Scalable Computing: Practice and Experience
Volume 6, Number 1, pp. 99–109. http://www.scpe.org

ISSN 1895-1767

DISTRIBUTED DATA MINING

VALÉRIE FIOLET[†] [‡], AND BERNARD TOURSEL[†],

Abstract. Knowledge discovery in databases, also called Data Mining, is an increasing valuable engineering tool. The huge amount of data to process is more and more significant and requires parallel processing.

Special interest is given to the search for association rules, and a distributed approach to the problem is considered. Such an approach requires that data be distributed to process the various parts independently. The research for association rules is generally based on a global criterion on the entire dataset. Existing algorithms employ a large number of communication actions which is unsuited to a distributed approach on a network of workstations (NOW).

Therefore, heuristic approaches are sought for distributing the database in a coherent way so as to minimize the number of rules lost in the distributed computation.

Key words. Data Mining, Association Rules, Distributed Processing, Heuristic.

1. Introduction.

Data Mining stands for the process of knowledge discovery in databases. Here, particular attention is given to the problem of association rules which exhibit a dependence on attributes of the database.

The computation of association rules has an exponential complexity with regard to the number of attributes of the database, so having recourse to parallelism could increase the size of manageable databases.

Parallel Algorithms for Shared Memory machines have been used in the past exploiting Synchronous Communication at each step of the process. However, existing parallel algorithms are clearly suited to parallel shared memory computers (SMP) because of the many synchronized communications they produce between each step of the process.

An algorithm running on a network of workstations, in which little asynchronous communication actions can be tolerated, is proposed here. The main problem of such an algorithm is now to distribute data to treat the obtained data fragments independently.

1.1. Association Rules Problem.

A well-known application of the search for association rules is the "Market-Basket Analysis" problem. The problem is to identify relationships between products that tend to be bought together.

The principal interest of the method is the clarity of the produced results, which can be easily understood by professionals of the data domain.

1.1.1. Problem definition.

Given a set of n records (each composed of items—or attributes) and m distinct items ($\in I$); the search for association rules consists in producing dependence rules to find relations between those items which predict the occurrence of other items.

An association rule is then an implication of form $X \Rightarrow Y$, where X and Y are itemsets belonging to I and where $X \cap Y = \emptyset$ [1]. (X is the condition or antecedent; Y is the conclusion or consequence).

We call k-itemset a set of k items.

The number of possible association rules is $O(m2^m)$. The computation complexity is $O(nm2^m)$.

In order to reduce the complexity of the problem, statistical measures are generally associated to computations, and help to reduce the size of the search space:

- Associated with each itemset is a **support**. The support of an itemset is the percentage of records in a database, D, which contains this itemset. (The support measures how interesting the itemset is, that is, its frequency in the dataset).
- Each association rule has an associated **confidence** measure:

$$\text{confidence}(X \Rightarrow Y) = \text{support}(X \cup Y)/\text{support}(X).$$

(The confidence of a rule measures a real causality between the condition and the conclusion).

[†] Laboratoire d'Informatique Fondamentale de Lille (UPRESA CNRS 8022), University of Lille1, Cité Scientifique, 59655 Villeneuve D'ascq CEDEX, FRANCE (tél. 03.20.43.45.39). {Fiolet, Toursel}@lifl.fr

[‡] Service Informatique—University of Mons-Hainault 6, Avenue du Champs de Mars, 7000 MONS, BELGIUM (tél. 065.37.34.46, Valerie.Fiolet@umh.ac.be)

[1] Formulation of the problem proposed by Agrawal and al. [1] and [2]

Table 1.1
Apriori Algorihm

```
Input: the database, the support threshold (minsup)
Output: $F_k$: set of frequent k-itemsets

$F_1$ = frequent 1-itemsets
for($k = 2; F_{k-1} \neq \emptyset; k++$)do
  $C_k$ = AprioriGen($F_{k-1}$)
  for each element $c$ in $C_k$ do
    count support for $c$.
  end for
  $F_k = \{c \in C_k | c.support \geq minsup\}$
end for
return $\cup F_k$
```

The search for association rules consists in finding rules of support and confidence greater than the given thresholds. The computation of itemsets with supports greater than the threshold is the most expensive part of a process.

