Distributed Data Mining by associated rules: Improvement of the Count Distribution algorithm

Hadj-Tayeb karima[1], Hadj-Tayeb Lilia[2]

[1] English Department, Es-Senia University, 31000\Oran-Algeria

[2] Computer science Department, Es-Senia University, 31000\Oran-Algeria

Abstract

Today, most large systems are overwhelmed by a flood of data that is stored daily in databases distributed. It is in this context that the distributed data mining is used by offering many parallel and distributed algorithms to extract crucial information.

Among the most popular techniques, we are interested in our work at association rules technique by focusing on the distributed approach "the Count Distribution (CD)". We aim for our contributions to improve this algorithm by reducing the number of exchanged messages, and the number of generated candidates.

Our algorithm is based on the sequential algorithm AClose of the closed frequents itemsets approch. The experimental results showed that the proposed algorithm meets the expected objectives by presenting a performance gain greater than the CD algorithm in which the last points are important performance factors in determining the quality of an algorithm for extraction rules.

Keywords: *Association rules, distributed system, the distribution count approach, closed itemsets.*

1. Introduction

With the explosion of distributed data, the evolution of data mining applications is critical. It is in this context, that the distributed data mining has emerged by offering multitude parallel and distributed technique. These techniques are too fast to extracting the implicit and useful knowledge hidden in these databases.

Among these most popular techniques, we are interested in association rules technique. It is used to generate a set of association rules that can discover meaningful relationships between attributes.

The algorithms based on this technique, research frequent itemsets among all the generated candidates. This generation is an important factor in the extraction process to measure the effectiveness of retrieval algorithms.

To achieve this, we are particularly interested in one of these distributed algorithms namely the algorithm (Count Distribution). This algorithm suffers mainly from the

exorbitant generation of candidates and the high cost of communication by their transmission.

In this context, we propose a new distributed approach to improve the performance of this algorithm.

2. Basics [4] [5]

To better understand the basic concepts related to the process of extraction association rules, we present the following few concepts:

- Item (pattern): corresponds to a value of an attribute in the database. All items of size k is a k-Itemset.
- Support noted Supp (X): the proportion of transactions of D containing X.
- The Set of frequent itemsets is the set of items whose support is above the minimum threshold Minsupp.

$$F = \{l \subseteq I \,/\, l \neq \varnothing \wedge \sup(l) \geq Min\sup\} \quad (1)$$

- The set of closed frequent itemsets FF is defined as follows (2):

$$FF = \{l \subseteq I \,/\, l \neq \varnothing \wedge \gamma(l) = l \wedge \sup(l) \geq Min\sup\}$$

Where γ is a closure operator of the Galois connection and the minimum threshold Minsupp.

- The set of maximal frequent itemsets FM is defined as follows

$$FM = \{l \in F \,/\, \forall l' \supseteq l, \sup(l') < Min\sup\} \quad (3)$$

3. Different approaches for extraction frequent itemsets [5][7][9]

The first approach is the approach of extraction of frequent itemsets. During each iteration, a set of candidate patterns is created. The supports of those patterns are calculated and the patterns no frequents are deleted. The reference algorithm is the Apriori algorithm.
This approach suffers from the generation of a large number of candidates, especially for contexts highly correlated. It can produce 2^k-2 rules for k-itemsets frequent. This approach provides a set of rules difficult to use by the expert.
To overcome the disadvantages of this approach to extract a condensed set of frequent itemsets, *the approach of extraction closed frequent itemsets* and *the approach of extraction maximal frequent itemsets* have emerged.
The maximal frequent itemsets are frequent itemsets on which all supersets are not frequents. This approach generates a small subset of association rules, but has the distinction of being incomplete (loss of information). The algorithm of reference based on this approach is the Max-Miner algorithm.
Hand against the closed algorithms from the theory of formal concepts, propose to generate a compact and generic subset of association rules. This subset has the advantage of being comprehensive in terms of knowledge (no loss), while being reduced in terms of size.

