Composition of Mining Contexts for Efficient Extraction of Association Rules

C. T. Diop[1,3], A. Giacometti[1], D. Laurent[1], N. Spyratos[2]

October 1, 2001

[1] LI - Université de Tours - 41000 Blois FRANCE
{giaco,laurent}@univ-tours.fr
[2] LRI - Université Paris 11 - 91405 Orsay Cedex FRANCE
spyratos@lri.fr
[3] Université Gaston Berger - Saint-Louis SENEGAL
cdiop@ugb.sn

Abstract

Association rule mining often requires the repeated execution of some extraction algorithm for different values of the support and confidence thresholds, as well as for different source datasets. This is an expensive process, even if we use the best existing algorithms. Hence the need for *incremental* mining, whereby mining results already obtained can be used to accelerate subsequent steps in the mining process.

In this paper, we present an approach for the incremental mining of *multi-dimensional* association rules, i.e., association rules that involve one or more tables from a given relational database. In our approach, association rule mining takes place in a *mining context* which specifies the form of rules to be mined.

More precisely, a mining context specifies the objects of interest, e.g., AllCustomers, as well as attributes of these objects, e.g., CustJob and ItemBought and conditions on those attributes, e.g. CustJob = Professor or Lawyer, ItemBought = Tea or Bread. Within a context, a mining task consists in discovering all association rules whose support and confidence are above user-defined thresholds.

Incremental mining is obtained by combining mining contexts using relational algebra operations. More precisely, we show how to accelerate the computation of a mining task when its context is defined by composition of contexts of already executed mining tasks.

Key words: Association rule, Relational algebra, Mining context, Incremental computation.

1 Introduction

Association rule mining often requires the repeated execution of some extraction algorithm for different values of the support and confidence thresholds, as well as for different source datasets. This is an expensive process, even if we use the best existing algorithms. Hence the need for *incremental* mining, whereby mining results already obtained can be used to accelerate subsequent steps in the mining process.

In this paper, we present an approach for the incremental mining of *multi-dimensional* association rules, i.e., association rules that involve one or more tables from a given relational database [11]. In our approach, association rule mining takes place in a *mining context* which specifies the form of rules to be mined. Incremental mining is obtained by combining mining contexts using relational algebra operations.

1.1 Association Rules and Mining Contexts

Different formalisms can be used to represent multi-dimensional association rules and specify mining tasks. Whereas most of these formalisms [6, 11, 20] are based on Dalalog and first-order logic, we shall use relational algebra to define association rules and mining tasks.

In our approach, a mining context, or simply a context, specifies the form of rules to be mined for a data mining task. More precisely, a context specifies the objects of interest, as well as attributes of these objects and conditions on those attributes. Let us explain these concepts through an example that will serve as a running example throughout the paper.

Example 1 *Consider a database $DBSales$ containing the following tables:*

- $Cust(Cid, Cjob, Caddr)$*, where* Cid*,* $Cjob$ *and* $Caddr$ *are the ids, jobs and home addresses of customers, respectively,*

- $Prod(Pid, Ptype)$*, where* Pid *and* $Ptype$ *are the ids and the types of products, respectively,*

- $Store(Sid, Sname, Saddr)$*, where* Sid*,* $Sname$ *et* $Saddr$ *are the ids, names and addresses of stores, respectively,*

- $Sales(Cid, Pid, Sid, Date)$*, where a tuple* $\langle c, p, s, d\rangle$ *in the Sales table means that customer* c *issued a transaction concerning product* p *in the store* s *on date* d*.*

Assuming that the objects of interest are all customers whose ids are stored in the Cust table, we consider the relational expression $AllCust = \pi_{Cid}(Cust)$*.*
Suppose now that the attributes of interest for customers are their jobs, addresses and the types of products they buy. We can specify this by a relational expression, call it B_1*, that is the join of the tables Cust, Sales and Prod, i.e.* $B_1 = Cust \bowtie Sales \bowtie Prod$*. Indeed,* B_1 *involves the attributes just mentioned, i.e.* $Cjob$*,* $Caddr$ *and* $Ptype$*.*
Moreover, if we are only interested in association rules dealing with customers whose job is professor or lawyer, and who buy tea or milk, then we can consider the set of selection conditions Σ_1 *defined by* $\Sigma_1 = \{Cjob = Professor, Cjob = Lawyer, Ptype = Tea, Ptype = Milk\}$*.*

