Hadoop based Mining of Distributed Association Rules from Big Data

Marwa Bouraoui
Signal, Image and Technology of Information Laboratory, National Engineering School of Tunis, Tunis El Manar University
Tunis, Tunisia
bourawimarwa@gmail.com

Ines Bouzouita
Computer Science Department, Faculty of Sciences of Tunis, Tunis El Manar University
Tunis, Tunisia
ines.bouzouita@yahoo.fr

Amel Grissa Touzi
Computer Science Department, Faculty of Sciences of Tunis, Tunis El Manar University
Tunis, Tunisia
grissa.touzi@topnet.tn

***Abstract*—Data analysis techniques need to be improved to allow the processing of data. One of the most commonly used techniques is the Association Rule Mining (ARM). These rules are used to detect facts that often occur together within a dataset. Though several methods have been suggested for the extraction of association rules, problems arise when data becomes large. To overcome such issue, we propose, in this paper, an efficient approach for ARM based on MapReduce framework, adapted for processing large volumes of data. Furthermore, because real-life databases lead to huge number of rules including many redundant rules, our algorithm propose to mine a compact set of rules with no loss of information. The results of experiments tested on large real world datasets highlight the relevance of mined data.**

Keywords—Association rules mining; closed itemsets; map reduce; hadoop; big data

I. INTRODUCTION

The recent years are characterized by hardware and software technologies revolution. This evolution offer to the users the opportunity to generate a huge volume of transactional data. This data is referred to Big Data ; it increases exponentially over years and requires extra space memory and computation. 3Vs (volume, variety and velocity) are three defining properties or dimensions of big data. Volume refers to the amount of data, variety refers to the number of types of data and velocity refers to the speed of data processing. To interpret such data, statistical analysis application needs more efficient and appropriate technologies to handle the large volume of data. To cope with these exigencies, data mining techniques were introduced. Data mining has been thoroughly studied in the recent years, particularly the association rules mining filed. ARM has become one of the core data mining tasks with many real world applications such as selective marketing, fraud detection in web, economic census. It aims to discover associations among transactions encoded in a database. An association rule is a probabilistic rule which implies certain association relationship among a set of objects in the form of "if-then" statement. These relationships help for decision-making.

Association rules mining has been investigated extensively in centralized datasets. However, when the data is distributed and grows large, the efficiency decreases since the communication cost and the memory usage become improper. Researchers are focusing on accelerate the processing of the ARM algorithms by parallelizing and executing them on multiple clusters. This solution boosts performance but introduces other drawbacks such that data partitioning, optimizing communication cost, workload balancing, collecting global information from the local one and errors provoked by failure that occurred in some nodes. Furthermore, since a crushing number of redundant rules can be generated from datasets with a reasonable size, relevance and utility of obtained rules is becoming a priority when dealing with optimizing ARM algorithms.

To conquer the above drawbacks, we introduce a distributed algorithm for mining generic basis of ARM based on the cloud computing framework MapReduce. Indeed, generic basis offer a compact set of informative and non-redundant rules, with the guarantee of no loss of information. These basis are generated from closed frequent itemsets . The MapReduce framework presents a parallel programming model which offer a distributed computing, required for mining large scale data. To evaluate the performance and efficiency of the proposed approach, we performed experiments over big real datasets. The obtained results highlight our approach is very valuable when dealing with ARM from large datasets.

This paper is formulated as follows. In section 2, we cite preliminary knowledges. In the section 3, we present related works. We detail our approach in section 4. In section 5, we cite results and analyses of our experiments. In section 6, we conclude our contribution.

II. BASIC CONCEPTS

A. Association Rules Mining

Suggested by Agrawal et al. [1], mining association rules have described the formal model of this problem as follows.

Let's consider a finite set of items I presented as I={i1,i2,..in}. We call k-itemset; the subset K={i1,i2,..ik} ⊆ I. Let D = {T1, T2, T3 ... Tn} be a finite set of data transactions.

978-1-5386-1084-8/17/$31.00 ©2017 IEEE

A transaction Ti consists of a set of items where Ti ⊂ I. For I1, a subset of I, the support of an itemset X1 is the percentage of transactions in the D database that contain X1.

