Distributed Frequent Closed Itemsets Mining

Chun Liu, Zheng Zheng, Kai-Yuan Cai, Shichao Zhang
School of Automation Science and Electrical Engineering
Beijing University of Aeronautics and Astronautics, Beijing, 100083, China
liuchun@asee.buaa.edu.cn, {zhengz, kycai}@buaa.edu.cn, zhangsc@mailbox.gxnu.edu.cn

Abstract
As many large organizations have multiple data sources and the scale of dataset becomes larger and larger, it is inevitable to carry out data mining in the distributed environment. In this paper, we address the problem of mining global frequent closed itemsets in distributed environment. A novel algorithm is proposed to obtain global frequent closed itemsets with exact frequency and it is shown that the algorithm can determine all the global frequent closed itemsets. A new data structure is developed to maintain the closed itemsets. Then an efficient implementation is provided based on the structure. Experimental results show that the proposed algorithm is effective.

1. Introduction
Association rule mining (ARM), one of the most popular topics in the KDD field, regards the extractions of association rules from a transaction database. In the past years, a lot of work has been dedicated to ARM and it has been successfully exploited in many fields such as market basket analysis and user visiting pattern analysis in network. ARM includes two phases. The first and the most computationally expensive step is frequent itemsets Mining (FIM). When the FIM is complete, it is straightforward to obtain association rules. Therefore, most of the researches of ARM focus on FIM. One of the main issues emerging from these researches concentrates on the combination explosion, i.e. the size of the collection of frequent itemsets will increase exponentially as the number of single items increases. To deal with this problem, closed itemsets mining has attracted the attention of many researchers in this community because it can not only provide complete but also condensed patterns.

With the advance in data gathering and storage technologies, the companies or organizations including many branches are inevitable to face the problem of mining distributed datasets. The reasons are as follows [1]. First, it might be unrealistic to collect all data from different data sources for centralized processing due to the huge volume of data. Second, it may be that some data sources are not willing to share their data but the analysis results because of data privacy. Finally, local analysis is also useful for the local data sources in real-world application. Moreover, as the scale of the data becomes larger and larger, the need of scalable and high performance systems by parallel and distributed manner is also increasing. As a result, the development of the parallel and distributed data mining algorithms becomes more and more appealing.

In this paper, we address the issue of mining frequent closed itemsets in a distributed setting. There is little work reported to deal with this problem. It is first discussed in [2]. In [2], the author has proved that all the global closed itemsets can be obtained by merging the local closed itemsets in the condition that the min-support is 0. However, the conclusion does not go for the case in which the min-support is larger than 0, and the implementation is not provided in that work. Moreover, the naive way of sending all the local closed itemsets to the master might cause heavy communications and the large sets of closed itemsets will also make the central mergence unfeasible. In this paper we deal with the problem regardless of the value of parameter min-support. We focus on synthesizing local closed itemsets to global closed itemsets since local closed itemsets can be easily generated by the existing techniques in each partition.

The main contributions of this paper are as follows. First, we propose an algorithm called DFCIM (Distributed Frequent Closed Itemsets Mining) to obtain global frequent closed itemsets with exact frequency in distributed environment and we prove that it can determine all the global frequent closed itemsets. Second, a new data structure FCItrees (Frequent Closed Itemsets trees) is developed to maintain all the closed itemsets. The characteristic of this structure facilitates to locate and output the closed itemsets. Finally, we provide an effective implementation based on the structure and the experimental results have shown the validity.

The rest of the paper is organized as follows. Section
2 presents the related work. In section 3, we provide the preliminaries which are used throughout the paper. Section 4 proposes the algorithm DFCIM. Section 5 discusses the implementation based on FCITrees in detail. And finally, the validity and the conclusion are provided in section 6 and section 7, respectively.

2 Related work

Frequent itemsets mining (FIM) has been extensively studied in the past years. Apriori [3] and FP-growth [4] are two most representative algorithms. Apriori is a level-wise algorithm which can generate many candidate itemsets and need repeated data scans. The algorithm FP-Growth first compresses the dataset into a highly condensed, much smaller data structure, called FP-tree and adopts a pattern fragment growth strategy to avoid the repeated data scans and candidate itemsets generations. The experiments show that this technique is more efficient than the Apriori algorithm. Most of the work in this field is based on the two algorithms.

