Parallel Regular-Frequent Pattern Mining in Large Databases

G Vijay Kumar, Dr V Valli Kumari

Abstract—Mining interesting patterns in various domains is an important area in data mining and knowledge discovery process. A number of parallel and distributed frequent pattern mining algorithms have been proposed so far for the large and/or distributed databases. Occurrence frequency is not the only criteria to mine the patterns but also occurrence behavior (regularity) of a pattern may also be included in mining process. A pattern is *frequent* if the occurrence frequency is not less than the user given support threshold and this pattern is *regular* if it satisfies the user given regularity measure. So far a few efforts have been made to mine regular patterns but there is no suitable algorithm to mine *regular-frequent* patterns in parallel and distributed environment. Therefore, in this paper we introduced a new method called PRF-method (Parallel Regular Frequent Pattern-method) to discover *regular-frequent* patterns in large databases using vertical data format which requires only one database scan. PRF-method works in parallel at each local site in order to reduce I/O cost and inter-process communication and generates all *regular-frequent* patterns in the final phase. The experimental results show that our PRF-method is highly efficient in large databases.

Index Terms— Regular patterns, frequent patterns, vertical data format, parallel algorithms, large databases.

1 INTRODUCTION

The fundamental and important area in data mining research is Frequent pattern mining. Apriori algorithm [1] [2] was the fundamental algorithm to mine frequent item sets that was introduced by Agrawal et.al in 1993. Apriori algorithm uses prior knowledge and employs an iterative approach known as level-wise search to generate frequent patterns (FP). In the first scans of the database, it generates 1-itemsets, recursively it generates 2-itemset and then 3-itemset and continues until all the frequent item sets were generated. To generate k-itemset, it requires k database scans and also large amount of memory is needed which reduces the efficiency of the algorithm. Han et.al [3] introduced frequent pattern tree (FP-tree) and FP-growth algorithm to generate frequent patterns without candidate generation in the year 2000. This algorithm has been found to be efficient in memory as well as in execution time when compared to apriori algorithm because it needs only two database scans to discover the complete set of frequent items from the database. When the occurrence frequency of an item is not less than the user given minimum support threshold (i.e support) then it is said to be frequent itemset.

Tanbeer et al. [4, 5] introduced a new problem called regular patterns which follow temporal regularity of a pattern in a transactional database. The pattern is supposed to be a regular pattern when its occurrence behavior is less than or equal to user given maximum regularity threshold (i.e., max_reg). RP-tree [4] was introduced which is used to mine regular patterns in a transactional database with two database scans like FP-tree [3]. In the first scan they identify length-1 regular items for a given max_reg by creating item header table called R-table with regularity and support. Next, RP_tree is constructed in the second scan in support descending order in order to mine the complete set of regular patterns. The importance of such type of patterns with occurrence frequency and regularity can be used in a wide range of applications in the real world.

There are a number of parallel and distributed FP mining algorithms [5-8] which are developed based on apriori and FP-tree algorithm. There was much attention among the authors to mine frequent patterns, sequential patterns in large databases from the last decade but a very little attention on regular pattern mining which was recently introduced. Therefore, among the available field of work we initiate to mine *regular-frequent* patterns in large databases using vertical data format. Therefore in this paper, we introduced an efficient method named PRF-method to mine *regular-frequent* patterns on large databases with one database scan using vertical data format [9, 10, 11]. In the first step we partition the database into several small equal parts and assign to each processor at their respective local sites. In the next step we require a database scan to convert the transactional database into vertical format. Now, our method individually mines all local vertical databases in parallel towards discovering all possible global *regular-frequent* patterns with PRF-local table and PRF-header table without any inter-process communication between the processors.

The rest of the paper is organized as follows. Section 2 is the related work which describes about frequent patterns, regular patterns and parallel and distributed frequent pattern mining algorithms. In section 3 we described the problem definition to mine *regular-frequent* patterns in parallel and distributed environment. In section 4 the mining procedure of PRF-method is discussed in detail. In section 5 our experimental results are shown. Finally we conclude the paper in section 6.

