Frequent Set Mining

Hafida Belbachir , Sid Ahmed Rahal, and Salim Khiat

* Faculty of science, Mohamed Boudiaf U.S.T.O University of Science and Technology, Oran, ALGERIA.

rahalsa2001@yahoo.com, Salim_khiat@caramail.com,

Abstract :

An efficient association rules algorithm was divided in two sub-problems: frequent set mining from Databases and association rules generation. Frequent sets lie at the basis of many Data Mining algorithms. As a result, hundreds of algorithms have been proposed in order to solve the first problem: the frequent set mining problem. In this paper, we attempt to survey the most successful algorithms and techniques that try to solve this problem efficiently.

***Keywords**: Frequent Set Mining, Association Rule, Support, Apriori.*

1. INTRODUCTION

Among the research tasks in excavation of data, the extraction of the rules of association is undoubtedly the task "headlight" which drew more the attention of the researchers and for which many work were carried out. On the one hand this technique allows the discovery of understandable and exploitable rules in a whole of bulky data, rules expressing of associations between items or attributes in a data base.

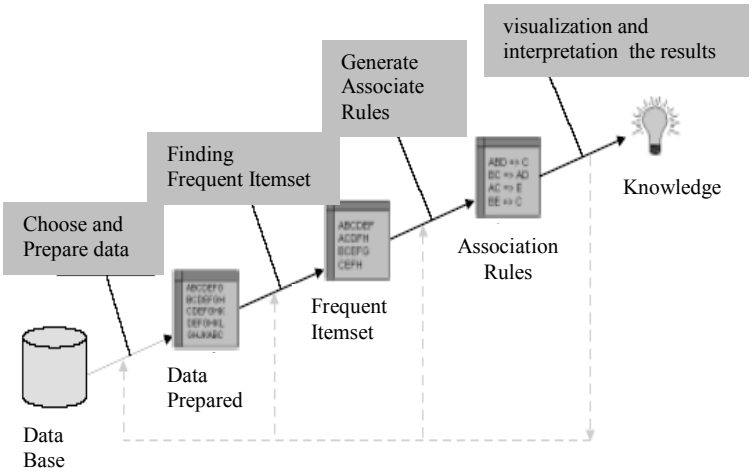
In addition, which return also the search for associations a gravitational subject of research and very credit is its broad field of application to various fields such as marketing, industry, assistance with the medical diagnosis, telecommunications, analysis of space data, telephony, etc.

The purpose of the extraction of the rules of association is to discover significant relations between binary attributes extracts of the data bases. An example of rule of association extracted from a data base of sales of supermarket is: "cereals and sugar → milk (support 7%, confidence 50%)". This rule indicates that the customers who buy cereals and sugar also tend to buy milk. The measurement of *support* defines the range of the rule, i.e. the proportion of customers who bought the three articles, and *confidence* measures defines the precision of the rule, i.e. the proportion of customers who bought milk among those which bought cereals and sugar. The extraction of rules of association consists in extracting the rules whose support and confidence are at least equal to minimal thresholds of support and confidence defined by the user.

The extraction of rules of association is an iterative process and interactive made up of several phases active of the selection and the preparation of the data until interpretation of the results, while passing by the phase of research of knowledge (***extraction of the frequent sets of attributes and generation of the rules of association***).

Two major problems for the use of the extraction of the rules of association gave place to many researches: the problem of time of extraction of the rules of association starting from the data file and the problem of the relevance and the utility of the extracted rules of association. This justifies that many work concerned the research of the frequent itemset and of the relevance of the rule of association, we draw up of it a state of the art in the following section on the first problem.

the process of extraction of rules of association.



2. RELATED WORKS

This section draws up a non exhaustive state of the art of the research tasks in the field of the extraction of the frequent itemset. One finds in the literature a broad range of algorithms considered as alternatives of Apriori, allowing to generate all the frequent itemsets in a compromise base.

However, one finds a very great number of frequent itemsets, which reduces considerably, not only the effectiveness but also the utility of the task. Indeed, a great number of frequent reasons leads to many rules of association, because recall that starting from only one k-frequent itemset one can generate 2^k rules. This forces the user to excavate (still!) in the rules to find the rules most interesting.