The triple $\mathcal{C}_1 = \langle AllCust, B_1, \Sigma_1\rangle$ *is an example of what we call a* mining context*, or simply a* context*. Moreover,* $AllCust$*,* B_1 *and* Σ_1 *are called the* reference*, the* basis *and the* domain *of* $\mathcal{C}_1$*, respectively. Given the context* $\mathcal{C}_1 = \langle AllCust, B_1, \Sigma_1\rangle$*, the following rule is an example of association rule that can be considered:*

$$AllCust \bowtie \sigma_{Cjob=Professor}(B_1) \Rightarrow AllCust \bowtie \sigma_{Ptype=Tea}(B_1)$$

This rule means intuitively that if a customer is a professor, then he (or she) buys tea. More formally, since Cid *is the attribute of the reference* $AllCust$*, we consider that this rule is satisfied in an instance* I *of* $DBSales$ *if* $(\pi_{Cid}(AllCust \bowtie \sigma_{Cjob=Professor}(B_1)))(I) \subseteq (\pi_{Cid}(AllCust \bowtie \sigma_{Ptype=Tea}(B_1)))(I)$*, i.e. if every identifier of customer in the answer to* $\pi_{Cid}(AllCust \bowtie \sigma_{Cjob=Professor}(B_1))$ *in* I*, is also present in the answer to* $\pi_{Cid}(AllCust \bowtie \sigma_{Ptype=Tea}(B_1))$ *in* I*.*

More generally, a context is a triple $\mathcal{C} = \langle R, B, \Sigma\rangle$ where R and B are two relational expressions and Σ is a set of selection conditions. Given a context $\mathcal{C} = \langle R, B, \Sigma\rangle$, we consider two types of patterns:

1. A *query of* $\mathcal{C}$ is a query of the form $R \bowtie \sigma_S(B)$ where S is a conjunction of selection conditions in Σ.

2. An *association rule of* $\mathcal{C}$ is an expression of the form $q_1 \Rightarrow q_2$ where q_1 and q_2 are queries of $\mathcal{C}$. Moreover, we say that a rule $q_1 \Rightarrow q_2$ is satisfied in an instance I if $(\pi_K(q_1))(I) \subseteq (\pi_K(q_2))(I)$, where K is the set of attributes of the reference table R.

Similarly to traditional approaches, interestingness of association rules is evaluated according to two thresholds, namely the support and confidence thresholds. More precisely, let $\mathcal{C} = \langle R, B, \Sigma \rangle$ be a context and let I be an instance of the database. Denoting by K the set of attributes of the reference R, the interestingness in I of association rules of $\mathcal{C}$ is defined as follows:

1. Let q be a query of $\mathcal{C}$. The *support* of q in I is the ratio $|(\pi_K(q))(I)|/|R(I)|$. If this ratio is greater than a given support threshold, then we say that q is *frequent* in I.

2. Let $q_1 \Rightarrow q_2$ be a rule of $\mathcal{C}$. The *confidence* of $q_1 \Rightarrow q_2$ in I is the ratio $|(\pi_K(q_1) \cap \pi_K(q_2))(I)|/|(\pi_K(q_1))(I)|$. If this ratio is greater than a given confidence threshold, and if q_1 and q_2 are frequent in I, then we say that $q_1 \Rightarrow q_2$ is *interesting* in I.

As in other approaches, given a context $\mathcal{C}$, a minimum support threshold and a minimum confidence threshold, the problem is to find all interesting association rules of $\mathcal{C}$. This problem can be solved in a two-steps process:

1. Find all frequent queries of $\mathcal{C}$.

2. Generate all interesting rules of $\mathcal{C}$ from the frequent queries in $\mathcal{C}$.

The overall performance of the mining task is determined by the first step. Clearly, in implementing our approach, we can use any of the existing level-wise algorithms [1, 3, 17] to compute the frequent queries of a context. In the following, we show how the computation of the frequent queries of a context using these algorithms can be accelerated, assuming that the results of previously computed tasks are stored.

