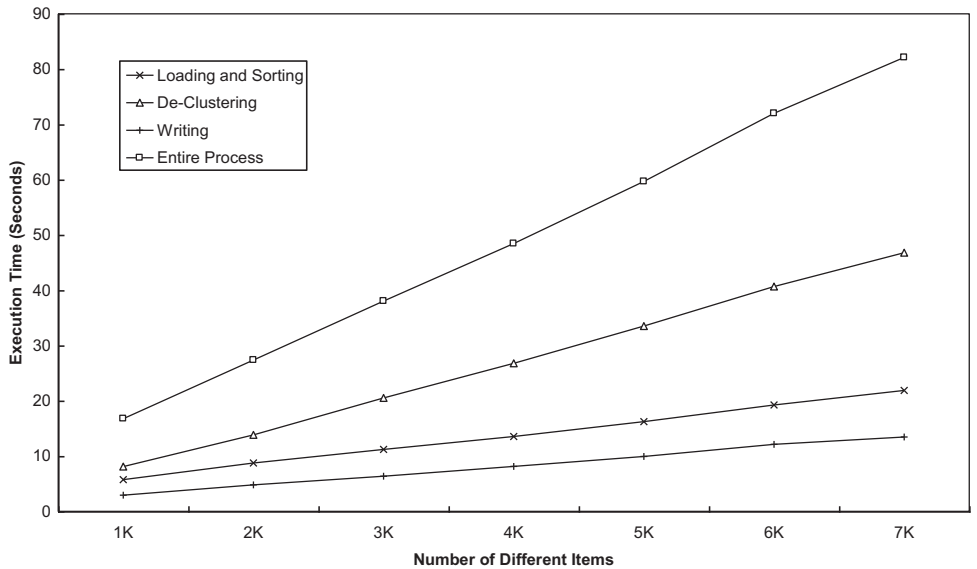


Fig. 8. De-clustering performance obtained by gradually increasing the number of different items from 1K to 7K (D200K.T10.I4).

the de-clustered datasets in the testing environment to measure the speed improvement of the mining algorithm with respect to the number of sites. The third experiment measured the effectiveness and correctness of the proposed de-clustering algorithm according to the precision and recall rates. In the fourth experiment, the data obtained from the de-clustering algorithm are compared with those obtained from random sampling approaches to show the effectiveness and superiority of the proposed method. Finally, two approaches are proposed to improve the aggregate mining results. This experiment also reveals the possibility of refining the distributed mining results.

4.1. Performance study of data de-clustering

The first experiment evaluated the stability of the proposed approach on data with various sizes and complexities, as listed in Table 1.

In this test bed, six datasets with different numbers of transactions were evaluated (200K transactions were added to each level). The proposed method was then used to de-cluster these datasets into subgroups for distributed mining. Notice that

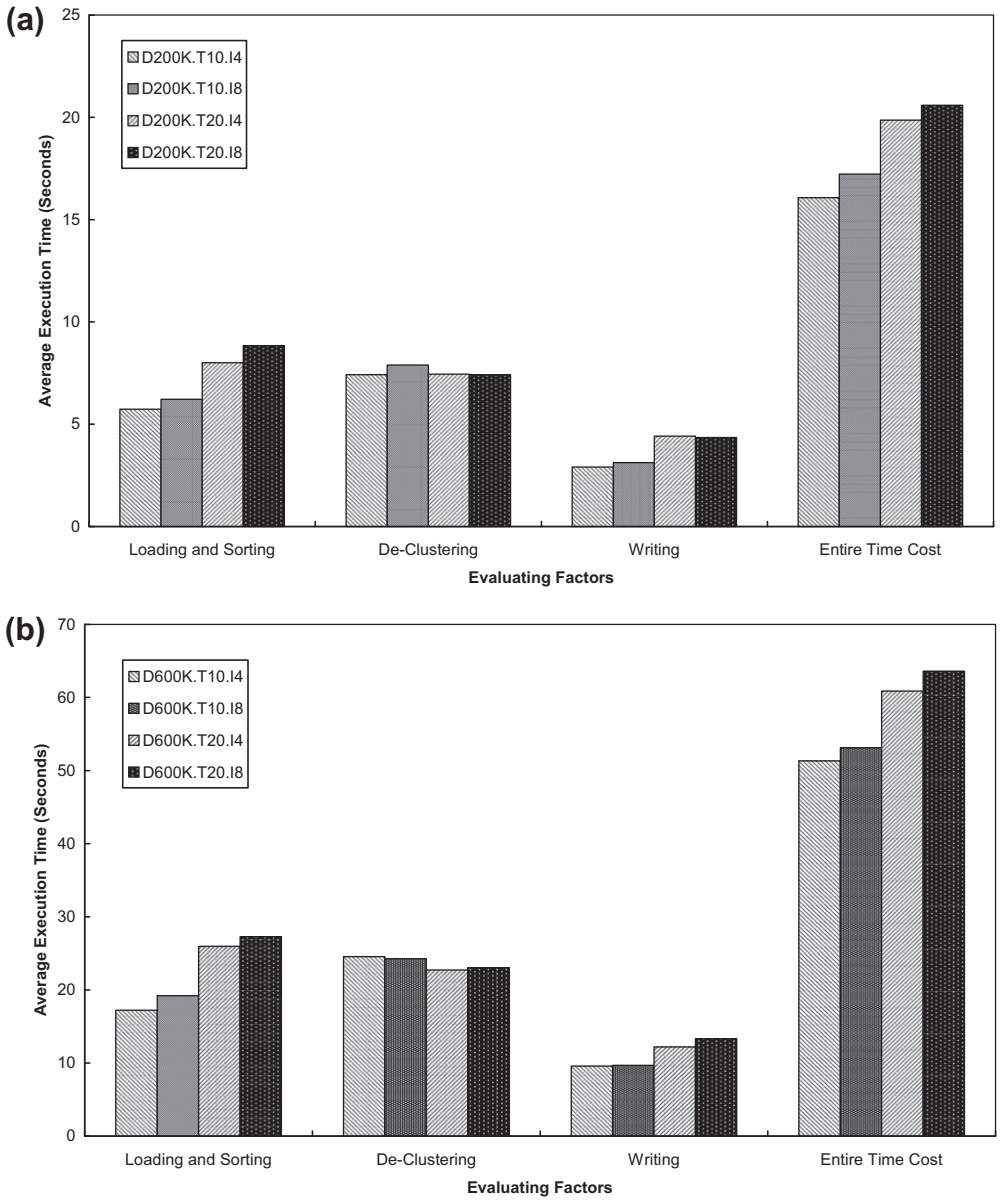


Fig. 9. Detailed time cost analysis of data de-clustering. Two experiments were performed with different data sizes (200K *vs.* 600K), average transaction lengths (10 *vs.* 20), and average sizes of maximal frequent itemsets (4 *vs.* 8).

the input transaction set has to be sorted before data de-clustering. The sorting algorithm adopted is not restricted by the proposed approach. As for the implementation, QuickSort was chosen to sort the dataset before de-clustering. According to the results shown in Table 2, the number of sites did not affect the performance of de-clustering. Figs. 7 and 8 show that only the data size and the number of items affect the performance of de-clustering. Therefore, increasing both factors gradually, as shown in Figs. 7 and 8, increased execution times linearly, which is acceptable. Fig. 9 shows the detailed execution time of the de-clustering task, where two datasets were employed to evaluate the performance. From Fig. 9a and b, the two de-clustering processes with a given data size require almost the same time; they are not affected by the attributes of average transaction length and average size of maximal frequent itemsets. For the "Loading and Sorting" and "Writing" parts in Fig. 9, the transactions were loaded from files and then sorted using QuickSort. Subsequently, the data were de-clustered by finding their approximate SSP, and written into files directly. The performances of "Loading and Sorting" and "Writing" usually depend on the hardware performance, e.g., hard disk read/write speed, CPU, RAM, and so on. The data are listed here only for benchmark illustration. For example, the proposed approach may be implemented as a built-in de-clustering function in

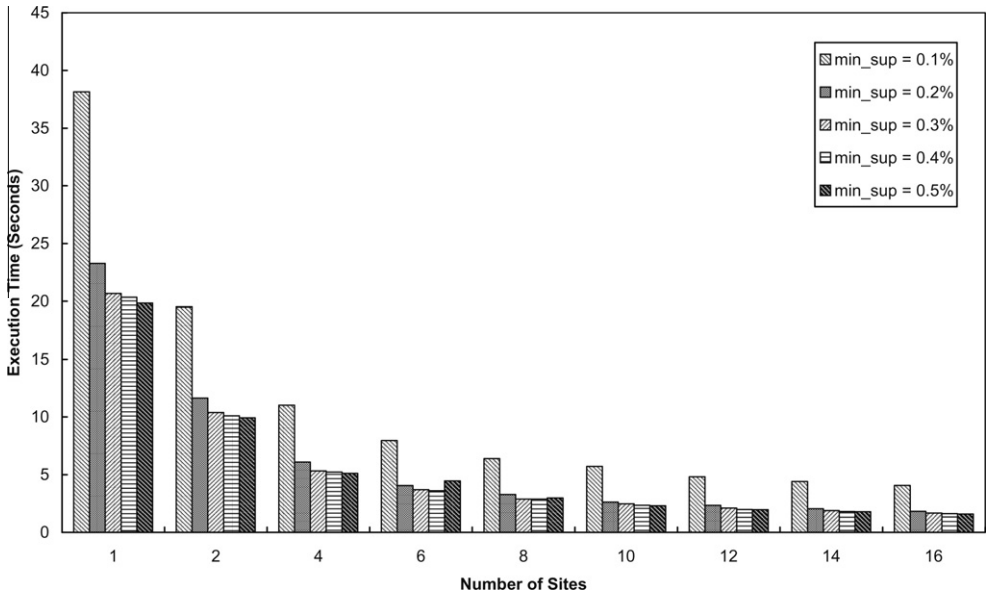


Fig. 10. Mining execution time with fixed dataset (D1200K.T10.I4) and minimum support thresholds ranging from 0.1% to 0.5%.

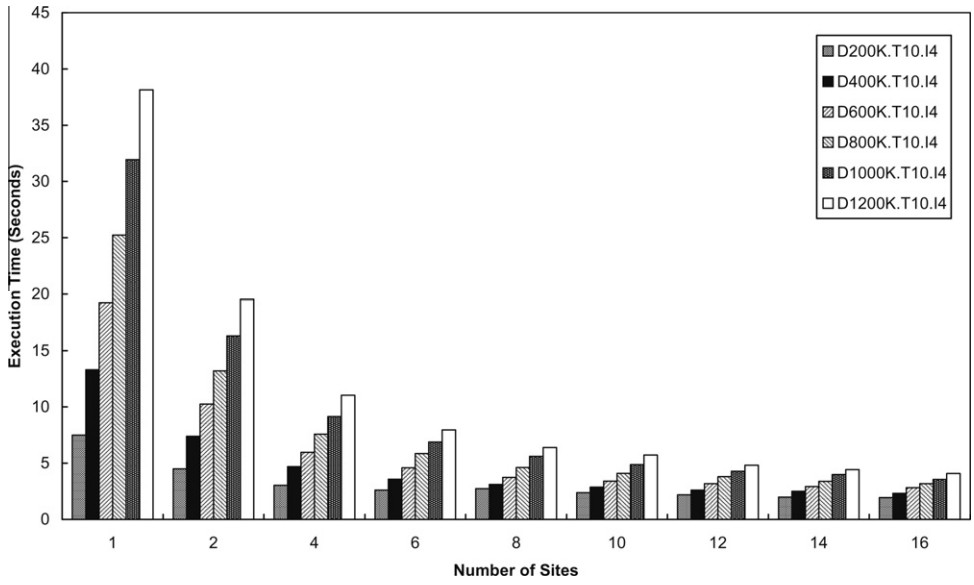
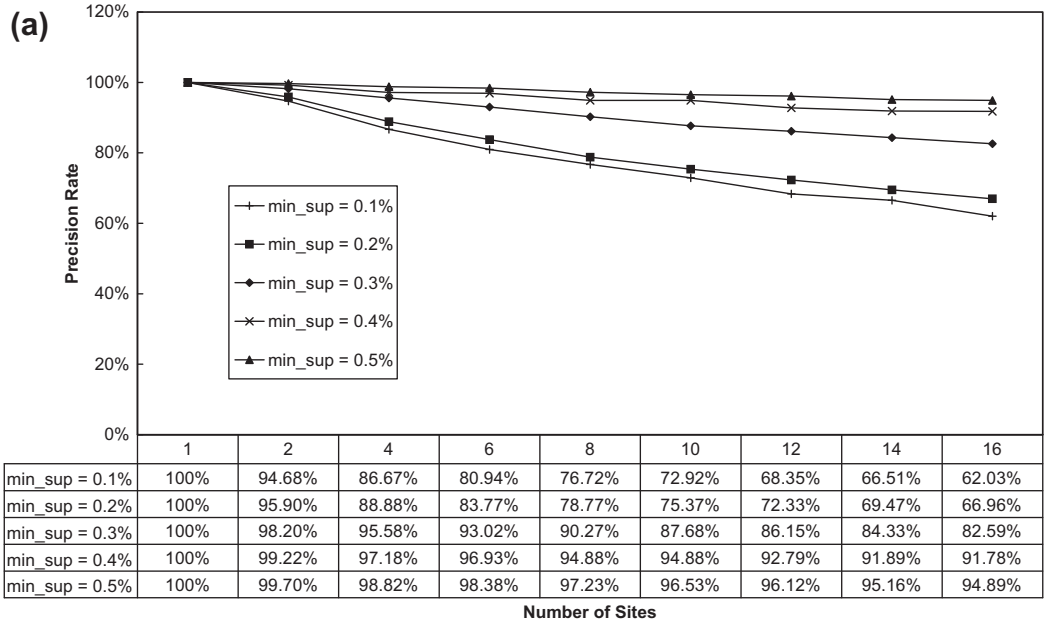


