DARCI: Distributed Association Rule Mining Utilizing Closed Itemsets

Suad Alramouni
Hashemite University, Jordan
suad@hu.edu.jo
Jae Young Lee
Colorado School of Mines, USA
jaelee@mines.edu

ABSTRACT

A distributed rule mining algorithm must minimize the communication cost to reduce the communication bandwidth use and to improve the scalability. There are a few distributed rule mining algorithms reported in the literature. In this paper we propose a new distributed association rule mining algorithm, called DARCI, which reduces the communication cost by up to 40% of the best known algorithm. The first step of mining global rules is to find globally frequent itemsets, and it involves the exchange of local supports of potentially frequent itemsets. DARCI reduces the communication cost by (1) sending frequent closed itemsets, which are supersets of locally frequent itemsets, instead of sending individual itemsets and (2) using a novel pruning technique to reduce the probability of having to send local supports.

Key Words: distributed data mining, association rules, transactional databases.

1. Introduction

In a transactional database, an association rule models the relationship between items in the database. Assume a database consists of a collection of baskets, each of which represents a set of items purchased by a customer in a grocery store. A rule *milk* → *eggs* models an association between *milk* and *eggs*, which implies that if a customer buys *milk* it is very likely that he/she buys *eggs* too. The interestingness of a rule is specified by its *support* and *confidence*. Let $I = \{a_1, a_2, \ldots, a_N\}$ be the set of all items. A database D is a set of transactions, $D = \{t_1, t_2, \ldots, t_m\}$,where $t_i \subseteq I$. An itemset α is any subset of I, $\alpha \subseteq I$. An itemset with k items is called *k-itemset*. Support of an itemset α is the number of transactions which contain α : $sup(\alpha) = |T|$, where $T = \{t \mid \alpha \subseteq t, t \in D\}$. Sometimes a support is expressed as a fraction $|T| / |D|$. Given a rule $R{:}\alpha \rightarrow \beta$, $\alpha \subseteq I$, $\beta \subseteq I$, and $\alpha \cap \beta = \phi$, the support of the rule is defined as $sup(R) = sup(\alpha \cup \beta)$ and the confidence of the rule is defined as $conf(R) = sup(\alpha \cup \beta) / sup(\alpha)$. We are interested in rules whose support and confidence satisfy certain minimum thresholds. Let s be a minimum support in fraction and c be a minimum confidence specified by the user. An itemset α is said to be *frequent* (or *large*) iff $sup(\alpha) \geq s \cdot |D|$, and a rule R is said to be *strong* iff $conf(R) \geq c$. Typically, we first identify frequent itemsets. Then, strong rules are mined from these frequent itemsets. The first efficient algorithm to mine association rules from a large database was proposed in 1994, which is called *Apriori* [1]. Since then, many other association rule mining algorithms were proposed, such as [2, 3, 4, 5, 6, 7, 8].

Let a distributed system consist of n local sites, $S = \{S^1, S^2, \ldots, S^n\}$. We assume there is no shared memory and there is no one site which acts as a coordinator in distributed rule mining. Each site S^i keeps a database, $D^i = \{t \mid t \subseteq I\}$. The number of transactions in each local site is $|D^i|$. The global database $D = \{D^1, D^2, \ldots, D^n\}$, and $|D| = \sum_{i=1}^{n} |D^i|$. For an itemset $\alpha \subseteq I$, $sup^i(\alpha)$ denotes the local support of α at site S^i, and the global support of α is, $sup(\alpha) = \sum_{i=1}^{n} sup^i(\alpha)$. Let s be a global minimum support (in fraction). Then, an itemset α is globally frequent iff $sup(\alpha) \geq s \cdot |D|$. A task of distributed rule mining is first to identify all globally frequent itemsets and, then, to mine strong rules from these itemsets.

There have been some distributed association rule mining algorithms proposed in the literature: CD [9], FDM [10], and DDM [11]. Among these, DDM was shown to have the lowest communication cost. In this paper, we present a new distributed algorithm, called DARCI (Distributed Association Rule mining utilizing Closed Itemsets), which reduces the communication cost by as much as 40% of that of DDM. Underlying idea of DARCI is as follows. One essential step of determining globally frequent itemsets is to exchange local supports of potentially frequent itemsets. In previous algorithms, local supports of itemsets were broadcast to other sites individually. If, however, we send frequent closed itemsets (along with their supports), we can reduce the communication cost. Since a frequent closed itemset is a superset of frequent itemsets, sending one frequent closed itemset will have the effect of sending multiple frequent itemsets with less communication cost. Another idea we employ is a pruning method by which the number of local supports that need to be broadcast is reduced. We conducted a series of experiments and showed significant saving in the communication cost.

The paper is organized as follows. Section 2 reviews previous distributed rule mining algorithms and Section 3 reviews frequent closed itemsets. Section 4 discusses the proposed DARCI algorithm. Section 5 analyzes communication cost. Section 6 shows the results of our experiments. Section 7 considers implementation issues, and Section 8 concludes the paper.