An association rule is a conditional implication modelized as r: X1 -> X2 between two itemsets X1, X2 ⊂ I where X1 ∩ X2 = Ø. In order to generate only the significant relationships between the itemsets, frequency measure, the support, denoted by Support (X1 -> X2) and precision measure, the confidence, denoted by confidence (X1 -> X2) are associated to each rule. The r rule support corresponds to the support of the union of its associated itemsets, and defined by:

$$Support (X1 \rightarrow X2) = support (X1 \cup X2) \quad (1)$$

The confidence of the rule r stipulates how frequently Y appears in transactions that contain X. It is defined by:

$$Confidence (X1 \rightarrow X2) = support (X1 \cup X2) / support (X1) \quad (2)$$

The generated rules are those whose support >= minSupport and confidence >= minConfidence, where minSupport and minConfidence are minimum thresholds defined by users based on their objectives and the type of processed data. The basic algorithms for mining association rules are Apriori [2], FP-Growth [3], Eclat and Clique [4].

B. Closed itemset

Definition 1. An extraction context. Is presented as a triplet E = (Ob, It, Rb). Ob denotes a finite set of object, It denotes a finite set of itemset and Rb denotes a binary relation (Rb ⊆ Ob × It). For a definite pair (ob, it) ∈ R signify that the object ob ∈ Ob contain the item it ∈It.

The closure operator (“) describe the closure operator φ ◦ ψ s.t. (φ, ψ) represents the pair of operators modelized by ψ : P(It) → P(Ob) s.t. ψ(It) = {ob ∈ Ob| ∀ it ∈ It, (ob, it) ∈ R} and φ : P(Ob) → P(It) s.t. φ(Ob) = {it ∈ It| ∀ ob ∈ Ob, (ob, it) ∈ R}. An equivalence classes [5] is a collection of disjoint subsets partitioned from items joined by equivalence relation according to the closure operator.

Definition 2. Closed itemset. Is the largest item in each equivalence classes and characterized by It”=It. Supp(It) is the sum of the number of object in E that contain It.

C. Hadoop Map-Reduce

MapReduce [6] is a parallel programming paradigm for handling data with a very large volume. Programs adopting this model are automatically parallelized and run on computer clusters. A typical MapReduce application processes several terabytes of data and operates several thousand machines. It is usually divided into two parts as shown:

- The mission of the mapper is to read and process data stored on disk.
- The mission of the reducer is to unify the results from the mapper and write them on disk.

Hadoop [7] is an open source framework based on Java, implementing the Map-Reduce paradigm. It allows the storage and treatment of excessively large volume of data scattered over distributed computing environments. Its distributed file system eases fast data transfer rates between nodes and manage the treatment to guarantee a continues processing even if a failure occurs in a node. across nodes prepared for parallel processing.

III. RELATED WORKS & MOTIVATION

The Apriori algorithm [2] is one of the most famous association rules mining algorithms. But when it comes to mine voluminous data, it fails to prove scalability and effectiveness. Now days many algorithms have been proposed on parallel and distributed platforms [8,9] to overcome these limitations. Unfortunately, these algorithms have major architectural weaknesses on communication and synchronization levels [10]. Moreover they add some overheads like data partitioning and task balancing.

To deal with these problems, MapReduce has emerged as the de facto solution for processing massive data on large clusters. Key advantages of MapReduce programming model are the high-throughput data processing and the fault-tolerant storage.

In addition, with the proposition of closed itemset [11] with many related algorithms [10], memory is no longer exhausted with frequent but not interesting and redundant itemsets. In fact, such methods guarantee great performance whenever the dataset and threshold. Yet, research on parallel environment for the big data context are not well expanded. In [12], based on FP-Growth algorithm Pfp [13], authors propose a method that divides a process into independent parallel process and have proved that it offers a better speedup. Yet, this approach was tested on small dataset and don’t prove its scalability. In [14], authors present a parallel algorithm for closed itemsets itemset mining (CFI) called CloPN. It consists on convert data into numerical representation based on prime number. From the new representation, it uses only division and multiplication operations to mine CFI. Thus, the running time is shorter but generated rules are still heavy and hard to employ.

Obviously, existing approaches are focusing on mining rules from big distributed data, drawing attention to the execution time, without focusing on the relevance and usefulness of the generated rules. As contribution, we introduce a method for extracting a compact non-redundant set and with no loss of information, from such voluminous data.

IV. NEW APPROACH

In this section, we detail the proposed method for distributed mining of association rules from big data.