This is why, other alternatives were proposed, in particular in the extraction of the condensed representations of **the frequent itemset**, the generation of **the closed itemset** and **the maximum itemset** as we will see it in this paragraph.

2.1. FREQUENT ITEMSET (F)

Algorithms of extraction of the frequent itemset according to the strategy adopted for the course of the lattice, and according to the way of calculating the support of the itemsets in the data base. The algorithm pioneer of search for frequent itemset is Apriori. This last, has the advantage of the simplicity of the implementation: one can immediately apply, by using all the references produced as together catalogues, without it being necessary to build as a preliminary a typology of these products. But this one presents two major disadvantages: the number of candidates phenomenal to generate during each iteration and the problem of expensive scan of the base of data for each iteration, which increases complexity. A certain number of optimizations of the Apriori algorithm was proposed. The fundamental idea which emerges consists in reducing the number of master keys in the base. Algorithms Apriori-TID, Partition, Sampling, DIC, Glare, FP-GROWTH are briefly explained in the continuation, where one uses indifferently the reason term or itemset.

Apriori-TID[1]

In order to improve the performances of Apriori which requires a number of significant course of the base of transactions, the authors of Apriori proposed in [1], the Apriori-TID algorithm. The two algorithms generate the candidates by using the same strategy.

They differ in calculation from the support from the itemsets candidates. Indeed, Apriori-TID uses a whole $\overline{C_k}$ of the form (TID, {ck}) where {ck} is the list of the itemsets contained in the transaction identified by TID. To be more precise, an element of $\overline{C_k}$ is form (TID, {idk}) where {idk} is the list of the identifiers of the itemsets of size K. The support of the itemsets of Ck is equal to the number of appearances of each itemset in $\overline{C_k}$. For K = 1, $\overline{C_1}$ corresponds to the base of transactions: the elements of each transaction are singletons representing the 1-itemsets candidates. For K > 1, $\overline{C_k}$ is built while using $\overline{C_{k-1}}$ and C_k:an element of $\overline{C_k}$ consists of an identifier TID of an element of $\overline{C_{k-1}}$ and list of the k-itemsets Ck contained in transaction TID. If this list is empty, this element is removed. At the time of the first iterations, $\overline{C_k}$ can be very large what causes a problem of storage of the lists of itemsets. In the other hand, the number of elements $\overline{C_k}$ can become smaller compared to the number of transactions in D, especially when K becomes larger. The interest of $\overline{C_k}$ is that the base of transactions D is not used any more to calculate the support of the k-itemsets.

Partition[12]

As the main drawback of Apriori is its slow and iterative support counting mechanism, Eclat has the drawback that it requires large parts of the (vertical) database to fit in main memory. To solve these issues, Savasere et al. Proposed the Partition algorithm [12].

The main difference in the Partition algorithm, compared to Apriori and Eclat, is that the database is partitioned into several disjoint parts and the algorithm generates for every part all sets that are relatively frequent within that part. This can be done very efficiently by using the Eclat algorithm (originally, a slightly different algorithm was presented). The parts of the database are chosen in such a way that each part fits into main memory. Then, the algorithm merges all relatively frequent sets of every part together. This results in a superset of all frequent sets over the complete database, since a set that is frequent in the complete database must be relatively frequent in one of the parts. Finally, the actual supports of all sets are computed during a second scan through the database.

Although the covers of all items can be stored in main memory, during the generation of all local frequent sets for every part, it is still possible that the covers of all local candidate k-sets can not be stored in main memory. Also, the algorithm is highly dependent on the heterogeneity of the database and can generate too many local frequent sets, resulting in a significant decrease in performance. However, if the complete database fits into main memory and the total of all covers at any iteration also does not exceed main memory limits, then the database must not be partitioned at all and the algorithm essentially comes down to Eclat.

DIC[3]

Algorithm DIC was proposed by Brin et al. in [3] reducing the number of courses of the data base. DIC partitionne the data base in blocks of M transactions. During the calculation of the supports of the k-itemsets, after the course of a partition of size M of D, one checks the k-itemsets candidates which already reached the minimum support, DIC uses them then to generate candidates of size (k+1), and starts to count their supports. Thus the support of candidates of different sizes is calculated during the same courses of D. In the other hand of the reduction in the number of sweepings of the data base, DIC simultaneously considers itemsets candidates of different sizes. This generate the problem of storage of the itemsets candidates treated simultaneously and the cost of calculation of the supports of the candidates which is more significant than for Apriori.