2. Previous work

CD[9], FDM[10], and DDM[11] are distributed versions of *Apriori*. Let C_k and L_k denote global candidate k-itemsets and global frequent k-itemsets, respectively. They first mine C_1 and L_1, then C_2 and L_2, and so on. At each level k, local sites broadcast local supports of some or all of C_k and each site adds up all local supports of an itemset to decide whether it is globally frequent. The differences among the three are as follows. CD broadcasts the local supports of all C_k. The communication complexity is $O(|C| \cdot n)$. Here $C = \cup_{all\ k} C_k$. In FDM, however, each site broadcasts the local supports of only those itemsets that are locally frequent at that site. Its communication complexity is $O(Pr_{potential} \cdot |C| \cdot n)$, where $Pr_{potential}$ is the probability that a candidate itemset is frequent at least at one site. DDM further reduces the communication cost by employing a negotiation scheme. DDM improves the communication cost to $O(Pr_{above} \cdot |C| \cdot n)$, where Pr_{above}, which is smaller than $Pr_{potential}$ in most cases, is the probability that a candidate itemset is locally frequent at a particular local site.

The negotiation scheme of DDM is as follows. Each site broadcasts the local support iff an itemset satisfies a certain condition. This condition is specified with two hypotheses: global hypothesis H and private hypothesis P. Suppose, at a certain point in time, that some sites have broadcast the local support of an itemset α. Let X_α be $\{D^i |\ sup^i(\alpha)$ has been broadcast$\}$ and Y_α be $\{D^i \mid sup^i(\alpha)$ has not yet been broadcast$\}$, $X_\alpha \cup Y_\alpha = D$. Then, H and P are defined as follows:

$H(\alpha) = 0 \qquad\qquad \text{if } X_\alpha = \phi,$

$$\frac{\sum_{D^i \in X_\alpha} sup^i(\alpha)}{\sum_{D^i \in X_\alpha} |D^i|} \bullet |D| \quad \text{otherwise} \qquad (1)$$

$P(\alpha, D^j)$ =

$$\sum_{D^i \in X_\alpha} sup^i(\alpha) + \frac{sup^j(\alpha)}{|D^j|} \bullet \sum_{D^i \notin X_\alpha} |D^i| \qquad (2)$$

H estimates the global support of α based on already broadcasted local supports and it is the same for all sites. On the other hand, *P*, which is defined for each local site, uses known values of those sites which already broadcasted their local supports (the first term of *P*), and uses its own local support for those sites which have not broadcast their local supports yet (the second term of *P*).

Each site S^i computes *H* and *P* for all candidate itemsets and broadcasts the local support of an itemset α if and only if the following condition is satisfied: $cond_{HP} = (H(\alpha) < min\_sup \cdot |D| \leq P(\alpha, D^i))$ or $(P(\alpha, D^i) < min\_sup \cdot |D| \leq H(\alpha))$ (3)

In this way, the communication cost of DDM is greatly reduced. To further reduce the communication cost, another parameter called *contribution*, denoted as *R*, was introduced in [11]. $R(\alpha, D^i)$ estimates how much S^i *w*ould contribute to the change of *P* on other sites if it broadcasts $sup^i(\alpha)$. Each site first identifies all candidate itemsets that satisfy the *condHP*. Then, for these itemsets, it also computes all *R* estimates, and broadcasts only the itemset which has the largest *R* value. This process, called *negotiation proces*s, continues until there is no more itemset that needs to be sent at all sites. Once the negotiation process is finished, each site has all globally frequent itemsets identified. Next, since we need the global supports of all globally frequent itemsets to mine strong global rules, the local supports of those frequent itemsets whose local supports were not broadcast are broadcast now. Interested readers are referred to [11] for more details about this algorithm.

3. Frequent closed itemset

Since DARCI utilizes frequent closed itemsets [12,13,14], we briefly review the concept here.

Definition 1. An itemset α is *closed* if and only if there exists no itemset α' such that (1) α' is a proper superset of α, and (2) every transaction containing α also contains α'.

Definition 2. A closed itemset α is a *frequent closed itemset* if and only if its support is no less than the given minimum support threshold.

Definition 3. An itemset α is said to be *derivable* from a set of frequent closed itemsets Φ if and only if there exists an itemset β such that $\beta \in \Phi$ and $\alpha \subseteq \beta$.

Definition 4. Let Φ be a set of frequent closed itemsets. Then $allFI(\Phi) = \{\alpha \mid \alpha$ is derivable from $\Phi\}$.

The following properties hold for frequent closed itemsets [12, 13].

Property 1. Let *L* be the set of all frequent itemsets and Φ be the set of all frequent closed itemsets in a database D. Then *L* is identical to allFI(Φ). ■

Property 2. The support of a frequent itemset α is the same as the support of the smallest frequent closed itemset that is a superset of α. ■

Property 3. The number of frequent closed itemsets is orders of magnitude smaller than the number of all frequent itemsets. ■

Example 1 demonstrates these three properties of frequent closed itemsets.