A well-known algorithm for the computation of frequent itemsets is the Apriori algorithm (see Table 1.1) proposed by Agrawal and Srikant ([2], [9]). It is used as follows:

- to compute the supports of items, and then to identify frequent items (frequent 1-itemsets)
- to generate candidate 2-itemsets, to count their supports, and then to identify frequent 2-itemsets
- to generate candidate 3-itemsets, to count their supports, and then to identify frequent 3-itemsets, and so on...

The guiding principle is that: **Every subset of a frequent itemset has to be frequent**.

This principle is used to prune many candidates using the AprioriGen algorithm, which provides candidate k-itemsets (C_k) by computation on F_{k-1} (frequent (k-1)-itemsets).

1.2. Existing Parallel Algorithms. Existing parallel algorithms based on Apriori can be classified as:

- those that propose a replication of candidate k-itemsets: making it necessary to synchronize after each k^{th} step so as to communicate local supports counts. In this way, it is possible to identify globally frequent k-itemsets which are useful for the next $(k+1)^{th}$ step (Count Distribution [3][10], Parallel PARTITION [8], PDM [7]);
- those that propose a partitioning of candidate k-itemsets: making it necessary to communicate data fragments to each processor to count supports and to synchronize the process at the end of each step (Data Distribution [3][10], Intelligent Data Distribution [5], DMA [4]).

In these two kinds of parallel algorithms, it is necessary to have many synchronous communications between two steps (the k^{th} and the $((k+1)^{th})$). These kinds of solutions are clearly not suited to a system with distributed memory and costly communications.

1.3. A Distributed Solution. Computation of association rules is very expensive (the cost is exponential). Parallelism may offer a way to bring this cost under control and so permit the processing of larger databases. In two ways:

- storing the database,
- computing frequent k-itemsets on the database.

Since existing parallel algorithms use many synchronous communications, they are inappropriate on a network of workstations.

A real distributed method is proposed to solve the problem of the search for frequent itemsets and the computation of association rules.

The count of supports for itemsets could be distributed (as it is in existing parallel algorithms), but the major difficulty for a real distribution of the process is the necessity to access to the whole database to count an itemset support. Furthermore, all the information in a step (computation of 1-itemsets for example) is useful for the next step (computation of 2-itemsets, for example).

These requirements appear as a global criterion that must be taken into account. Potentially all items can appear together in an itemset; each record can have an influence on the support of an itemset.

2. An Entirely Distributed Process.

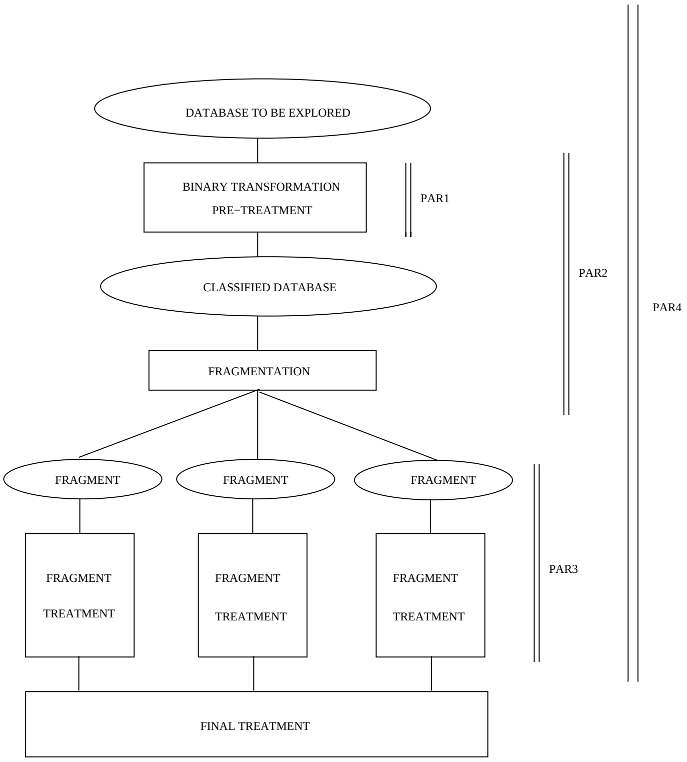

Fig. 2.1. *General schema*

2.1. General Schema. A general schema is suggested (see Figure 2.1) to find association rules on a dataset. Taking into account that a data mining process (the search for association rules, for example) is part of a KDD process (Knowledge Discovery in Data: cleaning, pre-processing, data mining, results validation . . .), and that the algorithm has been tested on a real medical database, it was decided to introduce into the schema the pre-processing phase that aims to format data for the problem. Assuming that the database is initially distributed in a vertical split (non-overlapping subsets of attributes on each workstation):

- The first stage consists in pre-treating the data (see PAR1 on Figure 2.1).