Sampling[13]

Another technique to solve Apriori's slow counting and Eclat's large memory requirements is to use sampling as proposed by Toivonen [13].

The presented Sampling algorithm picks up a random sample from the database, then finds all relatively frequent patterns in that sample, and then verifies the results with the rest of the database. In the cases where the sampling method does not produce all frequent sets, the missing sets can be found by generating all remaining potentially frequent sets and verifying their supports during a second pass through the database. The probability of such a failure can be kept small by decreasing the minimal support threshold. However, for a reasonably small probability of failure, the threshold must be drastically decreased, which can cause a combinatorial explosion of the number of candidate patterns. Nevertheless, in practice, finding all frequent patterns within a small sample of the database can

be done very fast using Eclat or any other efficient frequent set mining algorithm. In the next step, all true supports of these patterns must be counted after which the standard levelwise algorithm could finish finding all other frequent patterns by generating and counting all candidate patterns iteratively. It has been shown that this technique usually needs only one more scan resulting in a significant performance improvement [13].

Eclat[16]

Eclat suggested by Zaki and Al in [16] uses the vertical format of the data base, where for each itemset one has sound tidset, i.e. of the whole of all the transactions containing this itemset. The vertical format has the advantage of returning the calculation of the simpler support since it is a question of carrying out in this case of the intersections of the tidsets. Moreover this, the size of the data base reduces automatically since only the transactions concerning a itemset are used for the intersection.

Eclat carries out a research of the frequent itemset initially in-depth and is based on the concept of classes of equivalence.

Two k-itemsets belong to the same class of equivalence if they have in common a prefix of size (k-1).For example ABC and ABD belong to the same class of equivalence. Each class can be treated separately in memory, which makes it possible to break up the lattice into sub-lattice where each sub-lattice represents a class of equivalence.

In-depth research initially starts with the 1-itemsets and keeps the intersection of the tidsets itemsets for the same class of equivalence.

Contrary to Apriori, Eclat does not know all the frequent itemsets on a level given before considering the candidates of the following level, which decreases the effectiveness, because the property of antimonotonicity is not usable any more to prune the space of research. This remains acceptable for the small data bases, but pruning remains poor and deteriorates the performances when it is a question of treating data bases of significant size.

The advantage of this approach, as underlines its authors, is that it remains easily parallelisable, since one can separately seek the frequent itemset in the various classes of equivalence.

FP-growth[5]

One of the most cited algorithms proposed after Apriori and Eclat is the FP-growth algorithm by [5]. Like Eclat, it performs a depth-first search through all candidate sets and also recursively generates the so called i-conditional database Di, but in stead of counting the support of a candidate set using the intersection based approach, it uses a more advanced technique.

This technique is based on the so-called FP-tree. The main idea is to store all transactions in the database in a tri based structure. In this way, in stead of storing the cover of every frequent item, the transactions themselves are stored and each item has a linked list linking all transactions in which it occurs together. By using the tri structure, a prefix that is shared by several transactions is stored only once. Nevertheless, the amount of consumed memory is usually much more as compared to Eclat.

The main advantage of this technique is that it can exploit the so-called single prefix path case. That is, when it seems that all transactions in the currently observed conditional database share the same prefix, the prefix can be removed, and all subsets of that prefix can afterwards be added to all frequent sets that can still be found (Han et al., 2004), resulting in significant performance improvements. As we will see later, however, an almost equally effective technique can be used in Eclat, based on the notion of closure of a set.

Discussion

The number of swaping of the data files realized and the numbers of it itemsets candidates considered by the algorithm of extraction of frequent are two principal factors of the effectiveness so the response times of these algorithms. The importance of the number of swapings carried out is related to the cost of the operations of input/output. The importance of the number of itemset candidates comes owing to the fact that the operations relating to the latter constitute major the part of the computing time CPU of the algorithm.

