

Domain and Data Partitioning for Parallel Mining of Frequent Closed Itemsets

Peiyi Tang

Dept of Computer Science
University of Arkansas at LR
2801 S. University Ave.
Little Rock, AR 72204

ptang@titius.compsci.ualr.edu

Li Ning

Dept of Computer Science
University of Arkansas at LR
2801 S. University Ave.
Little Rock, AR 72204

lxning@ualr.edu

Ningning Wu

Dept of Information Science
University of Arkansas at LR
2801 S. University Ave.
Little Rock, AR 72204

nxwu@ualr.edu

ABSTRACT

In this paper, we propose an algorithm to partition both the search space and the database for the parallel mining of frequent closed itemsets in large databases. The partitioning of the search space is based on splitting the power set lattice of the total item set to two sub-lattices. Conditional databases are used to partition the large database. The combination of the search space and database partitioning allows parallel processors to mine the frequent closed itemsets independently and thus minimizes the interprocessor communication and synchronization. The partitioning also ensures the load balance among the parallel processors.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications—*Data mining*; D.1.3 [Programming Techniques]: Concurrent Programming—*Parallel programming*

General Terms

Algorithms, Performance, Design

Keywords

Parallel Algorithm, Parallel Data Mining, Partitioning, Conditional Database, Frequent Closed Itemset, Association Rule

1. INTRODUCTION

Since Agrawal et al [1] defined the data mining of association rules in large databases in 1993, there has been tremendous amount of interest in research and practice in this area [2, 3, 4, 5, 6, 7].

Data mining of association rules from sets of items is very expensive in terms of the time and resource requirements. In the real-world databases, both the size of the database (the number of transactions in the database) and the size of

the total item set (the number of attributes in the database) are very large.

Mining of association rules is based on discovering frequent itemsets first. The domain of the search space for frequent itemsets is the power set of the total item set. Its size grows exponentially with the size of the total item set. The size of database also has a huge impact on the time of data mining. The large sizes of the database and the total item set also require large memory and disk spaces. As the size of the modern databases increases, it is increasingly difficult to use a single processor for the data mining tasks. Here is where the parallel processing technique can come in and help. Recently, we see the increasing interest in using parallel processing technique for data mining [8, 9, 6, 10].

In the recent years, there is a lot of interest in discovering frequent *closed* itemsets as a concise representation of frequent itemsets without loss of information [11]. In this paper, we propose an algorithm to partition both the search space and the database for the parallel mining of frequent closed itemsets in large databases. The partitioning of the search space is based on two sub-lattices of the power set lattice of the total item set. Conditional databases are used to partition the large database. The combination of the search space and database partitioning allows parallel processors to mine the frequent closed itemsets independently and thus minimizes the interprocessor communication and synchronization. The partitioning also ensures the load balance among the parallel processors.

The organization of the paper is as follows. The related work is described in Section 2 first. In Section 3, we provide the definitions and preliminaries for the rest of the paper. The search space partitioning is discussed in Section 4. In Section 5, we discuss the conditional databases as the data partitioning for our parallel mining algorithm. Our parallel mining algorithm for frequent closed itemsets are presented in Section 6. Section 7 concludes the paper.

2. RELATED WORK

Parallel data mining has been studied extensively in [8, 12, 9, 6, 10]. All of these papers are about parallel data mining of frequent itemsets and try to parallelize the *Apriori* algorithm [1].

There are basically three ways to parallelize the *Apriori* algorithm:

- *count distribution* where each processor counts the partial support.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

43rd ACM Southeast Conference, March 18-20, 2005, Kennesaw, GA, USA.

Copyright 2004 ACM 0-58113-000-0/00/0004 ...\$5.00.

- *data distribution* where the candidate set is partitioned among the parallel processors and the entire database is replicated across all processors.
- *candidate distribution* where the candidate set is partitioned and the database is replicated selectively.

In this paper, we partition the domain of candidates of frequent closed itemsets using two sub-lattices of the k -prefix and the remaining suffix of the total item set. We use conditional databases for database partitioning. We proved that the mining of closed itemsets in each pair of domain-data partitions can be carried out independently and, thus, the interprocessor communication and synchronization can be minimized.