As FIM usually causes the large size of frequent itemsets, many approaches have been investigated to deal with the issue. The emerging work can be categorized into maximal frequent itemsets mining and closed itemsets mining. An itemset is called a maximal itemset when it has no superset in the given set regardless of the frequency information [5, 6]. As a result, when we deduce the subsets from the maximal itemset, we can not get the exact frequency information and only know that it is larger or equal to that of the maximal itemset. Nevertheless, closed itemsets mining also can provide condensed itemsets but no information loss. Much excellent work has been dedicated to it [7-13] such as A-close [12], CLOSE [8], CHARM [13], CLOSE+ [9], FPclose [11], TFP [7] and DCI-CLOSED [10]. However, all these algorithms focus on mining closed itemsets from the datasets but not synthesizing local frequent closed itemsets to global frequent closed itemsets, so they are not suitable for closed frequent itemsets mining in the distributed environments.

As an effective method to cope with the large size of data, parallel and distributed Mining algorithms have attracted much attention of the community. [14-19] have focused on parallel and distributed algorithms for mining association rules, but not the closed itemsets mining. In [1], an algorithm of synthesizing high frequency rules from different data sources based on weight model is proposed. Each branch sends the rules but the frequent itemsets to headquarter. It may cause excess communications due to that a frequent itemset can generate many rules. In [20], the author presents a method using kernel estimation to synthesize local patterns to global patterns.

3 Preliminaries

The problem of mining frequent itemsets from a transaction dataset can be stated as follows. Let \( I = \{x_1, x_2, \ldots, x_m\} \) be a set of \( m \) elements, called items. A subset \( X \subseteq I \) is called an itemset. The \( k \)-itemset is an itemset with \( k \) different items. All the items in any itemset are in ascending lexicographic order, and we call the smallest item, head item. A transaction is denoted as \( t \) and \( t \) is also an itemset. Transaction dataset \( DS \) is a finite sequence of transactions. Given itemsets \( X \) and the transaction \( t \), if \( X \subseteq t \), it is said that \( X \) is contained by the transaction. The support of itemset \( X \), \( \text{SUP}(X) \), is the number of transactions that contain itemset \( X \).

**Definition 1.** An itemset \( X \) is called a frequent itemset if \( \text{SUP}(X) \geq ms \times n \), where \( n \) is the total number of transactions and \( ms \) is the user specified parameter i.e. min-support.

**Definition 2.** An itemset \( X \) is a closed itemset if there exist no itemset \( X' \) such that: 1), \( X \subset X' \); 2), \( \text{SUP}(X) = \text{SUP}(X') \).

Apparently, given an itemset \( X \), if \( \text{SUP}(X) \) is larger than that of any superset of \( X \), then \( X \) is a closed itemset. From the definition of closed itemset, we deduce four important lemmas that are going to be useful in the following sections.

**Lemma 1.** Given an itemset \( X \), if it has two supersets \( Y_1 \) and \( Y_2 \) such that \( Y_1 \not\subset Y_2 \) and \( Y_2 \not\subset Y_1 \) and \( Y_1 \cap Y_2 = X \), then \( X \) must be a closed itemset.

**Proof.** We suppose that \( X \) is not a closed itemset, then it holds that \( \exists Y, X \subset Y, \text{SUP}(X) = \text{SUP}(Y) \).

Since \( X \subset Y_1 \Rightarrow \text{SUP}(X) \geq \text{SUP}(Y_1) \),

\( X \subset Y_2 \Rightarrow \exists t_k, Y_1 \not\subset t_k, X \subset t_k \),

then \( \text{SUP}(X) \geq \text{SUP}(Y_1) + 1 \). Similarly, we can deduce \( \text{SUP}(X) \geq \text{SUP}(Y_2) + 1 \). Therefore, \( \text{SUP}(Y) > \text{SUP}(Y_1), \text{SUP}(Y) > \text{SUP}(Y_2) \). Since \( \text{SUP}(Y) > \text{SUP}(Y_1), X \subset Y \), and \( X \subset Y_1 \Rightarrow Y \subset Y_1 \) (otherwise, \( \text{SUP}(X) > \text{SUP}(Y) \)). Similarly, we have \( Y \subset Y_2 \). Thus, \( Y \subseteq X \). It conflicts with the hypothesis, so \( X \) must be a closed itemset. \( \square \)

**Lemma 2.** Given an itemset \( X \) which is not a closed itemset, if it has two supersets \( Y_1 \) and \( Y_2 \), then it holds that \( Y_1 \subset Y_2 \) or \( Y_2 \subset Y_1 \).