4. Parallel algorithms for mining association rules [2][6][8]

There are two different approaches in parallel association rules which are based on these algorithms: data parallelism or task parallelism.
Both approaches require that the database is divided horizontally on all nodes. In the first approach, each node performs the same calculation independently on its local base and the local support is exchanged with other nodes.
As against the second approach is to divide the candidates into disjoint groups and assign them to different nodes. Each node trait independently the assigned candidates and accede to local and no local transactions. We present in the following, the most popular algorithms:

4.1 Parallel algorithms derived from the Apriori algorithm

Introduced in 94, this algorithm is the first algorithm for extraction rules based on the approach of frequent itemsets.
This algorithm has been parallelized in 96, proposing the Count Distribution algorithm (CD) and the Data Distribution algorithm (DD). The CD requires communication between nodes to generate the global support. It is based on the parallelism of the data. We will present in detail this algorithm in **Section V** and the proposed improvements in **section VI**. Otherwise, the DD algorithm is based on the parallelism of the task. It requires the communication between nodes for the exchange of local data. To generate the global support, each node must scan the entire database.

4.2 FDM Algorithm (Fast Distributed Mining)

This algorithm minimizes the number of candidates generated using two pruning techniques: local and global pruning.
The local pruning is to remove the element X of all candidates if X is not locally frequent. Once that's done each site sum received local support to generate the globally frequent itemsets, this technique is called global pruning.
FDM is to reduce the number of messages sent between sites

4.3 *GridDMM Algorithm (Grid-Based Distributed Max-Miner)*

The maximal algorithms explore simultaneously the trellis of itemsets down to up (research level) and up to down (in order to identify the maximal large sizes)
*GridDMM is b*ased on the Max-Miner algorithm. This algorithm has been proposed for the extraction of maximal frequent itemsets in a distributed database on a data grid. GridDMM reduces the cost of communication through the communication to n-cube (n: number of nodes). These nodes can exchange and merge the information of local computing through more dimensional links between them. This algorithm generates reduced set of candidates based on the pruning of subsets and supersets.

5. Count Distribution Algorithm (CD) [1][3][8]

This algorithm is the parallel version of the sequential algorithm Apriori. This algorithm partitions and distributes horizontally and equitably the database on all processors. This algorithm is detailed as follows.

5.1 Principle

At each step, each processor P_i generates independently support candidates by accessing its local database D_i. It transmits the following the local support of the candidates by a broadcast to other processors that calculate the global support of itemset. Once all global frequent itemsets L(k) has been determined, each processor constructs all candidates C (k +1) in parallel with the next step. This process is repeated until all frequent itemsets are found.

5.2 Pseudo-code of the CD algorithm

Initially, all candidates of size 1 (C1) is the set of items.

Input: Database D, N sites, and minimal support minsup
Output: all frequent itemsets F

1. The database D is partitioned horizontally into N partitions.
2. The N data partitions are distributed over the N sites.
3. For i = 1, each site j in parallel:
   {
   - Scans its database DJ to calculate the local support from all candidates of size 1;
   - Diffuse the calculation to all other sites;
   - Determine the frequent itemsets of size 1, L1;
   }
4. Repeat
   {
   - i = i + 1;
   - Ci = Apriori-Gen (Li-1), (for i> = 2)
   - Each site calculates the support of Ci by accessing at its local database
   - Each site broadcasts the local calculation at all other sites;
   - Each site determines the frequent itemsets of size i, Li;
   }
   Until all frequent itemsets (F) are found;

5.3 Performance

This algorithm presents the following limits:

- It generates a very important set of candidates at each step.
- Each processor generates the same set of global frequent itemsets at each step; this fact, the replication of the calculation degrades performance.
- It has a simple system of communication messages, but it suffers from high communication cost due to the broadcast of generated candidates.

6. Distributed approach proposed for the extraction of closed frequent itemsets [1] [9]

To overcome the disadvantages of the CD algorithm, we propose some modifications to improve the performance of this algorithm.

6.1 Pseudo-code of the proposed algorithm

Initially, the master processor has its level of all items which are considered as candidate itemsets of size 1 (C1). This set will be distributed to all processors. He also has at its global database.