3. PRELIMINARIES

Let $I = \{i_1, \dots, i_n\}$ be the set of all items called *the total item set*. Without loss of generality, we assume the items in I are sorted in lexicographic order. Any subset of I (including itself) is called *itemset* and can be written as a string of items in lexicographic order. The empty itemset is written as empty string ϵ . Given an itemset x , $|x|$ is the length or cardinality of x . The length of the empty itemset ϵ is 0. For a non-empty itemset x , the i -th ($1 \leq i \leq |x|$) element of x is denoted by $x[i]$. Its substring from i -th through j -th elements of x ($1 \leq i \leq j \leq |x|$) is denoted by $x[i : j]$. If $|x| \geq 1$, the first k ($1 \leq k \leq |x|$) items of x , $x[1 : k]$, is called k -prefix of x and denoted by $pre(k, x)$.

A database D is a set of transactions each of which is a subset of I . Given an itemset x , the *support* of x in D , denoted by $sup_D(x)$ is the number of transactions in D that contains x . That is,

$$sup_D(x) = |\{t \in D \mid x \subseteq t\}| \quad (1)$$

An itemset is *frequent* in D if its support is greater than or equal to a threshold m , i.e. $sup_D(x) \geq m$. To find all the frequent itemsets in a database is the first step in mining association rules. It is also the most resource- and time-consuming step. In real world data mining applications, both the size of the database, $|D|$, and the size of the total item set, $|I|$, are very large. It is, thus, important to partition the large problem to be solved by parallel processing.

A frequent itemset x is *closed* if and only if there is no proper superset of x , x' , such that $sup_D(x) = sup_D(x')$. In other words, a frequent itemset x is *closed* if and only if $sup_D(x) > sup_D(x')$ holds for every x' such that $x \subset x' \subseteq I$. Note that for any x' such that $x \subset x'$, $sup_D(x) \geq sup_D(x')$ holds, because any transaction t in D containing x' also contains x .

Since any subset of a frequent itemset is also frequent, we only need to consider the frequent items in I , i.e. the $i \in I$ such that $sup_D(\{i\}) \geq m$. Without loss of generality, we assume that I is the set of frequent items.

4. PARTITIONING OF POWER SET LATTICE

The search space of the frequent closed itemsets is the power set of I : the set of all subsets of I , denoted by 2^I . Let $|I| = n$. The size of 2^I is $|2^I| = 2^n$. The power set 2^I is a complete boolean lattice, $\langle 2^I, \cup, \cap \rangle$, where the join and meet operations are the set union and intersection, and the top and bottom elements are I and ϵ , respectively.

Assume that we have 2^k parallel processors. We want to partition the power set lattice 2^I into 2^k small search spaces of equal size. We first split I into $pre(k, I) = I[1 : k]$ and $I - pre(k, I) = I[k + 1, n]$. We form two power set lattices: $\langle 2^{I[1:k]}, \cup, \cap \rangle$ and $\langle 2^{I[k+1:n]}, \cup, \cap \rangle$. For each $x \in 2^{I[1:k]}$, we define a partition denoted by P_x as follows:

$$P_x = \{x \cdot y \mid y \in 2^{I[k+1:n]}\} \quad (2)$$

where \cdot is the concatenation operator. Each $x \in 2^{I[1:k]}$ is called the *handle* of partition P_x . Obviously, $P_\epsilon = 2^{I[k+1:n]}$. From the definition (2) above, we have:

$$P_{x_1} \cap P_{x_2} = \emptyset, \forall x_1, x_2 \in 2^{I[1:k]} \wedge (x_1 \neq x_2)$$

The size of each partition P_x is the same: $|2^{I[k+1:n]}| = 2^{n-k}$.

Given an itemset $I = abcde$ and $k = 2$, Figure 1 shows how lattice 2^I is partitioned to four partitions: P_{ab}, P_a, P_b and P_ϵ . The elements of the lattice are shown in lexicographic order. The lines show the partial order of set inclusion (cover), but the lines between the subsets in the different partitions are omitted.

