Distributed Mining of Frequent Closed Itemsets: Some Preliminary Results

Claudio Lucchese
Ca' Foscari University of Venice
clucches@dsi.unive.it

Salvatore Orlando
Ca' Foscari University of Venice
orlando@dsi.unive.it

Raffaele Perego
ISTI-CNR of Pisa
perego@isti.cnr.it

Abstract

In this paper we address the problem of mining frequent closed itemsets in a distributed setting. We figure out an environment where a transactional dataset is horizontally partitioned and stored in different sites. We assume that due to the huge size of datasets and privacy concerns dataset partitions cannot be moved to a centralized site where to materialize the whole dataset and perform the mining task. Thus it becomes mandatory to perform separate mining on each site, and then merge the local results do derive a global knowledge. This paper shows how frequent closed itemsets, mined independently in each site, can be merged in order to derive globally frequent closed itemsets. Unfortunately, such merging might produce a superset of all the frequent closed itemsets, while the associated supports could be smaller than the exact ones because some globally frequent closed itemsets might be not locally frequent in some partition. A post-processing phase is thus needed to compute exact global results.

1 Introduction.

The frequent itemset mining (FIM) problem has been extensively studied in the last years. Several variations to the original Apriori algorithm [3], as well as completely different approaches, have been recently proposed (see, for example, the papers presented at the last two FIMI workshops [1, 2]). Recently it has been shown that *frequent closed itemsets* [10, 11, 4, 9, 15, 14, 6] are particularly interesting because they provide qualitatively the same information, by guaranteeing at the same time better performance, non redundant results, and concise representation.

In this paper we investigate a novel topic: distributed mining of frequent closed itemsets. While some papers address the problem of mining all the frequent itemsets in a distributed environments, at our best known, no proposal for distributed closed itemset mining exists. We figure out a distributed framework in which one (virtually single) transactional dataset $\mathcal{D}$ is horizontally partitioned into N parts. Each transaction of the dataset consists of an ordered list of items in the set of possible items $\mathcal{I}$. Each partition $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_N$ is made up of a subset of the transactions in $\mathcal{D}$, and there are no overlaps among the partitions.

In order to mine $\mathcal{D}$ we could devise a very simple strategy based on the "moving data" approach. We could choose a central site, which will receive every partition and, once materialized the whole dataset, will perform the mining task. The main drawback of this approach is that real datasets are usually huge, and it is not feasible to send an entire partition to a central site because of communication costs. Moreover, a central site could not be able to mine it as a whole, but it should have to partition it again in some way. In addition, if we have to deal with privacy concerns, we cannot move sensible data from one site to another. Hence, it is mandatory to employ a "moving results" approach, according to which frequent closed itemsets are independently extracted on each site, and then are collected and merged to get the global solution. Such algorithms are more suitable for *loosely-coupled distributed settings*, since they limit the number of communications/synchronizations between distributed nodes.

In the last years researchers have proposed many data mining algorithms that can efficiently be run in a loosely-coupled setting [8, 5], even if a few of them regards the FIM problem[1]. In particular, *Partition* [12], i.e. an algorithm for mining all the frequent

[1] On the other hand, frequent itemset mining has been extensively studied in the context of tightly-coupled environments, like parallel clusters.

itemsets, can be straightforwardly implemented in a loosely-coupled distributed setting, i.e. with a limited number of communications/synchronizations (see for example [7]).

In this paper we want to explore whether a *Partition*-like approach can also be exploited to mine frequent closed itemsets in a distributed environment. In order to demonstrate the feasibility of this approach, we show how the various local results, extracted independently on each site with the preferred closed itemset mining algorithm, can be merged in order to obtain the set of all the globally frequent closed itemsets.

Unfortunately, similarly to what happen in the distributed extraction of all the frequent itemsets, merging local results may produce a not exact set of all the frequent closed itemsets, while the associated supports could be unknown. A post-processing phase is thus necessary to compute exact global results.

The rest of the paper is organized as follows. Section 2 discusses some preliminary results concerning the merging function proposed to join the local results computed independently in the various distributed sites. Section 3 describes the additional steps needed to retrieve the exact supports of closed itemsets, and, finally, Section 4 draws some conclusions and future work.

2 Closed Itemsets Extraction

Our goal is to find a distributed algorithm that minimizes synchronizations and communications in mining frequent closed itemsets.