- *G Vijay Kumar is currently pursuing phd degree in computer science and engineering, Acharya Nagarjuna University, India, PH-9247411556. E-mail: gvijay_73@yahoo.co.in*
- *Dr. V Valli Kumari is currently working as a Professor in the department of CS&SE, AU college of engineering, India, PH-9440065256. E-mail: vallikumari@gmail.com*

2 Related Work

In this section we discuss on frequent pattern mining, regular pattern mining, vertical data format and parallel and distributed frequent pattern mining. Frequent pattern mining [1-3] was the fundamental research area in data mining. A good number of algorithms have been developed so far to mine frequent patterns in different domains. A good number of algorithms are based on apriori [6, 7] and FP-tree [8, 9, 10] in parallel and distributed databases. The *apriori* algorithm was the first technique to find frequent patterns. In this algorithm it generates k frequent item set from previously generated k-1 frequent itemset. It requires k database scans to generate length-k frequent item set and to handle the candidates generation it needs large amount of memory. To overcome the above disadvantages a tree based datastructure called FP-tree and FP-growth algorithm introduced by Han et. al [3] in the year 2000. In this method the database was captured with a database scan to find a set of distinct items along with their respective support count value. While scanning second time, it constructs FP-tree in the form of frequency-descending order. Each node of the FP-tree contains item name and its count which represents how many times that an item was occurred in the transactions from root node to the tail node. The FP-growth algorithm can be applied to mine frequent patterns without any additional database scan. Recently, Tanbeer et. al [4] introduced a new problem to mine regular patterns in a database which follows a temporal regularity in the occurrence behavior of a pattern. A tree-based data structure called RP-tree which captures user-given maximum regularity threshold and mines regular patterns from the database with two database scans. It creates an item header table, called regular table (R-table) to store items with respective regularity and support in the first scan. In the second scan the RP-tree is constructed in support-descending order only for the regular itemsets in the R-table. Most of the construction process of an RP-tree is similar to FP-tree construction technique however in RP-tree no node maintains the support count instead it maintains the transaction_ids.

The main advantages of using vertical data format [11][12] are it requires only one database scan, it reduces input/output operations, it uses simple operations like union, intersection, deletion etc., it judges non regular itemsets before generating candidate itemsets. So, in this paper we used vertical data format (i.e., *itemset* : *tid_set*) to mine *regular-frequent* patterns with PRF-method. Most of the efficient frequent pattern algorithms in parallel and distributed environment were based on FP-tree and FP-growth mining process. The algorithms FP-Forest [8], Load Balancing FP-tree [9] and Parallel Pattern-tree [10] distributes large database into several small parts to each of the local sites. Later they construct their local FP-trees separately at each site according to their FP-tree construction procedure and mine for global frequent patterns at the final stage. In this paper we consider [5] [10] to mine global *regular-frequent* patterns in large databases using vertical database format in parallel and distributed environment.

3 Problem Definition

In this section we describe the concepts of *regular-frequent* pattern mining and define the fundamental definitions of the problem to obtain complete set of *regular-frequent* patterns in parallel and distributed environment.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items. A set $X = \{i_j, \ldots, i_k\} \subseteq I$, where $j \leq k$ and $j, k \in [1, n]$ is called a pattern(or an itemest). A transaction $t = (tid, Y)$ is a couple where *tid* is a transaction-id and Y is a pattern. Let size(t) be the size of t, i.e., the number of items in Y. A transaction database DB over I is a set of transactions $T = \{t_1, \ldots, t_m\}$, $m = |DB|$ is the size of DB, i.e., the total number of transactions in DB. If $X \subseteq Y$, which means that t contains X or X occurs in t and denoted as t_j^X, $j \in [1, m]$. Therefore, $T^X = \{t_j^X, \ldots, t_k^X\}$, $j \leq k$ is and $j, k \in [1, m]$ is a set of all transactions where pattern X occurs in DB.

3.1 Definition 1 (Frequent pattern X)

The total number of transactions in a DB that contains pattern X is called the support of X, $Sup(X)$. Hence $Sup(X) = |T^X|$, where $|T^X|$ is the size of T^X. If $Sup(X) \geq min\text{-}sup$ then X is a frequent pattern.