The algorithms presented in this section were developed for applications concerning of the commercial data bases. The data files used for their experimentation are built starting from data bases of sales of supermarkets and data files synthetic generated according to characteristics of the data of sales. These data are scattered and slightly correlated and the execution times obtained on these data files are weak, about a few seconds at a few minutes. These response times are relatively weak bus in the data of this longest type frequent itemset contain only one limited number of items (the value of μ is weak in front of m) and the total number of frequent itemset (of which the number of itemsets candidates considered depends) is reduced. In the case of contexts for which longest frequent itemset is large, i.e. for a value of μ raised, the performances of these algorithms are degraded considerably:

- Apriori carries out μ iterations and thus μ swapings of the context to extract the frequent itemsets. For high values of μ , this involves significant execution times, the operations of input/output being very expensive in time. This problem is essential in the case of data dense for which the size of the data files is more significant for a number of objects and a number of items identical.
- AprioriTid requires during its execution the storage of the lists of OID associated with the itemsets whose total volume can be more significant than the size of the context of extraction. This problem of space of storage requires the use of the disc and involves significant execution times, particularly at the time of the first iterations.
- For Partition and Sampling, the number of k-itemsets candidates for $1 \leq K \leq \mu$ is more significant than for the Apriori algorithm. This involves a problem of memory capacity necessary to the storage of all the candidates of all sizes at the time of the final phase of the algorithm. Moreover, the last sweeping of

the context which makes it possible to calculate the supports of all the itemsets candidates requires a significant computing time whose share is useless because it relates to infrequent itemset overall.

- For DIC, after $\frac{|B|}{M}^{ème}$ the iteration, $\frac{|B|}{M}$ sets of itemsets candidates (of different sizes) is considered simultaneously at the time of each iteration. Moreover, each one of these Ck sets of k-itemsets candidates is a superset of the Ck unit generated by Apriori. Arise then the difficulties of the space of storage of the sets of itemsets candidates treated simultaneously during the readings of the data file and the total cost of the calculation of the supports of the candidates which is more significant than for Apriori. Indeed, DIC determines the supports of some itemsets candidates infrequents which are not considered by Apriori.

The data bases characterized by a value of μ high represent a significant share of the real data bases. It is thus necessary to develop new algorithms of extraction of the frequent itemsets which make possible the extraction of rules of association starting from this type of data in reasonable times. The dense or correlated data are also characterized by a value of μ high. However, in this type of data, the problem of the effectiveness of the algorithms of extraction of the frequent itemsets is exacerbated by a strong density of the frequent itemsets of big size.

2.2.MAXIMUM FREQUENT ITEMSET(MF)

The algorithms such as Apriori and its derivatives were developed for compromise data bases of sales and were tested mainly on synthetic data files. These data scattered and are slightly correlated what gives a relatively weak execution time. In this type of data bases, longest itemsets frequent contains only one limited number of items, compared to the number of transactions. But the performances of these algorithms are degraded considerably, when they are dense data bases, where the frequent itemsets can be very long, as in the case of the biological or stock exchange data bases. The longest itemsets among the frequent itemsets are in fact the maximum itemsets constituting the positive edge.

$$MF= \{l \in F \mid \forall l' \supseteq l, supp(l') < \gamma \}$$

The maximum itemsets are frequent itemsets of which all the supersets are not very frequent. The problem of extraction of the frequent itemsets breaks up then as follows:

1. Extraction of the whole of the maximum frequent itemsets MF;
2. Calculation of the supports of the subsets of the maximum MF itemsets by carrying out only one course of the data base.

Generally, the algorithms dedicated to the extraction of the maximum frequent itemsets carry out simultaneously a course upwards (research by levels) and from top to bottom (in order to quickly identify maximum big sizes) in the lattice of the itemsets.

Several algorithms were proposed. Among them let us quote MaxMiner [2], Pincer Search [8], MaxEclat [16], MaxClique [16], and GenMax [14].