A. General Principle

We introduce a distributed approach for mining generic basis of association rules based on the cloud computing framework MapReduce. Indeed, generic basis offer a compact set of informative and non-redundant rules, with the guarantee of no loss of information.

Our algorithm is iterative and takes place in three Map-Reduce jobs as shown in Algorithm 1 & Fig. 1. The first one consists on mining the 1-frequent itemsets. Then, an iterative process occurs to mine frequent candidates (generators). The last phase is responsible of the closure computation. Finally

we generate basis from the obtained closed itemsets and their frequent generators.

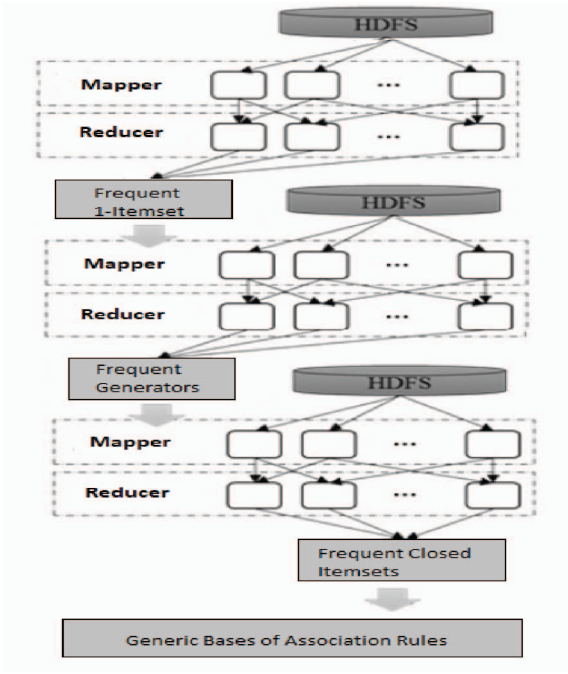

Fig. 1. General Principle

Algorithm 1: General process

```
BEGIN
Input: Dataset, min threshold, min Confidence
F1 <- Frequent1-MapReduceJob
while(frequentK.size()>1){
k++;
Ck <- generatekCandidate(k)
Fk <- FrequentK-generator-MapReduceJob
}
ClosedItemset <-ClosureCalculate-MapReduceJob
Rule basis <- RuleBasisGeneration()
Output: Rule basis
END
```

B. Frequent Generators Mining

For mining frequent generators, we have to mine initially the frequent 1-itemsets from all nodes as described in Algorithm 2 and Algorithm 3.

Since each mapper is charged to an HDFS block, the input information defined by the key value couple is (key=null, value=oi). The mapper tokenize each object oi into item (token) to constitute a key value couple with all the items in the input data file (the SPLIT() procedure). The keys present the items from the oi and value is '1'. In the reduce phase, all the keys are collected together and the values with the same keys are summed to find the occurrences for a definite item. In the reducer phase, the output of the mapper are taken as input and then the reducer reduces the key-value pairs to unique keys with values added up. As our goal is to mine only the frequent items, we exclude the pairs whose value lower than min threshold.

Algorithm 3: Mapper

```
Procedure Mapper(key, Value = oi)
oi[] <- SPLIT(oi)
For all oi[j] in oi[] do
Call OUTPUT(Key = oi[j], Value = 1)
End For
End Procedure
```

Algorithm 3: Reducer

```
Procedure Reducer(key = oi[j],  Value = List(oi[j]k))
sum=0
For all L in List(oi[j]k) do
sum = sum + L
End for
If (sum >= minSupport)
Then
 Call OUTPUT (Key = oi[j], Value = sum)
FrequentK <- SaveFrequentK()
End Procedure
```

Once getting the frequent 1-itemsets, the next phase is to mine the frequent generators. We operate an iterative process composed by two steps: generate candidates and mine the frequent one. For a definite iteration k, the candidates are generated by applying the famous Apriori-Gen algorithm to obtain the k-candidate Ck from the frequent itemset Fk-1. Then, we apply a mapreduce job for the mining of the frequent one. This process is described in Algorithm 4 and Algorithm 5.

The mapreduce process for mining the frequent candidates is similar the frequent 1-itemset, but here the items are the generated candidates according to the current iteration.