3.2 Definition 2 (Regularity of a frequent pattern X)

Let t^X_{j+1} and t_j^X, $j \in [1, (m - 1)]$ be two consecutive transactions where X appears. The difference between these two consecutive transactions can be defined as a period of X, say p^X (i.e., $p^X = t^X_{j+1} - t^X$, $j \in [1, (m - 1)]$). For simplicity, to calculate the period of a pattern, we consider the first transaction in the DB as null i.e., $t_{first} = 0$ and the last transaction is the m^{th} transaction i.e., $t_{last} = t_m$. Let for a T^X, P^X be the set of all periods of X i.e., $P^X = \{p_1^X, \ldots, p_r^X\}$, where r is the total number of periods in P^X. Then the regularity of X can be denoted as $reg(X) = max\{p_1^X, \ldots, p_r^X\}$. A pattern X is called a *regular-frequent* pattern if it satisfies both of the following criteria's: (a) the support of a pattern X is not less than the user given minimum support threshold say *min_sup*, δ, and (b) The regularity of a frequent pattern X is not greater than the user given regularity threshold say *max_reg*, λ, in percentage of $|DB|$.

3.3 Definition 3 (Parallel Regular Frequent)

Let $S = \{S_1, \ldots, S_n\}$be n-number of sites in homogeneous distributed system. The database DB is divided into n equal partitions as $db_1, db_2, \ldots, db_n$, and each partition db_i is assigned to a site S_i. Let $Sup_i(X)$ denotes the support count of X in db_i *and* $Reg_i(X)$ denotes the regularity of a patter X in db_i. Hence, we call $Sup_i(X)$ and $Reg_i(X)$ as the local support count and local regularity of a pattern X in db_i respectively and $Sup(X)$ and $Reg(X)$ as the global support count and global regularity of a pattern X in DB respectively. For a given *min_sup*, δ (support) and *max_reg*, λ (regularity), we contain a header table (Table-3) which collects all δ's and λ's from the local sites to find global support and global regularity respectively. X is global *regular-frequent* pattern if it satisfies the given two thresholds (i.e., $(Sup(X) = Sum(Sup_1, Sup_2, \ldots, Sup_n) \geq \delta)$ and $(Reg(X) = Max(Reg_1, Reg_2, \ldots, Reg_n) \leq \lambda)$).

4 Parallel and Distributed Regular Frequent Pattern Mining

In this section we explain parallel and distributed *regular-frequent* pattern mining with our proposed PRF-method. In this we consider a distributed environment containing different sites (locations) where each site contains a processor, memory and other needful resources. Divide the large database into small, non-overlapping and equal partitions in order to distribute for n-number of sites. We explain parallel and distributed *regular-frequent* pattern mining with our example database *DB* (Table 1).

TABLE 1
Example Database *DB*

S_0		S_1	
Tid	*Transaction*	*Tid*	*Transaction*
1	a, b, c, d	1	b, c, d, e
2	a, b	2	a, c, d
3	a, c, d, e	3	a, d, e
4	b, d, e	4	a, b, c, d, e
5	a, c, d, e	5	a, c, d, e
6	b, c, d	6	c, d
7	a, d, e	7	b, c, d
8	a, b, c, d, e	8	b, d, e
9	a, b, c, e	9	a, b, c

TABLE 2
Vertical Database

S_0		S_1	
Items	*Transaction_id*	*Items*	*Transaction_id*
a	1, 2, 3, 5, 7, 8, 9	a	2, 3, 4, 5, 9
b	1, 2, 4, 6, 8, 9	b	1, 4, 7, 8, 9
c	1, 3, 5, 6, 8, 9	c	1, 2, 4, 5, 6, 7, 9
d	1, 3, 4, 5, 6, 7, 8	d	1, 2, 3, 4, 5, 6, 7, 8
e	3, 4, 5, 7, 8, 9	e	1, 3, 4, 5, 8

Table 1 is the horizontal database that contains two sites S_0 and S_1, each one having nine transactions. We require only one database scan to mine *regular-frequent* patterns by using PRF-method. At each local processor the horizontal database converts into vertical database with one database scan which are shown in Table 2. Let us consider the global minimum support *min_sup*, $\delta = 8$ and the global maximum regularity of a pattern *max_reg*, $\lambda = 4$ are two measures to mine global *regular-frequent* patterns with the help of Table 3 (i.e., PRF-header table). There is a header table array which is somewhat similar to PP-tree [10] header table that collects all local *max_reg's* and all *min_sup's* respectively.