Fig. 11. Mining execution time with fixed support threshold (0.1%) and data sizes ranging from 200K to 1200K.

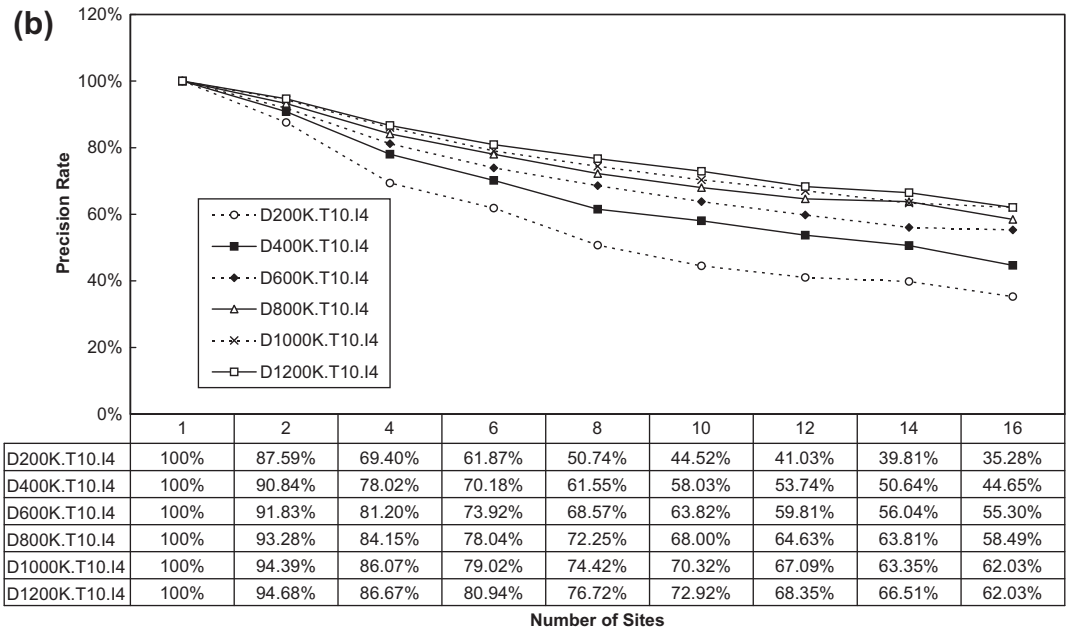
DBMS. Then, the time of "Loading and Sorting" would be substantially reduced by the effective indexing functionalities of DBMSs. The de-clustered itemsets may also be directly synchronized with the other databases in a distributed environment.

4.2. Performance study of distributed data mining

In the second experiment, the test bed consisted of 1–16 sites. The purpose was to evaluate the distributed mining performance with respect to data size and support threshold. As each site has its own local disk, each part of the de-clustered dataset was loaded on its local disk before the experiment started. Fig. 10 shows the distributed execution time for mining the dataset D1200K.T10.I4 with the support threshold ranging from 0.1% to 0.5%; the obtained speedup is significant. Fig. 11



	1	2	4	6	8	10	12	14	16
min_sup = 0.1%	100%	94.68%	86.67%	80.94%	76.72%	72.92%	68.35%	66.51%	62.03%
min_sup = 0.2%	100%	95.90%	88.88%	83.77%	78.77%	75.37%	72.33%	69.47%	66.96%
min_sup = 0.3%	100%	98.20%	95.58%	93.02%	90.27%	87.68%	86.15%	84.33%	82.59%
min_sup = 0.4%	100%	99.22%	97.18%	96.93%	94.88%	94.88%	92.79%	91.89%	91.78%
min_sup = 0.5%	100%	99.70%	98.82%	98.38%	97.23%	96.53%	96.12%	95.16%	94.89%



	1	2	4	6	8	10	12	14	16
D200K.T10.I4	100%	87.59%	69.40%	61.87%	50.74%	44.52%	41.03%	39.81%	35.28%
D400K.T10.I4	100%	90.84%	78.02%	70.18%	61.55%	58.03%	53.74%	50.64%	44.65%
D600K.T10.I4	100%	91.83%	81.20%	73.92%	68.57%	63.82%	59.81%	56.04%	55.30%
D800K.T10.I4	100%	93.28%	84.15%	78.04%	72.25%	68.00%	64.63%	63.81%	58.49%
D1000K.T10.I4	100%	94.39%	86.07%	79.02%	74.42%	70.32%	67.09%	63.35%	62.03%
D1200K.T10.I4	100%	94.68%	86.67%	80.94%	76.72%	72.92%	68.35%	66.51%	62.03%

Fig. 12. Precision rates of aggregate mining results. (a) Precision rates of aggregate mining results with fixed dataset (D1200K.T10.I4) and minimum support thresholds (*min_sup*) ranging from 0.1% to 0.5%. (b) Precision rates of aggregate mining results with fixed minimum support threshold (0.1%) and dataset sizes ranging from 200K to 1200K.

shows the results for six datasets with data sizes ranging from 200K to 1200K, different data complexities, and a fixed support threshold (0.1%). Based on the results, the performance in such a distributed mining environment increased with increasing data size.

4.3. Mining result verification

The third experiment was designed to verify the distributed mining results. To verify the correctness of the mining results, two effectiveness measurements, the precision and recall rates [9], were employed. The precision rate p can be evaluated using Eq. (1) and the recall rate r can be evaluated using Eq. (2). In Eqs. (1) and (2), the parameter α represents the set of frequent itemsets generated by a single site method or the proposed approach; the parameter β is the set of frequent itemsets that cannot be generated by a single site method, but can be derived by the proposed approach; and the parameter γ denotes the set of frequent itemsets generated by a single site method but that cannot be derived by the proposed approach

$$p = \frac{|\alpha|}{|\alpha| + |\beta|} \tag{1}$$

$$r = \frac{|\alpha|}{|\alpha| + |\gamma|} \tag{2}$$

Additionally, the concept of aggregate mining results M is given in Eq. (3) to represent the union result of all distribution sites ($M = \alpha + \beta$). In Eq. (3), m_i is the mining result of s_i, where $S = \{s_1, s_2, \ldots, s_n\}$ represents the distributed environment consisting of n sites

$$M = \bigcup_{i=1}^{|S|} m_i \tag{3}$$

In Fig. 12a, the experiment used dataset D1200K.T10.I4 and support thresholds ranging from 0.1% to 0.5% to estimate the precision of aggregate mining results. The purpose was to evaluate the relationship between the support threshold and the precision rate of aggregate mining results. Fig. 12b shows the precision rates of aggregate mining results with a fixed support threshold of 0.1% for data sizes ranging from 200K to 1200K. It can be seen that the precision decreases with increasing number of sites. The recall rates of the aggregate mining results of the two experiments are shown in Tables 3 and 4, respectively. Both tables indicate that the proposed approach achieves 100% recall rates for all situations.

To conduct more experiments with different data complexities, eight datasets with different numbers of transactions, different average transaction lengths, and different average sizes of maximal frequent itemsets with a fixed support threshold of 0.3% were prepared. The verification results are shown in Table 5, which indicates that the aggregate recall rates can always be completely guaranteed.

The aggregate precision rate seems inadequate. When the data size is small, the support threshold is low, and the number of distribution sites is large. Fortunately, the proposed de-clustering approach can be regarded as a sampling method, where

Table 3
Precision and recall verification for mining results of D1200K.T10.I4 with support thresholds.

Sites	1 Sites					2 Sites					4 Sites				
min_sup (%)	$\|\alpha\|$	$\|\beta\|$	$\|\gamma\|$	p	r	$\|\alpha\|$	$\|\beta\|$	$\|\gamma\|$	p	r	$\|\alpha\|$	$\|\beta\|$	$\|\gamma\|$	p	r
0.1	20,705	0	0	100	100	20,705	1164	0	94.68	100	20,705	3185	0	86.67	100
0.2	2549	0	0	100	100	2549	109	0	95.90	100	2549	319	0	88.88	100
0.3	1039	0	0	100	100	1039	19	0	98.20	100	1039	48	0	95.58	100
0.4	759	0	0	100	100	759	6	0	99.22	100	759	22	0	97.18	100
0.5	668	0	0	100	100	668	2	0	99.70	100	668	8	0	98.82	100
	6 Sites					8 Sites					10 Sites				
0.1	20,705	4875	0	80.94	100	20,705	6282	0	76.22	100	20,705	7691	0	72.92	100
0.2	2549	494	0	83.77	100	2549	687	0	78.77	100	2549	833	0	75.37	100
0.3	1039	78	0	93.02	100	1039	112	0	90.27	100	1039	146	0	87.68	100
0.4	759	24	0	96.93	100	759	41	0	94.8S	100	759	41	0	94.88	100
0.5	668	11	0	98.38	100	668	19	0	97.23	100	668	24	0	96.53	100
	12 Sites					14 Sites					16 Sites				
0.1	20,705	9589	0	68.35	100	20,705	10424	0	66.51	100	20,705	12,674	0	62.03	100
0.2	2549	975	0	72.33	100	2549	1120	0	69.47	100	2549	1258	0	66.96	100
0.3	1039	167	0	86.15	100	1039	193	0	84.33	100	1039	219	0	82.59	100
0.4	759	59	0	92.79	100	759	67	0	91.89	100	759	68	0	91.78	100
0.5	668	27	0	96.12	100	668	34	0	95.16	100	668	36	0	94.89	100

min_sup, minimum support threshold (=0.1–05%).
p, precision rate (%).
r, recall rate (%).

each de-clustered subgroup can be considered as a sample, such that all of its features are extracted from the source dataset. Moreover, the mining result of any de-clustered subgroup is very similar to that of the complete set. These characteristics

Table 4
Precision and recall verification for mining results with support threshold = 0.1%.