1.2 Composition of Mining Contexts

As mentioned earlier, the objective of this paper is to reuse results of already executed mining tasks in order to mine more efficiently the interesting rules of new mining tasks. More precisely, the main contribution of this paper is to show the following: Suppose that a number of tasks have already been performed and that their frequent queries have been stored. Then any new mining task, whose context is defined by composition of the contexts of old tasks, can be performed more efficiently than if this new mining task had been defined without reference to the old tasks. To this end, we define operations over contexts similar to those of the relational algebra, and we study their properties. The following example shows how contexts are composed.

Example 2 *In the database DBSales of Example 1, assume that we want to discover the relationships between the jobs of customers living in Paris and the types of products they buy. Using the context* $\mathcal{C}_1$ *given previously, a context* $\mathcal{C}_1^*$*, denoted by* $\sigma^b_{Caddr=Paris}(\mathcal{C}_1)$*, can be defined by applying the selection condition* $(Caddr = Paris)$ *on the basis of* $\mathcal{C}_1$*. That is,* $\mathcal{C}_1^* = \langle AllCust, \sigma_{Caddr=Paris}(B_1), \Sigma_1 \rangle$*.*

Given the new context $\mathcal{C}_1^*$, let $q_1^* = AllCust \bowtie \sigma_S(\sigma_{Caddr=Paris}(B_1))$ be a query of $\mathcal{C}_1^*$. We can see easily that the answer to q_1^* is always included in the answer to the query $q_1 = AllCust \bowtie \sigma_S(B_1)$ of $\mathcal{C}_1$. More generally, it can be seen that every query q_1^* of $\mathcal{C}_1^*$ is contained (in the sense of query containment [21]) in a query q_1 of $\mathcal{C}_1$.
Therefore, the support of every query q_1^* of $\mathcal{C}_1^*$ is less than or equal to the support of a query q_1 of $\mathcal{C}_1$. It follows that if a query q_1^* of $\mathcal{C}_1^*$ is frequent, then it is contained in a frequent query q_1 of $\mathcal{C}_1$. Conversely, if a query q_1^* of $\mathcal{C}_1^*$ is not contained in a frequent query q_1 of $\mathcal{C}_1$, then it can not be frequent.
As a consequence, if we assume that the frequent queries of $\mathcal{C}_1$ are stored, this property can be used to prune some of the candidate queries of $\mathcal{C}_1^*$ when computing its frequent queries. Moreover, this new pruning step can be done without any access to the database, implying that the computation of the frequent queries of $\mathcal{C}_1^*$ can be accelerated.

Suppose now that we are interested in rules relating the jobs of customers and the names of the stores where they issue their transactions. If we are only interested in rules dealing with professors or lawyers, and concerning stores called $Carrefour$ or $Ikea$, then we can define the context $\mathcal{C}_2 = \langle AllCust, B_2, \Sigma_2\rangle$, with the same reference as $\mathcal{C}_1$, but where $B_2 = Cust \bowtie Sales \bowtie Store$ and $\Sigma_2 = \{Cjob = Professor, Cjob = Lawyer, Sname = Carrefour, Sname = Ikea\}$. Let us assume that the frequent queries of $\mathcal{C}_2$ have already been computed and stored.
Then, based on $\mathcal{C}_1$ and $\mathcal{C}_2$, a new context $\mathcal{C}_{12}$ can be defined by taking the join of the bases and the union of the domains of $\mathcal{C}_1$ and $\mathcal{C}_2$. In other words, $\mathcal{C}_{12}$ is defined as follows:

$$\mathcal{C}_{12} = \langle AllCust, B_1 \bowtie B_2, \Sigma_1 \cup \Sigma_2\rangle.$$

This new context allows to consider the relationships between the jobs of customers, the types of products they buy and the names of stores where their transactions are issued. As we shall see later, the computation of the frequent queries of $\mathcal{C}_{12}$ can be optimized, based on the sets of frequent queries of $\mathcal{C}_1$ and $\mathcal{C}_2$.