Input: Database D, the number of sites N, and minimal support minsup
Output: all frequent closed itemsets FF

1. The master processor partitions the database into N horizontal partitions.
2. The master processor distributes the data partitions on N sites
3. Repeat
   {
   - The master processor generates and distributes candidates of size i (Ci) at each site j, where: Ci = Apriori-Gen (Li-1), (for i> = 2)
   - In parallel, each site j:
     {
     - scans its database to calculate the local DJ support of all local candidates of size i;
     - Diffuse local support to master processor;
     }
   - The master processor collects the local supports for all candidates of each site j and generates the global support of each candidate.
   - For each candidate x in C (k)
     {
     If the support of x is less than the minsup and x is also frequent that one of its k-subsets of size (i-1) then x is not frequent; else x is a frequent item;
     }
   - i= i + 1;
   }
   Until all the frequent itemsets F are found;

4. For each F belonging to the set of frequent itemsets, the master processor computes its closure and generates all the frequent closed itemsets FF.

6.2 Performance of proposed algorithm.

We present the following suggestions:

- Reduce the set of candidates generated

The CD algorithm is an algorithm based on the approach of extraction of frequent itemsets; thus, it suffers from the exorbitant generation of candidates. Therefore, we propose to take advantage of sequential algorithms based on the 'closed frequent itemsets'. These algorithms generate a set of closed patterns which is a subset of all frequent patterns. They are based on new closure heuristics to prune all candidates.
After a thorough study of these algorithms closed, we choose algorithm AClose. This algorithm eliminates the repetitive calculation of the closure of frequent itemsets in each iteration, which may increase the cost of transfer.
It includes a new heuristic for the pruning phase of all candidates is to eliminate a candidate to step (k +1), if it is also one of its frequent k-subset to step (k). This criterion reduces the number of candidates generated at each step.

- Eliminate the duplicated calculation by CD algorithm

The CD algorithm requires that each processor performs the same calculation on all candidates to produce ultimately the same k-frequent itemsets, this degrades performance. To resolve this problem, we propose to designate a master processor (coordinator), which relieves the others processors who made the duplicated calculation. This processor will be solely responsible for generating the candidates of the next step and circulate it to all processors.

- Reduce the cost of communication between processors

The CD algorithm requires a broadcast support local of candidates for all nodes. It then requires a high communication cost which is equal to (| Ck | × (N-1) × N), where | Ck | is the number of candidates in step k and N is the number of nodes. To reduce this cost, we propose a new communication system in which the nodes exchange information only with the coordinator.

- Reduce the number of steps and extraction time

The closed frequents are the feature to run in a number of steps lower than frequents algorithms. In addition, reducing the previous two points, we minimize the local calculations, and thus, it reduces the number of steps and time required for implementation

7. Evaluation experiment

Our aim is to evaluate the performance of two algorithms (CD and our algorithm) for a comparative study. To implement the two distributed algorithms, we have a database that has partitioned and distributed horizontally and equitably across all nodes. The database used is "Hypothyroid" [7] characterized by a high density of 66% and consists of 3247 records. This database is managed under access previously treated for tests performed on the

Threshold= 0.2			
Count Distribution algorithm(CD)		**Proposed Algorithm**	
Number of candidats	Number of frequent itemsets	Number of candidats	Number of closed frequent itemsets
21623	20623	13600	13367

Data Mining. We implemented our own simulation of distributed systems using the Java language.
The experiments were performed on a 1.6 GHZ Pentium PC, with a 512 MB RAM and Windows XP.

In the first experiment, after several changes of the threshold, we fixed this threshold to 0.2, and we vary the numbers of processors in steps of 2.

- **Reduce the number of candidates and the number of frequent itemsets using the frequent closed**

The results are represented graphically:

Threshold= **0.2**				
Number of nodes	**distributed Algorithm**	**Number of steps**	**Time of execution *10^3 (Ms)**	**Cost of communication**
3	CD	14	57.663	129738
	Proposed	13	51.954	81600
5	CD	14	61.673	432460
	Proposed	13	51.519	136000
7	CD	14	65.303	908166
	Proposed	13	51.779	190400
9	CD	14	69.649	1556856
	Proposed	13	51.854	244800
11	CD	14	73.661	2378530
	Proposed	13	51.609	299200
13	CD	14	77.827	3373188
	Proposed	13	51.834	353600
15	CD	14	81.978	4540830
	Proposed	13	52.250	408000