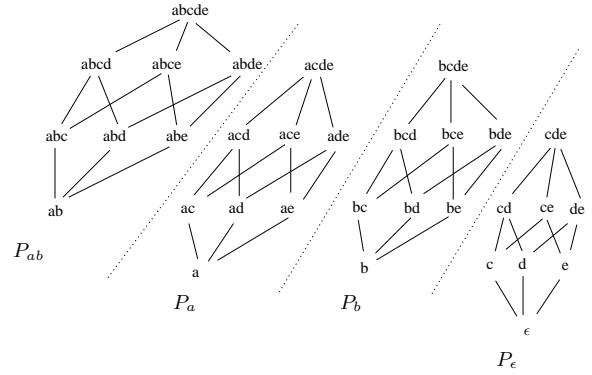


Figure 1: Partition of Lattice 2^{abcde}

The handles of all the partitions can be obtained by recursive bisection of $2^{I[1:k]}$. The algorithm of the recursive bisection is shown in Figure 2.

```

function biSec( $x$ )
 $x$ : a set of item
return: the power set of  $x$ ,  $2^x$ 

 $k := |x|$ ;
 $P := \emptyset$ ;
if ( $k > 0$ ) then
   $Q := biSec(pre(k - 1, x))$ ;
  for each  $q \in Q$  do
     $P := P \cup (q \cdot x[k])$ ;
     $P := P \cup q$ ;
  endfor
endif
return  $P$ ;

```

Figure 2: Bisection of 2^x

5. CONDITIONAL DATABASES

Once the search space 2^I is partitioned to 2^k partitions, the problem can be reduced to find all the frequent closed itemsets in each of the partitions independently on parallel processors.

To discover all the frequent closed itemsets in partition P_x does not need to use the entire database. Instead, only those transactions containing x are needed. The set of these transactions is called *conditional database* denoted by D_x and defined as follows:

$$D_x = \{t - x \mid t \in D \wedge x \subseteq t\} \quad (3)$$

where $t - x$ means $t \cap \bar{x}$ and \bar{x} is the complement of x , i.e. $\bar{x} \cup x = I$ and $\bar{x} \cap x = \epsilon$. To obtain conditional database D_x , we simply need to go through database D once to check every transactions. The algorithm for this is shown in Figure 3. Note that if D has a transaction $t = x$, D_x will have an empty transaction $\epsilon = t - x$.

```

function condDB( $x, D$ )
inputs:
   $x$ : an itemset
   $D$ : database
return: conditional database  $D_x$ 

 $D_x := \emptyset$ ;
for each  $t \in D$  do
  if ( $x \subseteq t$ ) then
     $D_x := D_x \cup \{t - x\}$ ;
  endif
endfor
return  $D_x$ ;

```

Figure 3: Conditional Database

If $x \neq \epsilon$, the size of D_x will be smaller than that of D . This will help reduce the time of searching as well as the memory requirement of the mining for the partition P_x .

We now show that all the frequent closed itemsets in P_x can be discovered by using conditional database D_x only.

The following propositions about the power set lattice $\langle 2^I, \cup, \cap \rangle$ will be used in our discussion.

PROPOSITION 1. *Given t and x in 2^I , if $x \subseteq t$, $(t - x) \cup x = t$ is true.*

PROOF. $(t - x) \cup x = (t \cap \bar{x}) \cup x = (t \cup x) \cap (\bar{x} \cup x) = (t \cup x) \cap I = (t \cup x)$. Since $x \subseteq t$, $t \cup x = t$. \square

PROPOSITION 2. *Given t and x in 2^I , if $t \cap x = \epsilon$, $(t \cup x) - x = t$ is true.*

PROOF. $(t \cup x) - x = (t \cup x) \cap \bar{x} = (t \cap \bar{x}) \cup (x \cap \bar{x}) = (t \cap \bar{x}) \cup \epsilon = (t \cap \bar{x})$. Since $t \cap x = \epsilon$, we have $t \subseteq \bar{x}$. Thus, we have $(t \cap \bar{x}) = t$. \square

PROPOSITION 3. *Given t, x and y in 2^I , if $y \cap x = \epsilon$ and $(y \cup x) \subseteq t$, $y \subseteq t - x$ is true.*

PROOF. Since $(y \cup x) \subseteq t$, we can have $(y \cup x) \cap \bar{x} \subseteq t \cap \bar{x} = t - x$. $(y \cup x) \cap \bar{x} = (y \cap \bar{x}) \cup (x \cap \bar{x}) = (y \cap \bar{x}) \cup \epsilon = y \cap \bar{x}$. Since $y \cap x = \epsilon$, we have $y \subseteq \bar{x}$ and thus, $y \cap \bar{x} = y$. Thus, $y \subseteq t - x$ is true. \square