Discussion

The algorithms presented in this section seek the maximum frequent itemsets simultaneously bottom to the top and the top downwards among the frequent itemsets by maintaining a unit containing largest itemsets frequent possible. The algorithms of extraction of the frequent itemsets by levels require μ iterations in order to determine all the frequent itemsets, μ being the size of the frequent itemsets (maximum) longest. By identifying these itemsets by the research top to the bottom, the algorithms of extraction of the maximum frequent itemsets reduce the total number of iterations necessary to the extraction of the frequent itemsets in the case when μ is high. Moreover, when a maximum itemset frequent is identified, its subsets do not have to be considered any more what reduces the number of itemsets generated candidates and thus the computing times CPU. The results of the experiments confirm the reduction of times of extraction of the frequent itemsets when an algorithm of extraction of the maximum frequent itemsets is used. They also show that the algorithm Max-Miner is overall more powerful than the algorithms To pincer-Search, MaxEclat and MaxClique in terms of execution time.

The method used in the algorithm Pincer-Search generate two significant problems. The first is the determination, at the end of each iteration, of the itemsets maximum candidates containing of the infrequent itemsets which is a Np-difficult problem [2]. The second is the generation of the itemsets candidates of the following iteration which requires many tests of inclusions in the maximum frequent itemsets because of the suppression of the itemsets candidates which are maximum frequent subsets of these itemsets. These two problems, which require significant computing times CPU, involve in certain cases of the definitely higher response times for the algorithm Pincer-Search compared to the algorithm Max-miner.

The MaxEclat algorithm is subjected to the problem of the lack of precision of the itemsets maximum candidates that it uses. These candidates, who are generated by combining the identical frequent itemsets except for their last item, are supersets of the maximum frequent itemsets whose many items must be removed to obtain the maximum frequent itemsets. Consequently, the MaxEclat algorithm must carry out more iterations that the algorithms Max-Miner and Pincer-Search to identify the maximum frequent itemsets. Each iteration corresponding to a swaping of the context and the calculation of the support of all the candidates, these iterations represent a considerable difference in the execution times.

The MaxClique algorithm is subjected to a problem of performance due to the computing times necessary for the generation of the itemsets maximum candidates that it uses.It generates the click maximum graph whose tops are the frequent items and the arcs are the frequent k-itemsets. The itemsets resulting, which are supersets of the maximum frequent itemsets, are the itemsets maximum candidates used for the research top downwards. The problem of the enumeration of click maximum of a graph being a Np-difficult problem, the number of operations

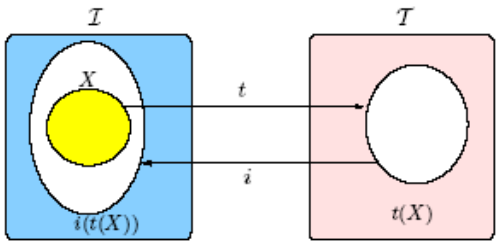
necessary to their generation is very significant and the response times of the algorithm are degraded considerably when the frequent itemsets (and thus click them) are long.
The results of the experiments of comparison algorithm Max-Miner with the algorithms of extraction of the frequent itemsets show that it makes it possible to reduce times of extraction of the frequent itemsets in the case of values of μ high, i.e. if longest itemsets frequent maximum contains a significant number of items. The experiments of the algorithms Close and A-Closed also shows that these two algorithms reduce times of extraction of the frequent itemsets for such data files. However, the algorithm Max-Miner requires a significant quantity of memory in order to store information concerning the candidates groups of each iteration. Moreover, this algorithm is subjected to the problem of the extraction of rules of association starting from dense and strongly correlated data related to the average size of the frequent itemsets which is high for the data files of this type.

2.3. CLOSED FREQUENT ITEMSET (CF)

Example :

$\mathcal{D}$	
Tid	Items
1	$x_1, x_2, \ldots, x_{100}$
2	$x_1, x_2, \ldots, x_{50}$