- The second stage (included in PAR2 on Figure 2.1), consists in fragmenting the data to search for association rules. The quality of the distribution will influence the quality of the final results.

- The third stage (see PAR3 on Figure 2.1), computing frequent itemsets and association rules, could then run totally independently on each fragment of data, using a classical sequential algorithm for association rules (APriori, for example) on each component.

- After the end of this third step, a decision must be taken for obtaining the results (FINAL TREATMENT on Figure 2.1):

1. either the results are considered on each fragment independently (what is linked to the profiles concept, see Sections 2.2.2 and 3);
2. or the results are brought together and a ”corrective” method is applied to them.

To distribute the processing phase (the data mining process), it was decided to execute all the processes in a distributed environment, therefore, the global PAR4 process (see Figure 2.1), consisting of computing all phases on a NOW, is considered. A component approach of the PAR4 process is adopted (see Figure 2.2). The process is broken up in three stages, each being carried out by a distinct kind of component.

- The **Pre-processing step** (the binary transformation of the database) is carried out by Clustering Objects according to the algorithm used for the binary transformation (see 2.2).

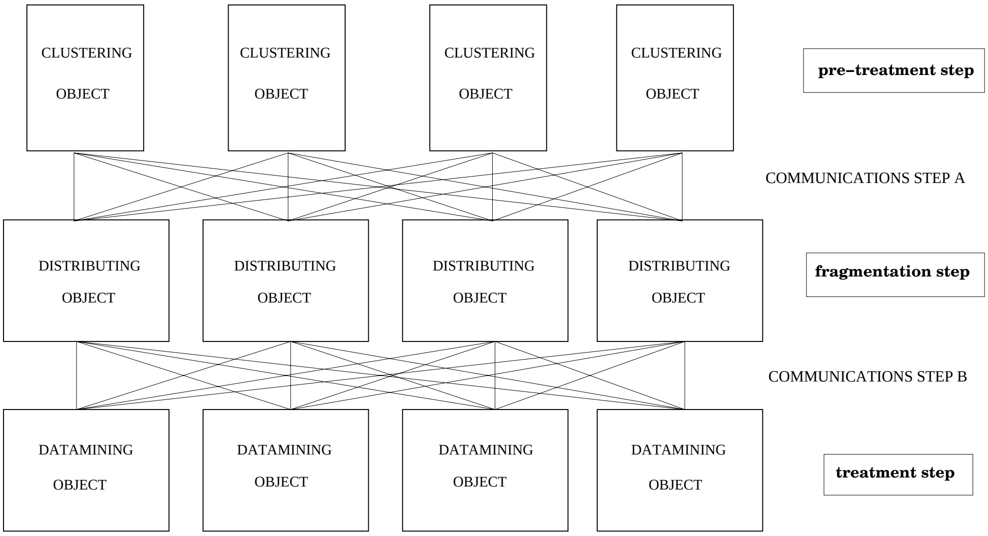

FIG. 2.2. *A Components Vision*

- The **Fragmentation step** by Distributing Objects.
- The effective **Treatment step**, the computation of association rules, by Datamining Objects.

Each stage of the general schema is described in the next section.

2.2. Successive Steps.

2.2.1. Pre-processing step. The pre-processing stage consists in a binary transformation of the database. Each attribute of the database is composed of a set of continuous values. Database attributes are classified to format the data for the problem. Since no classification exists for most of the database attributes, classes must be identified for each. A clustering algorithm (K-Means) that aims to identify groups (clusters) of values is used, those groups will be used to classify the database. This algorithm looks for groups of values:

- values in a group that are most similar,
- values in distinct groups that are least similar.

Let G_j be the j^{th} identified group; c_j, the medium value for this group; r_i, the i^{th} record of the database; and b_{ij} the bit that represents the presence of the j^{th} group for the i^{th} record. Values of a G_j group are then replaced by c_j in the database. Then for each record, r_i, of the database, D, and for each identified c_j, a b_{ij} bit can be used to represent c_j, the presence for the r_i record (see Section 3.3 for specific results on the database).

In this way, a binary database is obtained. For each attribute from the initial database, x binary columns are obtained (with only one bit equal to 1, one class for each initial attribute of a record).