LEMMA 1. *Given x and y in 2^I , if $y \cap x = \epsilon$, $sup_{D_x}(y) = sup_D(y \cup x)$ is true.*

PROOF. Recall that $D_x = \{t - x \mid t \in D \wedge x \subseteq t\}$ in (3). For each $t' = t - x$ in D_x containing y , we have $y \subseteq t' = t - x$ and $x \subseteq t$. According to PROPOSITION 1, we have transaction $t = t' \cup x$ in D . Since $y \subseteq t'$, we have $y \cup x \subseteq t' \cup x = t$. That is, transaction $t = t' \cup x$ in D contains $y \cup x$. Thus, the number of transactions in D containing $y \cup x$ is at least as large as the number of transactions in D_x containing y , i.e. $sup_{D_x}(y) \leq sup_D(y \cup x)$. On the other hand, for each t in D containing $y \cup x$, i.e. $(y \cup x) \subseteq t$, we have $x \subseteq (y \cup x) \subseteq t$. Note that $y \cap x = \epsilon$. According to PROPOSITION 3, we now have $y \subseteq (t - x)$. Thus, we have transaction $t - x$ in D_x contains y . Therefore, the number of transactions in D_x containing y is at least as large as the number of transactions in D containing $y \cup x$, i.e. $sup_{D_x}(y) \geq sup_D(y \cup x)$. Therefore, $sup_{D_x}(y) = sup_D(y \cup x)$. \square

Following are the two major theorems for using conditional databases to discover all the frequent closed itemsets.

THEOREM 1. *If $y \in P_x$ is a frequent closed itemset in database D , $y - x$ is also a frequent closed itemset in the conditional database D_x and $y = (y - x) \cup x$.*

PROOF. Note that $(y - x) \cap x = y \cap (\bar{x} \cap x) = y \cap \epsilon = \epsilon$. According to LEMMA 1, we have $sup_{D_x}(y - x) = sup_D((y - x) \cup x)$. Since $y \in P_x$, we have $x \subseteq y$ according to the definition of partition P_x in (2). According to PROPOSITION 1, we have $(y - x) \cup x = y$. Therefore, we have $sup_{D_x}(y - x) = sup_D(y)$. Since y is a frequent itemset, we have $sup_{D_x}(y - x) = sup_D(y) \geq m$. So, $y - x$ is a frequent itemset in D_x . We now prove that $y - x$ is also a closed frequent itemset in D_x .

Suppose that $y - x$ is not a closed frequent itemset in D_x . Then there is an itemset z such that $y - x \subset z$ and $z \subseteq I - x$ (i.e. $z \subseteq \bar{x}$), and $sup_{D_x}(y - x) = sup_{D_x}(z)$. We already have $sup_{D_x}(y - x) = sup_D(y)$. Since $z \subseteq \bar{x}$, we have $z \cap x = \epsilon$. According to LEMMA 1 again, we have $sup_{D_x}(z) = sup_D(z \cup x)$. Therefore, we have $sup_D(y) = sup_D(z \cup x)$.

We have $y - x \subset z$ and want to prove $(y - x) \cup x \subset (z \cup x)$. First, $y - x \subset z$ implies $y - x \subseteq z$, from which we can have $(y - x) \cup x \subseteq (z \cup x)$. We can prove that $(y - x) \cup x \neq (z \cup x)$. Suppose that $(y - x) \cup x = (z \cup x)$. Then we have $((y - x) \cup x) \cap \bar{x} = (z \cup x) \cap \bar{x}$. Note that we have both $z \cap x = \epsilon$ and $(y - x) \cap x = \epsilon$. According to PROPOSITION 2, we have both $(z \cup x) \cap \bar{x} = z$ and $((y - x) \cup x) \cap \bar{x} = y - x$. Therefore, we have $z = y - x$, which contradicts $y - x \subset z$ we had. Thus, $(y - x) \cup x \subset (z \cup x)$ is true. Note that $y \in P_x$. According to the definition of partition P_x in (2), we have $x \subseteq y$. According to LEMMA 1 again, $(y - x) \cup x = y$. In other words, $y \subset (z \cup x)$ is true.

Now we found a $z \cup x$ such that $y \subset (z \cup x)$ and $sup_D(y) = sup_D(z \cup x)$. This means that y is not a closed frequent itemset. This proves that $y - x$ is a closed frequent itemset in D_x .