First of all, candidates list is loaded from the HDFS. The mapper tokenize each object oi into item (token) to form a key value pair with all the items in the input data file (the SPLIT() procedure). The goal here is to check if the current transaction oi holds the current candidate. If it is the case, the key is the candidate and the value is '1'. In this phase, the reducer collect the keys together and the values with the same keys are summed to find the occurrences for a definite candidate. In the reducer phase, the output of the mapper are taken as input and then the reducer reduces the key-value pairs to unique keys with values added up. As our goal is to mine only the frequent candidates, we exclude the pairs whose value lower than min threshold. The obtained frequent candidates are saved in the HDFS to be accessible for all nodes.

Algorithm 4: Mapper

```
Procedure Mapper(key, Value = oi)
candList <- LoadKCandidates()
oi[] <- SPLIT(oi)
For ( c in candList) do
     For all oi[j] in oi[] do
     If c ∈ oi[j]
    Then Call OUTPUT (Key = c, Value = 1)
    End For
End For
End Procedure
```

Algorithm 5: Reducer

```
Procedure Reducer(key = c,  Value = List(ci))
sum=0
For all L in List(ci) do
sum = sum + L
End for
If (sum >= minSupport)
Then Call OUTPUT (Key = c, Value = sum)
FrequentK <- SaveFrequentK()
End Procedure
```

C. *Generic basis generation*

In this phase, we generate generic basis of association rules. These basis are generated from the frequent closed itemsets. This process is described in Algorithm 6 and Algorithm 7. In the mapper, the frequent generators (itemsets) list and the closed itemsets (initially empty) list are loaded from the HDFS. Here, only frequent itemsets that present a subset of the current object are retained. We apply then the Gen-Closure function to the current object to calculate the closure of the retained itemsets (generators). Obtained closed itemsets and their supports are saved in the HDFS to be updater in the next iteration. In the reducer phase, the output of the mapper are taken as input and then the reducer reduces the key-value pairs to unique keys with values added up. The keys are the closed itemsets and the values are their support.

Algorithm 6: Mapper

```
Procedure Mapper(key, Value = oi)
    FrequentK <- LoadFrequentK()
    G= FrequentK ⊂ oi
    For g ∈ G
        ClosedItemset <- LoadClosedItemsetMap()
        If  g ∈ ClosedItemset
        Then g.closure = g.closure ∩ oi
      Else g.closure = oi
    End For
ClosedItemset <- SaveClosedItemset()
For c in ClosedItemset
Call OUTPUT(Key = c, Value= c.support)
End For
End Procedure
```

Algorithm 7: Reducer

```
Procedure Reducer(key = c,  Value = List(c.support))
sum=0
For all L in List(c.support) do
sum = sum + L
End for
Call OUTPUT (Key = c, Value = sum)
ClosedItemset <- UpdateClosedItemsetMap(c)
End Procedure
```

Once mine the closed itemsets and their associated generators, we can generate the basis of association rules. We use the IGB basis. It is described as follows:

Definition 7. Let us denote the closed frequent itemsets by CFI, and the minimal generators of the frequent itemsets that are equal or included in a CFI fc by Gmf.

IGB = {Rule: grs ⇒ (fc1 - grs) | fc, fc1 ∈ CFI and (fc - grs) ≠ Ø and gs ∈ Gmf ∧ fc1 =< fc ∧ Confidence(Rule) ≥ minconf ∧ ∄ grs' ⊂ grs where Confidence(grs' ⇒ fc1-grs') ≥ minconf}.

The IGB basis [15] is characterized by three important criteria. The first one is its compactness since it is more dense than other basis like the pair (GBE,GBA). The second criteria is that it guarantee the generation of rules with no loss of information [15]. And the last one is its covering of the maximum of important information. Indeed these basis are generated from closed frequent itemsets and has a smaller premise.

V. EXPERIMENTAL RESULTS

To evaluate the efficiency of our approach, we performed on two real datasets that we have downloaded from [16]. The first one is the "webdocs" with a size of 1.4 Gigabyte and it consists of 1,692082 transactions with 5,267,656 distinct items. The second is the "connect" dataset which present game events report with a size of 8.8 MegaBytes.

Our experimentations were all carried out on three nodes. Each one is equipped with Hadoop 2.7.3 version. Each node is characterized by 4 Intel Core processors with 4GB RAM and 500G hard disk. We ran all nodes on Ubuntu 16.04. As we have three nodes, one node was designed as master and the two other nodes were designed as slaves. To implement our algorithm, we have used the java programming language with the JDK-8 version.

We have compared our approach to a naïve adaptation of the Apriori to the MapReduce.