Example 1. Let *D* be a transactional database containing 6 transactions as shown in Table 1, and the minimum support threshold, *min_sup,* be 50% (i.e., 3 transactions). The complete set of frequent itemsets, *L,* contains 19 itemsets, *L* = {*b*: 6, *e*: 5, *a*: *4*, *c*: 4, *d*: 4, *ab*: 4, *ae*: 4, *bc*: 4, *bd*: 4, *be*: 5, *abe*: 4, *ad*: 3, *ce*: 3, *de*: 3, *abd*: 3, *ade*: 3, *bce*: 3, *bde*: 3, *abde*: 3}, while the set of frequent closed itemsets, Φ, contains only 7 itemsets, Φ = {*b*: 6, *be*: 5, *abe*: 4, *bd*: 4, *bc*: 4, *abde*: 3, *bce*: 3}.

Table 1. Example database *D*.

Transaction ID	Items
1	*A, b, d, e*
2	*B, c, e*
3	*A, b, d, e*
4	*A, b, c, e*
5	*A, b, c, d, e*
6	*B, c, d*

It is easy to verify the three properties for frequent closed itemsets. First, deriving all subsets from Φ results in *L*. Next, *d* has two supersets in Φ: *bd* and *abde,* with *bd* being the shortest. So, the support of itemset *d* can be determined by that of *bd*, which is 4. Finally, we can see that $|\Phi|$ is significantly smaller than $|L|$.■

4. DARCI

4.1. Underlying concepts of DARCI

In what follows, we use the notations shown in Table 2, in addition to those already defined in previous sections. Note when we say "an *itemset* or *support* is broadcast", it means "an itemset and its local support" are broadcast. We will often use *message* to refer to the both.

One way of reducing the number of messages is to bundle up multiple messages into a single message. DARCI achieves this by broadcasting *LFCI*'s, early in the course of the algorithm, instead of sending out individual itemsets.

Lemma 1. *Any frequent itemset* $\alpha \in LL^i$ *at* S^i *is derivable from* $LFCI^i$.

Proof. By property 1, *allFI*($LFCI^i$) is identical to LL^i. It follows that $\alpha \in$ *allFI*($LFCI^i$) . So, by definition 4, α is derivable from $LFCI^i$. ■

Lemma 2. *FCI provides an upper bound on global frequent itemsets L.*

Proof. We will prove that $L \subseteq$ *allFI*(*FCI*). For all $\alpha \in L$, there exists *i* such that $\alpha \in LL^i$. By Lemma 1, α is derivable from $LFCI^i$. By Definition 3, there exists β such that $\alpha \subseteq \beta$ and $\beta \in LFCI^i$. Since $FCI = \cup_{\forall i} LFCI^i$, $\beta \in FCI$. So, α is also derivable from *FCI*. By Definition 4, $\alpha \in$ *allFI*(*FCI*). Hence, $L \subseteq$ *allFI*(*FCI*). ■

Table 2. Notations.

N	number of local sites.
D^i	local database at site S^i.
D	Global database (= { $D^1, D^2, \ldots, D^n$ }).
σ	global minimum support in number of transactions (= *min_sup* · $\|D\|$).
$sup^i(\alpha)$	local support of itemset α at S^i.
$sup(\alpha)$	global support of itemset α (= $\sum_{i=1}^{n} sup^i(\alpha)$).
k-itemset	itemset of length *k*.
LL_k^i	local frequent *k*-itemsets at S^i.
LL^i	all local frequent itemsets at S^i (= $\cup_{\forall k} LL_k^i$).
L_k	Global frequent *k*-itemsets.
L	all global frequent itemsets (= $\cup_{\forall k} L_k$).
C_k	Global candidate *k*-itemsets.
C	all global candidate itemsets (= $\cup_{\forall k} C_k$).
$LFCI^i$	local frequent closed itemsets, of length greater than or equal to 2, at S^i.
FCI	= $\cup_{\forall i} LFCI^i$.
X_k^i	*k*-itemsets that need to be broadcast by site S^i.
$sup_{best}(\alpha)$	best scenario global support of α.

So, broadcasting *LFCI*'s has the same effect as broadcasting all local frequent itemsets without losing support information and with considerably low communication cost.

Another approach we employ to reduce the communication cost is to devise a pruning method which removes messages that do not have to be broadcast. DARCI uses a pruning method called *best scenario pruning* to reduce the number of messages to be broadcast. As described earlier, if an itemset

α is locally frequent at S^i, it is broadcast as a part of $LFCI^i$. So, the whole system can be partitioned into two groups: (1) those which broadcast α in their *LFCI*'s and (2) those that did not include α *i*n their *LFCI's*. Let S_{α}^{sent} be the former and $S_{\alpha}^{not-sent}$ be the latter, with $S_{\alpha}^{sent} \cup S_{\alpha}^{not-sent} = S$.