Since $x \subseteq y$, we have $y = (y - x) \cup x$ according to PROPOSITION 1. \square

THEOREM 2. *If z is a frequent closed itemset in the conditional database D_x , $z \cup x$ is also a frequent closed itemset in database D .*

PROOF. Similar to the proof of THEOREM 1. \square

```

procedure pMine( $I, k, D, C, m$ )
in:
   $I$ : the total item set with  $|I| = n$ 
   $k$ : size of prefix to partition the search space,  $1 \leq k \leq n$ 
   $D$ : the database
   $m$ : the minimum support threshold
in-out:
   $C$ : the set of frequent closed itemsets

(1)  $n := |I|$ ;
(2)  $I' := I[k+1 : n]$ ;
(3)  $X := biSec(k, I[1 : k])$ ;
(4) forall  $x \in X \wedge x \neq \epsilon$  do parallel on  $2^k - 1$  processors
(5)    $D_x := condDB(x, D)$ ;
(6)    $C_x^{I'} := closed(D_x, x, I', m)$ ;
(7)    $C_x := \{x \cup z \mid z \in C_x^{I'}\}$ ;
(8)    $C := C \cup (\bigcup_{(x \in X) \wedge (x \neq \epsilon)} C_x)$ ;
(9) endforall
(10) if ( $k \leq |I'|$ ) then
(11)   call pMine( $I', k, D, C, m$ );
(12) else
(13)   do  $k := k - 1$  while  $k > |I'|$ ;
(14)   if ( $k > 0$ )
(15)     call pMine( $I', k, D, C, m$ );
(16)   endif
(17) endif

```

Figure 4: Parallel Mining of Closed Itemsets

6. PARALLEL MINING OF FREQUENT CLOSED ITEMSETS

6.1 Partitioning of Frequent Closed Itemsets

According to THEOREM 1, every frequent closed itemset y in partition P_x can be obtained by mining the corresponding frequent closed item set $y - x$ first and then union it with x , i.e. $(y - x) \cup x$.

THEOREM 2 says that for every frequent closed itemset z mined from D_x , $z \cup x$ is also a frequent closed itemset in D . But, $z \cup x$ may not in partition P_x .

In fact, we only need to mine those closed itemsets z from D_x such that $z \subseteq I' = I[k+1 : n]$. In other words, the search space should be restricted to $2^{I[k+1:n]}$ rather than $2^{(I-x)}$. Let the set of such closed itemsets from D_x be denoted by $C_x^{I'}$. Then, the set of all closed itemsets of database D in P_x , denoted by C_x , is

$$C_x := \{x \cup z \mid z \in C_x^{I'}\} \quad (4)$$

And we have

$$C_{x_1} \cap C_{x_2} = \emptyset, \forall x_1, x_2 \in 2^{I[1:k]} \wedge (x_1 \neq x_2) \quad (5)$$

We can split the task of finding the frequent closed itemsets in the database D to 2^k independent tasks to find C_x from conditional databases D_x for each $x \in 2^{I[1:k]}$. The set of all frequent close itemsets in the database D , denoted by C , can be obtained as follows:

$$C = \bigcup_{x \in 2^{I[1:k]}} C_x \quad (6)$$

6.2 Parallel Mining Algorithm

Note that each conditional databases D_x for $x \neq \epsilon$ is smaller than the original database D and the search space is

reduced from 2^I to $2^{I[k+1:n]}$. The search time and the memory requirement to find C_x from D_x will be smaller than those to find the entire C directly from D . Since each task of finding C_x from D_x is independent with others, they can be executed in parallel on parallel processors.

The parallel algorithm to discover the frequent closed itemsets is shown as procedure pMine(I, k, D, C, m) in Figure 4. Here, I is the total itemset, D the database, k the size of prefix of I used to partition the search space, and m the minimum support threshold. C is the input-output parameter for the set of frequent closed itemsets. The main function of the data mining (not shown) simply calls pMine(I, k, D, C, m) with an empty C . In Figure 4, the set of the handles of the partitions, x , are obtained in the lines 1 through 3. Lines 4 through 9 are the parallel **forall** loop to mine frequent closed itemsets in each partition P_x for $x \neq \epsilon$ from conditional database D_x using $2^k - 1$ parallel processors. Within the parallel loop, D_x , $C_x^{I'}$ and C_x are all local variables. Line 5 calls function *condDB*() in Figure 3 to form the local conditional database D_x . Function call *closed*(D_x, x, I', m) in line 6 is to find the closed itemsets as subsets of I' (including the possible empty set ϵ) from D_x . Line 8 is the reduction to collect all closed itemsets mined by parallel processors and add them to the global variable C .