Sites	1 Sites					2 Sites					4 Sites				
Datasets	$\|\alpha\|$	$\|\beta\|$	$\|\gamma\|$	p	r	$\|\alpha\|$	$\|\beta\|$	$\|\gamma\|$	p	r	$\|\alpha\|$	$\|\beta\|$	$\|\gamma\|$	p	r
D200E.T10.I4	20,519	0	0	100	100	20,519	2906	0	87.59	100	20,519	9047	0	69.40	100
D4OOK.T10.I4	20,479	0	0	100	100	20,479	2064	0	90.84	100	20,479	5770	0	78.02	100
D600K.T10.I4	20,606	0	0	100	100	20,606	1833	0	91.83	100	20,606	4772	0	81.20	100
D800E.T10.I4	20,821	0	0	100	100	20,821	1500	0	93.28	100	20,821	3921	0	84.15	100
D1000K.T10.I4	20,691	0	0	100	100	20,691	1230	0	94.39	100	20,691	3350	0	86.07	100
D1200K.T10.I4	20,705	0	0	100	100	20,705	1164	0	94.68	100	20,705	3185	0	86.67	100
	6 Sites					8 Sites					10 Sites				
D200K.T10.I4	20,519	12,648	0	61.87	100	20,519	19,920	0	50.74	100	20,519	25,570	0	44.52	100
D400E.T10.I4	20,479	8700	0	70.18	100	20,479	12,792	0	61.55	100	20,479	14,810	0	58.03	100
D600K.T10.I4	20,606	7270	0	73.92	100	20,606	9444	0	68.57	100	20,606	11,681	0	63.82	100
D800K.T10.I4	20,821	5858	0	78.04	100	20,821	7995	0	72.25	100	20,821	9796	0	68.00	100
D1000E.T10.I4	20,691	5494	0	79.02	100	20,691	7113	0	74.42	100	20,691	8732	0	70.32	100
D1200K.T10.I4	20,705	4875	0	80.94	100	20,705	6282	0	76.72	100	20,705	7691	0	72.92	100
	12 Sites					14 Sites					16 Sites				
D200E.T10.I4	20,519	29,489	0	41.03	100	20,519	31,017	0	39.81	100	20,519	37,645	0	35.28	100
D4OOK.T10.I4	20,479	17,632	0	53.74	100	20,479	19,959	0	50.64	100	20,479	25,391	0	44.65	100
D600E.T10.I4	20,606	13,848	0	59.81	100	20,606	16,167	0	56.04	100	20,606	16,656	0	55.30	100
D800K.T10.I4	20,821	11,397	0	64.63	100	20,821	11,810	0	63.81	100	20,821	14,775	0	58.49	100
D1000K.T10.I4	20,691	10,148	0	67.09	100	20,691	11,972	0	63.35	100	20,691	12,668	0	62.03	100
D1200E.T10.I4	20,705	9589	0	68.35	100	20,705	10,424	0	66.51	100	20,705	12,674	0	62.03	100

p, precision rate (%).
r, recall rate (%).

Table 5
Precision and recall verification for mining results with support threshold = 0.3%.

Sites	1 Sites					2 Sites					4 Sites				
Datasets	$\|\alpha\|$	$\|\beta\|$	$\|\gamma\|$	p	r	$\|\alpha\|$	$\|\beta\|$	$\|\gamma\|$	p	r	$\|\alpha\|$	$\|\beta\|$	$\|\gamma\|$	p	r
D200K.T10.I4	1043	0	0	100	100	1043	55	0	94.99	100	1043	158	0	86.84	100
D200K.T10.I8	1642	0	0	100	100	1642	132	0	92.56	100	1642	318	0	83.78	100
D200E.T20.I4	13,807	0	0	100	100	13,807	1173	0	92.17	100	13,807	3404	0	80.22	100
D200K.T20.I8	17,246	0	0	100	100	17,246	1183	0	93.58	100	17,246	3122	0	84.67	100
D600K.T10.I4	1046	0	0	100	100	1046	20	0	98.12	100	1046	72	0	93.56	100
D600K.T10.I8	1624	0	0	100	100	1624	69	0	95.92	100	1624	187	0	89.67	100
D600K.T20.I4	13,471	0	0	100	100	13,471	817	0	94.28	100	13,471	1896	0	87.66	100
D600K.T20.I8	17,207	0	0	100	100	17,207	624	0	96.50	100	17,207	1729	0	90.87	100
	6 Sites					8 Sites					10 Sites				
D200K.T10.I4	1043	255	0	80.35	100	1043	375	0	73.55	100	1043	539	0	65.93	100
D200K.T10.I8	1642	536	0	75.39	100	1642	791	0	67.49	100	1642	1048	0	61.04	100
D200K.T20.I4	13,807	5133	0	72.90	100	13,807	6928	0	66.59	100	13,807	8866	0	60.90	100
D200K.T20.I8	17,246	4795	0	78.25	100	17,246	6599	0	72.33	100	17,246	8509	0	66.96	100
D600K.T10.I4	1046	121	0	89.63	100	1046	157	0	86.95	100	1046	204	0	83.68	100
D600K.T10.I8	1624	289	0	84.89	100	1624	385	0	80.84	100	1624	505	0	76.28	100
D600K.T20.I4	13,471	3063	0	81.47	100	13,471	3973	0	77.22	100	13,471	5032	0	72.80	100
D600K.T20.I8	17,207	2581	0	86.96	100	17,207	3501	0	83.09	100	17,207	4333	0	79.88	100
	12 Sites					14 Sites					16 Sites				
D200K.T10.I4	1043	674	0	60.75	100	1043	807	0	56.38	100	1043	851	0	55.07	100
D200K.T10.I8	1642	1290	0	56.00	100	1642	1552	0	51.41	100	1642	1745	0	48.48	100
D200K.T20.I4	13,807	10,395	0	57.05	100	13,807	12,000	0	53.50	100	13,807	12,927	0	51.65	100
D200K.T20.I8	17,246	10,347	0	62.50	100	17,246	11,684	0	59.61	100	17,246	13,094	0	56.84	100
D600K.T10.I4	1046	258	0	80.21	100	1046	296	0	77.94	100	1046	333	0	75.85	100
D600K.T10.I8	1624	600	0	73.02	100	1624	652	0	71.35	100	1624	764	0	68.01	100
D600E.T20.I4	13,471	5617	0	70.57	100	13,471	6282	0	68.20	100	13,471	6872	0	66.22	100
D600K.T20.I8	17,207	5082	0	77.20	100	17,207	5651	0	75.28	100	17,207	6450	0	72.74	100

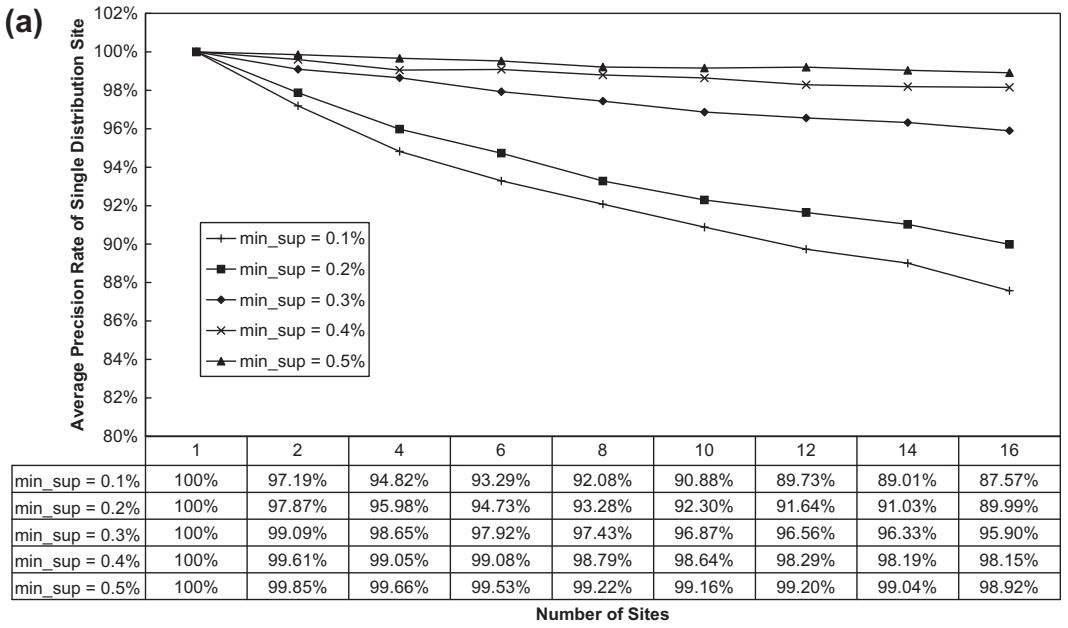
p, precision rate (%).
r, recall rate (%).

can be employed to alleviate the problem of low aggregate precision rate. The sampling approach based on de-clustering is described below.

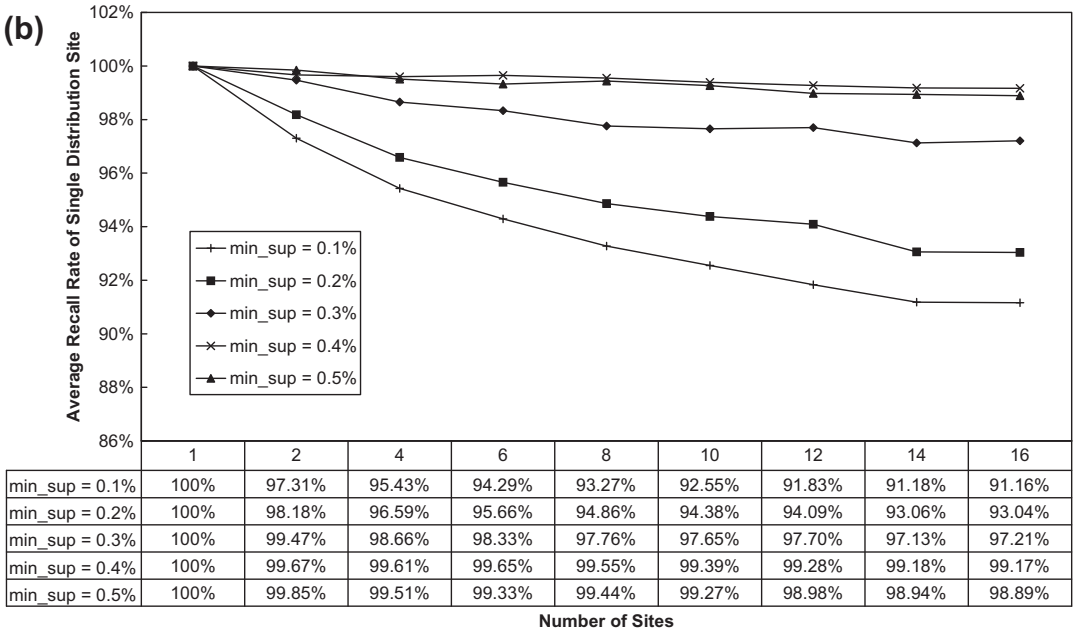
Using the settings from previous experiments, experiments with different support thresholds and data sizes were conducted, with results shown in Figs. 13 and 14, respectively. To verify the effectiveness of the sampling approach, the mean of the results of all distribution sites in the following experiments was measured.

Fig. 13 shows that if the data size is large enough, the support threshold has no effect on the correctness (the measured precision and recall rates are over 90%). In Fig. 14, the sampling approach achieves precision rates of over 70% for the worst case, where dataset, *min_sup*, and the number of sites are set to D200K.T10.I4, 0.1%, and 16, respectively. It thus outperforms the worst aggregate mining results, as shown in Fig. 12b.

Additionally, from Fig. 14, the precision and recall rate of each participant increased with increasing data size. This means that the proposed method is very effective for the processing of extremely large sets of data.