For the sake of simplicity, we limit the following discussion to a transactional dataset $\mathcal{D}$ that is partitioned in only two parts. More formally, let $\mathcal{D}_1$ and $\mathcal{D}_2$ be the two disjoint horizontal partitions of $\mathcal{D}$, where $\mathcal{D}_1 \cup \mathcal{D}_2 = \mathcal{D}$, and $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$. However, we can easily generalize our results to n partitions, for an arbitrary value of n.

Moreover, in this section we do not consider the issues concerning the (relative) minimum support threshold used to locally mine the various partitions. Remember that some globally frequent (closed) itemsets might not be extracted at all from some partitions, while some globally infrequent ones might be mined from others. In other words, the method discussed below surely works when we mine all the partitions with an (absolute) support $supp = 1$, i.e. when we consider as frequent any itemset which occur at least once in $\mathcal{D}$. We will take in account the issues concerning the minimum support constraint in the last part of the paper.

2.1 Preliminaries. Given a transactional dataset $\mathcal{D}$, let T and I, $T \subseteq \mathcal{D}$ and $I \subseteq \mathcal{I}$, be subsets of all the transactions and items appearing in $\mathcal{D}$, respectively.

The concept of *closed itemset* is based on the two following functions f and g:

$$f(T) = \{i \in \mathcal{I} \mid \forall t \in T, i \in t\}$$
$$g(I) = \{t \in \mathcal{D} \mid \forall i \in I, i \in t\}.$$

Function f returns the set of itemsets included in all the transactions belonging to T, while function g returns the set of transactions supporting a given itemset I. We can also consider $g(i)$ and $g(X)$ to be *tid-lists*, i.e. lists of identifiers associated with all the transactions in $\mathcal{D}$ set-including item i and itemset X, respectively.

DEFINITION 2.1. *An itemset I is said to be* closed *if and only if*

$$c(I) = f(g(I)) = f \circ g(I) = I$$

where the composite function $c = f \circ g$ is called Galois operator *or* closure operator.

Given a dataset partition $\mathcal{D}_j$, let T_j and I_j, $T_j \subseteq \mathcal{D}_j$ and $I_j \subseteq \mathcal{I}_j$, be subsets of all the transactions and items appearing in $\mathcal{D}_j$, respectively. We can thus redefine the two following functions:

$$f_j(T_j) = \{i \in \mathcal{I}_j \mid \forall t \in T_j, i \in t\}$$
$$g_j(I_j) = \{t \in \mathcal{D}_j \mid \forall i \in I_j, i \in t\}.$$

Therefore we can consider $g_j(i)$ and $g_j(X)$ be *tid-lists* only referring to transactions in $\mathcal{D}_j$ which set-include item i and itemset X, respectively. Similarly, we can define $c_j(I_j) = f_j(g_j(I_j))$, $\forall I_j \subseteq \mathcal{I}_j$.

Let $\mathcal{C}$ be the collection of all the closed itemsets in $\mathcal{D}$, and $\mathcal{C}_1$ and $\mathcal{C}_2$ be the two sets of closed itemsets mined from $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. Before introducing the theorems stating the properties concerning $\mathcal{C}_1$ and $\mathcal{C}_2$, and their relationship with $\mathcal{C}$, let us introduce a couple of important lemmas concerning the closure operator $c()$ with respect to the dataset $\mathcal{D}$. The same lemmas also hold for operators $c_1()$ and $c_2()$ relative to $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively.

LEMMA 2.1. *Given an itemset X and an item $i \in \mathcal{I}$, $g(X) \subseteq g(i) \Leftrightarrow i \in c(X)$.*

Proof. Proof.
$(g(X) \subseteq g(i) \Rightarrow i \in c(X))$:

> Since $g(X \cup i)$[2] $= g(X) \cap g(i)$, $g(X) \subseteq g(i) \Rightarrow g(X \cup i) = g(X)$. Therefore, if $g(X \cup i) = g(X)$ then $f(g(X \cup i)) = f(g(X)) \Rightarrow c(X \cup i) = c(X) \Rightarrow i \in c(X)$.

[2] For the sake of readability, we will drop parentheses around singleton itemsets, i.e. we will write $X \cup i$ instead of $X \cup \{i\}$, where single items are represented by lowercase characters.

$(i \in c(X) \Rightarrow g(X) \subseteq g(i))$:

> If $i \in c(X)$, then $g(X) = g(X \cup i)$. Since $g(X \cup i) = g(X) \cap g(i)$, $g(X) \cap g(i) = g(X)$ holds too. Thus, we can deduce that $g(X) \subseteq g(i)$.