For $x = \epsilon$, the conditional database D_ϵ is the same as database D . The direct data mining for C_ϵ on D_ϵ will take longer time than other tasks. To avoid load imbalance, we do not use another parallel processor to do the job. Instead, we run the same parallel algorithm recursively with I' as the total item set to find all the closed itemsets in $2^{I'}$ if $|I'| \geq k$ (line 11). If $|I'| < k$, we reduce k to make $|I'| = k$ and call pMine(I', k, D, C, m) if $k > 0$ (line 15). If $|I'| = k = 0$, we do not need to call pMine(I', k, D, C, m), because only possible candidate to be added to C is the empty itemset ϵ .

function closedIS(D_x, x, I', m)

inputs:

D_x : Conditional Databases

x : the hand of partition P_x

I' : restricted item set

m : the minimum support threshold

return:

the set of closed itemsets as subsets of I' from D_x

$B := \emptyset$;

$C := \emptyset$;

call traditional mining algorithm to find all closed itemsets with minimum support m in D_x and put them in B ;

for each $c \in B$ **do**

if ($c \cap (I - x - I') = \emptyset$) **then**

$C := C \cup c$;

endif

endfor

return C ;

Figure 5: Local Mining

Function *closed*(D_x, x, I', m) is use to find all the closed itemsets in D_x in the domain $2^{I'}$. Note that the transactions in D_x may contain items from $I - x$ and we have $I - x \supseteq I'$. Note that the candidate closed itemsets in $2^{I'}$ should also include the empty set $\epsilon \in 2^{I'}$. The empty set ϵ is a closed

itemset of D_x if and only if there is no item in $I - x$ which appears in every transactions of D_x . In any case, we will not discuss the efficient algorithm to find the closed itemsets in the restricted domain in this paper. Instead, we use the traditional closed itemset mining algorithms such as A-close [11] or CLOSET [13] to find all closed itemsets in the domain 2^{I-x} and then filter through them to find the closed itemsets in $2^{I'}$. The algorithm is shown in Figure 5.

Let us use an example to illustrate our parallel mining algorithm. Suppose we have a database

$$\{bcade, bfd, cde, bcae, cde\}$$

and the minimum support is $m = 2$. We first pre-process the database to find the frequent items, remove the infrequent items from the database, and then order the frequent items in the increasing order of their supports. The result is the database

$$D = \{abcde, bd, cde, abce, cde\}$$

with the total item set $I = abcde$. Suppose that $k = 2$. Figure 6 illustrates the parallel mining of this example.