In the remaining of this paper, we show how the example above can be generalized and how the computation of frequent queries in composed contexts can be optimized.

1.3 Related Works

So far, many SQL based query languages have been proposed in order to facilitate the specification of mining tasks [9, 10, 20], but to our knowledge, none of these languages allows to combine such tasks based on properties of the relational algebra. We can also note that our notion of context is closed to the notion of query flock introduced in [20]. However, the objective of the approach presented in [20] is not to store and use the results of mining tasks in order to optimize new mining tasks, but rather to optimize the computation of one mining task using query containment.

On the other hand, storing and using the results of mining tasks in order to optimize new mining tasks has been studied in [14, 15, 16]. However, these approaches compare only mining tasks defined over a single reference and a single basis (using our terminology), and they do not propose any composition operations as we do.

In [14, 15], the authors consider the traditional case where the source database contains only one table (the basis in our terminology), whose rows contain sets of items.
In [14], mining tasks are compared with respect to three criterions: the support thresholds, the selection conditions on the source dataset and the constraints on the patterns. Moreover, the authors use materialized views to store the results of mining tasks already computed.

In [15], mining tasks are mainly compared with respect to their support thresholds, and the authors propose to store partially the results of mining tasks. More precisely, assuming that the size of the cache storing the results of mining tasks is limited, the authors present different caching strategies aiming at storing only the most useful frequent itemsets that can be used to answer new mining tasks. In [16], the same authors present an extension of their approach, whereby the source database is not the traditional table, but a data cube. Moreover, in [16], mining tasks are not only compared with respect to their supports, but also with respect to group-by conditions on the data cube.

In this paper, we dot not compare mining tasks with different support thresholds. We note that our formalism allows for such comparisons. However, we do not consider them in the present paper because we focus on the composition of mining contexts. Hence, we consider the case where the source database contains *multiple tables.* It follows that we can specify and compare mining tasks with *different basis* or with *different references.* We recall that these comparisons allow for optimizations when computing the answers to mining tasks. Finally, in our storage technique, only the positive boundaries of the results of mining tasks can be stored.

The rest of the paper is organized as follows: In Section 2, we formally define the notion of mining context that have been roughly introduced previously, and we provide definitions and properties for the measures of support and confidence. The operations for combining mining contexts are studied in Section 3, and in Section 4, we show how to use the properties given in Section 3 in order to improve the efficiency of association rule discovery. Section 5 concludes the paper and outlines further research based on this work. Proofs are omitted due to a lack of space; the interested reader is referred to [8] for more details.

2 Mining Contexts and Mining Tasks

2.1 Definitions and Notations

The formalism used in this paper is based on the relational model of databases ([21]). We recall that a relational database schema is a set of relation names, each of which is associated with a set of attributes. We call schema of a relation R the set of attributes associated with R and we denote it by $sch(R)$. For each attribute A, the possible values for A belong to a specific set of values, called the domain of A and denoted by $dom(A)$.

In the remainder of this paper, we assume a fixed database schema, $DB = \{R_1, R_2, \ldots, R_n\}$, and we call instance of DB any set of relations $I = \{r_1, r_2, \ldots, r_n\}$, where r_i is a relation over $sch(R_i)$, for every $i = 1, 2, \ldots, n$. Given a relational expression, or a query q, over DB, we call schema of q the set of attributes over which q is defined, and we denote it by $sch(q)$. Moreover, we denote by $q(I)$ the answer to q in I.

As in [4], if q_1 and q_2 are two queries such that $sch(q_1) = sch(q_2)$, we say that q_1 is *contained* in q_2, or that q_1 is *more specific* than q_2, denoted by $q_1 \sqsubseteq q_2$, if for every instance I we have $q_1(I) \subseteq q_2(I)$. Queries q_1 and q_2 are said *equivalent,* denoted by $q_1 \equiv q_2$, if both $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$ hold.