LEMMA 2.2. *If $Y \in \mathcal{C}$, and $X \subset Y$, then $c(X) \subseteq Y$.*

Proof. Note that $g(Y) \subseteq g(X)$ because $X \subseteq Y$. Moreover, Lemma 2.1 states that if $j \in c(X)$, then $g(X) \subseteq g(j)$. Thus, since $g(Y) \subseteq g(X)$, then $g(Y) \subseteq g(j)$ holds too, and from Lemma 2.1 it also follows that $j \in c(Y)$. So, if $j \notin Y$ held, Y would not be a closed itemset because $j \in c(Y)$, and this is in contradiction with the hypothesis.

2.2 Local vs. global closed itemsets Our goal is to show that it is possible to perform independent computations on each partition of the dataset, and then join the local result by using an appropriate merging function $\oplus$ in order to obtain the global results. In this section we describe such merging function, and show that $\mathcal{C} \equiv \overline{\mathcal{C}}$, where $\overline{\mathcal{C}} = \mathcal{C}_1 \oplus \mathcal{C}_2$.

THEOREM 2.1. *Given the two sets of closed itemsets $\mathcal{C}_1$ and $\mathcal{C}_2$, mined respectively from the two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, we have that:*

$$\overline{\mathcal{C}} \;=\; \mathcal{C}_1 \oplus \mathcal{C}_2 \;=$$
$$(\mathcal{C}_1 \cup \mathcal{C}_2) \;\cup\; \{X_1 \cap X_2 \mid (X_1, X_2) \in (\mathcal{C}_1 \times \mathcal{C}_2)\} \equiv \mathcal{C}.$$

Therefore we can obtain $\mathcal{C}$ by collecting the closed itemsets contained in $\mathcal{C}_1$ and $\mathcal{C}_2$, and intersecting them to obtain further ones. We prove the above theorem by showing that the double inclusion holds.

THEOREM 2.2.
$$(\mathcal{C}_1 \cup \mathcal{C}_2) \subseteq \mathcal{C}$$

Proof. By definition, X is a closed itemsets in $\mathcal{D}$ *if and only if* X occurs in $\mathcal{D}$, and

$$\forall i \notin X \;:\; g(X) \not\subseteq g(i).$$

Therefore,

$$X \in \mathcal{C}_1 \Rightarrow \forall i \notin X \;:\; g_1(X) \not\subseteq g_1(i),$$

Since

$$g_1(X) \not\subseteq g_1(i) \;\Rightarrow\; g(X) \not\subseteq g(i),$$

we have that

$$X \in \mathcal{C}_1 \;\Rightarrow\; \forall i \notin X \;:\; g(X) \not\subseteq g(i),$$

and therefore X is closed in $\mathcal{D}$, i.e. $X \in \mathcal{C}$.

The same clearly holds also if $X \in \mathcal{C}_2$.

THEOREM 2.3. *Given $X_1 \in \mathcal{C}_1$ and $X_2 \in \mathcal{C}_2$, we have that*

$$Z = (X_1 \cap X_2) \;\in\; \mathcal{C}$$

Proof. If $Z \in \mathcal{C}_1$ or $Z \in \mathcal{C}_2$, then $Z \in \mathcal{C}$ because $(\mathcal{C}_1 \cup \mathcal{C}_2) \subseteq \mathcal{C}$ (see Theorem 2.2).

So in the following we will consider the *non-trivial* case, i.e. $Z \notin \mathcal{C}_1$ and $Z \notin \mathcal{C}_2$. In other terms, we are interested to the case in which $Z \subset X_1$ and $Z \subset X_2$.

By absurd, assume that Z is not closed, so that $\exists i \notin Z \mid g(Z) \subseteq g(i)$. So, we have that:

$$g(Z) \subseteq g(i) \;\Rightarrow\; g_1(Z) \subseteq g_1(i)$$
$$g(Z) \subseteq g(i) \;\Rightarrow\; g_2(Z) \subseteq g_2(i).$$

or, equivalently, $i \in c_1(Z)$ and $i \in c_2(Z)$.

By Lemma 2.2, it follows that $c_1(Z) \subseteq X_1$ and $c_2(Z) \subseteq X_2$, since X_1 and X_2 are closed in $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. So, the only way to choose an i that belongs to both $c_1(Z)$ and $c_2(Z)$, where $c_1(Z) \subseteq X_1$ and $c_2(Z) \subseteq X_2$, is that $i \in Z = (X_1 \cap X_2)$. But this is in contradiction with the hypothesis by absurd, i.e. $i \notin Z$.