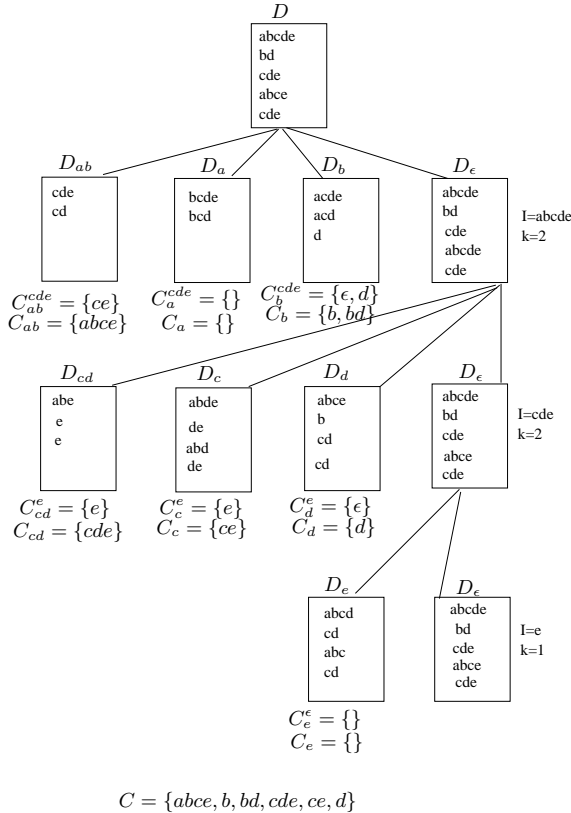


Figure 6: Parallel Data Mining of Example

The first round of parallel mining is carried out by partitioning the search space 2^{abcde} into four partitions. The set of handles of partitions is $X = \{ab, a, b, \epsilon\}$ and $I' = cde$. This partitioning is shown in Figure 1. The four conditional databases, D_{ab}, D_a, D_b, D_e are shown in Figure 6. The local set of closed itemsets, $C_x^{I'}$ and C_x for each $x \in 2^{pre(k, I)}$, mined from the conditional databases D_x are also shown below them in Figure 6.

In the second round of parallel mining, $pMine()$ is called with $I = cde$. The partitioning of the lattice 2^{cde} with $k = 2$

is shown in Figure 7 (again, the cover relation between the subsets in different partitions is not shown). The parallel

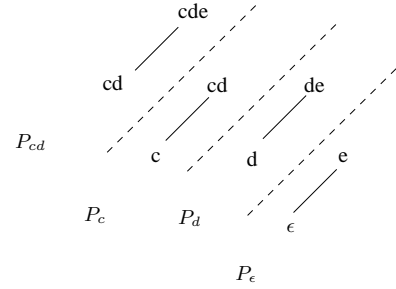


Figure 7: Partition of Lattice 2^{cde}

mining continues with the conditional databases D_{cd}, D_c, D_d and D_e , which are also shown in Figure 6.

The third round of parallel mining is similar, but with $I = e$ and $k = 1$. The partitioning of the lattice 2^e is shown in Figure 8

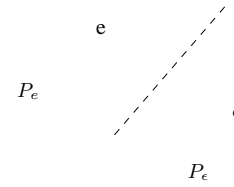


Figure 8: Partition of Lattice 2^e

The conditional database D_e can be found in Figure 6 again. Note that we have $I' = \epsilon$ this time and only possible closed itemset from 2^ϵ is ϵ .

The set of all closed frequent itemsets mined and collected from the parallel processors after the three rounds of parallel mining is

$$C = \{abce, b, bd, cde, ce, d\}$$

7. CONCLUSION

We have presented a parallel algorithm for mining frequent closed itemsets from large databases. By partitioning both the search space and the database, we are able to parallelize the data mining problem into $2^k - 1$ completely independent small tasks with good load balance.

We are currently running performance evaluation of our parallel algorithm. The preliminary data show that a good sub-linear speedup can be achieved.

8. REFERENCES

- [1] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Database*, pages 207–216, 1993.
- [2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.
- [3] Hannu Toivonen. Sampling large databases for association rules. In *In Proceedings of 1996*

International Conference on Very Large Data Bases, pages 134–145, September 1996.

- [4] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 283–286, 1997.
- [5] Jr. Reberto J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 85–93, 1998.
- [6] Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.
- [7] Karam Gouda and Mohammed J. Zaki. Efficiently mining maximal frequent itemsets. In *Proceedings of the First IEEE International Conference on Data Mining*, pages 163–170, 2001.
- [8] R. Agrawal and J. C. Shafer. Parallel mining of association rules. *IEEE Transactions On Knowledge And Data Engineering*, 8:962–969, 1996.
- [9] Mohammed Javeed Zaki, Srinivasan Parthasarathy, and Wei Li. A localized algorithm for parallel association mining. In *Proceedings of the 1997 ACM Symposium on Parallel Algorithms and Architectures*, pages 321–330, 1997.
- [10] Dejiang Jin and Sotirios G. Ziavras. A super-programming approach for mining association rules in parallel on pc clusters. *IEEE Transactions on Parallel and Distributed Systems*, 15:783–794, 2004.
- [11] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proceeding of the 7th International Conference on Database Theory*, pages 398–416, 1999.
- [12] M. J. Zaki, M. Ogihara, S. Parthasarathy, and W. Li. Parallel data mining for association rules on shared-memory multiprocessors. In *Proceedings of the 1996 ACM/IEEE conference on Supercomputing*. Article No. 43, 1996.
- [13] Jian Pei, Jiawei Han, and Runying Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *Proceedings of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.