2.2 Mining Contexts

In our approach, the form of rules to be mined for a data mining task is specified by a *mining context.*

Definition 1 - Mining Context. *A* mining context $\mathcal{C}$*, or simply a* context*, is a triple* $\mathcal{C} = \langle R, B, \Sigma \rangle$ *where:*

- *R is a relational expression, called the* reference *of $\mathcal{C}$.*
- *B is a relational expression such that $sch(R) \subseteq sch(B)$, called the* basis *of $\mathcal{C}$.*
- *Σ is a set of atomic selection conditions, called the* domain *of $\mathcal{C}$. These selection conditions are of the form $(A = a)$ where A is an attribute of B not present among the attributes of R, and a is a constant in $dom(A)$.*

We note that the notion of reference is very important, since it specifies the values that are subject to counting when searching for frequent queries (see Definition 3). Note that our notion of reference is similar to that of *key atom* introduced in [6].

We now define the kinds of patterns and rules that we are interested in.

Definition 2 - Search Space. *Given a context $\mathcal{C} = \langle R, B, \Sigma \rangle$*

1. *Σ^* denotes the set of all selection conditions of the form $(A_1 = a_1) \wedge \ldots \wedge (A_n = a_n)$ where $(A_i = a_i)$ with $i = 1 \ldots n$ is an atomic selection condition in Σ.*
2. *$\mathcal{Q}(\mathcal{C})$ denotes the set of all queries of the form $R \bowtie \sigma_S(B)$ where $S \in \Sigma^*$. Queries in $\mathcal{Q}(\mathcal{C})$ are called* candidate queries of $\mathcal{C}$, *or simply queries of $\mathcal{C}$.*
3. *$\mathcal{R}(\mathcal{C})$ denotes the set of all rules of the form $q_1 \Rightarrow q_2$ where $q_1 = R \bowtie \sigma_{S_1}(B)$ and $q_2 = R \bowtie \sigma_{S_1 \wedge S_2}(B)$ are queries of $\mathcal{C}$. Rules of $\mathcal{R}(\mathcal{C})$ are called* candidate rules of $\mathcal{C}$, *or simply rules of $\mathcal{C}$. Moreover, we say that a rule $q_1 \Rightarrow q_2$ is* satisfied *in an instance I if $(\pi_K(q_1))(I) \subseteq (\pi_K(q_2))(I)$, where $K = sch(R)$.*

As $S_1 \wedge S_2$ is stronger than S_1, we always have $\pi_K(q_2) \sqsubseteq \pi_K(q_1)$. As a consequence, the rule $q_1 \Rightarrow q_2$ is satisfied in I if and only if $(\pi_K(q_1))(I) = (\pi_K(q_2))(I)$. On the other hand, we could have considered more general rules of the form $q_1 \Rightarrow q_2$ where $q_1 = R \bowtie \sigma_{S_1}(B)$ and $q_2 = R \bowtie \sigma_{S_2}(B)$ are queries of $\mathcal{C}$. However, as we shall see in Section 2.3, the confidence of such rules would not be the support of $q_1 \cap q_2$ devided by the support of q_1.

2.3 Support and Confidence

All standard measures of interest for association rules as given in [12], can also be defined in our approach. For the purposes of this paper, however, we are only interested in the support and confidence of a rule.

Definition 3 - Support of a Query. *Let $\mathcal{C} = \langle R, B, \Sigma \rangle$ be a context and q be a query such that $sch(R) \subseteq sch(q)$. For every instance I, the* support of q relatively to $\mathcal{C}$ and I, *denoted by $Sup(q \mid \mathcal{C}, I)$, is the following ratio:*

$$Sup(q \mid \mathcal{C}, I) = \frac{|(\pi_K(q))(I)|}{|R(I)|}, \text{ where } K = sch(R).$$

Given a support threshold α, we denote by $freq_\alpha(\mathcal{C}, I)$ the set of all queries in $\mathcal{Q}(\mathcal{C})$ whose support relatively to $\mathcal{C}$ and I is greater than or equal to α, i.e.

$$freq_\alpha(\mathcal{C}, I) = \{q \in \mathcal{Q}(\mathcal{C}) \mid Sup(q \mid \mathcal{C}, I) \geq \alpha\}.$$

*Queries in $freq_\alpha(\mathcal{C}, I)$ are called α-*frequent *in I, or simply* frequent *if α and I are understood.*

The following proposition states that the support as defined above is monotonic with respect to query containment, thus generalizing the basic property of monotonicity used for standard association rules in [1].