Lemma 3. $sup(\alpha) \leq sup_{best}(\alpha)$, where

$$sup_{best}(\alpha) = \sum_{S^i \in S_{\alpha}^{sent}} sup^i(\alpha) + \sum_{S^i \in S_{\alpha}^{not-sent}} (min\_sup \cdot |D^i| - 1) \quad . \qquad (3)$$

Proof. The upper bound of $sup(\alpha)$ is obtained if we assume the largest possible values for local supports that were not broadcast. If the local support of α was not broadcast, it is infrequent at that site and it is at most one less than the local minimum support, ($min\_sup \cdot |D^i|$ -1). Therefore, sup (α) cannot be larger than $sup_{best}(\alpha)$, which uses $sup^i(\alpha)$ from S_{α}^{sent} and ($min\_sup \cdot |D^i|$ - 1) from $S_{\alpha}^{not-sent}$.■

DARCI's best scenario pruning method is based on Lemma 3. Suppose that all *LFCI*'s have been broadcast. Then, every site now has *FCI* and must determine whether each itemset $\alpha \in C$ ($C = allFI(FCI)$) is globally frequent or not. To do that, each site S^i examines each itemset α and decides whether it needs to broadcast α's local support. An itemset α falls into one of the following three cases. **Case 1:** α is frequent at all sites; **Case 2**: α is infrequent at all sites; and **Case 3**: α is frequent at some sites.

In Case 1, the local support of α has already been broadcasted as a part of $LFCI^i$. It is not necessary to broadcast the local support in Case 2 either because α is globally infrequent; thus, we do not need to consider it at all. In Case 3, α 's local support was broadcast by S_{α}^{sent} but not by $S_{\alpha}^{not-sent}$. Two subcases are possible. **Case 3-1**: If α is frequent at S^i, S^i does not have to do any thing because the support of α was broadcast in $LFCI^i$. **Case 3-2**: If α is infrequent at S^i, S^i has to decide whether to broadcast α's local support or not.

An important observation is that S^i should not broadcast α's local support if it knows α cannot be globally frequent even in the best possible scenario. Note that $sup_{best}(\alpha)$ is the best possible scenario of the global support of α. Therefore, it is sufficient for S^i to compute $sup_{best}(\alpha)$ using Equation 3 and check it against the global minimum support σ. If $sup_{best}(\alpha) < \sigma$, then α can never be globally frequent. So, S^i broadcasts α's support if and only if $sup_{best}(\alpha) \geq \sigma$. Example 1 illustrates the best scenario pruning method.

Example 2. Let us consider a small distributed system consisting of three local sites: S = {S^1, S^2, S^3}, D = {D^1, D^2, D^3}, $|D^1|$ = 35, $|D^2|$ = 25, $|D^3|$ = 40, $|D|$ = 100, and global minimum support *min_sup*, in fraction, is 0.4 (or σ = 40). Suppose that the local supports of an itemset α are: $sup^1(\alpha)$ = 10 , $sup^2(\alpha)$ = 12, and $sup^3(\alpha)$ = 10. Then, S^2 has broadcast the local support of α in $LFCI^2$, but S^1 and S^3 did not. Now S^1 tries to determine whether it needs to broadcast its local support of α. S^1 computes the best scenario global support of α, $sup_{best}(\alpha)$, as follows: $sup_{best}(\alpha)$ = (its local support) + (local support of α at S^2) + ($min\_sup \cdot |D^3|$ -1) = 10 + 12 + (0.4$\times$ 40 -1) = 37. Note that S^1 did not broadcast α's support but it uses its own support (instead of $min\_sup \cdot |D^1|$ - 1). Since $sup_{best}(\alpha)$ is smaller than σ (= 40), S^1 does not broadcast its local support. ■

As we will see in section 5, broadcasting *LFCI's* instead of individual frequent itemsets and the use of the best scenario pruning method, combined, greatly reduces the communication cost.

4.2 DARCI algorithm

The DARCI algorithm is shown in Figure 1. Each site first mines local frequent 1-

itemsets LL_1^i and broadcasts it to all other sites (lines 2 and 3). The union of all LL_1^i becomes the global candidate 1-itemsets C_1 (line 4). To determine whether an individual itemset in C_1 is globally frequent or not, each site needs to exchange its local support. However, broadcasting the local supports of all candidate itemsets is too costly.

```
    **Algorithm 1.** DARCI
1.  For each site $S^i$, $1 \leq i \leq n$ do
2.    Mine locally frequent 1-itemsets $LL_1^i$
3.    Broadcast $LL_1^i$
4.    $C_1 = {}_{i=1}^{n} LL_1^i$ /* global candidate 1-itemsets*/
5.    $X_1^i = Prune(C_1, LL_1^i)$
6.    For each $\alpha \in X_1^i$ do
7.        Broadcast $\alpha$
8.    End For
9.    Determine $L_1$
10.   Compute $LFCI^i$
11.   Broadcast $LFCI^i$
12.   $k = 2$
13.   Repeat
14.       $C_k = Apriori\text{-}gen(L_{k-1})$
15.       $X_k^i = Prune(C_k, LL_k^i)$
16.       For each $\alpha \in X_k^i$ do
17.           Broadcast $\alpha$
18.       End for
19.  /* at this point all local supports of $\alpha$ are
        available at all sites */
20.       Determine $L_k$ : $L_k = \{ \alpha \mid sup(\alpha)$ (=
          $\sum_{i=1}^{n} sup^i(\alpha)$ ) $\geq \sigma\}$
21.       $k = k + 1$
22.   Until $L_{k-1} = \phi$
23. End for
```

Figure 1. DARCI algorithm.