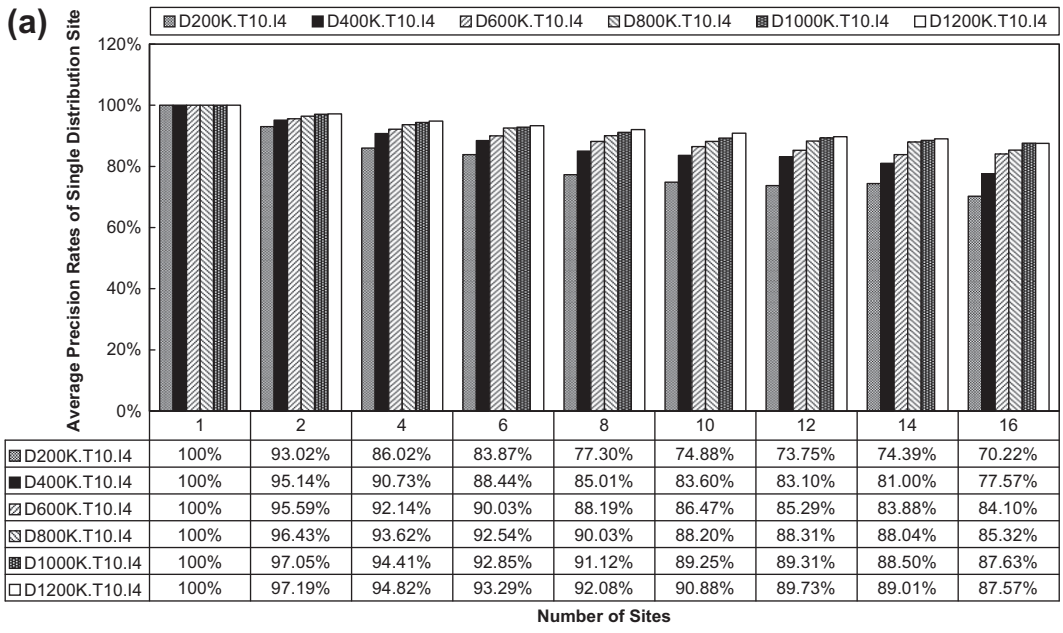


	1	2	4	6	8	10	12	14	16
min_sup = 0.1%	100%	97.19%	94.82%	93.29%	92.08%	90.88%	89.73%	89.01%	87.57%
min_sup = 0.2%	100%	97.87%	95.98%	94.73%	93.28%	92.30%	91.64%	91.03%	89.99%
min_sup = 0.3%	100%	99.09%	98.65%	97.92%	97.43%	96.87%	96.56%	96.33%	95.90%
min_sup = 0.4%	100%	99.61%	99.05%	99.08%	98.79%	98.64%	98.29%	98.19%	98.15%
min_sup = 0.5%	100%	99.85%	99.66%	99.53%	99.22%	99.16%	99.20%	99.04%	98.92%

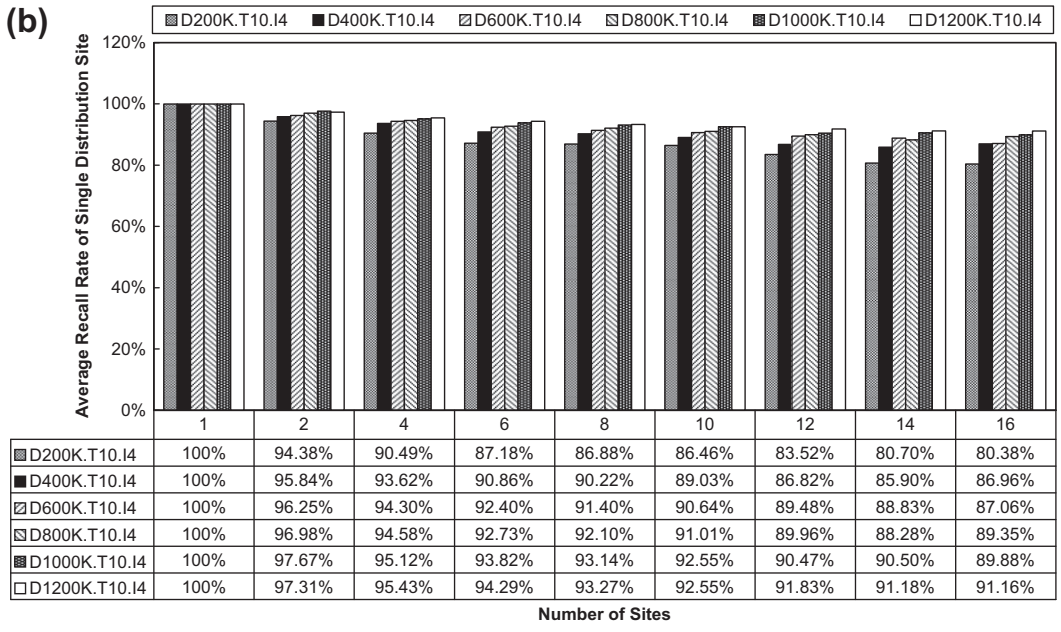


	1	2	4	6	8	10	12	14	16
min_sup = 0.1%	100%	97.31%	95.43%	94.29%	93.27%	92.55%	91.83%	91.18%	91.16%
min_sup = 0.2%	100%	98.18%	96.59%	95.66%	94.86%	94.38%	94.09%	93.06%	93.04%
min_sup = 0.3%	100%	99.47%	98.66%	98.33%	97.76%	97.65%	97.70%	97.13%	97.21%
min_sup = 0.4%	100%	99.67%	99.61%	99.65%	99.55%	99.39%	99.28%	99.18%	99.17%
min_sup = 0.5%	100%	99.85%	99.51%	99.33%	99.44%	99.27%	98.98%	98.94%	98.89%

Fig. 13. Experiment used to verify the de-clustering method considered as sampling with fixed dataset (D1200K.T10.I4) and support thresholds (*min_sup*) ranging from 0.1% to 0.5%. (a) Precision rate of a single site mining result. (b) Recall rate of a single site mining result.



	1	2	4	6	8	10	12	14	16
D200K.T10.I4	100%	93.02%	86.02%	83.87%	77.30%	74.88%	73.75%	74.39%	70.22%
D400K.T10.I4	100%	95.14%	90.73%	88.44%	85.01%	83.60%	83.10%	81.00%	77.57%
D600K.T10.I4	100%	95.59%	92.14%	90.03%	88.19%	86.47%	85.29%	83.88%	84.10%
D800K.T10.I4	100%	96.43%	93.62%	92.54%	90.03%	88.20%	88.31%	88.04%	85.32%
D1000K.T10.I4	100%	97.05%	94.41%	92.85%	91.12%	89.25%	89.31%	88.50%	87.63%
D1200K.T10.I4	100%	97.19%	94.82%	93.29%	92.08%	90.88%	89.73%	89.01%	87.57%



	1	2	4	6	8	10	12	14	16
D200K.T10.I4	100%	94.38%	90.49%	87.18%	86.88%	86.46%	83.52%	80.70%	80.38%
D400K.T10.I4	100%	95.84%	93.62%	90.86%	90.22%	89.03%	86.82%	85.90%	86.96%
D600K.T10.I4	100%	96.25%	94.30%	92.40%	91.40%	90.64%	89.48%	88.83%	87.06%
D800K.T10.I4	100%	96.98%	94.58%	92.73%	92.10%	91.01%	89.96%	88.28%	89.35%
D1000K.T10.I4	100%	97.67%	95.12%	93.82%	93.14%	92.55%	90.47%	90.50%	89.88%
D1200K.T10.I4	100%	97.31%	95.43%	94.29%	93.27%	92.55%	91.83%	91.18%	91.16%

Fig. 14. Experiment used to verify the de-clustering method considered as sampling with fixed support threshold (0.1%) and dataset sizes ranging from 200K to 1200K. (a) Precision rate of a single site mining result. (b) Recall rate of a single site mining result.

The following observations were made:

(1) If the source data size is moderate, then mining any subgroup of the de-clustered result is sufficient. The approximation is as satisfactory as that derived in a distributed environment by just running a single site.
(2) The precision and recall rate of each participant increases with the data size. When the data size tends to infinity, the asymptotic curve may approach the ideal case.
(3) If the resources are restricted due to hardware conditions, budget, and/or time constraints, the proposed method is recommended as a pre-testing sampling tool to filter out infeasible settings. Generally, when doing association rule mining, the support threshold needs to be modified to tune the results. However, the number of derived frequent itemsets increases with decreasing support threshold, which implies a time-consuming process. However, in the very

beginning of the analysis, the proper support threshold is unknown, and it takes time and resources to repeat the analysis. Therefore, analyzing a subgroup of the source data to filter out infeasible analysis settings substantially improves the mining process. Notice that a very small support threshold of 0.1% was intentionally used in the experimental setting; it produced more than twenty thousand frequent itemsets in Table 4. The setting was used for the stress test of the proposed de-clustering approach, with a satisfactory outcome.

Although applying the proposed method as a sampling approach can obtain approximate mining results, it should be noted that the sampling method cannot completely replace distributed data mining to effectively extract knowledge from extremely large datasets. From the results, it was found that the sampling method cannot return 100% recall, which implies that some information may be lost. Generally speaking, if the discovered association rules have high support values, they are usually well-known facts and not interesting. Therefore, the interesting rules to be discovered usually have moderate or less support, which indirectly indicates that the interesting patterns might be ignored by the sampling method. It can thus be concluded that adopting the proposed method as a sampling approach can only provide summary information of the entire

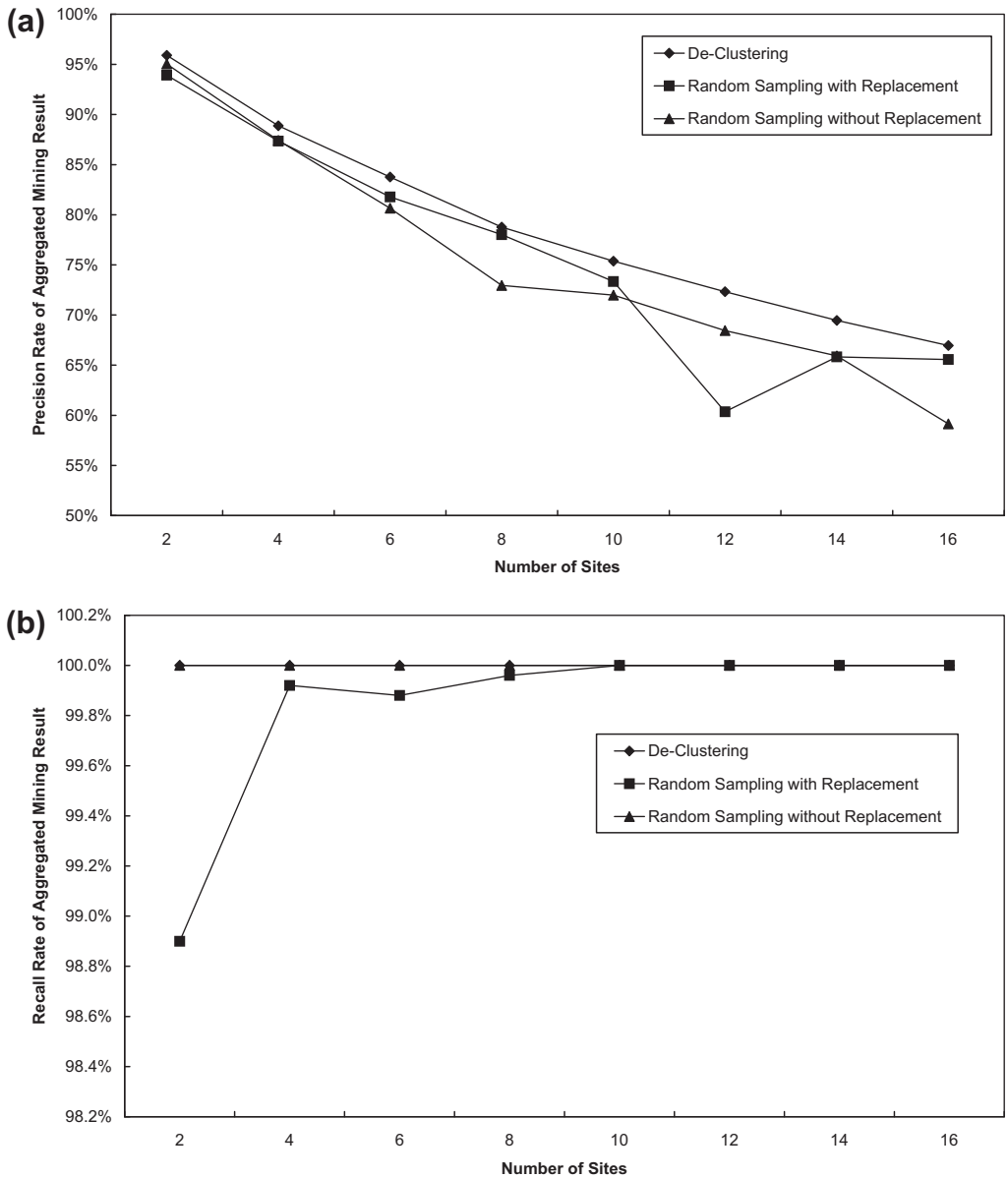


Fig. 15. Precision and recall rates of aggregate mining results for various levels of site distribution. The experiment was based on the dataset D1200K.T10.I4 with a minimum support threshold of 0.2%. Note that all points of the de-clustering and the random sampling without replacement overlap each other in Fig. 15b.

data. If it is necessary to realize the overall picture of the target dataset, then distributed data mining, rather than sampling, is recommended. Although the data sampled by data de-clustering can only be regarded as summary information, it still has a high level of confidence by itself. The next experiment shows that the proposed method is superior to the random sampling approach.