2.2.2. Fragmentation step. The entire process is based on the fragmentation stage which consists in identifying groups or profiles of records. Data is then fragmented according to these profiles.

The distribution of processes with regards to profile fragmentation is related to work [6] in which association rules are computed, and groups (profiles) are then generated on the new compact representation of the data (association rules). This approach is not adapted to large databases (because of the exponential cost of association rules computation). It is proposed to approach the problem by building on the work of Lent [6]. The aim is to identify groups of similar records and then to compute association rules on these groups.

This step of data distribution will be discussed later.

2.2.3. Process step—Association rule computation. In this step of the schema, the database is fragmented in binary fragments, representing groups of records so that records in a fragment are most similar, and records in distinct fragments are least similar (the fragmentation respects clustering criteria).

The effective process could then run independently on each data fragment, using the classical sequential Apriori algorithm described in Section 1.1. Association rules are computed on local data.

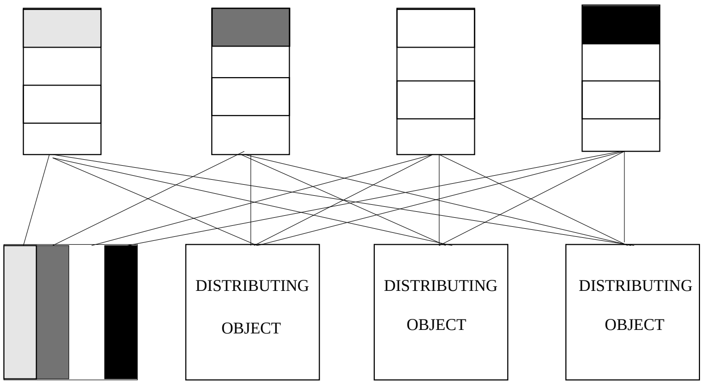

FIG. 2.3. *Communications between Clustering and Distributing Objects: Records combining.*

2.3. About Communication Evolution. Contrary to existing parallel algorithms that produce many synchronous communications between each step, the approach suggested here tends to minimize the number of communications until a quasi-zero-communication effective process.

Communications are limited to the displacement of data between each step: after pre-processing before distribution (Communications step A)) and before process by Apriori algorithm on fragments (Communications step B).

Let C_i be the i^{th} Clustering Object; D_i, the i^{th} Distributing Object; Da_i, the i^{th} Datamining Object; Fa_i, the i^{th} data vertical fragment from initial database; and Fb_{ij}, the j^{th} binary data horizontal fragment sent by C_i.

2.3.1. Data communications. Between each processing phase, the data is communicated to the following objects.

As each attribute can be treated independently, the distribution for the pre-processing step can simply be derived from the initial distribution of data (assuming that the initial distribution comes from a vertical split of the database). A C_i Clustering Object will then treat the Fa_i vertical data fragment. Therefore no communications are needed at the beginning of computation.

The first step of communications consists in communicating data from pre-processing (clustering) objects to Distributing Objects (see Figure 2.2 - Communications step A). Data has been split vertically for the pre-processing, the distribution criterion is based on entire records. The distribution is here based on an horizontal split, so data has to be recombined as shown in Figure 2.3.b. Each C_i Clustering Object will send horizontal data fragment F_{ij} to the D_j Distributing Object. D_j Distributing Object will then merge each F_{kj} fragment from Clustering Objects by a vertical recombining.

The second step of communications consists in communicating data from Distributing Objects to Datamining Objects (see Figure 2.2—Communications step B). The data distribution is provided by computations on Distributing Objects. Potentially each Distributing Object has to communicate data to each Datamining Object.

2.3.2. Pipeline. The pre-processing and distribution processes can be pipelined. Since pre-processing of each attribute of the database runs independently of every other attribute, as soon as the pre-processing of one attribute is finished, data for this attribute may be communicated to the Distributing Objects, meanwhile another attribute can be pre-treated. In this way, part of communication time can overlap with the computation.

This overlapping of communications can also take place between Distributing and Datamining Objects (see Figure 2.4).

2.3.3. Criteria for communications. The difference between synchronous and asynchronous communications, for the problem, must not be forgotten:

- synchronous communications could lead to an optimal choice for fragmentation from a global state.
- asynchronous communications allow a communication overlapping that decreases the latency time, but that does not permit an optimal split.