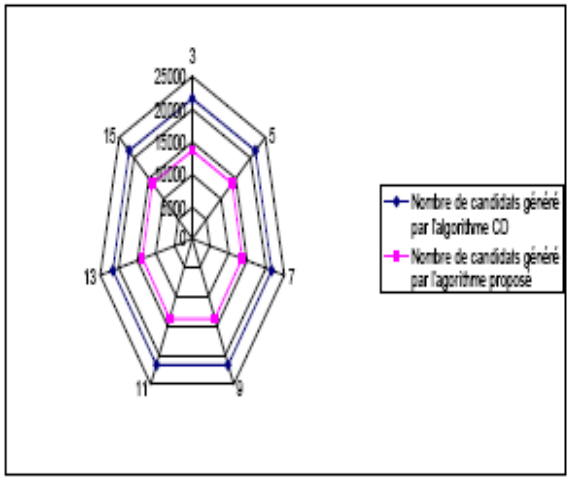


Fig. 1 Reduce the number of generated candidates

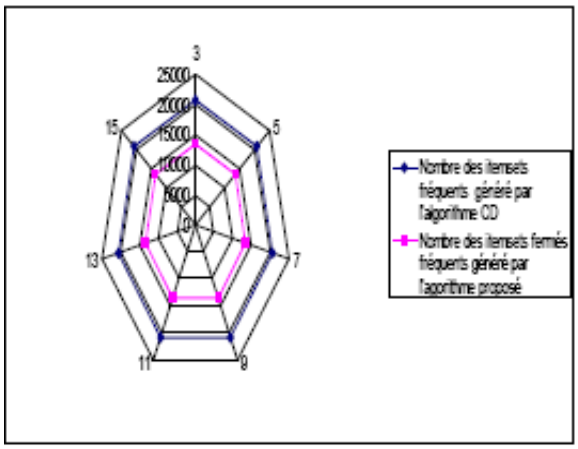


Fig. 2 Reduce the number of frequent itemsets

- **Reduce the number of steps, the cost of communication and response time**

The results are represented graphically

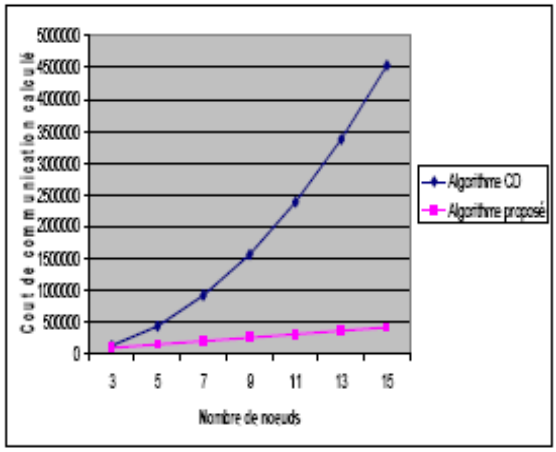


Fig.3 Reduce the cost of communication

Number of nodes = 5				
	CD Algorithm		Proposed Algorithm	
Threshold	Number of candidats	Number of frequents itemsets (F)	Number of candidats	Number of closed frequents itemsets (FF)
0.1	32767	32767	16386	16383
0.2	21623	20703	13600	13367
0.3	12852	12715	10523	10453
0.4	9627	9355	8350	8169
0.5	5683	5596	5683	5596