4.4. Comparison between de-clustering and random sampling

Based on the experiment in Section 4.3, the de-clustering algorithm can be adopted as a sampling approach to obtain the approximate association rule mining results. To perform a thorough comparison, the experiment was designed to compare the following performance measurements: the precision and recall rates of aggregate mining results, the best precision and recall rates of individual sites, the average precision and recall rates, the worst precision and recall rates of individual sites, and the stability of precision and recall rates. All the measurements were tested using different data parameter settings for different numbers of site distributions. However, for conciseness, only one set of results is here presented.

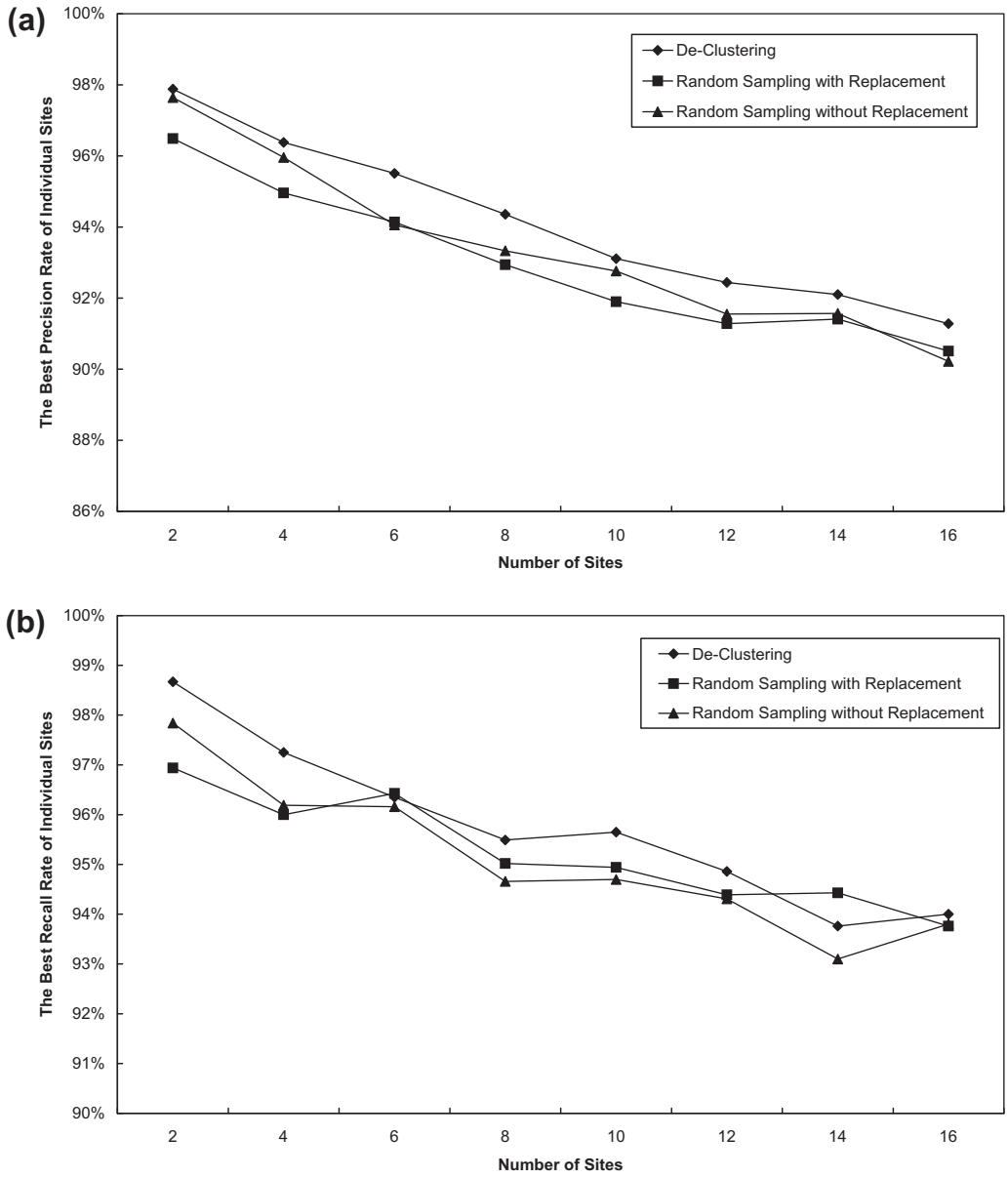


Fig. 16. Best precision and recall rates of individual sites for various levels of site distribution. The experiment was based on dataset D1200K.T10.I4 with a minimum support threshold of 0.2%.

The parameter settings of this experiment were as follows. The representative dataset was D1200K.T10.I4 with a support threshold of 0.2%. The precision, recall, and aggregate mining results all follow the definitions in Section 4.3. In addition, all mining result validations take into account the situations of data de-clustering and random sampling with and without replacement. Although all experimental results are not shown, the significant trends are discussed.

Fig. 15 shows the precision and recall rates of the aggregate mining result. In Fig. 15a, the de-clustering approach outperforms the other two random sampling approaches with regard to the precision of the aggregate mining result, although this may not always be the case when other experimental settings are adopted. Based on all experiments, the precision rates of the aggregate mining results obtained by a random sampling approach may on occasion slightly outperform the proposed approach, but the differences are bounded by 1% when the number of site distributions is low. In most cases, the proposed approach outperforms the other two random sampling approaches. Although rival approaches may randomly sample a superior dataset when the sampling matrices are large, random sampling cannot be applied to a large number of site distributions. The main reason is that a high number of site distributions cannot support enough samples for an individual site,

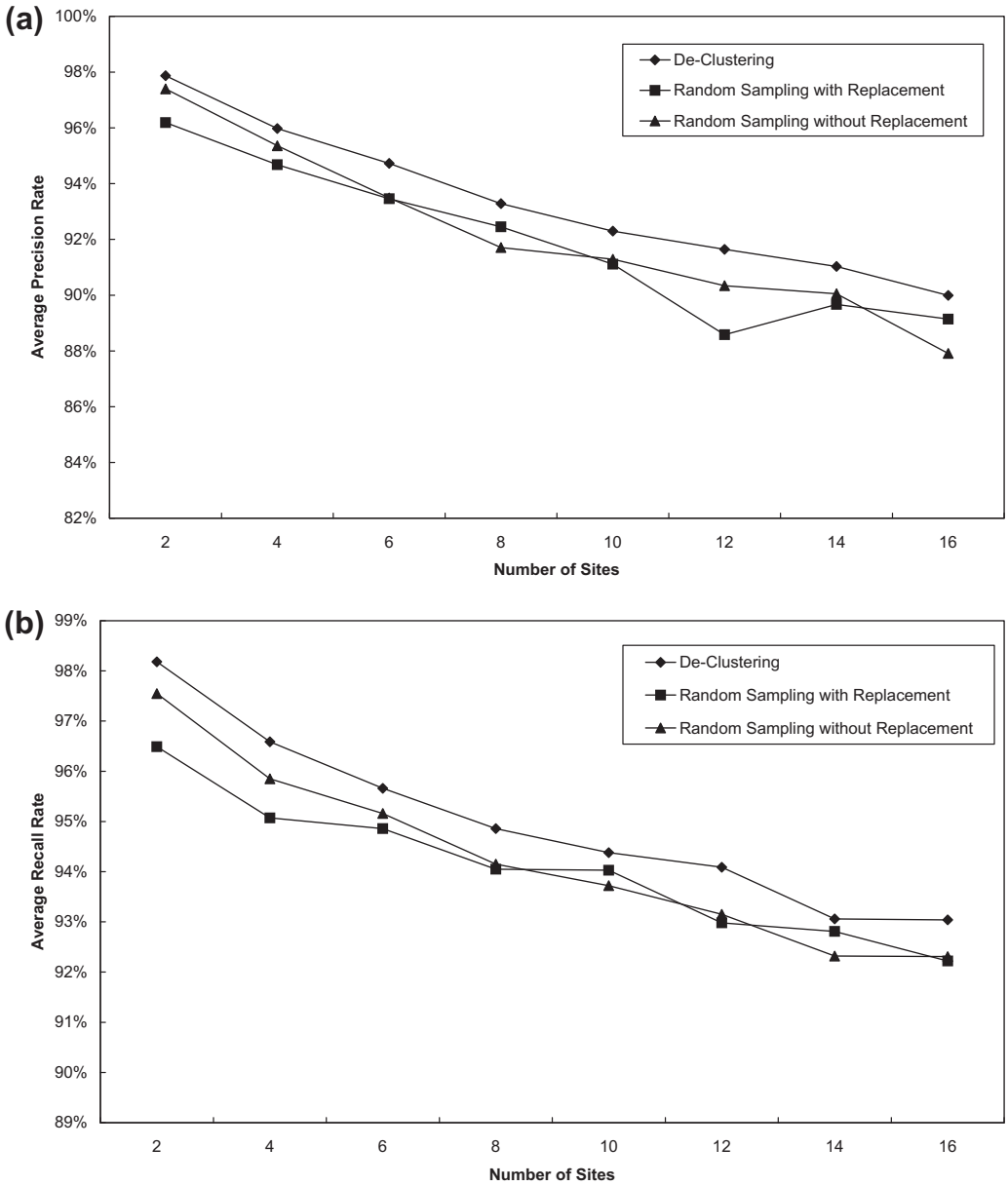


Fig. 17. Average precision and recall rates of individual sites for various levels of site distribution. The experiment was based on dataset D1200K.T10.I4 with a minimum support threshold of 0.2%.

which causes unstable data quality in all sites. This can be observed in Fig. 15a when the number of distribution sites is higher than 8.

According to Fig. 15b, the de-clustering and random sampling without replacement methods can always generate the complete and correct association rules. In contrast, the random sampling with replacement cannot always offer a 100% recall rate, which implies such the approach is not applicable to distributed data analysis.

Figs. 16–18 illustrate the performance evaluation of individual sites for various numbers of site distributions. In Fig. 16(Fig. 18), each point in the chart is the maximum (minimum) precision/recall rate of all individual sites for the various site distribution settings. Fig. 17 shows the means of all sites' precision and recall rates to evaluate the average performance of all approaches for various numbers of site distributions. The results indicate that the proposed data de-clustering approach can obtain the best results in almost all cases. In only a few situations, the random sampling approach slightly outperforms the proposed approach, but the differences are bounded by 1%. In summary, the proposed de-clustering approach can obtain better results in most situations, although it only outperforms the other methods by about 3%-10% in most cases. Random sampling cannot be applied to distributed data analysis because its performance is quite unstable.