Fig. 2 and Fig. 3 illustrate the experimental results related to the execution time, tested on both datasets under different values of minimum support and with 80% minConfidence. Results show that the more the minSupport decreases the more the execution time increases, and this is due to the increase in the number of generated rules.

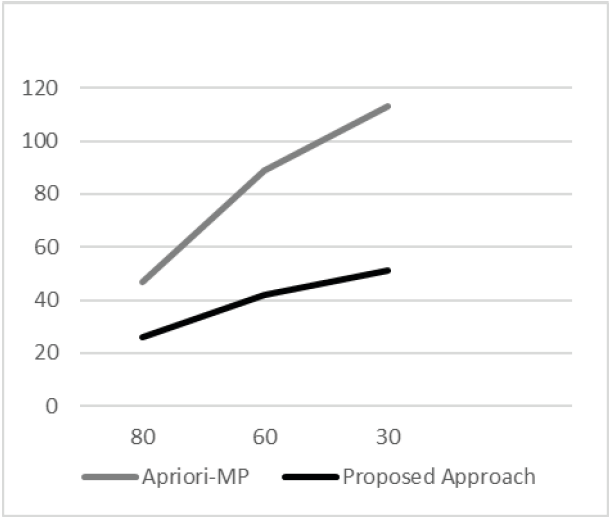

Fig. 2 .Running time on "Connect" dataset

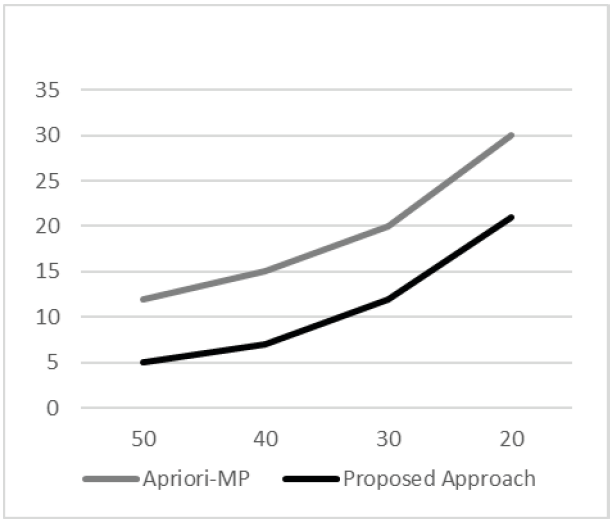

Fig. 3 .Running time on " webdocs" dataset

Fig. 4 and Fig. 5 illustrate the experimental results related to the number of generated rules on both datasets.

The results affirm that our method reduces significantly the number of rules. The reason is that we mine generic basis of rules generated from frequent closed itemsets. These basis presents a set of compact rules with no loss of information.

In addition, by comparing the rules generated by our proposed approach compared to those generated by the Apriori algorithm under the Hadoop platform, we can conclude that we extracted fewer number of rules without losing information and favoring the rules with small premise, which helps in decision-making.