The pruning of C_1 is done by the function *Prune,* which implements the *best scenario pruning* method. *Prune* examines each itemset in C_1 and returns X_1^i, which includes only those itemsets that need to be broadcast (line 5). Once the local supports of selected itemsets are broadcast (lines 6 – 8), each site now can determine all global frequent 1-itemsets, L_1 (line 9). Using L_1, each site mines local frequent closed itemsets $LFCI^i$ and broadcasts it (lines 10 and 11). The collected *LFCI*'s are used by each site later when pruning *Ck* ($k \geq 2$). Within the loop of lines 13 and 22, global frequent itemsets are mined for $k \geq 2$. For each level k, C_k is constructed from L_{k-1} using the *Apriori-gen* function of the Apriori algorithm [1] (line 14). In line 15, C_k is pruned by the *Prune* function. Then, X_k^i broadcast and L_k is determined (lines 16 – 19). This process is repeated until there is no more frequent itemsets.

The *Prune* function is shown in Figure 2. Given candidate k-itemsets C_k, the *Prune* function prunes all itemsets from C_k that don't need to be broadcast. As discussed in Section 4.1, an itemset α can be in one of three cases, and we need to consider only Case 3-2, α is frequent at some sites and infrequent at S^i.

```
/* best scenario pruning */
Function Prune ($C_k$, $LL_k^i$ )
1.  $X = \phi$
2.  For each $\alpha \in C_k$
3.      if $\alpha \notin LL_k^i$ AND $\exists\ \alpha \in LL_k^j$
4.      if $sup_{best}(\alpha) \geq \sigma$
5.      $X = X \cup \{\alpha\}$
6.      End if
7.  End for
8.  Return $X$
Function $sup_{best}(\alpha)$
9.  $support = sup^i(\alpha)$
10. $support = support + \sum_{S^i \in S_\alpha^{sent}} sup^i(\alpha)$
11. $support = support + \sum_{S^i \in S_\alpha^{not-sent}}$
      $\lceil min\_sup \cdot |D^i| \rceil - 1$
12. Return support
```

Figure 2. Prune function.

Line 3 checks for this condition. Note that when determining whether α satisfies this condition, the *LFCI*'s collected from all other sites are used. If α is frequent at S^j (or $\alpha \in LL_k^j$), it must be in $LFCI^j$. So, it is sufficient for S^i to examine $LFCI^j$ to check the condition in line 3. In line 4, it calls the $sup_{best}(\alpha)$ function to compute the best scenario global support of α, and compares the result with the global minimum support, σ. If it is greater than or equal to σ, α is included in *X*, which will include only those itemsets that need to be broadcast. Lines 9 – 16 show the function $sup_{best}(\alpha)$. It is the implementation of Lemma3, and the same as described in Section 4.1.

5. Communication cost

In DARCI, communication occurs in lines 3, 7, 11 and 17. In line 3, local frequent 1-itemsets, LL_1^i, are broadcast, and line 7 broadcasts additional candidate 1-itemsets that were returned from the *Prune* function. The cost is represented by $cost_{DARCI}(k=1) = |LL_1^i| + |X_1^i|$. Line 11 sends out $LFCI^i$. Let this cost be $cost_{DARCI}(LFCI^i) = |LFCI^i|$.

Line 17 broadcasts local supports of some itemsets from C_k ($k \geq 2$). There are three cases again, as discussed in Section 4.1, and we need to consider only Case 3: α is frequent at some sites. α is broadcast if it satisfies the conditions in lines 3 and 4 of the *Prune* function, and the cost can be expressed as $cost_{DARCI}(k \geq 2) = Pr_{prune} \cdot |C_{k\geq 2}|$, where Pr_{prune} is the probability that α is frequent at some sites and is potentially globally frequent, and $C_{k\geq 2} = \cup_{\forall k \geq 2} C_k$.

The total communication cost of DARCI is $cost_{DARCI} = cost_{DARCI}(k=1) + cost_{DARCI}(LFCI^i) + cost_{DARCI}(k \geq 2)$, or $|LL_1^i| + |X_1^i| + |LFCI^i| + Pr_{prune} \cdot |C_{k\geq 2}|$.

We estimate the communication cost of DDM in the same way. When $k = 1$, all sites exchange some of candidate 1-itemsets to determine L_1. This cost can be expressed as $cost_{DDM}(k=1) = Pr_{above} \cdot |C_1|$. DDM does not have a cost corresponding to $cost_{DARCI}(LFCI^i)$. The cost for the case when $k \geq 2$ is the same for k=1, and is expressed as $cost_{DDM}(k \geq 2) = Pr_{above} \cdot |C_{k\geq 2}|$. In addition, DDM has one more communication cost component. After the negotiation is completed, DDM still needs to collect local supports of un-broadcast frequent itemsets as discussed in Section 2.1. Let this cost be represented as $cost_{DDM}(post\text{-}neg)$. The total cost of DDM is $cost_{DDM} = cost_{DDM}(k=1) + cost_{DDM}(k \geq 2) + cost_{DDM}(post\text{-}neg)$, or $Pr_{above} \cdot |C_1| + Pr_{above} \cdot |C_{k\geq 2}| + cost_{DDM}(post\text{-}neg)$. Table 3 summarizes the components of communication cost in DDM and DARCI. Note that LL_1^i, X_1^i, $LFCI^i$, Pr_{above}, Pr_{prune}, $C_{k\geq 2}$, and $cost_{DDM}(post\text{-}neg)$ are all determined by the distribution of items in the whole system and are not analytically quantifiable. However, our experiments showed the cost of DARCI is lower than that of DDM; we will discuss this cost comparison issue again in Section 6.3 using experimental data.