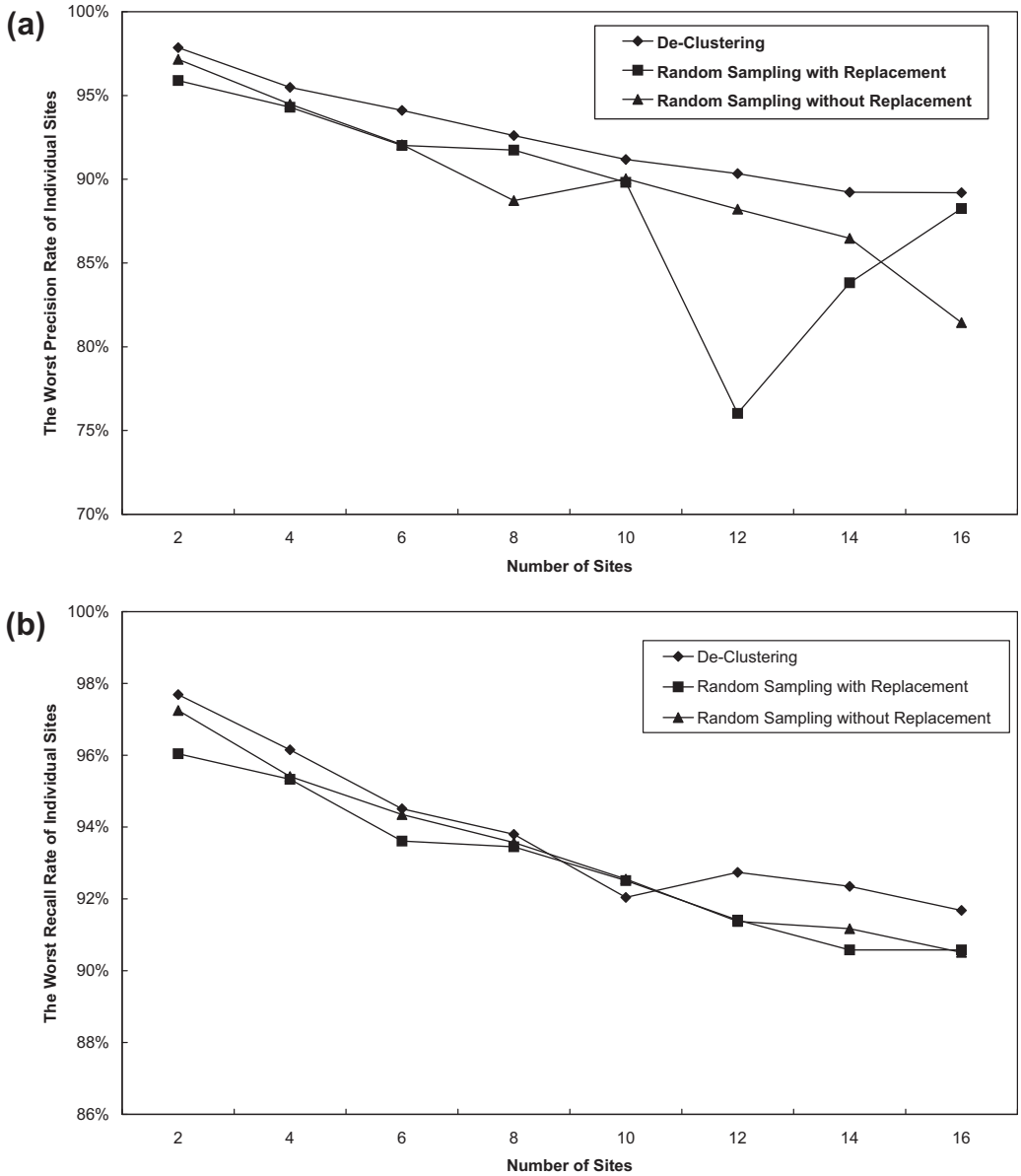


Fig. 18. Worst precision and recall rates of individual sites for various levels of site distribution. The experiment was based on dataset D1200K.T10.I4 with a minimum support threshold of 0.2%.

To evaluate the stability of the tested approaches, the variance measurement of statistics (see Eq. (4)) was used to measure the stabilities, where lower variance means higher stability

$$\text{variance } \mathrm{Var}(X) = E((X-\mu)^2) \tag{4}$$

where $\mu = E(X)$ is the expected value of random variable X.

Fig. 19 shows the stability of precision and recall rates for various numbers of site distributions. With regard to precision, the stability of random sampling with/without replacement is very poor, which means that the random sampling approach cannot guarantee the data quality of each individual site. Although the stability of the recall rate of random sampling seems to be stable in Fig. 19b (the variance is less than 1), the random sampling approach obtained a high variance in other experimental settings (some of the variances are higher than 5). Conversely, the data de-clustering approach always produced stable precision and recall rates for all tested situations, which guarantees the data quality of each individual site (i.e., all the distributed data are similar to the original dataset and to each other). Consequently, the random sampling approach cannot provide any guarantee of the minimal quality of the distributed mining result, which is detrimental to discovering frequent itemsets. Moreover, the experimental results reveal that the random sampling approach cannot even be adopted as a sampling method to extract summary information from an analysis database.

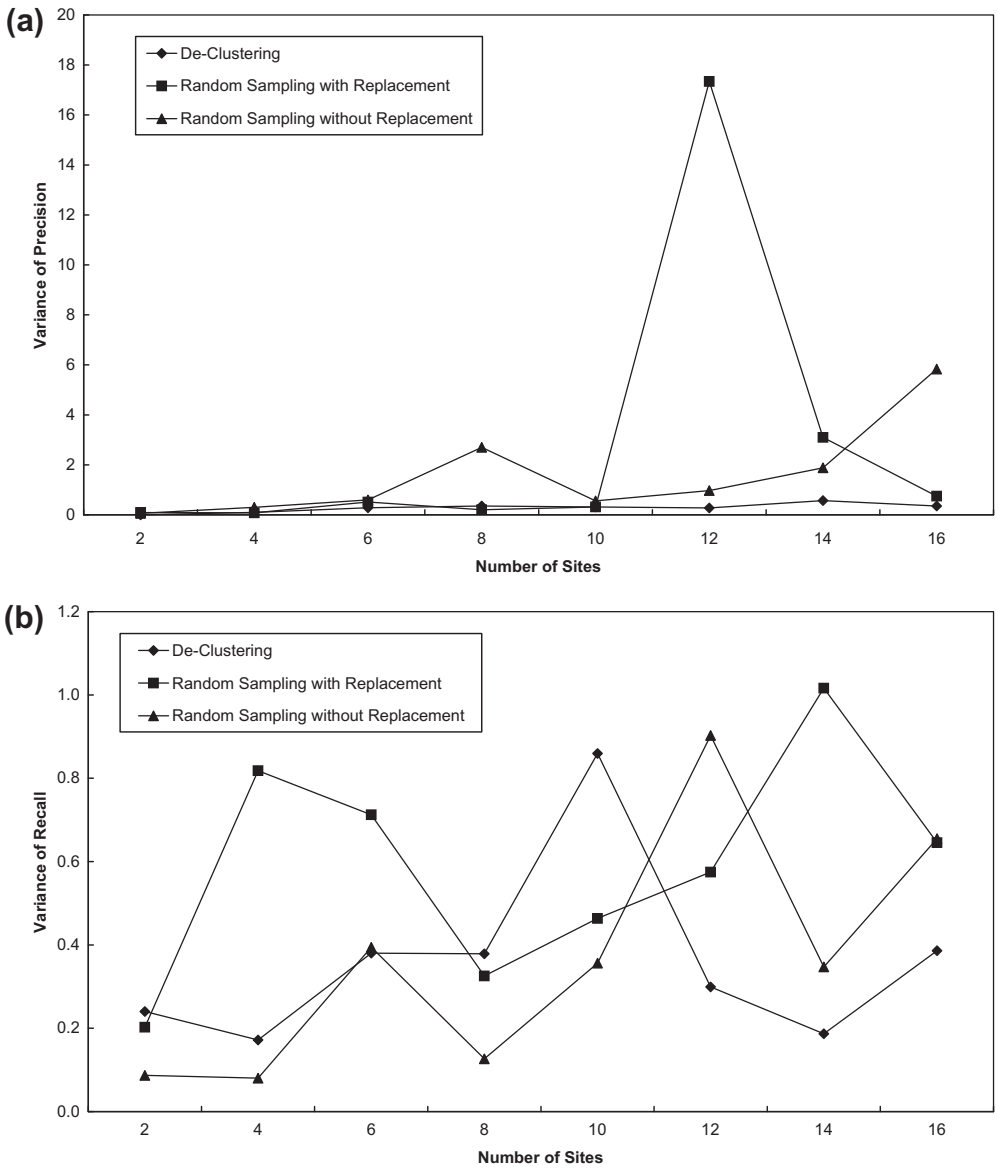


Fig. 19. Stability of precision and recall rates of all individual sites for various levels of site distribution, where higher variance means lower stability. The experiment was based on dataset D1200K.T10.I4 with a minimum support threshold of 0.2%.

4.5. Performance improvement

Based on the results of the experiment in Section 4.3, the data de-clustering algorithm may produce a less precise aggregate mining result when the minimum support threshold is low, the size of dataset is small, and the number of site distributions is high. To improve the precision rate of the aggregate mining result, two refining processes are proposed to check the possibility of improving the precision rate by trading off an acceptable decrease in the recall rate. For consistency, the three worst cases from the experiment in Section 4.3 are selected as the test bed: (1) D1200K.T10.I4 with a support of 0.1% (Table 3), (2) D200K.T10.I4 with a support of 0.1% of (Table 4), and (3) D200K.T10.I8 with a support of 0.3% (Table 5). In addition, the experiment results show the all curves of the precision rate were improved from the original value to 100%. As an improvement of the precision rate may deteriorate the recall rate, it is difficult to measure the effectiveness of the process. Therefore, another measurement is needed to evaluate the effectiveness of improvement that considers both precision and recall rates at the same time. *F*-Measure [9], shown in Eq. (5), is adopted as a harmonic mean to evaluate the effectiveness of the total improvement

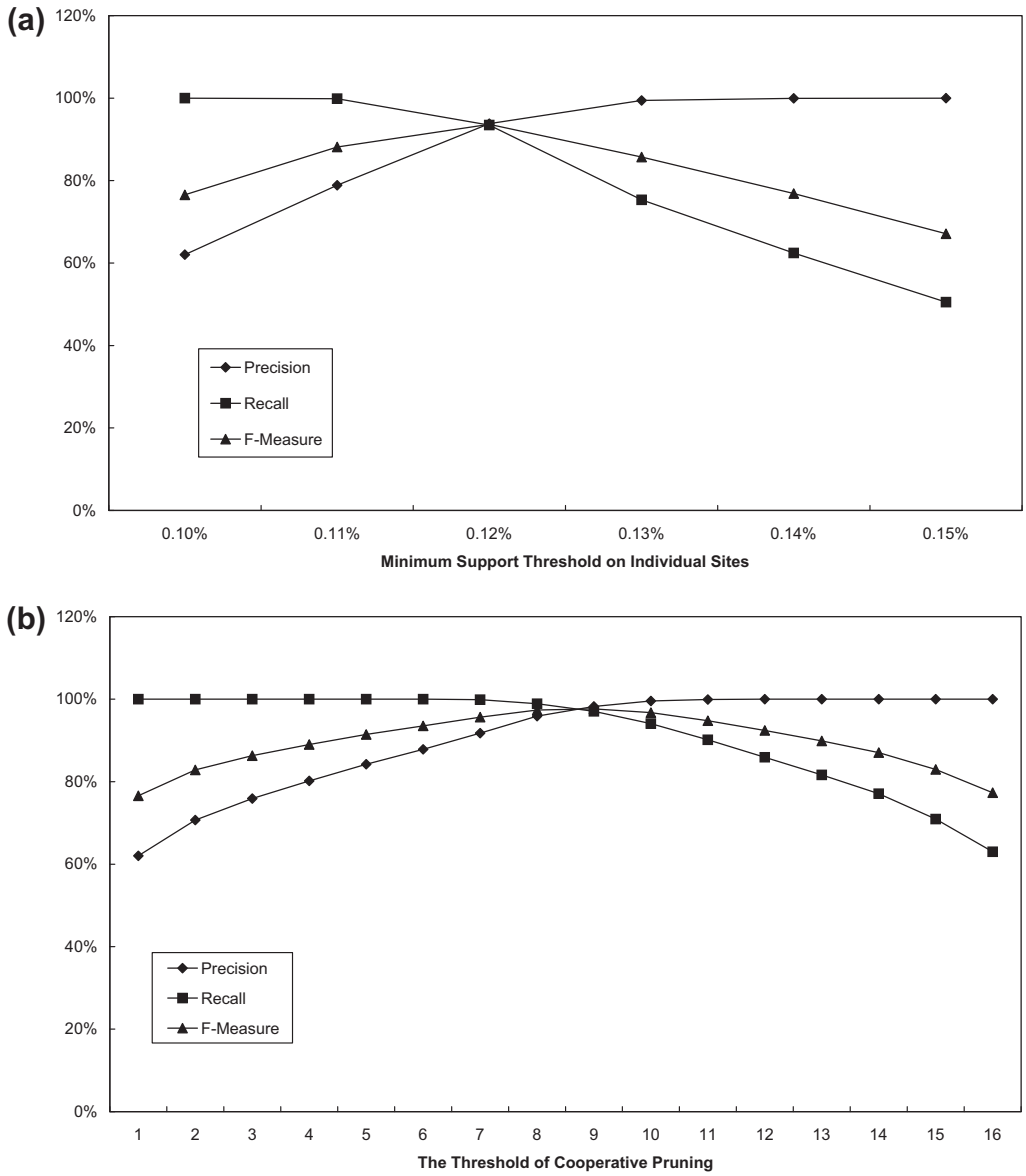


Fig. 20. Performance improvement based on dataset D1200K.T10.I4 with a minimum support threshold of 0.1% and 16 distribution sites. The leftmost points are the baseline results and other points are improved results. (a) Improvement obtained by applying higher support threshold to individual sites process. (b) Improvement obtained by cooperative pruning.

$$F\text{-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)$$

The two improvement processes are described below.