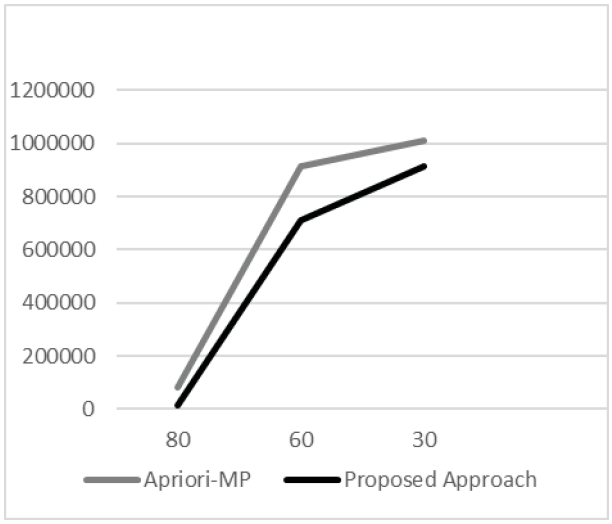

Fig. 4 .Number of rules generated with "Connect" dataset

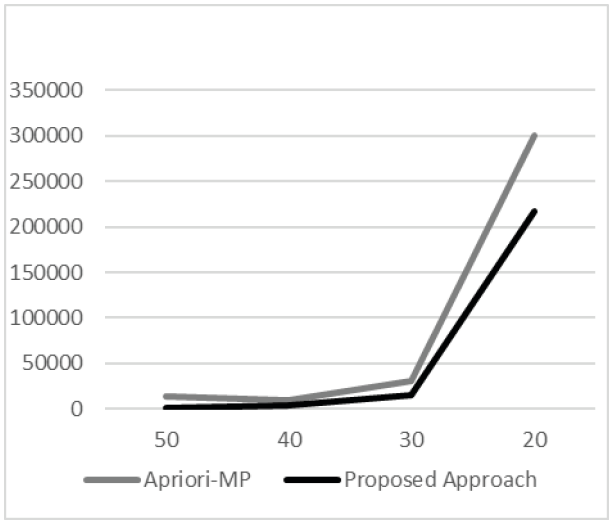

Fig. 5 .Number of rules generated with " webdocs" dataset

VI. Conclusion

In this paper, we have introduced an approach for distributed mining of association rules from big data, based on the MapReduce framework. We proposed to mine generic basis of association rules. Indeed, generic basis offer a compact set of informative and non-redundant rules, with the guarantee of no loss of information. The obtained affirm the efficiency of our algorithm in ARM from large datasets.

References

[1] R. Agrawal, T. Imielinski, and A. Swami.: "Database mining: A performance perspective", IEEE Transaction on Knowledge and Data Engineering: Special issue on learning and discovery in knowledge-based databases, December 1993.

[2] R. Agrawil, R. Skirant.: "Fast algoritms for mining association rules", in Proceedings of the20th International Conference on Very Large Databases, pp. 478–499, June 1994.

[3] J. Han, J. Pei and Y. Yin.: "Mining Frequent Patterns without Candidate Generation", in ACM SIGMOD International Conference on Management of Data, vol. 29, no. 2, pp. 1-12, 2000.

[4] M.J. Zaki, S. Parthasarathy, M. Ogihara and W. Li.: "New Algorithms for Fast Discovery of Association Rules", in proceeding of the 3rd international conference on knowledge discovery and data maining, pp.283-286, 1997.

[5] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme and L. Lakhal.: "Mining frequent patterns with counting inrfence", in KDD cenference, pp.66-75, 2000.

[6] J. Dean and S. Ghemawat.: "Mapreduce: Simplified data processing on large clusters" Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[7] K. Shvachko, H. Kuang, S. Radia and R. Chansler.: "The hadoop distributed file system", in the IEEE 26th sysposium on mass storage, pp.1-10, 2010.

[8] O.R. Zaine, M. El-Hajj, P. Lu. : "Fast parallel association rule mining without candidacy generation", in the ICDM conference, pp.665-668, 2001.

[9] L. Liu, E. Li, Y. Zhang, Z. Tang. : "Optimization of frequent itemset mining on multiple core processor", in the VLDB cenference, pp.1275-1285, 2007.

[10] J. Wang, J. Han and J. Pei.: "Closet+: searching for the best startegies for mining frequent closed itemsets", in the KDD conference, pp. 236-245, 2003.

[11] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal. : "Discovering frequent closed itemsets for association rules", in ICDT, vol. 1540, pp. 398–416. Springer, Heidelberg 1998.

[12] S-Q. Wang, Y-B Yang, Y. Gao, G-P. Chen and Y. Zhang.: "Mapreduce-based closed frequent itemset mining with efficient redundancy filtering", in ICDM workshop, pp.449-453, IEEE Computer Society , 2012.

[13] H. Li, Y. Wang, D. Zhang, M. Zhang and E.Y. Chang.: "Pfp: parallel fp-growth for query recommendation", in ACM Conference on Recommender Systems (RecSys), pp. 107–114, 2008.

[14] M. Zitouni, R. Akbarinia, SB. Yahia, F. Masseglia.: "A prime number based approach for closed frequent itemset mining in big data", in the 26th International conference on database and expert systems applications (DEXA'2015), vol 9261, pp 509–516, 2015.

[15] G. Gasmi, S. BenYahia, E.M. Nguifo, Y. Slimani: "IGB: A new informative generic base of association rules", in Proceedings of the Intl. Ninth Pacific-Asia Conference on Knowledge Data Discovery (PAKDD'05), LNAI 3518, Hanoi, Vietnam, Springler-Verlag,pp. 81–90, 2005.

[16] http://fimi.cs.helsinki.fi/data.