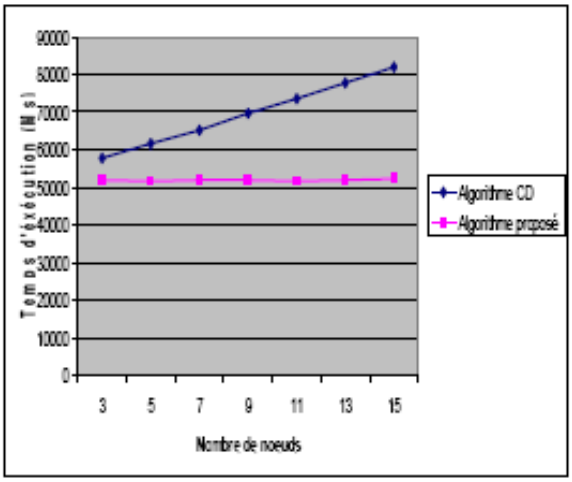


Fig. 4 Reduce the execution time

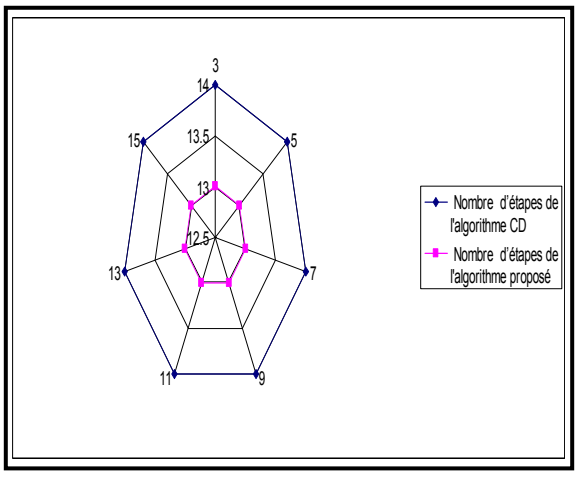


Fig. 5 Reduce the number of steps

In the second experiment, we fixed the number of processors to 5, and we vary the threshold to 2.

- **Reduce the number of candidates and the number of frequent itemsets using the frequent closed itemsets**

The results are represented graphically

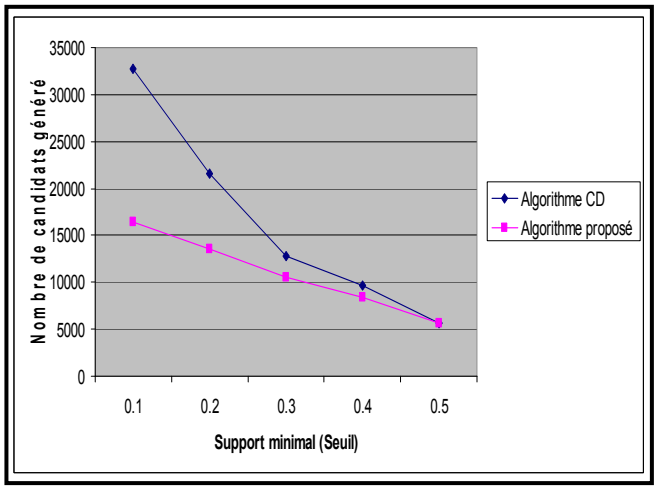


Fig.6 Reduce the number of generated candidates

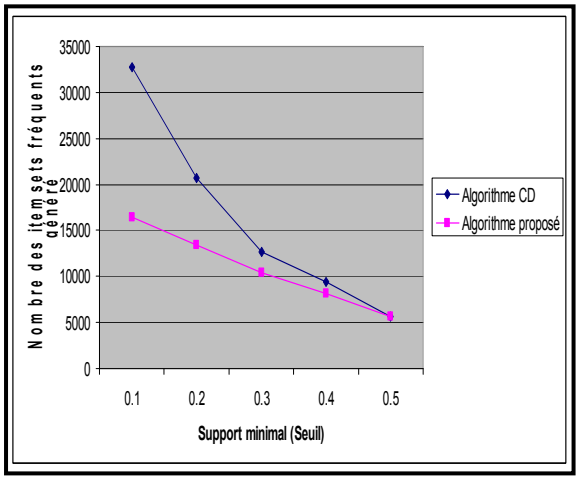


Fig. 7 Reduce the number of frequent itemsets

- **Reduce the number of candidates and the number of frequent itemsets using the frequent closed itemsets**