After converting the database into vertical format the mining process starts with the proposed PRF-method by finding all the *periods* of length-1 items at each of their individual local sites (i.e., P_0^X, P_1^X, . . . , P_n^X). To obtain the *periods* of a pattern from the above two processors in Table 1, let us consider the first transaction be a null transaction say $t_{first} = 0$ and the last transaction is the ninth transaction say $t_{last} = 9$. For example, in site S_1 item *e* occurs in 1, 3, 4, 5 and 8 transactions respectively. The *periods* computation of *e* are (1 - t_{first} =) 1, (3 - 1 =) 2, (4 - 3=) 1, (5 - 4 =) 1, (8 - 5 =) 3, (9 - 8 =) 1, where $t_{first} = 0$ and $t_{last} = 9$. Therefore the maximum *period* of item *e* in S_1 is called a regularity of a pattern i.e., max_e (1, 2, 1, 1, 3, 1) = 3 which can be seen in Table 4.

TABLE 3
PRF Header Table

Items	S_0	S_1	--	S_n	Total
i_1	$Sup_0(i_1)$	$Sup_1(i_1)$	--	$Sup_n(i_1)$	$\sum_i(\text{Sup}_i(i_1))$
	$Reg_0(i_1)$	$Reg_1(i_1)$	--	$Reg_n(i_1)$	$Max_i(Reg_i(i_1))$
i_2	$Sup_0(i_2)$	$Sup_1(i_2)$	--	$Sup_n(i_2)$	$\sum_i(Sup_i(i_2))$
	$Reg_0(i_2)$	$Reg_1(i_2)$	--	$Reg_n(i_2)$	$Max_i(Reg_i(i_2))$
⋮	⋮	⋮	--	⋮	⋮
i_m	$Sup_0(i_m)$	$Sup_1(i_m)$	--	$Sup_n(i_m)$	$\sum_i(Sup_i(i_m))$
	$Reg_0(i_m)$	$Reg_1(i_m)$	--	$Reg_n(i_m)$	$Max_i(Reg_i(i_m))$

TABLE 4
PRF Local Tables with P_i^X, Sup_i and Reg_i

S_0				S_1			
Items	P_0^X	Sup_0	Reg_0	*Items*	P_1^X	Sup_1	Reg_1
a	1, 1, 1, 2, 2, 1, 1	7	2	a	2, 1, 1, 1, 4	5	4
b	1, 1, 2, 2, 2, 1	6	2	b	1, 3, 3, 1, 1	5	3
c	1, 2, 2, 1, 2, 1	6	2	c	1, 1, 2, 1, 1, 1, 2	7	2
d	1, 2, 1, 1, 1, 1, 1, 1	7	2	d	1, 1, 1, 1, 1, 1, 1, 1, 1	8	1
e	3, 1, 1, 2, 1, 1	6	3	e	1, 2, 1, 1, 3, 1	5	3

PRF-header table (Table 5) collects all local supports and local regularities of length-1 itemset to identify according to the given two thresholds ($\lambda = 4$ and $\delta = 8$). In the process of PRF-method, it finds for global minimum support threshold from all the sum of supports ($Sup(X) = Sum(Sup_1, Sup_2, \ldots, Sup_n)$ available in the header table. For example $Sup(e) = 6 + 5 = 11$ which satisfies the support $\delta = 8$ and then it calculates maxi-

mum regularity measure among all local regularities (i.e., $Reg(X) = Max(Reg_1, Reg_2, \ldots, Reg_n) \leq\leq \lambda$) for the same example, in local site S_0, P_0^e is max_e (3, 1, 1, 2, 1, 1) = 3 which is less than 4 and in local site S_1, P_1^e is max_e (1, 2, 1, 1, 3, 1) = 3 and *Reg(e)* finds the global regularity is *Max* (3, 3) = 3. So, *Reg*(*e*) satisfies the regularity threshold (λ = 4) so *e* is a regular item. And then So in this example item *e* is a *regular-frequent* item. If *Reg*(*e*) in S_1 did not satisfies the *regularity* threshold or/and *support* threshold in PRF-header table it removes *e* from the table.