Let us consider the example above very simple compromise base. For a minimum support = 1, the process of extraction of the frequent itemsets generates 2^{100} -1 is nearly 10^{30} frequent itemsets, and consequently a very great number of rules of association. An alternative to the extraction of all the frequent itemsets was proposed by Pasquier and Al in [10].The idea consists in extracting a subset from itemsets frequent <closed> which constitutes <a minimal whole generating non redundant for all the frequent itemsets and their supports>. In other words, to extract a whole of itemsets frequent, known as closed, with their frequencies. These closed itemsets are an condensed representations of all the frequent itemsets. Thus, from closed, one can deduce the support from any frequent itemset without resorting to the course of the base of transactions.
Intuitively, an itemset is known as to close if it does not have any superset with the same support.
Thus in the example, only the itemsets which will be generated is the closed frequent itemset:{x1;x2;:::;x100} and {x1;x2;:::;x50} and only the rule of association:
x1,x2…,x50 → x51,x52…,x100 will be produced. All the other possible rules can be easily derived starting from this rule.
This approach not only makes it possible to improve the effectiveness of the extraction but also to reduce considerably the number of redundant rules which submerge the interesting rules. Three principal algorithms were proposed for the extraction of closed: Closed [10] A-closed [11], Charm [15], Closet [6] and Closet+ [7].



Closing of Item X

The algorithms Closet and Charm are similar in so far as they initially explore all the two frequent itemsets closed by an in-depth approach, with the difference that Closet is based on a tree (FP-TREE) which compresses the transactions of D.
In addition, a concept similar to the closed reason called key reason was proposed by Bastide and Al in the algorithm Pascal [4], like by Boulicaut and Al in the Min-Ex algorithm where it bears the free name of reason and a δ-free way more extent.

Close, A-Close et Close+ [9]
- During each iteration, ***the Close algorithm*** considers a whole of k-itemsets generating. It builds a whole of itemsets closed candidates who are closings of these k-generators and it then determines among these candidates the frequent itemsets closed according to the minimal threshold of support minsupport. Finally, it creates to them (k+1)-generators which will be used at the time of the following iteration in order to build the whole of itemsets closed candidates who are closings of (the k+1)-generators a swaping of the context of extraction is necessary during each iteration, in order to determine closings of the K-generators and to calculate their supports.
- the algorithm ***A-Closed*** starts by determining the frequent generating 1-itemsets. Then, during each iteration K, it generates a whole of K-generators candidates starting from (the k-1)-itemsets generating frequent. It determines the supports of these k-itemsets generating candidates and removes the non minimal generators infréquents and generators which are identified according to their supports. When all the frequent generators are determined, their closings which constitute them itemsets closed frequent is given. During each iteration, a swaping of the context is carried out in order to calculate the support of the K-generators candidates. An ultimate swaping of the context is then carried out at the time of the determination of closings of all the frequent generators.
- ***the Close+*** algorithm, makes it possible to determine the frequent itemsets closed among the whole of the frequent itemsets, without reaching the context of extraction. It makes it possible to extend a current implementation of extraction of the itemsets frequent, without having to modify it, in order to generate the frequent closed itemsets. The frequent k-itemsets which are frequent closed k-itemsets are identified by comparing their supports with the supports of their (k+1)-on sets.
This algorithm does not carry out swaping because it applies to the whole of the frequent itemsets and their already extracted supports.

Discussion

Several algorithms of extraction of the sets closed starting from a finished binary relation were proposed in the literature. Among the latter, we can quote the algorithm of Bordat, the algorithm of Carpineto and the algorithm of Ganter implemented in ConImp which is most general since it makes it possible to calculate the closed sets some is the operator of the closing used and most effective among these three algorithms. However, these algorithms are not applicable in the context of the KDD because they make it possible to calculate the sets closed in reasonable times only for contexts of extraction comprising with more few tens of attributes (items) and a few hundreds of objects. The contexts of extraction of the KDD are made up for the majority of several hundreds to several thousands of attributes and several tens of thousands to several million objects. These algorithms requiring in best case as many swapings of the context of extraction than there are sets closed in the context, they cannot be used within the framework of the KDD. Moreover, they do not take into account the support of the itemsets in order to limit the space of research and determine all the closed itemsets of which a significant proportion have weak supports and are thus not significant for the applications of the KDD. The algorithms Aprem and Impec also make it possible to calculate the closed sets some is the operator of closing used and it was shown that these algorithms are more effective than the algorithm of Ganter in terms of applicability and execution time since it can be used for relations of more significant sizes. However these algorithms have summers developed within a framework other than the KDD in order to solve problems of nature different from those involved in the field of the KDD and their application to this field poses problems of performances.