The following corollary is a simple consequence of the above Theorem.

COROLLARY 2.1.
$$\{X_1 \cap X_2 \mid (X_1, X_2) \in (\mathcal{C}_1 \times \mathcal{C}_2)\} \subseteq \mathcal{C}$$

Theorem 2.2 and Corollary 2.1 show that $\overline{\mathcal{C}} \subseteq \mathcal{C}$. Our merge function is thus *correct*, in the sense that any itemset $X \in \overline{\mathcal{C}}$ is also a closed itemset in the global dataset $\mathcal{D}$.

In the following we prove the opposite implication, i.e. $\mathcal{C} \subseteq \overline{\mathcal{C}}$, which allow us to show that our merge function is also *complete*, and therefore $\mathcal{C} \equiv \overline{\mathcal{C}}$.

THEOREM 2.4.
$$\mathcal{C} \subseteq \overline{\mathcal{C}}$$

Proof. Let $X \in \mathcal{C}$ be an itemset belonging to some transactions of $\mathcal{D}$. Therefore X is included in some transactions of either $\mathcal{D}_1$ or $\mathcal{D}_2$, or it is included in both $\mathcal{D}_1$ and $\mathcal{D}_2$.

If X only occurs in transactions of one partition, either $\mathcal{D}_1$ or $\mathcal{D}_2$, we can trivially show that $X \in \overline{\mathcal{C}}$. Suppose that this partition is $\mathcal{D}_1$. Therefore $g_1(X) = g(X)$, and $g_2(X) = \emptyset$.

Since $c(X) = X$ by hypothesis, then $\forall i \notin X \mid g(X) \not\subseteq g(i)$. Then it also holds that $\forall i \notin X \mid g_1(X) \not\subseteq g(i)$, because $g_1(X) = g(X)$. Since $g_1(i) \subseteq g(i)$, then it also holds that $\forall i \notin X \mid g_1(X) \not\subseteq g_1(i)$, or, equivalently, $X = c_1(X)$. Therefore, in this case $X \in \overline{\mathcal{C}}$ surely holds, because $X \in \mathcal{C}_1$.

If X appears in transactions of $\mathcal{D}_1$ and $\mathcal{D}_2$, then we can compute its closure in both of them, i.e. $X_1 = c_1(X)$ and $X_2 = c_2(X)$. So either $X = (X_1 \cap X_2)$ or $X \subset (X_1 \cap X_2)$ can hold.

If $X = (X_1 \cap X_2)$, then $X \in \overline{\mathcal{C}}$ by definition of $\overline{\mathcal{C}}$.

The second condition $X \subset (X_1 \cap X_2)$ can not hold. If $X \subset (X_1 \cap X_2)$, then $\exists i \notin X$ such that $i \in c_1(X)$ and $i \in c_2(X)$. Hence $g_1(X) \subseteq g_1(i)$ and $g_2(X) \subseteq g_2(i)$. Since $g(X) = g_1(X) \cup g_2(X)$ and $g(i) = g_1(i) \cup g_2(i)$, then $g(X) \subseteq g(i)$ also holds, i.e. $i \in c(X) = X$. But this is in contradiction with the hypothesis that $i \notin X$.

Note that from a theoretical point of view, itemsets in $\mathcal{C}$ form a lattice, i.e. for every $X, Y \in \mathcal{C}$ their join element $X \cup Y$ and their met element $X \cap Y$ belong to $\mathcal{C}$ [15, 14]. We have just shown that if $X \in \mathcal{C}_1$ or $X \in \mathcal{C}_2$, then $X \in \mathcal{C}$. Moreover, in order to complete the lattice $\mathcal{C}$, it is also needed to set-intersect each pair $(X_1, X_2) \in (\mathcal{C}_1 \times \mathcal{C}_2)$, and add $X_1 \cap X_2$ to $\mathcal{C}$. The proofs of both the implications show that this is enough to complete the lattice $\mathcal{C}$, i.e. $\mathcal{C} \equiv \overline{\mathcal{C}}$.

Till now we have defined a merging function $\oplus$, used to obtain $\mathcal{C}$ as $\mathcal{C}_1 \oplus \mathcal{C}_2$. This result can be generalized to the case of N partitions of $\mathcal{D}$.