TABLE 5
PRF Header Table with Global Regularities and Global Supports

Items	Sup_0	Sup_1	Sup_0+Sup_1	Reg_0	Reg_1	$Max(Reg_0, Reg_1)$
a	7	5	**12**	2	4	**4**
b	6	5	**11**	2	3	**3**
c	6	7	**13**	2	2	**2**
d	7	8	**15**	2	1	**2**
e	6	5	**11**	3	3	**3**

TABLE 6
Length 2 Itemsets

S_0		S_1	
Items	*Transaction_id*	*Items*	*Transaction_id*
a, b	1, 2, 8, 9	a, b	4, 9
a, c	1, 3, 5, 8, 9	a, c	2, 4, 5, 9
a, d	1, 3, 5, 7, 8	a, d	2, 3, 4, 5, 9
a, e	3, 5, 7, 8, 9	a, e	3, 4, 5
b, c	1, 6, 8, 9	b, c	1, 4, 7, 9
b, d	1, 4, 6, 8	b, d	1, 4, 7, 8
b, e	4, 8, 9	b, e	1, 4, 8
c, d	1, 3, 5, 6, 8	c, d	1, 2, 4, 5, 6, 7
c, e	3, 5, 8, 9	c, e	1, 4, 5
d, e	3, 4, 5, 7, 8	d, e	1, 3, 4, 5, 8

Table 5 summarizes all global support values and global regularity values of lenth-1 itemset for our running example. For further processing with simple *and operation* we mine for length-2 itemset ignoring all items that do not satisfy, since *regular-frequent* patterns follow the downward closure property. In the running example all length-1 items satisfies the given two thresholds, so all the items will be processed to form length-2 itemset. In Table 6, 7 and 8 we can see the process of mining *regular-frequent* patterns of length-2.

In Table 8 itemset (a, b) did not satisfied the two given measures, (b, c) did not satisfied regularity measure, (b, e) and (c, e) did not satisfied the support threshold. So all these four length-2 itemsets are not *regular-frequent* therefore they will be ignored. Again the same process continues until no *regular-frequent* itemsets found. The advantage of our proposed method is, it takes less time to convert the database into vertical format and the mining process takes place individually at all local sites without having any inter-process communication among the processors.

TABLE 7
PRF Local Tables with P_i^X, Sup_i and Reg_i

S_0				S_1			
Items	P_0^X	Sup_0	Reg_0	*Items*	P_1^X	Sup_1	Reg_1
a, b	1, 1, 6, 1	4	6	a, b	4, 5	2	5
a, c	1, 2, 2, 3, 1	5	3	a, c	2, 2, 1, 4	4	4
a, d	1, 2, 2, 2, 1	5	2	a, d	2, 1, 1, 1, 4	5	4
a, e	3, 2, 2, 1, 1	5	3	a, e	3, 1, 1, 4	3	4
b, c	1, 5, 2, 1	4	5	b, c	1, 3, 3, 2	4	3
b, d	1, 3, 2, 2	4	3	b, d	1, 3, 3, 1, 1	4	3
b, e	4, 4, 1	3	4	b, e	1, 3, 4, 1	3	4
c, d	1, 2, 2, 1, 2, 1	5	2	c, d	1, 1, 2, 1, 1, 1, 2	6	2
c, e	3, 2, 3, 1	4	3	c, e	1, 3, 1, 4	3	4
d, e	3, 1, 1, 2, 1, 1	5	3	d, e	1, 2, 1, 1, 3, 1	5	3

TABLE 8
PRF Header Table with Length-2 Global Regularities and Global Supports

5 Experiment Results

Items	Reg_0	Reg_1	$Max(R_0,R_1)$	Sup_0	Sup_1	Sup_1+Sup_2
~~a, b~~	~~6~~	~~5~~	~~6~~	~~4~~	~~2~~	~~6~~
a, c	3	4	4	5	4	9
a, d	2	4	4	5	5	10
a, e	3	4	4	5	3	8
~~b, c~~	~~5~~	~~3~~	~~5~~	4	4	8
b, d	3	3	3	4	4	8
~~b, e~~	4	4	4	~~3~~	~~3~~	~~6~~
c, d	2	2	2	5	6	11
~~c, e~~	3	4	4	~~4~~	~~3~~	~~7~~
d, e	3	3	3	5	5	10

In this section we are going to produce our results. Since

there is no existing algorithm to discover *regular-frequent* patterns, we only examine PRF-method performance which includes converting database into vertical format and local table and Header table and corresponding parallel mining process. We used our PRF-method over synthetic (*T1014D100K*) and real (*Kosarak*) datasets which are frequently used for frequent pattern mining experiments which are developed at IBM Almaden Quest research group and obtain from the website http://cvs.buu.ac.th/mining/Datasets/~synthesis_data/ and UCI Machine Learning Repository (University of California – Irvine, CA) respectively. We consider all local sites with identical configuration which consists of 2.66 GHz CPU with 2GB main memory running on Windows XP. All programs are written in NetBeans IDE 7.0. Message passing interface is established for communication among all the sites. We distributed