**Proof.** If \( Y_1 \not\subset Y_2 \) or \( Y_2 \not\subset Y_1 \), then we can get the result that \( X \) must be a closed itemsets from lemma 1. \( \square \)

In lemma 2, for the sake of simplicity, we limit the setting with two itemsets. From this lemma, it is easy to find that the one with larger frequency between \( Y_1 \) and \( Y_2 \) contains all the frequency information of \( X \). However, the result can be generalized to the case of \( K \) itemsets (\( K \geq 2 \)). We show
Lemma 3. Given an itemset \( X \) which is not a closed itemset, if there are \( K(K \geq 2) \) different itemsets containing \( X \), then the one with maximal frequency contains all the frequency information of \( X \).

Lemma 4. Given a set of closed itemsets \( S \) and a closed itemset \( X \), if \( X \in S \), then there is no new closed itemsets generated after \( X \) is added to \( S \).

Proof. Suppose that itemset \( U \) be generated. According to lemmas 3, there must exist itemset \( T \) in \( S \) which contains all the frequency information of \( U \) and satisfies that \( U \subseteq X \), \( T \subseteq X \), \( U \subseteq T \), and \( SUP(U) = SUP(T) \). When \( X \) is added, since \( U \subseteq X,T \subseteq X \Rightarrow SUP(U) = SUP(U) + 1 \) and \( SUP(T) = SUP(T) + 1 \) (\( SUP(X) \) denotes the updated support of itemset \( X \)). Thus, \( SUP(U) = SUP(T) \), and this means that \( U \) is not a closed itemset after \( X \) arrives. So we have this lemma proved. □

Definition 3. For an itemset \( X = \{x_1, x_2, \ldots, x_k\} \), its suffix-itemsets are \( x_1, x_1', x_2, x_2', \ldots, x_k, x_k' \) where \( x_1' = \{x_1, x_2, \ldots, x_k\} \); \( x_2' = \{x_2, x_3, \ldots, x_k\} \); \ldots; \( x_i' = \{x_i, \ldots, x_k\} \); \ldots; \( x_k' = \{x_k\} \). The corresponding suffix-itemset of item \( x_i (1 \leq i \leq k) \) is the suffix-itemset which contains the item \( x_i \) as the head item. For example, the corresponding suffix-itemset of \( x_2 \) is \( x_2' = \{x_2, \ldots, x_k\} \).

In this paper we discuss the problem of frequent closed itemsets mining in the distributed environment as depicted in figure 1. And we assume that there are \( N(N \geq 2) \) partitions and each partition has carried out the local closed itemsets mining with the same min-support. Our goal is to merge the local closed itemsets to obtain the global frequent closed itemsets with exact frequency. In the following sections, we suppose each partition owns the dataset \( D_i \). And we denote with \( SUP_i(X) \) the global frequency of itemset \( X \), with \( SUP_i(X) \) the frequency of itemset contained in \( i_{th} \) partition.

4 The algorithm DFCIM

In this section, we provide the algorithm DFCIM. In this algorithm, only the sets of frequent closed itemsets which are rather smaller than that of the total closed itemsets are transmitted between the partitions and the master.

In the following discussion, we denote with \( S_i \) the set of frequent closed itemsets, \( S_i' \) the set of infrequent closed itemsets in \( D_i \), and with \( S \) the set of global frequent closed itemsets. For simplicity, the following discussion is limited to the setting with only two data sources \( D_1 \) and \( D_2 \).

Given two sets of frequent closed itemsets \( S_1 \) and \( S_2 \), we first introduce two functions \( C_1(S_1, S_2) \) and \( C_2(S_1, S_2) \) which are defined as follows.