THEOREM 2.5. *Given the sets of closed itemsets $\mathcal{C}_1, \ldots, \mathcal{C}_N$ mined respectively from N disjoint horizontal partitions $\mathcal{D}_1, \ldots, \mathcal{D}_N$ of $\mathcal{D}$, we have that:*

$$\overline{\mathcal{C}} = (\ldots((\mathcal{C}_1 \oplus \mathcal{C}_2) \oplus \ldots \oplus \mathcal{C}_N)\ldots).$$

Proof. It is easy to give a proof by induction. We have already proved with Theorem 2.1 that the equality holds in the case $N = 2$.

Suppose that Theorem 2.5 holds for N, we want to show that it holds for $N+1$ as well. Given the $N+1$ partitions, by hypothesis we know that the closed itemsets in $\mathcal{D}' = \{\mathcal{D}_1 \cup \ldots \cup \mathcal{D}_N\}$ are $\mathcal{C}' = ((\mathcal{C}_1 \oplus \mathcal{C}_2) \oplus \ldots \oplus \mathcal{C}_N)$. At this point, we can think at the dataset $\mathcal{D}$ as it was made of two partitions only, i.e. $\mathcal{D} = \mathcal{D}' \cup \mathcal{D}_{N+1}$. Therefore, we can apply Theorem 2.1 and get $\overline{\mathcal{C}} = \mathcal{C}' \oplus \mathcal{C}_{N+1} = (\ldots((\mathcal{C}_1 \oplus \mathcal{C}_2) \oplus \ldots \oplus \mathcal{C}_N)\ldots)\ldots\mathcal{C}_{N+1}$.

While computing the union of the various $\mathcal{C}_i$ is straightforward, the same is not true for the intersections. In order to quantify the impact of intersections in the overall computation, we experimentally counted the number of closed itemsets which have to be calculate by intersections as a function of the number of partitions. Figure 1 plots this number in a real world case: the mushroom dataset mined with absolute minimum support 1. From the figure we can see that the number of *intersection itemsets* increases as the number of partition grows, even if this growth is less than linear.

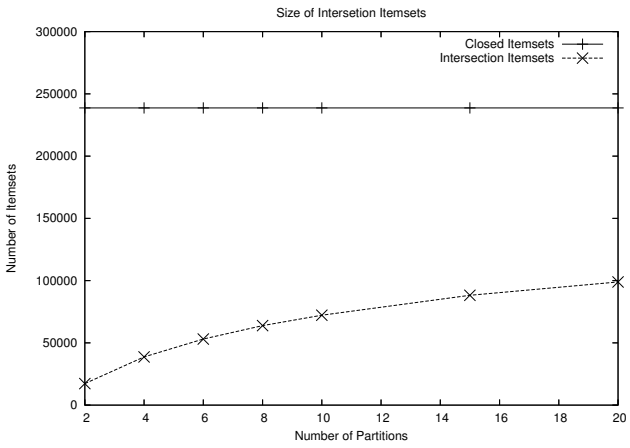

Figure 1: Intersection Itemsets in a real world case: the mushroom dataset mined with absolute minimum support 1.

When the dataset is split into 20 partitions, about 2/5 of all frequent closed itemsets have to be computed with intersections.

3 Computing the supports of frequent closed itemsets

Theorem 2.1 shows a way to devise the identities of all the closed itemsets in $\mathcal{D}$ given the closed itemsets found in its partitions $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_N$. In this section we show how to calculate the supports of such itemsets given the supports of all the itemsets in $\mathcal{C}_i, \forall i$.

Given $X \in \overline{\mathcal{C}}$, we denote with $\|X\|_i$ the support of X in partition $\mathcal{D}_i$. The global support of X is clearly $\sum_{i=1,\ldots,N} \|X\|_i$. Note that by construction of $\overline{\mathcal{C}}$, it may happen that $X \notin \mathcal{C}_i$ for some partition $\mathcal{D}_i$, and therefore, even if the support of every itemset in $\mathcal{C}_j$ for every partition j is known, the support $\|X\|_i$ may not be known. In fact, for each $X \in \overline{\mathcal{C}}$, we can distinguish between two cases:

$\exists i \mid X \in \mathcal{C}_i$,

i.e. X was obtained by union of the various $\mathcal{C}_j$. In this case $\|X\|_i$ is given, but we cannot say anything about $\|X\|_{j \neq i}$.