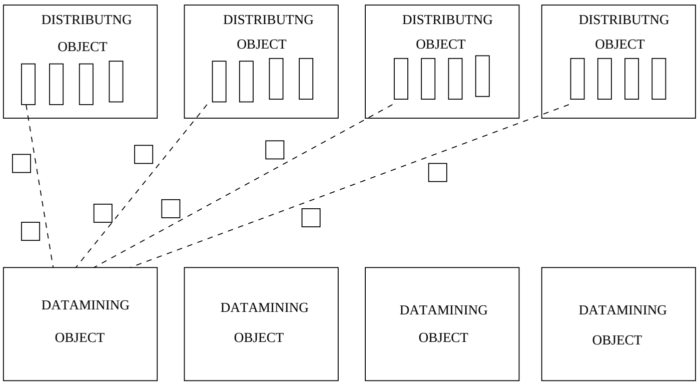

FIG. 2.4. *Pipeline between Distributing and Datamining Objects.*

Asynchronous communications are intended to be used since the purpose is to work on a NOW with irregular communication delays.

3. Data Distribution Step. The fragmentation problem consists in distributing the data in a way to keep similar data together, and least similar are split into distinct fragments. Using this kind of distribution, it could be expected that records in a fragment contain a subset of frequent items. Computation of frequent itemsets will then be less expansive .

Given n records and m items for all the databases; given n' records and m' items for a data fragment (with $n' < n$ because of fragmentation of records, and $m' < m$ as expected from profile fragmentation; distinct fragments have distinct sets of frequent items which are subsets of the whole set of items). The potential complexity (number of itemsets to be evaluated) for the process of the considered fragment (without pruning of infrequent itemsets) is then $O(n'2^{m'}) < On2^m)$.

So, the potential process complexity for K fragments is:

$$O(n'_k 2^{m'_k}) < O(n2^m) \text{ where } \sum_{k=1}^{K} n'_k = n.$$

A real decrease of global complexity could be expected. To conserve coherence of the results, data has to be distributed by identifying profiles. Then each data profile will be computed independently. The problem is that profiles are unknown.

As no global vision for the distribution of records exists, classical clustering algorithms can not be used to identify record profiles (clusters of records). The fragmentation supplied would not be optimal.

3.1. Fragmentation methods.

1. A first idea consists in using results of the pre-processing phase to distribute records. An attribute of the initial database gives X binary attributes in the binary database to be distributed. Only one of the binary attributes can be present for a record. Distributing data using the decomposition of one initial attribute and repeating this process using several initial attributes, to obtain uniform size fragments, could then be imagined.
   This proceed does not decrease the complexity of the problem. Only the attributes used for distribution become trivial. Other attributes (or items) must be computed, since attributes that classify the database do not exist (or at least are unknown).
2. A second idea consists of folding records (using binary logical functions : AND, XOR, OR...), and distributing records using the value of the obtained folding. The major difficulty is here that the local similarity between two records is lost in the folding, and two similar records may be separated in the distribution.
3. A third idea, that particular attention is given to, consists in using one (or several) pattern record(s), the same for all Distributing Objects, and to distribute data using a distance to this (these) pattern record(s).

Some pattern record(s) and a distance function (between patterns and records to distribute) should be chosen.

3.2. Fragmentation with regard to pattern records. The first parameter for the distribution step proposed consists in the choice of **pattern record(s)**.

Initially, a pattern record was randomly generated. This did not match with the database to be managed. The solution was not satisfying because of the random effect: the pattern did not represent the database structure.

Next, a pattern record was randomly elected in the database (one record is elected as a referential pattern record). Results were then relative to the database being worked on. The furnished distance values were acceptable with regards to other parameters. But this solution is not really satisfying because of the random choice of the pattern; a particular record may be chosen (as a pattern) that will clearly influence the split. A possible variation is to use a limited number of referential pattern records or to accept only records which agree with some quality function.

Work is now oriented on the computation of a pattern record that matches with the database (a kind of medium record for the database which conserve records structure), using all the information from pre-processing phase. For example, during pre-processing phase it is possible to obtain the proportion of bits equal to 1 for a binary attribute.

The second important parameter in the schema is the **distance function** used to distribute data according to distance from the pattern.

The first distance function that was included in the schema was the Hamming distance function. It counts equality of bits between the pattern and each record to be affected to a fragment. The Hamming distance function seems to be an obvious choice since a binary database is used at this stage of the process, but is too simple for the problem. Other distance and dissimilarity functions on binary data inspired from Jaccard, Salton and Cosine similarities were then used, as well as some using logical function properties. But these experimentations did not produce expected results. The obtained dissimilarity values were no more suited to the problem than those obtained with Hamming distance.