- \( C_1(S_1, S_2) \) returns all the closed itemsets after the two sets \( S_1 \) and \( S_2 \) are combined together.
- \( C_2 = \bigcup_{Y \in S_1} (Y \cap X \in S_2) \), where the symbol * means that if there are the same intersections with different frequency, then only the one with the maximal frequency is maintained.

From the above definitions, we have the following lemma.

Lemma 5. Given two sets \( S_1 \) and \( S_2 \) which are the sets of closed itemsets from different partitions, then it holds that

\[
C_1(S_1, S_2) = S_1 \bigcup S_2 \bigcup C_2(S_1, S_2)
\]

Proof. In order to merge the sets of \( S_1 \) and \( S_2 \), we add each itemset of \( S_1 \) to \( S_2 \). For a closed itemset \( X \) of \( S_1 \), if \( X \in S_2 \), from lemma 4, there would be no new closed itemsets generated. Otherwise, \( X \) must be a new closed itemset for \( S_2 \). And in the latter case, according to lemma 1, the intersections between itemsets of \( S_1 \) and those of \( S_2 \) might become new closed itemsets. Furthermore, for the same intersections with different frequency, only the one with maximal frequency is interesting according to lemma 3. Note that, \( C_2(S_1, S_2) \) can return all the intersections that would become new closed itemsets. Therefore, we can conclude that the combination result can be obtained by \( S_1 \bigcup S_2 \bigcup C_2(S_1, S_2) \). □

Lemma 6. A global frequent closed itemset must be a local frequent closed itemset or be subsumed by the local frequent closed itemsets.

Proof. Let \( X \) be the global frequent closed itemset. If \( X \) does not occur in any set \( S_i \), i.e. it is neither a local frequent closed itemset nor the one subsumed by the local frequent closed itemsets, then \( SUP_i(X) < ms \times n_i \) (\( n_i \) is the number of transactions in \( D_i \)) for all \( i = 1, 2, \ldots, N \), and \( SUP(X) < ms \times n(SUP(X)) = \sum_{i=1}^{N} SUP_i(X) \times n = \sum_{i=1}^{N} n_i \). As a result, \( X \) can not be a global frequent
closed itemset. Therefore, we have the lemma proved. □

From lemma 6, we can see that a global frequent closed itemset must be included in \( C_1(S_1, S_2) \) or subsumed by the itemsets in \( C_1(S_1, S_2) \). And we have it as the following remark.

**Remark 1.** A global frequent closed itemset must be included in \( C_1(S_1, S_2) \) or subsumed by the itemsets in \( C_1(S_1, S_2) \).

**Example:** Let \( D_1 = \{ ABC, ABC, ABC, CD \} \), \( D_2 = \{ CDE, CDE, CDE, ABC, CE \} \), and the min-support \( ms = 2 \). Then we can obtain \( S_1 = \{ C : 4, CDE : 3 \} \), \( S_2 = \{ CD : 1, CE : 4, CDE : 3 \} \), \( S = \{ C : 8, CD : 4, CE : 4, ABC : 4, ABC : 4 \} (ms = 4) \). However, it is easy to get that \( C_1(S_1, S_2) = \{ C : 8, CE : 4, ABC : 3, CDE : 3 \} \), which is carried in the master and global frequent closed itemset \( CD \) is included by the itemset \( CDE \).

Moreover, for these global frequent closed itemsets that still occur as subsets of the itemsets in \( C_1(S_1, S_2) \) like the itemset \( CD \) in example 1, we have the following lemma.

**Lemma 7.** For a global frequent closed itemset \( X \) that occurs only as subset of the itemsets in \( C_1(S_1, S_2) \), it must occur in one set \( S_i^0 (i = 1 \) or \( 2) \).