The methods used by the algorithms Close and A-Closed have several other significant advantages compared to the methods used by the algorithms of Bordat, Carpineto and Ganter. They make it possible on the one hand as far as possible to limit the number of swapings of the context necessary: the algorithms Close and A-Closed require a number of swapings equal to the size of largest of the generators of the frequent closed itemsets, increased by one for A-Closed. In addition, the generators being minimal within the meaning of inclusion and thus of the size, these two algorithms limit as much as possible the costs in time CPU of the operations on the itemsets, more particularly the tests of inclusions and the intersections, which depend directly on the size of the itemsets.

The experiments show that in many cases the algorithms Close and A-Closed make it possible to decrease times of extraction of the frequent itemsets and, for the algorithm Close, the memory capacity necessary to the extraction what increases the field of application by it. This is more particularly true for the dense and/or correlated data which represent a significant share of the existing data bases. Moreover, the frequent closed itemsets make it possible to generate the whole of the valid rules of association or many bases for the valid rules of association.

The problem of the determination of the approach and thus of the algorithm which will be most effective according to the data file used is complex. In the case of not correlated and scattered data, approach of the Apriori algorithm, i.e. the course of the lattice of the itemsets, makes it possible to obtain better response times. However, the extraction of the frequent itemsets starting from this type of data with the algorithms Closed and A-Closed, by the course of the lattice of the closed itemsets, gives acceptable response times. In the case of correlated and/or dense data, the Apriori algorithm gives response times very significant and quite higher than those of the algorithms Close and A-Closed. The nature of the data which constitute the data file thus makes it possible to evaluate a priori the most effective algorithm, the properties of correlation and density of many types of data having been studied in the literature. Thus, it was established that the data of sales of supermarkets scattered and are slightly correlated bus in this type of data, the average number of items by objects is weak in front of the total number of items and each item is contained only in one small number of objects. It was also established that a significant proportion of the real data bases consist of correlated or dense data. They are the statistical and space data, the collections of texts and images, the histories of access Internet, etc.

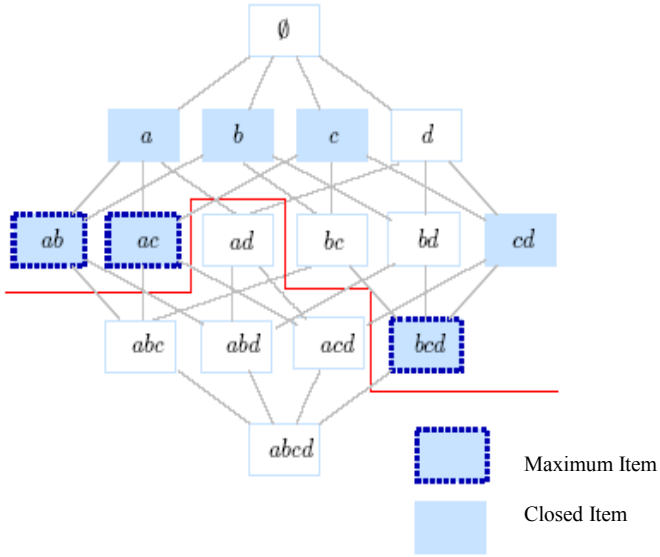
It is also possible to evaluate the effectiveness of the Close algorithms and A-Closed compared to the Apriori algorithm for a data file by determining the frequent proportion of 1-itemsets which is closed. This proportion constitutes an effective indicator of the proportion total of itemsets frequent which is closed and so almost all the frequent 1-itemsets are closed, it is probable that almost all the frequent itemsets are closed and thus calculations of closings realized by Close and A-Closed.

3. LET US SUMMARIZE . . .

The phase of extraction of the frequent itemsets remains an expensive phase in time and memory capacity and very often generates a very significant number of frequent itemsets.

Instead of considering all the frequent reasons one can consider only the closed frequent reasons or the maximum frequent reasons whose principal algorithms were quoted higher in this paragraph. Also, we have the relation of following inclusion between the categories of frequent:

$$MF \subseteq CF \subseteq F$$



It is stressed that in the dense data bases, the number of itemsets closed and maximum is definitely lower than the number of itemsets, from where interest to consider only the itemsets closed or maximum during the generation of the rules of association.
The figure shows the lattice of the itemsets for I={a;b;c;d}, closed and maximum frequent itemsets. Thus, we have 10 frequent itemsets, 7 frequent closed and 3 maximum.