The problem with these functions is that they do not distinguish between a match of items and a match of bits. As the aim is to obtain binary fragments in which records are most similar in term of identical items (bits equal to 1), particular attention should be given to this similarity of items (similarity of bits 1).

A dissimilarity function inspired from bioinformatic work for sequence alignment has been constructed. It gives a particular penalty to dissimilarity of bits, and gives a gain to similarity of items (bits 1). In this way, one dissimilarity of bits is accepted for x similarities of items. Dissimilarity values obtained in this way are much appropriate to the application.

As work is actually directed on the use of a pattern that describe the database (a medium record for the database), the dissimilarity function may be adapted to this kind of pattern.

Computation of patterns that match with the database (using information from pre-processing) could be done in two ways:

- A binary pattern record with bit value of the majority in the records for each attribute (or item).
- A float pattern with proportion of 1 in the records for each attribute (or item).

For the first pattern (the binary one), binary distance functions exposed before could be used.

For the second pattern (the float pattern), the following distance function could be used: for each attribute (or item) the absolute float value between proportion rate (float in the pattern) and bit value is measured; then float values on each attribute are merged using Euclid's method or summation methods.

4. Experiments.

4.1. Experiment Environment. Experiments were realized on real data, a medical database composed of 182 continuous attributes (before pre-processing phase) and 30.000 records. After the clustering phase, we obtained 480 binary attributes (or items). We have increased the width of the database but many rare binary items were pruned at the beginning of frequent itemsets generation. The complexity of the database for the considered problem comes from its large number of attributes.

Programs were realized using Java RMI, on a linux heterogeneous environment, pipeline mechanisms for the distinct phases were implemented.

4.2. First Experiment. During the tests, the computation of frequent itemsets was stopped after frequent 1-itemsets generation (because of the complexity of this generation for next steps).

Data fragments quality was evaluated with regards to criteria for a good split :

- most similar records in the same fragment;
- less similar records in distinct fragments.

A complete evaluation was realized for the Hamming distance function (using a randomly chosen record from the database as pattern record), on which pertinence of the distribution was computed using criteria exposed above.

Observations on the distinct steps during tests and on criteria of distribution serves to highlight some difficiencies in our distance function (Hamming distance), that is intended to be corrected in the current works.

From a human observation of obtained fragments (observations of binary tables), the distribution using Hamming distance does not fulfill the distribution criteria; the distance function is used to compute a mathematical evaluation of those distribution criteria that is inappropriate (it is not simple to look for clustering criteria on binary data).

In addition, a dramatic imbalance in partitions was achieved.

A comparison of generated frequent itemsets is not relevant since the generation of frequent itemsets was stopped after the first stage (1-itemsets).

Other experiments were realized using distinct binary distance functions, with distinct binary pattern records (a randomly generated pattern record, a randomly chosen pattern record, a pattern record with the bit of the majority for each attribute, and a pattern of proportion for each attribute (see Section 3.2).

A real problem of imbalance was due to the arbitrarily choice of intervals for the distribution steps (see Table 4.1). For ten arbitrary fixed intervals, a lot of empty or almost empty fragments were obtained. For some distance functions, such as Jaccard, for example, only one fragment was obtained independently of the pattern record used. For most of the distance functions, only four or five fragments where obtained when ten were expected.

4.3. Second Experiment.

- Because of the complexity of the process of computing the whole set of frequent itemsets and comparing the distributed results to a global (not distributed) process of data, the distinct distance functions and pattern records were tested, first on synthetic data, next on a small fragment of the medical database (20 continuous attributes, 600 records; 23 binary items were obtained after pruning infrequent).
  Then frequent itemsets could be computed on the entire dataset to compare these results to those obtained from the distinct distributed solutions (distinct from the perspective of distance functions and pattern records used).
- In the asynchronous schema, distance intervals are used to distribute data (to simulate a clustering method on distance values), but these intervals can not be fixed arbitrary as done in the first experimentation. Arbitrary values of intervals bring a dramatic imbalance in distribution, in particular, at the end, all records could be in the same fragment or in only four or five fragments (see Table 4.1).
  Intervals should be adjusted to the database using pre-processing information. To bypass this problem, a synchronization method was used during the distribution stage. A clustering algorithm was used on the obtained distance values (distance values to the pattern record(s)). Clusters of values were obtained, and then data was distributed according to those clusters of distance values (see Table 4.2).