The results are represented graphically

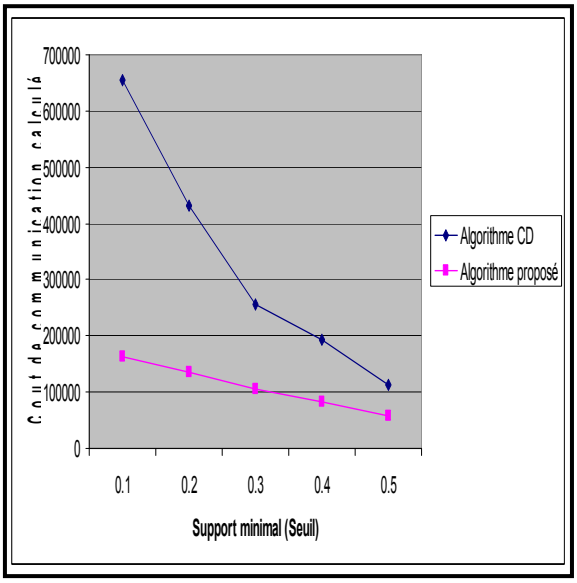


Fig. 8 Reduce the cost of communication

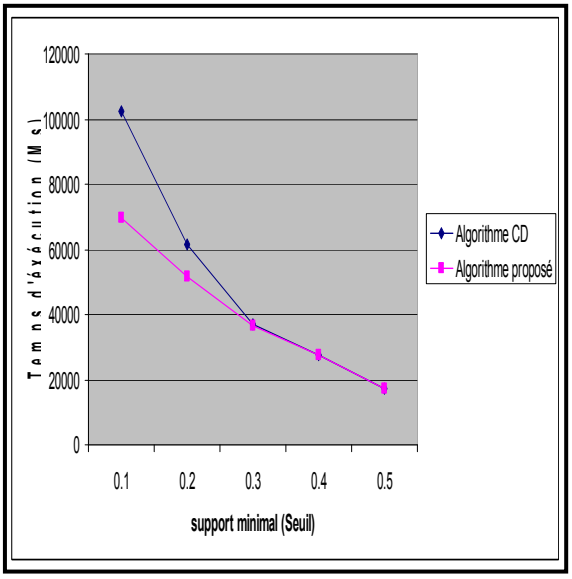
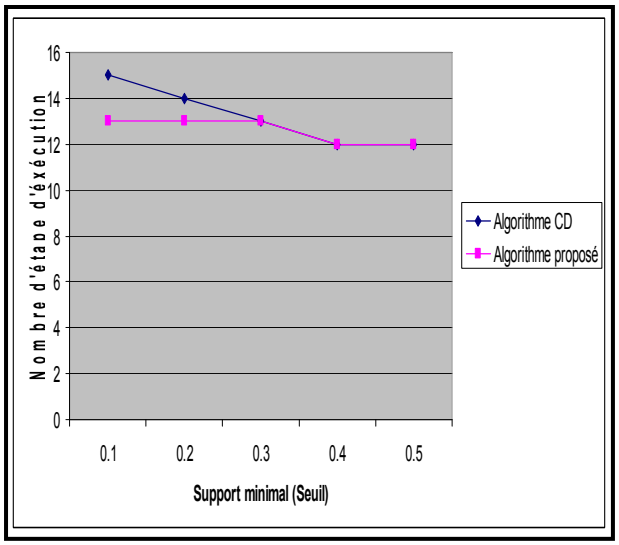


Fig. 9 Reduce the execution time



Number of nodes = 5				
Threshold	Distributed Algorithm	Number of steps	Time of exécution *10 ^{3} (Ms)	Cost of communication
0 .1	CD	15	102.557	655340
	Proposed	13	69.605	163860
0.2	CD	14	61.673	432460
	Proposed	13	51.519	136000
0.3	CD	13	36.883	257040
	Proposed	13	36.758	105230
0.4	CD	12	27.600	192540
	Proposed	12	27.455	83500
0.5	CD	12	17.214	113660
	Proposed	12	17.196	56830

Fig. 10 Reduce the number of steps

According to the results presented, the proposed algorithm meets the expected goals: reduce the number of candidates generated, reduce the cost of communication, and reduce the execution time.

Indeed, the generation of a minimum set of candidates based on the notion of closure, and the new system of communication which is based on our algorithm reduces the cost of communication.
We notice through the final results, that our algorithm runs in time smaller than the CD algorithm.