Table 3. Components of communication cost.

	$k = 1$	*LFCI*	$k \geq 1$	*post-neg*
DDM	Y	N	Y	Y
DARCI	Y	Y	Y	N

6. Experiment

6.1 Experiment set up

We generated two synthetic datasets using the same method described in [1]. The first dataset, DS1, is T10.I4.D10000K and the second one, DS2, is T20.I6.D10000K. Here, T is the average transaction size, I is the average maximal potentially frequent itemset size, and D is the number of transactions. Then each of them was divided into 100 smaller databases. The number of transactions at each site was about 100K. The number of distinct items was set to 1000 for both datasets. We wrote simulation programs for both DARCI and DDM and ran them on Linux machines with Intel CPU's. We used minimum support of s = 2%, 1.5%, 1%, and 0.5%, and ran the programs with n = 20, 40, 60, 80, and 100 sites.

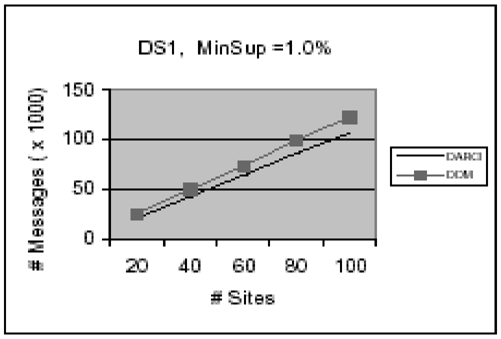
6.2 Experimental result

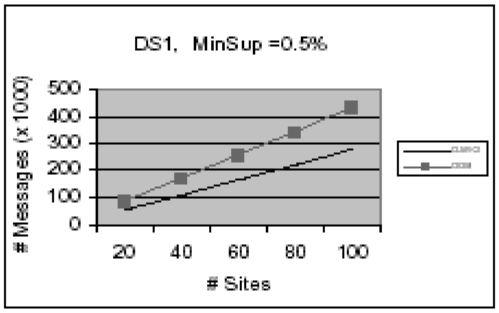
Figures 3 and 4 show some of our experimental results. Figures 3 shows the communication cost over a range of n = 20 to 100 with s = 1% and 0.5%. It shows DARCI's communication cost is smaller than that of DDM. For example, for DS2 with n = 100 and s = 0.5%, DARCI's cost is about 59% of DDM's cost. With both algorithms, the communication cost increases linearly with the number of sites. However, DARCI scales slightly better than DDM (slope of DARCI's graph is smaller than that of DDM). Figure 4 shows the communication cost over minimum supports. Again, we can see that DARCI performs better than DDM. As the minimum support decreases, the number of candidate/ frequent itemsets increases and this adversely affects the performance of DDM more than it does that of DARCI.

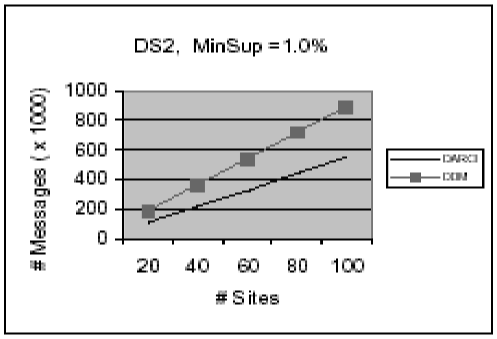
6.3 Communication cost revisited

First, $cost_{DARCI}(k = 1)$ and $cost_{DDM}(k = 1)$ can be considered comparable. $|LL_1^i| + |X_1^i|$ is smaller than $|C_1|$. However, since only the fraction Pr_{above} of $|C_1|$ is broadcast, we can consider them comparable. This was confirmed by our experiment (refer to Table 4). $cost_{DARCI}(LFCI)$ is not included in DDM. We will examine $cost(k \geq 2)$ more closely, here $cost_{DDM}(post\text{-}neg)$ will be considered together with $cost_{DDM}(k \geq 2)$.