**Proof.** From lemma 6, \( X \) must occur in one set \( S_i \). Thus, let us suppose that it occurs in set \( S_1 \). Clearly, it is meaningless to consider that \( X \) occurs in \( S_2 \). If \( X \) is not in \( S_2 \), we can see that \( X \) cannot be a global frequent closed itemset because \( S_1 \) contains all its frequency, and if \( X \) is a global frequent closed itemset, it must occur as a closed itemset in \( C_1(S_1, S_2) \). If \( X \) does not occur in \( D_2 \), then it has no chance to become a global frequent closed itemset according to the definition of closed itemsets. So for the global frequent closed itemsets \( X \), it should only occur in \( S_2 \). Similarly, if \( X \) occurs in \( S_2 \), it must occur in \( S_1 \). □

From above lemma, we can see that, the reason for the global itemsets that still occur as subsets in \( C_1(S_1, S_2) \) is that they are not frequent in some partitions and contained by the corresponding sets \( S_i^0 \). In example 1, the global closed itemset \( CD \) is not frequent in \( D_1 \) and contained in \( S_1 \). So we can obtain all the global frequent closed itemsets by the following theorem.

**Theorem 8.** Let \( T_1 = C_2(C_1(S_1, S_2), S_i^0) (i = 1, 2) \) and \( T_0 = \bigcup_{i=1}^{2}(T_i - C_2(T_i, S_i)) \), then \( S \subseteq C_1(S_1, S_2) \bigcup^* T_0 \).

**Proof.** From the definition of \( C_2(S_1, S_2) \), \( T_i \) contains all the intersections of the elements of \( C_1(S_1, S_2) \) and \( S_i^0 \). As a result, from lemma 7, the global frequent closed itemsets that only occur as subsets of some closed itemsets in \( C_1(S_1, S_2) \) must be included in \( T_i \). However, in \( T_i \) there may exist itemsets that also occur in \( S_i \). In that case, they should be eliminated by operation \( T_i - C_2(T_i, S_i) \) because its complete frequency information contained in \( D_i \) has been extracted in set \( S_i \). Notice that some closed itemsets may be frequent in some partitions but not in the others, and clearly they are also included in \( T_i \) in that case. And then by the operation \( \bigcup_{i=0}^{2}(T_i - C_2(T_i, S_i)) \) their exact frequencies can be obtained. So we can conclude that \( C_1(S_1, S_2) \bigcup^* T_0 \) can determine all the global frequent closed itemsets with their exact frequencies. However, some local frequent closed itemsets is not global frequent closed itemsets but included by \( C_1(S_1, S_2) \bigcup^* T_0 \), so it holds that \( S \subseteq C_1(S_1, S_2) \bigcup^* T_0 \). □

Considering the example 1, we can see that, \( T_1 = \{ CD \} \) in which \( CD \) is the global itemsets that occurs as subset in \( C_1(S_1, S_2) \), and \( T_2 = \{ ABC \} \) in which \( ABC \) is the itemset that occurs as frequent itemset in one partition but not in the other. In the meantime, the itemset \( CDE \) is not global frequent but contained in \( C_1(S_1, S_2) \bigcup^* T_0 \). We have obtained the above result in the setting with only two partitions, however, the above result can be extended to the case of \( N (N \geq 2) \) partitions by the following theorem.

**Theorem 9.** Let \( S_0 = C_2(S_1, S_2, \ldots, S_N) \), \( T_i = C_2(S_0, S_i^0) \), and \( T_0 = \bigcup_{i=1}^{N}(T_i - C_2(T_i, S_i)) \) \((i = 1, 2, \ldots, N)\), then \( S \subseteq S_0 \bigcup^* T_0 \) and \( S_0 \bigcup^* T_0 \) can determines all the global frequent closed itemsets.

**DFCIM Algorithm.** From the theorem 9, to obtain the global frequent closed itemset we can adopt the algorithm DFCIM shown in figure 2.

<table>
<thead>
<tr>
<th>Algorithm DFCIM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1:</strong> Each partition mines its partition to obtain the local closed itemsets.</td>
</tr>
<tr>
<td><strong>Step 2:</strong> The master merges the local frequent closed itemsets and hands out the result of mergence to each partition.</td>
</tr>
<tr>
<td><strong>Step 3:</strong> Each partition carries out the operation ( T_i - C_2(T_i, S_i) ) and sends the result to the master.</td>
</tr>
<tr>
<td><strong>Step 4:</strong> The master combines the results of ( T_i - C_2(T_i, S_i) ) and updates the set of ( S_i ). And then, a reduction operation is made to eliminate these itemsets which are not global frequent in the master.</td>
</tr>
</tbody>
</table>

Fig. 2. The algorithm DFCIM.