(1) Applying higher support threshold to individual sites.
Intuitively, a higher minimum support threshold can be applied to individual sites to refine their precision rates. For instance, if a support threshold of 0.1% is applied to the original dataset, then a finer precision rate of mining results can be expected by applying a slightly higher support threshold, such as 0.15%, to each individual site in the distributed mining environment.

(2) Cooperative pruning.
In the proposed approach, since all partitions are similar to the original dataset and to each other, it can be assumed that with the same support threshold, the frequent itemsets of one distribution site might be similar to those of other sites. Cooperative pruning is thus proposed to filter out itemsets that are frequent in just one or a few sites. For example, pruning can preserve just the itemsets that are frequent in more than 50% of the distribution sites.

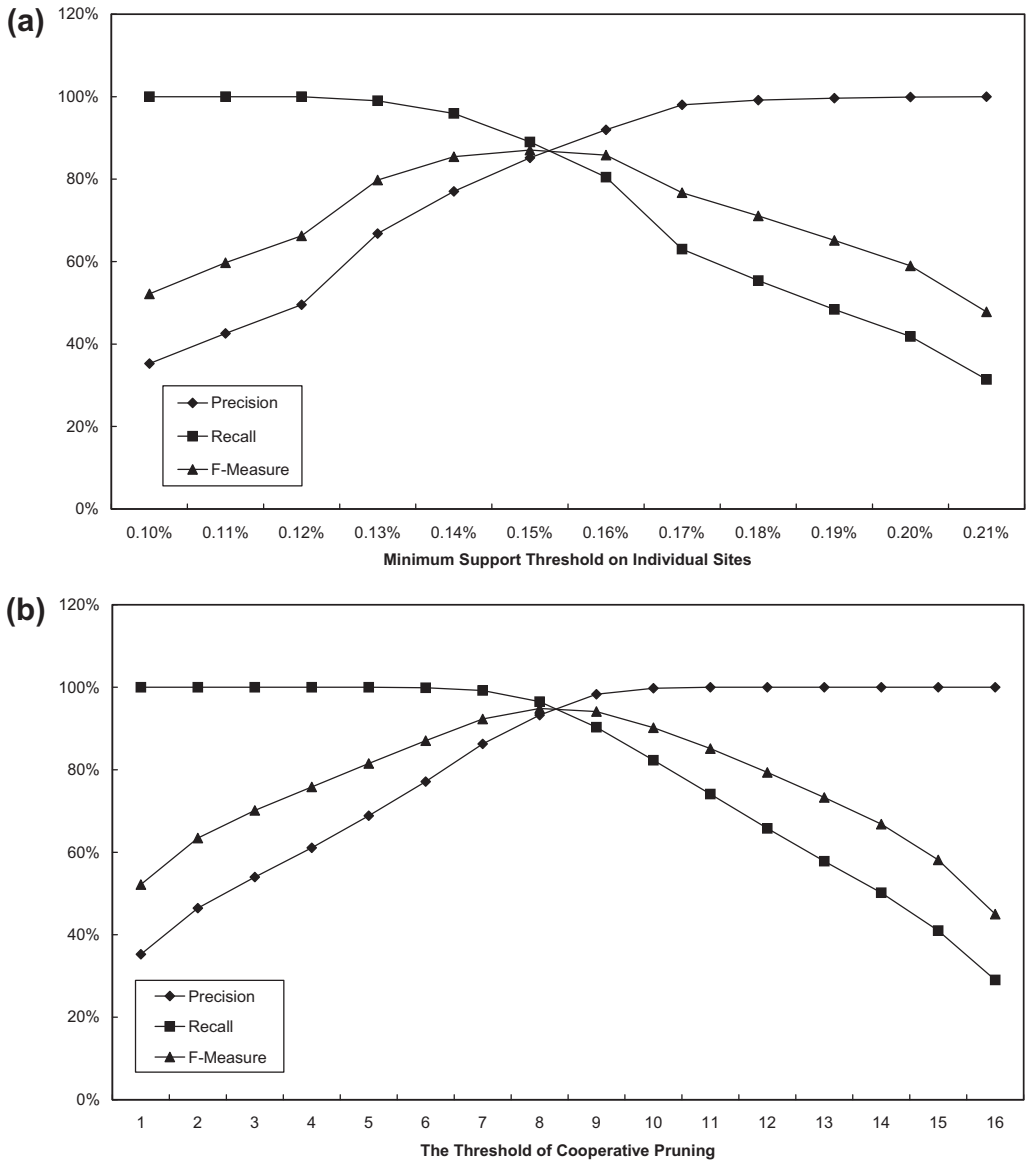


Fig. 21. Performance improvement based on dataset D200K.T10.I4 with a minimum support threshold of 0.1% and 16 distribution sites. The leftmost points are the baseline results and other points are improved results. (a) Improvement obtained by applying higher support threshold to individual sites process. (b) Improvement obtained by cooperative pruning.

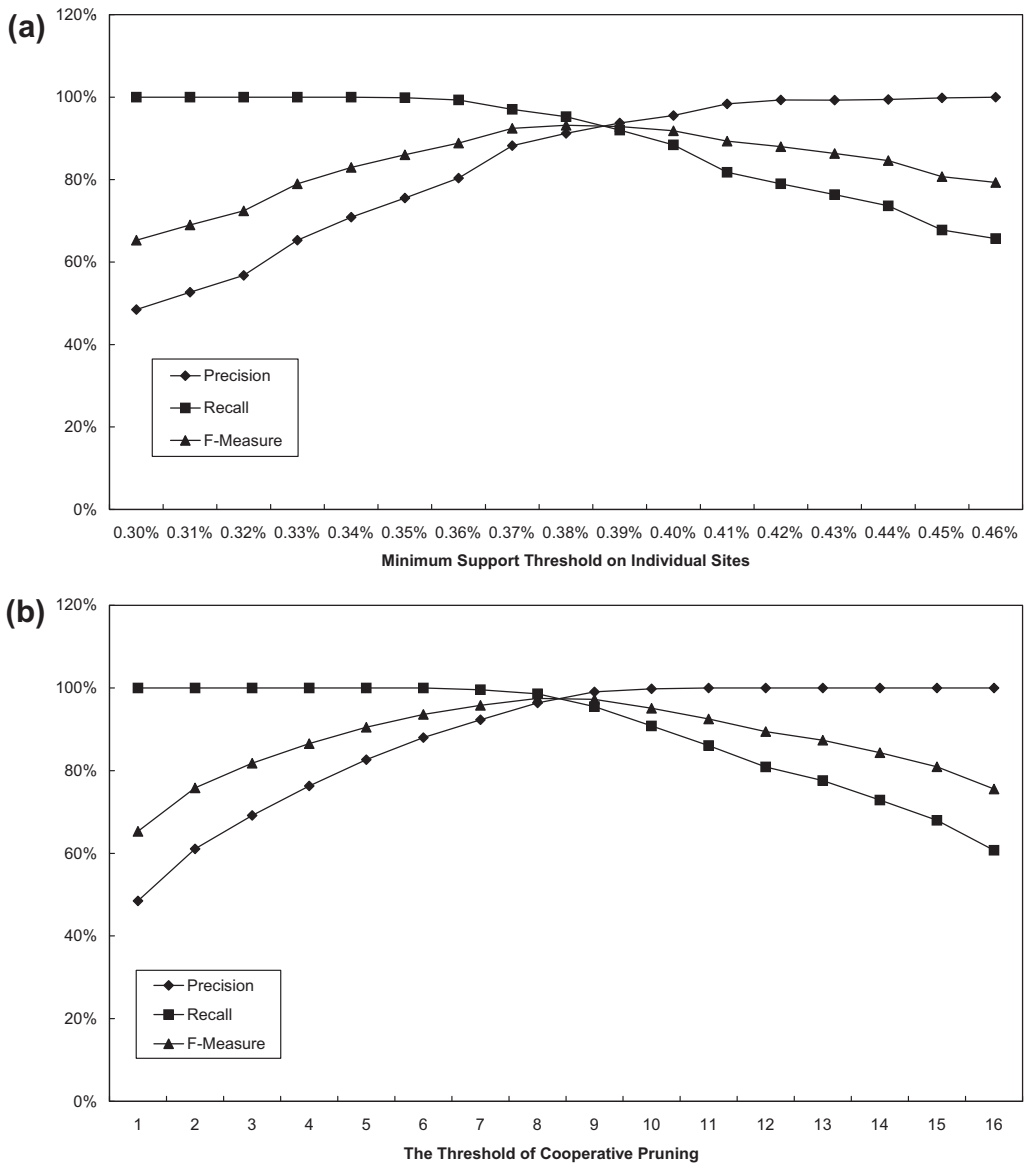


Fig. 22. Performance improvement based on dataset D200K.T10.I8 with a minimum support threshold of 0.3% and 16 distribution sites. The leftmost points are the baseline results and other points are improved results. (a) Improvement obtained by applying higher support threshold to individual sites process. (b) Improvement obtained by cooperative pruning.

The experimental results are shown in Figs. 20–22. The results in part (a) depict the first process (applying higher support threshold to individual sites) and those in part (b) illustrate the second process (cooperative pruning). In the experimental design of the first process, the minimum support threshold was gradually increased by a fixed amount of 0.01% until the precision rate reached 100%. *F*-Measure was used to assess the effectiveness of improvement during the process. Similarly, the threshold of cooperative pruning was gradually increased and the changes were evaluated using *F*-Measure.

The experimental results show that the proposed processes substantially improve the effectiveness of the mining results. Moreover, the adopted three worst cases of the experiment in Section 4.3were upgraded to a satisfactory level, as the precision and recall rates both exceeded 90%. The advantages and disadvantages of the two processes are discussed below.

(1) The advantages/disadvantages of the first process.
The major advantage of the first process is that it does not require any additional process to refine the mining result. However, it is difficult to find the support threshold which maximizes the *F*-Measure without a pre-test. Based on the experimental results, the different datasets and support threshold settings need different scales of support increase to maximize effectiveness.

(2) The advantages/disadvantages of the second process.
Cooperative pruning requires all distribution sites to vote for distinguishing frequent itemsets at the last stage of mining. This voting process exchanges the maximal frequent itemsets instead of the transactional data. This can be performed very quickly, and the cost can be ignored. From the effectiveness perspective, cooperative pruning can return almost perfect results (the precision and recall rates exceeded 95%). The results can maintain an impressive 100% recall to improve precision in many cases. Although it seems that finding the optimal cooperative pruning threshold is difficult, the obtained empirical rule suggests that using about 50% of the distribution sites as the cooperative pruning threshold produces acceptable results.