Proposition 1 *Let $\mathcal{C}_1 = \langle R_1, B_1, \Sigma_1\rangle$ and $\mathcal{C}_2 = \langle R_2, B_2, \Sigma_2\rangle$ be two contexts such that $K = sch(R_1) = sch(R_2)$, let q_1 and q_2 be queries of $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively.*
If $R_1 \sqsubseteq R_2$ and $\pi_K(q_2) \sqsubseteq \pi_K(q_1)$, then for every instance I, we have:

$$Sup(q_2 \mid \mathcal{C}_2, I) \leq Sup(q_1 \mid \mathcal{C}_1, I).$$

Based on this proposition and using properties of the relational algebra, we can easily show the following facts.

Corollary 1 *Let $\mathcal{C}_1 = \langle R_1, B_1, \Sigma_1\rangle$ and $\mathcal{C}_2 = \langle R_2, B_2, \Sigma_2\rangle$ be two contexts such that $K = sch(R_1) = sch(R_2)$, let q_1 and q_2 be two queries of $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, and let S be a selection condition in Σ_1^*. Then for every instance I, we have:*

1. $Sup(\sigma_S(q_1) \mid \mathcal{C}_1, I) \leq Sup(q_1 \mid \mathcal{C}_1, I)$,
2. $Sup(q_1 \bowtie q_2 \mid \mathcal{C}_i, I) \leq Sup(q_i \mid \mathcal{C}_i, I)$ *for* $i = 1, 2$.

Moreover, if $sch(q_1) = sch(q_2)$ then we also have:

3. $Sup(q_1 \cap q_2 \mid \mathcal{C}_i, I) \leq Sup(q_i \mid \mathcal{C}_i, I)$ *for* $i = 1, 2$,
4. $Sup(q_2 - q_1 \mid \mathcal{C}_2, I) \leq Sup(q_2 \mid \mathcal{C}_2, I)$,
5. $Sup(q_2 \cup q_1 \mid \mathcal{C}_i, I) \geq Sup(q_i \mid \mathcal{C}_i, I)$ *for* $i = 1, 2$.

Given a context $\mathcal{C} = \langle R, B, \Sigma\rangle$, the confidence of an association rule of $\mathcal{C}$ is defined as follows.

Definition 4 - Confidence of a Rule. *Let $\mathcal{C} = \langle R, B, \Sigma\rangle$ be a context and let q_1 and q_2 be two queries of $\mathcal{C}$. For every instance I, the* confidence of the rule $\mathcal{R}: q_1 \Rightarrow q_2$ relatively to $\mathcal{C}$ and I, *denoted by $Conf(\mathcal{R} \mid \mathcal{C}, I)$, is the following ratio:*

$$Conf(\mathcal{R} \mid \mathcal{C}, I) = \frac{|(\pi_K(q_1) \cap \pi_K(q_2))(I)|}{|(\pi_K(q_1))(I)|}, \text{ where } K = sch(R).$$

Given a support threshold α and a confidence threshold β, we denote by $int_{\alpha,\beta}(\mathcal{C}, I)$ the set of all rules $q_1 \Rightarrow q_2$ in $\mathcal{R}(\mathcal{C})$ such that q_1 and q_2 are frequent queries in $freq_\alpha(\mathcal{C}, I)$, and $Conf(q_1 \Rightarrow q_2 \mid \mathcal{C}, I) \geq \beta$, i.e.

$$int_{\alpha,\beta}(\mathcal{C}, I) = \{q_1 \Rightarrow q_2 \in \mathcal{R}(\mathcal{C}) \mid q_1, q_2 \in freq_\alpha(\mathcal{C}, I) \text{ and } Conf(q_1 \Rightarrow q_2 \mid \mathcal{C}, I) \geq \beta\}.$$

Rules in $int_{\alpha,\beta}(\mathcal{C}, I)$ are called (α, β)-interesting *in I, or simply* interesting *if α, β and I are understood.*

The following proposition states basic properties of the confidence.