### 5 Implementations

In the algorithm DFCIM, the main operations are \( C_1(S_1, S_2) \) which is carried in the master and \( C_2(S_1, S_2) \)
which is carried in the partitions. And we can see that the main tasks are to determine whether an itemset is included in a set and whether the itemset has intersections with the itemsets of the given set. Thus, how to organize the itemsets and how to locate the intersections quickly are important issues. In the following, we provide our method. First, we present the new data structure FCItrees representing the set of closed itemsets, and then we discuss the approach to implement the operations of $C_1(S_1, S_2)$ and $C_2(S_1, S_2)$ based on the FCItrees.

### 5.1 FCItrees

**Definition 4.** Frequent Closed Itemsets trees (FCItrees) is a lexicographic summary data structure shown in figure 3.

![Fig. 3. The FCItrees.](image)

Clearly, FCItrees is similar to FP-tree[15], but it differs from FP-tree in that the nodes in FCItrees are in lexicographic order. In addition, every node in the structure associates with a tag Flag which is useful in the implementation. Notice that, NP(node pointer) and RP(root pointer) in the header table are two pointers. And the integer beside each item maintains the frequency of the itemset represented by the very node in FCItrees.

**Definition 5.** All the same items and the corresponding NP in header table are linked together by broken line in FCItrees, and we call this chain ItemList of the item (ItemList(item) in short).

**Definition 6.** All the items which share the same head item make up a tree and the corresponding RP in header table points to the root. We call this tree ItemTree of the head item (ItemTree(item) in short).

In the following, we provide two properties without proof.

**Property 1.** In each ItemTree, the node’s frequency is at most equal to that of its ancient nodes.

**Property 2.** In each ItemTree, an itemset must be a closed itemset if the frequency of the node which represents the itemset is larger than that of its children nodes.

Because of the characteristic of prefix tree that overlapping closed itemsets share prefixes of the corresponding branches, the above two properties are easy to get. In addition, the properties show that we can output all the closed itemsets easily by top-down frequent itemsets discovery scheme.

### 5.2 Proposed method

In this section, we first discuss the implementation of the function $C_1(S_1, S_2)$ and then the function $C_2(S_1, S_2)$.

#### 5.2.1 Function $C_1(S_1, S_2)$

In order to obtain $C_1(S_1, S_2)$, we can add the closed itemsets of $S_1$ to $S_2$ one by one. For a closed itemset $X$, the algorithm for addition is as follows. If $X \in S_2$, we should update the frequencies of $X$ and all its subsets in $S_2$. Otherwise, we need to insert $X$ to $S_2$ and determine the set of intersections. The pseudo code is given in the figure 4, in which inter-list is a chain used to maintain all the subsets and intersections.

<table>
<thead>
<tr>
<th>Algorithm $C_1(S_1, S_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Construe FCItrees with $S_2$:</td>
</tr>
<tr>
<td>2. For each closed itemset $X$ of $S_1$;</td>
</tr>
<tr>
<td>3. Addition(FCItrees, $X$, inter-list);</td>
</tr>
<tr>
<td>4. Insert each itemsets of inter-list into FCItrees;</td>
</tr>
<tr>
<td>5. Generate the suffix-itemsets of $X$;</td>
</tr>
<tr>
<td>6. If $X \in FCItrees$</td>
</tr>
<tr>
<td>7. For each $x_i$ of $X$</td>
</tr>
<tr>
<td>8. Travel ItemTree($x_i$) using suffix-itemset $x_i'$ to obtain all the subsets and insert them to inter-list;</td>
</tr>
<tr>
<td>9. Else</td>
</tr>
<tr>
<td>10. For each $x_i$ of $X$</td>
</tr>
<tr>
<td>11. Travel ItemList($x_i$) using suffix-itemset $x_i'$ to obtain all the intersections and insert them to inter-list;</td>
</tr>
<tr>
<td>12. Insert $X$ to inter-list;</td>
</tr>
</tbody>
</table>

![Fig. 4. The algorithm $C_1(S_1, S_2)$.](image)
item $x_i$ as the head item(6). Similarly, we can visit each ItemTree of the given closed itemset $X$ to obtain its subsets (7-8). If the given closed itemsets is not included by FC-Itrees, we determine all the intersections. To deal with this problem, we travel each ItemList with deep-first strategy in the ascending lexicographic order(10-11). When we visit a node with the deep-first strategy, all the nodes under it are visited. To avoid repeated visit in the travel of each ItemList, we can use the Flag each node contains. The detailed strategy is as follows. When traveling ItemList($x_i$), for each node in the ItemList, if the Flag of the node have not been set, we visit all the nodes under it and set the Flag of the nodes that belong to $X$; otherwise, we jump over it and reset the Flag. With this strategy, each intersection that $X$ has with the elements of $S_2$ is checked only once(11).