5. Discussion

The experiment results show that the proposed de-clustering algorithm can be used with conventional association rule miners to perform distributed mining tasks. The primary contribution of the proposed approach is eliminating the inter-site communication cost during frequent itemsets generation, which is required in other distributed mining algorithms. Based on the results, only the number of different items and the data size affect the de-clustering performance, with both factors increasing linearly.

The proposed approach consumes almost the same amount of time to de-cluster the dataset into subgroups regardless of data complexity, which implies that the data de-clustering is very stable. The increased speed in discovering frequent itemsets is significant, especially when the dataset is enormous. The correctness verification indicates that even though the precision rate is poor when the data size is small, the aggregate mining results still fully cover the complete frequent itemsets. Fortunately, the de-clustering algorithm can be considered as a sampling method to produce approximate results for the complete frequent itemsets. In addition, with increasing source data size, the precision and recall rate of each participant asymptotically approach the ideal case.

Inadequate results can be further refined using the proposed two improvement processes to reach a satisfactory level without any modification of the algorithm. Therefore, the proposed approach efficiently provides comprehensive frequent itemsets generation for distributed association rule mining.

6. Conclusions and future research

Association rule mining is useful in selective marketing, decision analysis, and business intelligence. However, due to the huge target data volume, most algorithms are time-consuming. Therefore, distributed algorithms are required to boost performance. Existing distributed architectures and algorithms suffer from an overhead in inter-site communications among processors or require a large amount of space to maintain the local support counts of a large number of candidate sets. In this paper, a distributed approach that de-clusters the transaction database into partitions for further mining tasks was presented. The purpose is to eliminate inter-site communications after distributing all subtasks. That is, communications occur only at the very beginning and at the end of the process. As a result, there is no communication cost during the itemset generation process, which makes the proposed framework very efficient and effective.

The proposed approach is based on the concept of data de-clustering, in which all the transaction data are de-clustered into partitions, such that all partitions are not only similar to each other, but also to the original group. The subtasks of itemset generation are then assigned to the participating sites using a round-robin approach. With task assignment, the load of each site can be highly balanced and the frequent itemsets of each site can be quickly obtained independently.

An empirical study was conducted for the proposed approach. It was found that the data de-clustering is satisfactory and that the results obtained can cover the same itemsets as those obtained from a single computer site. In addition, an improvement in distributed data mining was apparent. The proposed method can be considered as a sampling approach to discover approximate mining results with limited computation and time costs. Finally, experiments indicate that inadequate mining results can be enhanced to an almost perfect level.

Although the proposed method has some advantages for mining association rules, there are still a number of issues worthy of further investigation. For example, since there is no opportunity for the partitions generated to be skewed or imbalanced, prior distributed association rule mining algorithms could be combined with the proposed method to avoid unstable or worst cases. Moreover, only cases regarding centralized database de-clustering were here considered. Further studies on distributed database de-clustering could be conducted. Finally, as the amount of data continues to grow, refining the proposed approach to be adaptive for incrementally mining association rules with temporal properties taken into account could be considered.

Acknowledgments

This work was supported in part by the National Science Council (NSC), Taiwan, ROC, under Grants NSC 95-2416-H-327-018, No. 98-2410-H-327-020-MY3, NSC 95-3113-P-006-004, and NSC 95-2221-E-006-307-MY3. This study was conducted under the Cloud Computing Technologies & Industry Development program of the Institute for Information Industry, which is subsidized by the Ministry of Economy Affairs of the Republic of China.

References

[1] C.C. Aggarwal, C. Procopiuc, P.S. Yu, Finding localized associations in market basket data, IEEE Transactions on Knowledge and Data Engineering 14 (1) (2002) 51–62.
[2] C.C. Aggarwal, P.S. Yu, A new approach to online generation of association rules, IEEE Transactions on Knowledge and Data Engineering 13 (4) (2001) 527–540.
[3] C.C. Aggarwal, P.S. Yu, Mining associations with the collective strength approach, IEEE Transactions on Knowledge and Data Engineering 13 (6) (2001) 863–873.
[4] C.C. Aggarwal, P.S. Yu, Redefining clustering for high-dimensional applications, IEEE Transactions on Knowledge and Data Engineering 14 (2) (2002) 210–225.
[5] R. Agrawal, J.C. Shafer, Parallel mining of association rules, IEEE Transactions on Knowledge and Data Engineering 8 (6) (1996) 962–969.
[6] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of the 20th International Conference on Very Large Data Bases, 1994, pp. 487–499.
[7] M.J. Atallah, S. Prabhakar, (Almost) Optimal parallel block access for range queries, Information Sciences 157 (2003) 21–31.
[8] S. Ayubi, M.K. Muyeba, A. Baraani, J. Keane, An algorithm to mine general association rules from tabular data, Information Sciences 179 (20) (2009) 3520–3539.
[9] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, 1999. pp. 74–84.
[10] F. Bodon, A fast APRIORI implementation, in: Proceedings of IEEE ICDM Workshop on Frequent Itemset Mining Implementations, 2003.
[11] C.C. Chang, C.Y. Lin, Reversible steganographic method using SMVQ approach based on declustering, Information Sciences 177 (8) (2007) 1796–1805.
[12] C.M. Chen, R. Bhatia, R.K. Sinha, Multidimensional declustering schemes using golden ratio and Kronecker sequences, IEEE Transactions on Knowledge and Data Engineering 15 (3) (2003) 659–670.
[13] C.M. Chen, C.T. Cheng, From discrepancy to declustering: near-optimal multidimensional declustering strategies for range queries, Journal of ACM 51 (1) (2004) 46–73.
[14] D.W. Cheung, J. Han, V.T. Ng, A.W. Fu, Y. Fu, A fast distributed algorithm for mining association rules, in: Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems, 1996, pp. 31–42.
[15] D.W. Cheung, V.T. Ng, A.W. Fu, Y. Fu, Efficient mining of association rules in distributed databases, IEEE Transactions on Knowledge and Data Engineering 8 (6) (1996) 911–922.
[16] D.W. Cheung, Y. Xiao, Effect of data distribution in parallel mining of associations, Data Mining and Knowledge Discovery 3 (3) (1999) 291–314.
[17] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, H. Kargupta, Distributed data mining in peer-to-peer networks, IEEE Internet Computing 10 (4) (2006) 18–26.
[18] D. Dudek, RMAIN: association rules maintenance without reruns through data, Information Sciences 179 (24) (2009) 4123–4139.
[19] M.T. Fang, R.C.T. Lee, C.C. Chang, The idea of de-clustering and its applications, in: Proceedings of 12th International Conference on Very Large Data Bases, 1986, pp. 181–188.
[20] H. Ferhatosmanoglu, A.S. Tosun, G. Canahuate, A. Ramachandran, Efficient parallel processing of range queries through replicated declustering, Distributed and Parallel Databases 20 (2) (2006) 117–147.
[21] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[22] J. Han, H. Cheng, D. Xin, X. Yan, Frequent pattern mining: current status and future directions, Data Mining and Knowledge Discovery 15 (1) (2007) 55–86.
[23] J. Han, M. Kimber, Data Mining: Concepts and Techniques, second ed., Morgan Kaufman, 2006.
[24] B. He, K.C.C. Chang, Automatic complex schema matching across web query interfaces: a correlation mining approach, ACM Transactions on Database Systems 31 (1) (2006) 346–395.
[25] T. Hu, S.Y. Sung, H. Xiong, Q. Fu, Discovery of maximum length frequent itemsets, Information Sciences 178 (1–2) (2008) 69–87.
[26] Y.M. Huang, J.N. Chen, S.C. Cheng, A method of cross-level frequent pattern mining for web-based Instruction, Educational Technology and Society 10 (3) (2007) 305–319.
[27] Y. Huang, H. Xiong, W. Wu, P. Deng, Z. Zhang, Mining maximal hyperclique pattern: a hybrid search strategy, Information Sciences 177 (3) (2007) 703–721.
[28] M. Kantarcioglu, C. Clifton, Privacy-preserving distributed mining of association rules on horizontally partitioned data, IEEE Transactions on Knowledge and Data Engineering 16 (9) (2004) 1026–1037.
[29] R.C.T. Lee, Clustering analysis and its applications, in: J.T. Toum (Ed.), Advances in Information System Science, vol. 8, Plenum Press, New York, 1981, pp. 169–292.
[30] A.J.T. Lee, C.S. Wang, An efficient algorithm for mining frequent inter-transaction patterns, Information Sciences 177 (17) (2007) 3453–3476.
[31] H. Liu, R. Setiono, Feature selection via discretization, IEEE Transaction on Knowledge and Data Engineering 9 (4) (1997) 642–645.
[32] H. Liu, X. Wang, J. He, J. Han, D. Xin, Z. Shao, Top-down mining of frequent closed patterns from very high dimensional data, Information Sciences 179 (7) (2009) 899–924.
[33] A. Nijenhuis, H. Wilf, Combinatorial Algorithms for Computers and Calculators, second ed., Academic Press, New York, 1978.
[34] K. Nørvåg, A study of object declustering strategies in parallel temporal object database systems, Information Sciences 146 (1–4) (2002) 1–27.
[35] M.E. Otey, S. Parthasarathy, C. Wang, A. Veloso, W. Meira Jr., Parallel and distributed methods for incremental frequent itemset mining, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 34 (6) (2004) 2439–2450.
[36] C. Papadimitriou, K. Steiglitz, Some complexity results for the traveling salesman problem, in: Proceedings of Eighth Annual ACM Symposium on Theory of Computing, 1976, pp. 1–9.
[37] B.H. Park, H. Kargupta, Distributed data mining: algorithms, systems, and applications, in: N. Ye (Ed.), The Handbook of Data Mining, Lawrence Erlbaum, 2003, pp. 341–358.
[38] R. Ramakrishnan, B.C. Chen, Exploratory mining in cube space, Data Mining and Knowledge Discovery 15 (1) (2007) 29–54.
[39] A. Schuster, R. Wolff, Communication-efficient distributed mining of association rules, Data Mining and Knowledge Discovery 8 (2) (2004) 171–196.
[40] A. Schuster, R. Wolff, D. Trock, A high-performance distributed algorithm for mining association rules, Knowledge and Information Systems 7 (4) (2005) 458–475.
[41] S. Skiena, Gray code (§1.5.3), in: Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Addison-Wesley, Reading, MA, 1990, pp. 42–43. 149.
[42] S.K. Tanbeer, C.F. Ahmed, B.S. Jeong, Y.K. Lee, Efficient single-pass frequent pattern mining using a prefix-tree, Information Sciences 179 (5) (2009) 559–583.
[43] S.K. Tanbeer, C.F. Ahmed, B.S. Jeong, Y.K. Lee, Sliding window-based frequent pattern mining over data streams, Information Sciences 179 (22) (2009) 3843–3865.
[44] H. Toivonen, Sampling large databases fro association rules, in: Proceedings of the 22nd International Conference on Very Large Data Bases, 1996, pp. 134–145.
[45] A.S. Tosun, Threshold-based declustering, Information Sciences 177 (5) (2007) 1309–1331.
[46] A.S. Tosun, Analysis and comparison of replicated declustering schemes, IEEE Transactions on Parallel and Distributed Systems 18 (11) (2007) 1578–1591.
[47] A.S. Tosun, Multi-site retrieval of declustered data, in: Proceedings of the 28th International Conference on Distributed Computing Systems, 2008, pp. 486–493.

[48] Y.J. Tsay, T.J. Hsu, J.R. Yu, FIUT: a new method for mining frequent itemsets, Information Sciences 179 (11) (2009) 1724–1737.
[49] F.H. Wang, On discovery of soft associations with "most" fuzzy quantifier for item promotion applications, Information Sciences 178 (7) (2008) 1848–1876.
[50] R. Wolff, A. Schuster, Association rule mining in peer-to-peer systems, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 34 (6) (2004) 2426–2438.
[51] M.J. Zaki, Parallel and distributed association mining: a survey, IEEE Concurrency 7 (4) (1999) 14–25.
[52] S. Zhong, Privacy-preserving algorithms for distributed mining of frequent itemsets, Information Sciences 177 (2) (2007) 490–503.