8. Conclusions

We presented the implementation of two distributed algorithms on a simulator of distributed system that we developed with the Java language, and in which we varied the simulation parameters for a variety of tests.

The aim of our work is to improve the performance of the distributed algorithm Count Distribution by proposing a new algorithm scalable and closed in a distributed environment. In other words, this algorithm tolerates the increased number of nodes and the size of the database.

Our algorithm is based on the sequential algorithm Aclose of the approach for extracting of frequent closed itemsets. This approach has the advantage of reducing the number of generated candidates during the extraction process based on the notion of closure.
To reduce of the cost of communication is an important factor for measuring the effectiveness of an algorithm, we proposed a new communication system in which all processors communicate only with the master processor (coordinator).

The study showed that our algorithm has a performance gain higher than the CD algorithm in which the cost of communication, the execution time and number of patterns generated are the important factors of performance in determining the quality of an algorithm for extracting rules.

It was noted that the approach to extracting frequent closed itemsets has the characteristic to generate a number of non-redundant and more compact association rules than the set of generated rules by the algorithms frequently. Therefore, for the extraction of distributed association rules, the proposed closed algorithm improves the relevance of the result of extraction of association rules by offering to the user a small set of rules covering all items of context,interesting and easy to handle.

Our algorithm is based on static distribution load and an equitable distribution of the database. We offer as a perspective to our work, our approach to implement a platform-type grid. This architecture will allow us to distribute the load and the database between resources depending on the speed and storage capacity of the nodes.

We propose to develop an algorithm for extracting frequent closed itemsets based on the parallelism of the task. We can then perform a comparative study of the performance of two algorithms based on two approaches to parallelism. This study will aim to identify one with a higher performance gain in terms of communication cost and execution time minimal.

References

[1] Cheung.D & AL. “Efficient Mining of Association Rules in Distributed Databases”. IEEE Transactions on Knowledge And Data Engineering, 911-922, 1996.
[2] Cheung.D & AL. **“**A Fast Distributed algorithm for Mining Association Rules”, In *Proceeding of 1996 International conference on Parallel and Distributed Information System* (PDIS’96), pages.31-44, Miami, FL,1996.
[3] L.Congnan and A. Pereira and C. Soon, “Distributed Mining of Maximal Frequent Itemsets on a Data Grid System”, *The Journal of Supercomputing,* 37. pp. 71- -90, 2006.
[4] N. Pasquier, “Data Mining: Algorithmes d'Extraction et de Réduction des Règles d'Association dans les Bases de Données,” Univ Clermont-Ferrand II, École Doctorale Sciences pour l'ingénieur de Clermont-Ferrand, 2000.
[5] O. Couturier, “Contribution à la fouille de données : règles d'association et interactivité au sein d'un processus d'extraction de connaissances dans les données, ” thése de doctorat, Dept d’Informatique., Univ Artois, 2005.
[6] S. Ben Yahia and M. Engelbert, “Approche d’extraction de règles d’associations basées sur la correspondance de Galois, Motifs dans les bases de données”, pp. 23- -55. RSTI, ISI 9, 2004.
[7] Sadok Ben Yahia & Engelbert Mephu Nguifo, « Approche d’extraction de règles d’associations basées sur la correspondance de Galois », RSTI - ISI – 9/2004. Motifs dans les bases de données, pages 23 à 55.
[8] Z. Mohammed, “Parallel and Distributed, Association Mining: A Survey”, *IEEE* 1999.
[9] www.cs.hku.hk/~dcheung/publication/pdis96.ps
[10] http://www.cs.kuleuven.be/~dtai/CP4IM/datasets/

Karima Hadj-Tayeb received the Magister degree in computer science from the University of Sciences and Technology of Oran (Algeria) in 2009. Currently, she is an assistant teaching in English department in the Es-Senia University of Oran and she is a doctorate candidate. Her research focuses on data mining and the distributed system.

Lilia Hadj-Tayeb received the Magister degree in computer science from the University of Es-Senia of Oran (Algeria) in 2008. Currently, she is an assistant teaching in the department of computer science in the Es-Senia University of Oran and she is a doctorate candidate. Her research focuses on data mining and the Grid.