Evaluation of a distribution

Multiple criteria are needed to evaluate a good method. Relevance of methods could be evaluated on three points:

1. The rate of preservation of frequent itemsets to maximize (or rate of loss to minimize). (loss of frequent itemsets: globally frequent and not frequent by distributed process).
2. The rate of false positive frequent itemsets to be minimized (false positive frequent itemsets: frequent in a distributed process and not globally frequent).
3. The complexity of process on each workstation (addition of complexities in a distributed schema must be less than the global complexity).

Table 4.1
Rate of distribution of records for some distance functions using intervals

Distance function	Rate of distribution of records (only not empty fragments appear in the table) (%)						
Binary randomly chosen pattern record							
Hamming	1,7	16,5	33,5	37,7	9,8	0,8	
Hamming variation (Items2)	1,3	7,8	28,8	43,3	16,5	2	0,17
Bioinformatic (2)	1,3	7,8	28,8	43,3	16,5	2	0,17
Salton	0,7	3,5	3,3	13,5	66,3	12,7	
Cosinus	0,7	1,8	2,17	22,7	22,8	38	11,8
Jaccard	0,17	99,8					
Binary randomly generated pattern record							
Hamming	1,7	1,7	10,8	50,17	37,17		
Hamming variation (Items2)	6,7	68,17	25	0,17			
Bioinformatic (2)	3,17	30,7	49,17	0,16	1		
Salton	2,3	97,7					
Cosinus	0,3	13,8	85,8				
Jaccard	99,7						
Bit for majority of records							
Hamming	7,8	45,17	39,3	6,5	1,17		
Hamming variation (Items2)	7,8	45,17	38,3	6,8	1,7	0,17	
Bioinformatic (2)	7,8	45,17	38,3	6,8	1,7	0,17	
Salton	47,3	28,5	8,8	15,3			
Cosinus	5,7	10,17	9,7	42,17	19,8	12,3	0,17
Jaccard	100						
Float pattern of proportion							
Summation	8	85	3,3	1,7			
Euclid	5,7	46	39,7	6,8	1,8		

Table 4.2
Rate of distribution of records using clustering of distance values

Distance function	Number of identified clusters	Rate of distribution of records (only not empty fragments appear in the table) (%)						
Bit for majority of records								
Hamming	5	14,8	17	10,7	10,3	3,5		
Hamming variation (Items2)	6	19,3	16,3	26,17	20,7	9,7	7,8	
Bioinformatic (2)	7	19,3	16,3	13,8	12,3	20,7	9,7	7,8
Salton	3	42,67	35,67	19,67				
Cosinus	5	52,67	17	21,5	4,5	4,33		
Jaccard	2	80,67	19,33					
Float pattern of proportion								
Summation	3	19,33	38	42,67				
Euclid	4	19,33	12,33	28,8	39,5			

Results

For the distance functions that give a sufficient discrimination of records (a sufficient numbers of clusters and a sufficient number of records in each cluster), a rate of preservation of frequent itemsets of 100 %, and a rate of false positive frequent itemsets between 15% and 20% (depending on fragment) of distributed generated frequent itemsets were reached.

Addition of complexities on each fragment is less than the global complexity because of non frequent items on fragments that permit to decrease the number of itemsets to be computed, and because of obvious items not included in the process (see Observations above). Less itemsets are computed on each fragment than in a global process and addition of complexities from each fragment is less than the global complexity.

Distinct Problems

1. Problems from a non uniform data distribution: Some distance functions do not permit the database to be broken into a sufficient number of distinct fragments (profiles), some like Jaccard, Salton, Summation can be immediately rejected on this basis(see Table 4.2).
2. Problems in the evaluation: Generated frequent itemsets are compared in a global process and in a distributed process, by merging results from fragments without paying attention to support values. The threshold for frequency in the distributed solution is relative to the size of fragments (in particular, little fragments give problems of complexity and should be treated separately). A good evaluation could not be made without a corrective method (see Section 2.1), that could consist in adjusting threshold support to local data using pre-processing information.

Observations

If a threshold of obviousness is used, beyond which an item could be considered to appear in all records, many items can be pushed aside, this will result in a decrease in the complexity of the computation. Those items will then be replaced at the end of the process, or be given separately from results.

In the distributed experiments, more obvious items were found on fragments than in a global process. This confirms the relevance of distribution (see Table 4.3).