Proposition 2 *Let $\mathcal{C} = \langle R, B, \Sigma\rangle$ be a context and let $\mathcal{R}: q_1 \Rightarrow q_2$ be a rule in $\mathcal{R}(\mathcal{C})$. Then, for every instance I:*

1. $Conf(\mathcal{R} \mid \mathcal{C}, I) = 1$ *if and only if $\mathcal{R}$ is satisfied in I.*

2. $Conf(\mathcal{R} \mid \mathcal{C}, I) = Sup(q_2 \mid \mathcal{C}, I)/Sup(q_1 \mid \mathcal{C}, I)$.

Note that the first part of Proposition 2 holds for any rule, whereas the second part holds only if q_2 is more specific than q_1 (which is true for every rule of a context). We point out in this respect that the second part of Proposition 2 is important from a computational point of view: it shows that the confidence of a rule can be computed without accessing the database (assuming that the supports of all frequent queries are available). Therefore, when we extract interesting association rules, the most important task is to compute the set of all frequent queries. In the following section, we shall see how context composition can optimize the computation of frequent queries.

3 Composition of Contexts

The operations define in this section are based on those of the relational algebra ([21]) and allow for the definition of new tasks from "old".

3.1 Manipulation and Comparison of Sets of Queries

First, we extend the usual operators of the relational algebra to sets of queries over the same schema, and we define a new operator that we call *reduction*.

Definition 5 - Operations on Sets of Queries. *Let $\mathcal{Q}$ be a set of queries over the same schema Sch.*

1. *Let X be a subset of Sch, the* projection of $\mathcal{Q}$ over X, *denoted by $\pi_X(\mathcal{Q})$, is defined by: $\pi_X(\mathcal{Q}) = \{\pi_X(q) \mid q \in \mathcal{Q}\}$.*
2. *Given a selection condition S containing only attributes from Sch, the* selection of $\mathcal{Q}$ with respect to S, *denoted by $\sigma_S(\mathcal{Q})$, is defined by: $\sigma_S(\mathcal{Q}) = \{\sigma_S(q) \mid q \in \mathcal{Q}\}$.*
3. *Given a selection condition S containing only attributes from Sch, the* reduction of $\mathcal{Q}$ with respect to S, *denoted by $\tau_S(\mathcal{Q})$, is defined by: $\tau_S(\mathcal{Q}) = \{q \in \mathcal{Q} \mid \sigma_S(q) \equiv q\}$. Intuitively, $\tau_S(\mathcal{Q})$ consists of those queries of $\mathcal{Q}$ that remain unchanged after selection following S.*

Let $\mathcal{Q}_1$ be a set of queries over schema Sch_1, and let $\mathcal{Q}_2$ be a set of queries over schema Sch_2.

4. *The* join of $\mathcal{Q}_1$ and $\mathcal{Q}_2$, *denoted by $\mathcal{Q}_1 \bowtie \mathcal{Q}_2$, is defined by: $\mathcal{Q}_1 \bowtie \mathcal{Q}_2 = \{q_1 \bowtie q_2 \mid q_1 \in \mathcal{Q}_1$ and $q_2 \in \mathcal{Q}_2\}$.*
5. *If $Sch_1 = Sch_2$ then let θ be a set theoretical operation (i.e. union, intersection or difference). Then we define: $\mathcal{Q}_1\ \theta\ \mathcal{Q}_2 = \{q_1\ \theta\ q_2 \mid q_1 \in \mathcal{Q}_1$ and $q_2 \in \mathcal{Q}_2\}$.*

Example 3 *We illustrate the difference between selection and reduction using Example 1. Consider the selection condition $S = (Caddr = Paris)$ and the set of queries $\mathcal{Q} = \{q_1,\ q_2\}$ where $q_1 = \sigma_{(Cjob=Lawyer \wedge Caddr=Paris)}(Cust)$ and $q_2 = \sigma_{Cjob=Professor}(Cust)$.*
Since selection over $\mathcal{Q}$ operates on every query, we have: $\sigma_S(\mathcal{Q}) = \{\sigma_S(q_1),\ \sigma_S(q_2)\}$, and replacing S, q_1 and q_2 by their respective definitions, we obtain:

$$\sigma_S(\mathcal{Q}) = \{\sigma_{(Cjob=Lawyer \wedge Caddr=Paris)}(Cust), \sigma_{(Cjob=Professor \wedge Caddr=Paris)}(Cust)\}.$$