For each itemset $\alpha \in C_k$, each site S^i decides whether to send it or not using certain criteria. There are, again, three cases. **Case 1**: α is frequent at all sites. With DARCI, supports are sent in *LFCI*. With DDM, since it is globally frequent, (1) it may be chosen to be sent during the negotiation process or (2) if it is not chosen during the negotiation process, it still needs to be broadcast during the post-negotiation process. Therefore, it is always broadcast with DDM. **Case 2**: α is infrequent at all sites. This is not sent out with both algorithms. **Case 3**: α is frequent at some sites. With DDM, (1) it is chosen during negotiation, (2) it is broadcast after negotiation, or (3) it is not broadcast at all. We found through experiment that the probability of either (1) or (2) ranged from







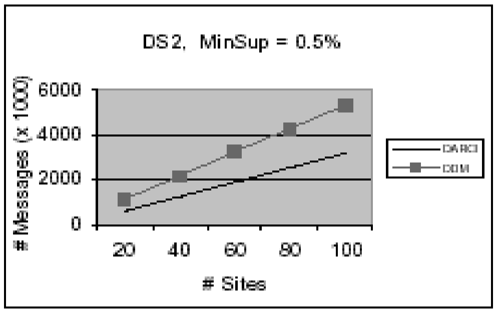
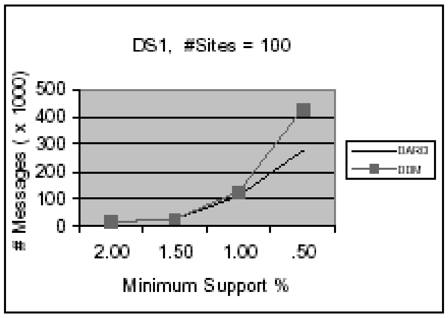


Figure 3. Communication cost vs. number of sites.



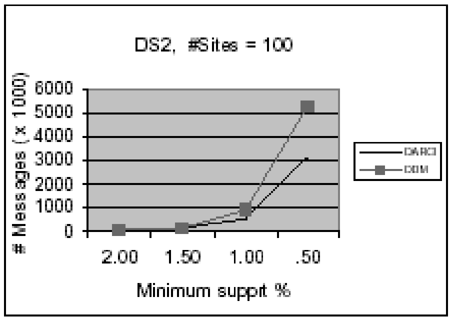


Figure 4. Communication cost vs. minimum support

Table 4. Communication cost.

	DS1		DS2	
	DARCI	DDM	DARCI	DDM
k =1	366	368	554	555
LFCI	651	0	4669	0
k = 2	11	240	81	1587
3	29	254	68	1456
4	18	197	104	1693
5	2	110	62	1561
6	1	42	14	1047
7	1	10	6	586
8	0	1	2	240
9	0	0	0	68
10	0	0	0	12
Total	1079	1222	5560	8805

about 30% to 65%. In the case of DARCI, this can be further divided into three subcases. **Case 3-1**: α is frequent at S^i. Then, it is not broadcast because it was already sent in $LFCI^i$. **Case 3-2**: α is infrequent at S^i, but $sup_{best}(\alpha)$ is less than the global minimum support. Again, we don't need to send it because it can never be globally frequent. **Case 3-3**: α is infrequent at S^i and $sup_{best}(\alpha)$ is equal to or greater than the global minimum support. Then, this must be broadcast. So, α is broadcast only in **Case 3-3**, and our experiment showed that this probability was between about 5% and 20%.

DARCI has an advantage with $cost_{DARCI}(k \geq 2)$, but is disadvantaged by $cost_{DARCI}(LFCI)$. Table 4 and Table 5 show part of our experimental results to illustrate this discussion. Table 4 shows the total communication cost (in number of messages per site) with n = 100 and *min_sup* = 1%. We can see, for DS2, that advantages of $cost_{DARCI}(k \geq 2)$, which is 7913, outweighs the disadvantage of $cost_{DARCI}(LFCI)$, which is 4669. The total cost of DARCI is about 63% of that of DDM for DS2.

Table 5. Breakdown of cost (k = 2), DS2.

	# items	DARCI	DDM
Case 1	1232	0	1232
Case 2	148601	0	0
Case 3	593	-	355
Case 3-1	267	0	-
Case 3-2	245	0	-
Case 3-3	81	81	-
Total	150426	81	1587

Table 5 shows further breakdown of the communication cost for DS2 (in Table 4) with k = 2. DDM broadcasts about 59% (355 out of 593) of Case 3 itemsets but DARCI broadcasts only 14% (81 out of 593) of them.

7. Implementation Issues

There are a couple of choices [12,13,14] to mine frequent closed itemsets. We used CLOSET [14] because it was known to be most efficient. During the execution of the algorithm each site needs to determine the supports of itemsets from *LFCI*'s (e.g., $sup^i(\alpha)$ in line 10 of *Prune*). A straightforward way is to find the smallest superset $\beta \in LFCI^i$ of α, and use the support of β as the support of α (by Property 2). If we do this every time we need the support of an itemset it is computationally expensive. Instead, we utilize a *hash tree*, like the one in *Apriori*, to efficiently extract the supports of candidate itemsets from *LFCI*'s. When each site receives *LFCI*'s, they are combined into *FCI* and stored in a data structure shown in Figure 5. In this example, there are three sites and total 7 frequent closed itemsets, with some of them coming from more than one site (e.g., *bcd* is from both S^2 and S^3). Their local supports are shown in columns 2, 3, and 4. In the data structure, frequent closed itemsets are sorted by their length (in Figure 5, shortest one is at the bottom).