We use the corresponding suffix-itemset of $x_i$ instead of $X$ to determine whether a node belongs to $X$ when traveling ItemList($x_j$) or ItemTree($x_j$)($i \leq j \leq n$), since in the travel we only check the nodes under $x_j$ and all nodes checked are larger than $x_j$. This way can speed up the process to some extent.

In this algorithm, we obtain each subset or intersection with the combined frequency derived from the two itemsets generation. And we maintain all the itemsets that need to be inserted to $S_2$ in the chain inter-list. In the end, we insert them to FC-Itrees. In the process of insertion, if the element of inter-list is already included by FC-Itrees and its frequency is larger than that in the FC-Itrees, we update the one in FC-Itrees. Otherwise, if it is not included in FC-Itrees, we construe new nodes to represent it. There may be the same intersections; however, this way can avoid repeated insertion. Especially, when $X$ is not included by FC-Itrees, it is also a new closed itemset for $S_2$ and may be not included in the set of intersections. As a result, we can insert $X$ into the chain inter-list after the process of finding intersections(12).

we would utilize the following example to illustrate the above process.

**Example 2**: we assume that the set $S_2 = \{ABCD : 1, ACD : 2, CD : 3\}$ which can be represented by FC-Itrees as in figure 3. And suppose that the set $S_1 = \{BCD : 2, CD : 3\}$. In order to implement the function $C_1(S_1, S_2)$, we add each itemset of $S_1$ to $S_2$. For $CD : 3$, it is easy to find that it is already included in $S_2$ by visiting ItemTree($C$) with top-down strategy. According to lemma 4, we then only need to find the subset: $CD : 6$ by visiting ItemTree($C$) which is also a closed itemset.

For itemset $BCD : 2$, the ItemTree($B$) have not been established, so it is not included in $S_2$. In this case, we should determine the intersections it has with each itemset of $S_2$. To achieve this objective, we travel each ItemList of itemset $BCD$ following the order: ItemList($B$), ItemList($C$) and then ItemList($D$). With the strategy we have mentioned, when traveling ItemList($B$), we get intersection: $BCD : 3$, and the tags Flag associated with the nodes $B$ and $D$ in the branch of $ABCD : 1$ are set to 1 shown in figure 5. For ItemList($C$), we obtain intersections: $CD : 2$, $CD : 3$. In the meantime, we discard $CD : 2$ according to lemma 3 and the status of FCtrees is shown in figure 6. For simplicity, we only show the main information and the tags Flag we have not depicted are all with the value of 0. When traveling ItemList($D$), we obtain nothing and just reset all the tags of node $D$ because all the tags of nodes in the chain have been set. So the elements in the chain inter-list are $CD : 6$, $BCD : 3$. In the end, we insert all the itemsets in inter-list into the FC-Itrees and the FC-Itrees is as in figure 7.

---

**Fig. 5. The FC-Itrees after traveling ItemList($B$).**

**Fig. 6. The FC-Itrees after traveling ItemList($C$).**

**Fig. 7. The FC-Itrees representing $C_1(S_1, S_2)$.**
5.2.2 Function $C_2(S_1, S_2)$

The algorithm for the function $C_2(S_1, S_2)$ is to find the intersections between the two sets of $S_1$ and $S_2$. According to the analysis of $C_1(S_1, S_2)$, we can easily give the pseudo code in figure 8.

```
Algorithm $C_2(S_1, S_2)$
1. construe FCtress with $S_2$;
2. for each itemsets $X$ of $S_1$
3. generate the suffix-itemsets of $X$;
4. for each item $x_i$ of $X$
5. travel ItemList($x_i$) using suffix-itemset $x_i'$ to obtain intersections and insert them to inter-list;
6. return inter-list;
```

Fig. 8. The algorithm $C_2(S_1, S_2)$.