The complexity to which results refer takes into account these obvious items (they have been pushed aside at the beginning of process on fragments).

Table 4.3
Rate of obvious items (threshold of obviousness : 95%)

Distance function	Rate of obvious items (only not empty fragments appear in the table)
Bit for majority of records	
Globally obvious	26,09
Hamming	43,48 56,52 26,09 34,8 4,3
Hamming variation (Items2)	69,56 43,48 52,17 34,8 26,09 4,3
Bioinformatic (2)	69,56 43,48 60,87 52,17 34,8 26,09 4,3
Salton	26,09 52,17 8,7
Cosinus	56,52 43,48 34,8 26,09 4,3
Jaccard	26,09 30,4
Float pattern of proportion	
Summation	69,56 43,48 17,4
Euclid	69,56 69,56 43,48 17,4

4.4. Future Works.

- We intend to compute interval values that could bring a good heuristic for this clustering method with regard to distance values. Maybe a clustering algorithm could be computed on a partition of the database. Work is in progress for this part.
- Corrective method : the support threshold on each site has to be adjusted to data on this site (this could clearly come from the beginning of computation on site by evaluating 1-itemset support, and with a comparison to global support (information that we get from pre-processing).

This corrective method and the arrangement of distributed threshold support would permit, we hope, a decrease complexity on some sites. It might be not be necessary to make a distinction for sites with too few records.

Tests show that it is necessary to draw a parallel between the relevance of frequent itemsets and the distribution of data (the size of distinct fragments). It will be necessary to test other databases to be sure that orientations derived from the observations made on the medical database are not specific to these data.

The schema uses many parameters (local threshold, distribution intervals, etc...) that should not be fixed arbitrarily. Values for those parameters have to be computed from pre-processing information.

5. Conclusion. It is necessary to remember that the schema will supply a heuristic for the problem of association rules. The distribution of data is the critical phase to obtain a good heuristic for the problem. To compute profiles without a global vision of data is of course a limitation to the process but the distribution could provide sufficient results with regard to speeding-up te process.

Many directions need to be explored, particularly for the choice of referential pattern records and the distance function. A lot of information can be obtained from the pre-processing phase and it is important to use it to distribute the data. This is the direction given to this work. It is intended to duplicate some records in the distribution. This will need a specific corrective method to take those duplications under consideration. It is also intend to combine the clustering phase and the distributing phase to avoid data communications between them.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami, *Database mining: A performance perspective,* IEEE Transactions on Knowledge and Data Engineering : Special issue on learning and discovery in knowledge-based databases 5(6):914-925 (December 1993).

[2] R. Agrawal and R. Srikant, *Fast algorithms for mining associations rules in large databases,* In Proc. of the 20th Int. Conf. on Very Large Data Bases (VLDB'94), pages 478-499 (September 1994).

[3] R. Agrawal and J.C. Shafer, *Parallel Mining of Association Rules,* IEEE Trans. on Knowledge and Data Eng., 8(6):962–969 (December 1996).

[4] D. W. Cheung, J. Han, V. T. Ng, A. Fu and Y. Fu, *A fast distributed algorithm for mining association rules,* In Proc. of the 4th Int. Conf. on Parallel and Distributed Information Systems, pages 31-42 (1996).

[5] E-H. Han and G. Karypis. Scalable Parallel Data Mining for Associations Rules, IEEE Trans. on Knowledge and Data Eng. , 2(3):337-352 (May-June 2000).

[6] B. Lent, A. Swami and J. Widom, *Clustering Association Rules,* In Proc. of the 13th Int. Conf. on Data Eng. (ICDE'97), pages 220-231 (April 1997)

[7] J. S. Park, M.-S. Chen, and P. S. Yu, *Efficient parallel data mining for association rules,* In Proc. of the 4th Int. Conf. on Information and Knowledge Management, pages 31-36 (1995).

[8] A. Savasere, E. Omiecinski and S. Navathe, *An efficient algorithm for mining association rules in large databases,* In Proc. of the 21st VLDB Int. Conf. (VLDB'95), pages 432-444 (September 1995)

[9] R. Srikant, *Fast algorithms for mining association rules and sequential patterns,* PhD thesis, University of Wisconsin (1996).

[10] Zaki, *Parallel and Distributed Association Mining: A survey,* (1999).

Edited by: Dan Grigoras, John P. Morrison, Marcin Paprzycki
Received: Semptember 30, 2002
Accepted: December 21, 2002