Fig. 1. Execution time over different δ and λ on T10I4D100K

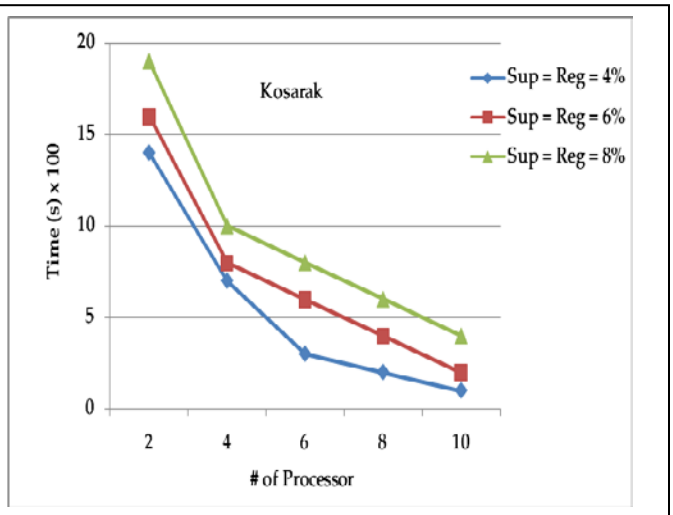

Fig. 2. Execution time over different δ and λ on Kosorak database

the partitioned database among the sites and the processor in the corresponding site will have complete access to its portion of the database.We report the results on *T1014D100K* synthetic dataset which contains 100K transactions, 870 items and 10.10 as the average transaction length. The above Figure 1 shows the execution time on different *Sup*() and *Reg*() values on increasing number of processors. We also report on the *Kosarak* real dataset that contains 990K transactions, 41,270 items and 8.10 as the average transaction length. Figure 2 shows the execution time on various *Sup*() and *Reg*() values.

6 Conclusion

Parallel computing is a necessary component in any large-scale data mining application. In large databases the performance of the parallel algorithms completely based on I/O cost and inter-process communication. In this paper we have introduced a new mining method called PRF-method in parallel and distributive environment to obtain the complete set of *regular-frequent* patterns reducing I/O cost and inter-process communication. It requires only one database scan to convert the horizontal database into vertical data format, so I/O cost reduced and also no inter-process communication takes place at the mining process between the processors. This shows the efficiency of our proposed method.

References

[1] Agrawal R., et al.: Mining association rules between sets of items in large databases. In ACM SIGMOD, pages. 207-216, 1993.
[2] Agrawal R and Srikanth R.: Fast algorithms for mining association rules. In VLDB, 489-499, 1994.
[3] Han J., et al. Mining frequent patterns without candidate generation. In ACM SIGMOD, 1-12, 2000.
[4] Tanbeer S K., et al. Mining regular patterns in transactional database. In IEICE Trans., 2568-2577, 2008.
[5] Tanbeer S K., et al. Discovering periodic-frequent patterns in transactional databases. In PAKDD 2009, pages. 242-253, 2009.
[6] Orlando S., et al. An efficient parallel and distributed algorithm for counting frequent sets. In VECPAR, 421-435, 2003.
[7] Savasere A., et al. An efficient algorithm for mining association rules in large databases. In VLDB 432-444, 1995.
[8] Hu J. and Yang-Li X. A fast parallel association rules mining algorithm based on FP-Forest. In 5th international symposium on Neural Networks, pages 40-49, 2008.
[9] Yu K.-M., Et al. Load balancing approach parallel algorithm for frequent pattern mining. In PaCT, 623-631, 2007.
[10] Tanbeer S K., et al. Parallel and distributed algorithms for frequent pattern mining in large databases. In IETE technical review, vol.26, pages 55-66, 2009.
[11] Yi-ming G., and Zhi-jun W. A vertical format algorithm for mining frequent itemsets. In IEEE transactions, pages 11-13, 2010.
[12] Zaki M. J. Parallel and distributed association mining: A survey. In IEEE concurrency, 14-25, 1999.