4. FUTURE WORKS

The results of this work presented in this article offer several prospects for later research at the theoretical level and the practical level. In this section, we briefly present some of these prospects which appear to us to be most interesting.

Techniques of implementation and structures of data

One can note during the implementations and the experiments of various algorithms that the structures of data and the techniques of implementation used have a particularly significant influence on the times execution and the memory capacity necessary to the execution of the algorithms. The study of the various techniques of implementation and the structures of data making it possible to improve the tasks of the extraction of knowledge in the data bases, according to their properties and of the properties of the various types of data, thus appears to us to be particularly significant for the resolution of the problems of the KDD.
The preliminary results of this study suggest that the structures of the data most effective are the structures of bitmaps, with the simple bitmaps and the hierarchical bitmaps, as well as the tree structures, with the trees of chopping, the trees of prefixes and the trees ternary.

Incremental maintenance of the whole of the frequent itemsets (F, MF, CF)

The incremental maintenance of the whole of the frequent itemset consists of reflected updates of the data of the data base (addition of a new object, a new attribute or new values of the attributes) on this unit. This approach is particularly useful for all the tasks of the KDD which can be realized starting from this whole (extraction of rules of association and time series, clustering and supervised classification) in the case of applications requiring of the frequent executions of these tasks. Operators of construction and incremental maintenance of the frequent lattices of concept, to which operators of construction of clusters are applied, were established in the DBMS directed objects O2.The experimental results show that this approach poses problems of performances in the case of data files of big sizes and the development of an effective algorithm of incremental maintenance of the whole of the frequent closed itemsets constitutes an interesting prospect for later work.

REFERENCES

[1] Agrawal Rakesh, Ramakrishnan Srikan
" Fast Algorithms for Mining Association Rules ".
1994.

[2] Bayardo Jr Roberto J.
"Efficiently Mining Long Patterns From DataBases "
1998.

[3] Brin Sergey, Rajeev Motwani, Jeffrey D.Ullman, Shalon Tsur.
" Dynamic Itemset Counting and Implication Rules for Market Basket Data. " 1997

[4] Bastide Yves, Rafik Taouil, Nocolas Pasquier, Gerd Stumme et Lotfi Lakhal
" PASCAL : Un algorithme d'extraction des motifs fréquents "

[5] Han Jiawi, Jina Pei, Yiwen Yin, Runying Mao,
" Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach "
2001

[6] Han Jiawi, Jina Pei et Runying Mao
" CLOSET: An efficient Algorithm for mining frequent closed itemsets " 2000

[7] Jianyong Wang, Jiawei Han, Jian Pei
" CLOSET+: Searching For The Best Strategies For Mining Fréquent Closet Itemsets " 2003

[8] Lin Dao-I, Zvi M. Kedem
" Pincer-Search: An Efficient Algorithm for discovering the Maximum Fréquent Set " July 1999.

[9] Nicolas Pasquier
" Data Mining : Algorithmes d'extraction et réduction des règles d'association dans les bases de données "
Thesis of doctorate January 2000

[10] Nicolas Pasquier, Yves Bastide, Rafik Taouil and Lotfi Lakhal
" Efficient Mining of association rules using closed itemset lattices "

[11] Nicolas Pasquier, Yves Bastide, Rafik Taouil and Lotfi Lakhal
"Discovering Frequent Closed Itemsets for Association Rules "

[12] Savasere Ashok, Adward Omiecinski, Shamkan Navathe.
" An Efficient Algorithm for Mining association Rules in Large Databases. " 1995.

[13] Toivonen Hannu
" Sampling Large databases for Association Rules "

[14] Zaki .M. J. et Gouda (K.)

" Efficiently mining maximal frequent itemsets"
Novembre 2001
[15] Zaki Mohammed .J, Ching Jui Hsiao
" Charm: An Efficient Algorithm For Closed Itemset
Mining " 2002
[16]: Zaki Mohammed Javeed, Srinivasan Parthasarathy,
Mitsunori Ogihara and Wei Li
" New Algorithms For Fast Discoverey of
Association Rules " July 1997.