What should explain here is that, in the process of $C_2(S_1, S_2)$ in each partition when obtaining the intersections, only the frequency information contained in the set $S_i$ is maintained. In this way, when all the intersections are returned to the master from each partition, their frequency information can be added together without repeated addition.

6 Experiment

We have performed extensive experiments to validate the algorithm DFCIM. In this section, we show some of them.

Datasets. In our experiments, synthetic datasets like T10I6D100K are used which are generated by IBM synthetic data producer in the same way as in [7, 8]. Here $T$ denotes the average length of each transaction, $I$ the average length of patterns and $D$ the number of transactions in datasets.

In all the experiments, the datasets were divided into three partitions. And for each dataset, we have carried out several experiments with different min-support. The algorithm FPclose which is available at [21] was taken as the reference to generate all the global frequent closed itemsets. And it is also used to generate the local closed itemsets. In all experiments, we find that the results of DFCIM are identical with that of FPclose for the same dataset and the same min-support. So it demonstrates the validity of DFCIM algorithm. In the following, for the dataset T10I6D100K, we will show the results of each operation denoted by the symbols we used in this paper.

In table 1, $m_s$ in the first column is the parameter min-support which is specified by users. And the symbol $S$ is used to denote the set of global frequent closed itemsets which is our goal. All the values in the table represent the number of itemsets in the set represented by the corresponding symbols in the first row.

<table>
<thead>
<tr>
<th>$m_s$</th>
<th>$T_0$</th>
<th>$T_1/C_2(T_1, S_1)$</th>
<th>$T_2/C_2(T_2, S_2)$</th>
<th>$T_3/C_2(T_3, S_3)$</th>
<th>$S_0 \cup^* T_0$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1823</td>
<td>1658/1532</td>
<td>1823/1784</td>
<td>1796/1834</td>
<td>1961</td>
<td>1804</td>
</tr>
<tr>
<td>0.2</td>
<td>485</td>
<td>485/475</td>
<td>485/458</td>
<td>485/424</td>
<td>498</td>
<td>452</td>
</tr>
<tr>
<td>0.5</td>
<td>27</td>
<td>27/27</td>
<td>27/26</td>
<td>27/27</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>0.8</td>
<td>3</td>
<td>3/2</td>
<td>3/2</td>
<td>3/3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1. The results for T10I6D100K.

From the table, we can see that the size of $S_0 \cup^* T_0$ is not less than that of $S_0$. It demonstrates that there exist global frequent closed itemsets that occur as subset of the itemsets in $S_0$ and become new closed itemsets in the end included by $T_0$. The size of $C_2(T_1, S_1)$ is larger than 0 and it verifies that some intersections included by $T_1$ also occur in the corresponding set $S_1$, which should be eliminated. And the size of $T_1$ is not less than that of $C_2(T_1, S_1)$ testifies there exist itemsets occurring in $S_0$ which also occur in the corresponding $S_1$, but not in the corresponding $S_i$, which we have claimed in lemma 7. Furthermore, that the size of $S$ is smaller than that of $S_0 \cup^* T_0$ shows that there exist itemsets that are not global frequent closed itemsets included by $S_0 \cup^* T_0$, which also should be eliminated in the end.

7 Conclusion

In this paper, we have discussed the theoretical basis for distributed closed itemsets mining. Based on the theoretical analysis, a novel algorithm is proposed to mine the frequent closed itemsets with exact frequency in the distributed environments. A new data structure is also developed to maintain the closed itemsets. The characteristic of the structure affords to locate the closed itemsets and the intersections between two closed itemsets. Based on the structure, an efficient implement is presented. The experiments have shown the validity of the algorithm DFCIM.

However, in this paper we have not taken into account the efficiency of DFCIM. In the following work, we will evaluate the efficiency and exploit more efficient implementation.

Acknowledgment

This work was supported by the Foundation of Natural Sciences under Grants 60633010 and 60774008, the Research and Development Project of High-Technology of China under Grant 863-2006AA01Z174.
References