FCI, β	$sup^1(\beta)$	$sup^2(\beta)$	$sup^3(\beta)$
abcd	5	5	
bcd		6	5
acd			5
cde			5
bd	7		
ab			5
cd			6

Figure 5. An example data structure to store *FCI*.

Then, a hash tree is built for each C_k ($k \geq 2$). The hash tree structure is exactly the same as the one that is used in *Apriori*, except that leaf nodes would store additional information (refer to [1] for details of hash tree). Then, each frequent itemset in *FCI*, say β, is run on the tree with the smallest one first moving upward in the *FCI* data structure. When β is run on the hash tree, all k-itemsets which are subsets of β are considered one at a time and each will lead to a leaf node. Once a candidate k-itemset, say α, reaches a leaf node, all the sites where β came from (or *origin sites*) and their supports are stored at the
leaf node. If, however, α has been processed before (e.g., as a subset of β', whose length is shorter than β), its information is not stored (refer to Property 2).

Example 3. Let us run two frequent closed itemsets, *cd* and *cde*, from Figure 5 on the hash tree of C_2. When *cd* is run, its origin site S^3 and support 6 are stored in the leaf node. For *cde*, all 2-itemset subsets are considered: *cd*, *ce*, and *de*. When *cd* is run, after reaching the leaf node, we find that *cd* has been processed. So, we do nothing this time. When considering *ce* and *de*, since they were not processed before, their origin site *S*3 and the support 5 are stored in their corresponding leaf nodes. After running all frequent closed itemsets in *FCI*, we will have extracted supports of all frequent itemsets in *allFI*(*FCI*) and their origin sites. Assume that we need *sup*3(*cd*) when executing line 10 of the *Prune* function. We run *cd* on the hash tree and arrive at its leaf node. Then, we retrieve its support of S^3 stored in the leaf node. ■

This hash tree is also used to check the condition in line 3 of the *Prune* function and to determine the local support of an itemset that is not frequent at that site, which is needed in line 9 of the *Prune* function. Computational cost of DARCI is slightly higher than that of DDM. It requires one more database scan for CLOSET (CLOSET requires two database scans, but the first database scan is for identifying frequent 1-itemsets and it replaces the database scan needed in line 2 of the DARCI algorithm). Afterwards, for each *k* (*k*≥2), both DDM and DARCI scan the database and run each transaction on the hash tree. DDM does this to determine the local supports of all candidate itemsets. DARCI needs to do this to determine the local supports of locally infrequent itemsets. DARC also needs to run *FCI*'s on the hash tree. Since an *FCI* fits in the main memory, this cost is not significant when compared with scanning the database. DDM requires a little bit more computation during negotiation because it may need to compute *P* multiple times for the same itemset, while DARCI processes an item only once. So, overall, we can conclude that the computational costs of DDM and DARCI after k≥2 are comparable but DARCI requires one more database scan at the beginning. But we believe reducing the communication cost is more important in a distributed system and this sacrifice in computational cost is worthwhile.

8. Conclusion

In this paper, we presented a new distributed association rule mining algorithm, called DARCI, which significantly reduces the communication cost of distributed rule mining. DARCI achieves this by utilizing the concept of frequent closed itemsets with using a novel pruning method. Our experiment showed that DARCI reduces the communication cost by as much as 40% of the best known algorithm, DDM.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," VLDB94, pp. 487-499.

[2] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," SIGMOD 2002, 1-12.

[3] J. Park, M. Chen, and P. Yu, "An effective hash based algorithm for mining

association rules," SIGMOD95, pp. 175-186.

[4] A. Savasere, E. Omiecinsk, and S. Navathe, "An efficient algorithm for mining association rules in large databases," VLDB95, pp. 432-443.

[5] H. Toivonen, "Sampling large databases for association rule mining," VLDB96, pp. 134-145.

[6] G. Webb, "Efficient search for association rules," in Proc. Int'l Conf. on Knowledge Discovery and Data Mining, pp. 99- 107, 2000.

[7] F. Coenen, G. Goulbourne, and P. Leng, "Computing association rules using partial totals," PKDD 2001, LNAI 2168,pp. 54-66.

[8] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Efficient mining of association rules using closed itemset lattice," in Information Systems, Vol. 24, No. 1, pp. 25-46, 1999.

[9] R. Agrawal and J. Shafer, "Parallel mining of association rules," in IEEE Trans. on KDE, 8(6):962-969, 1996.

[10] D.W. Cheung, J. Han, V. Ng, A. Fu, and Y. Fu, "A fast distributed algorithm for mining association rules," Int'l Conf. on Parallel and Distributed Information Systems, pp. 31-44, 1996.

[11] A. Schuster and R. Wolff, "Communication-efficient distributed mining of association rules," ACM SIGMOD01, pp.473-484.

[12] N. Pasquier, Y. bastide, T. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemsets For Association Rules," ICDT99, pp 398-416.

[13] M.J. Zaki and C. Hsiao, "CHARM: An Efficient Algorithm For Closed Itemset Mining," Technical Report TR 99-10, Computer Science, RPI, 1999.

[14] J. Pei, J. Han, and R. Mao, "CLOSET: An Efficient Algorithm For Mining Frequent Closed Itemsets," DMKD'00, pp 11-20.