$\neg\exists i \mid X \in \mathcal{C}_i$,

i.e. X was obtained from the intersection between two closed itemsets. In this case we don't know $\|X\|_j, \forall j$.

In both cases we need to derive $\|X\|_j$, i.e. the support of itemset X on partition $\mathcal{D}_j$, where $X \notin \mathcal{C}_j$.

The support of $\|X\|_j$ is equal to $\|c_j(X)\|$. Since every $Y \in \mathcal{C}_j$ is closed, we have that $c_j(X)$ is exactly equal to $\|c_j(Y)\|$ where $Y \in \mathcal{C}_j$ is the smallest itemset such that $X \subseteq Y$. Therefore, in order to calculate the support of all the itemsets in $\mathcal{C}$ it is sufficient a post-processing phase where subset searches are performed to calculate the contributes of each partition to the global support of some itemset.

In the above we have always discarded the minimum support constraint, i.e. we have used a minimum absolute support of 1. Unfortunately when we introduce a minimum support threshold $\sigma > 1$, we cannot guarantee the correctness of the algorithm. In fact it may happen that some global frequent itemset is not frequent in some partition, and since its local support in such partition is not retrieved, its global support cannot be derived. Similarly, some locally frequent itemsets may result to be globally infrequent.

Analogously to Partition [12], we can however devise the following strategy. Since each globally frequent itemset has to be frequent in at least one partition, we have that $\overline{\mathcal{C}}$ will contain all the global frequent closed itemsets, but some globally infrequent one as well. In order to retrieve the exact support of itemsets which are not frequent in all partitions, we have to compute their supports in all the sites where they were found to be not frequent. After recollecting these counts, we can calculate the exact support of every itemset in $\overline{\mathcal{C}}$ and get the correct solution $\mathcal{C}$.

4 Conclusion

We have addressed the problem of mining frequent closed itemsets in a distributed environment. In the distributed mining of frequent itemsets, a three steps algorithm is sufficient in order to get exact results. First, independent mining tasks are performed on each partition, then the results are merged to form a big candidate set, and, finally, an additional check is needed for each candidate to retrieve its actual support in the partitions where it was found to be infrequent. In this paper we investigate the merging step in the case of closed itemset mining. We have shown that in this case the merging step is completely different and surely more complex.

However, our preliminary results demonstrate the feasibility of the approach. Future works regards the actual implementation of the algorithm and its performance evaluation. Moreover, similarly to [13], we are also interested in studying an approximated version of the algorithm, which should not require an additional step for exactly counting itemsets supports.

References

[1] *Proc. of the 1st Workshop on Frequent Itemset Mining Implementations (FIMI'03)*. 2003.

[2] *Proc. of the 2nd Workshop on Frequent Itemset Mining Implementations (FIMI'04)*. 2004.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 1994.

[4] Gosta Grahne and Jianfei Zhu. Efficiently using prefix-trees in mining frequent itemsets. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, November 2003.

[5] H. Kargupta and P. Chan (Eds.). *Advances in Distributed and Parallel Knowledge Discovery*. AAAI Press/The MIT Press, 2000.

[6] C. Lucchese, S. Orlando, and R. Perego. Fast and Memory Efficient Mining of Frequent Closed Itemsets. Technical Report TR-CS-2004, Ca' Foscari University of Venice, Italy, 2004.

[7] A. Mueller. Fast sequential and parallel algorithms for association rules mining: A comparison. Technical Report CS-TR-3515, Univ. of Maryland, 1995.

[8] B. Park and H. Kargupta. Distributed Data Mining: Algorithms, Systems, and Applications. In *Data Mining Handbook*, pages 341–358. IEA, 2002.

[9] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.

[10] Jian Pei, Jiawei Han, and Runying Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *SIGMOD International Workshop on Data Mining and Knowledge Discovery*, May 2000.

[11] Jian Pei, Jiawei Han, and Jianyong Wang. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *SIGKDD '03*, August 2003.

[12] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases*, pages 432–444. Morgan Kaufmann, September 1995.

[13] C. Silvestri and S. Orlando. Distributed Approximate Mining of Frequent Patterns. In *Proc. of the 2005 ACM Symposium on Applied Computing, SAC 2005, special track on Data Mining*, 2005.

[14] Mohammed J. Zaki. Mining non-redundant association rules. *Data Min. Knowl. Discov.*, 9(3):223–248, 2004.

[15] Mohammed J. Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed itemsets mining. In *2nd SIAM International Conference on Data Mining*, April 2002.