On the other hand, the reduction of $\mathcal{Q}$ results in keeping q_1 and removing q_2 from $\mathcal{Q}$, because $\sigma_S(q_1) \equiv q_1$ and $\sigma_S(q_2) \not\equiv q_2$. Therefore we have: $\tau_S(\mathcal{Q}) = \{q_1\}$, i.e.

$$\tau_S(\mathcal{Q}) = \{\sigma_{(Cjob=Lawyer \wedge Caddr=Paris)}(Cust)\}.$$

We now define two different ways for comparing sets of queries. The first uses the standard query containment ([21]), whereas the second uses equivalence.

Definition 6 - Comparison of Sets of Queries. *Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two sets of queries over the same schema.*

- *$\mathcal{Q}_1$ is* included *in $\mathcal{Q}_2$, denoted by $\mathcal{Q}_1 \sqsubseteq \mathcal{Q}_2$, if for every q_1 in $\mathcal{Q}_1$, there exists q_2 in $\mathcal{Q}_2$ such that $q_1 \sqsubseteq q_2$.*
- *$\mathcal{Q}_1$ is* strongly included *in $\mathcal{Q}_2$, denoted by $\mathcal{Q}_1 \,\tilde{\subseteq}\, \mathcal{Q}_2$, if for every q_1 in $\mathcal{Q}_1$, there exists q_2 in $\mathcal{Q}_2$ such that $q_1 \equiv q_2$.*
  Moreover, $\mathcal{Q}_1$ is strongly equivalent *to $\mathcal{Q}_2$, denoted by $\mathcal{Q}_1 \sim \mathcal{Q}_2$, if $\mathcal{Q}_1 \,\tilde{\subseteq}\, \mathcal{Q}_2$ and $\mathcal{Q}_2 \,\tilde{\subseteq}\, \mathcal{Q}_1$.*

Clearly, if $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are two sets of queries over the same schema such that $\mathcal{Q}_1 \,\tilde{\subseteq}\, \mathcal{Q}_2$, then we have $\mathcal{Q}_1 \sqsubseteq \mathcal{Q}_2$, whereas the converse does not hold. Moreover, the following proposition holds.

Proposition 3 *Let $\mathcal{C}_1 = \langle R_1, B_1, \Sigma_1\rangle$ and $\mathcal{C}_2 = \langle R_2, B_2, \Sigma_2\rangle$ be two contexts such that $K = sch(R_1) = sch(R_2)$ and $sch(R_1 \bowtie B_1) = sch(R_2 \bowtie B_2)$. Then for every instance I, we have:*

1. *If $\mathcal{Q}(\mathcal{C}_1) \sqsubseteq \mathcal{Q}(\mathcal{C}_2)$ and $R_2 \sqsubseteq R_1$, then for every $q_1 \in \mathcal{Q}(\mathcal{C}_1)$, there exists $q_2 \in \mathcal{Q}(\mathcal{C}_2)$ such that $Sup(q_1 \mid \mathcal{C}, I) \leq Sup(q_2 \mid \mathcal{C}, I)$.*
2. *If $\mathcal{Q}(\mathcal{C}_1) \,\tilde{\subseteq}\, \mathcal{Q}(\mathcal{C}_2)$, then for every $q_1 \in \mathcal{Q}(\mathcal{C}_1)$, there exists $q_2 \in \mathcal{Q}(\mathcal{C}_2)$ such that*
$$\frac{Sup(q_1 \mid \mathcal{C}_1, I)}{Sup(q_2 \mid \mathcal{C}_2, I)} = \frac{|R_1(I)|}{|R_2(I)|}.$$

We note that when $R_1 = R_2 = R$, case 2 above reduces to $Sup(q_1 \mid \mathcal{C}_1, I) = Sup(q_2 \mid \mathcal{C}_2, I)$. This proposition will be used in Section 4 to decide whether a query is frequent without having to access the database.

3.2 Projection over the Basis of a Context

We now study how the set of frequent queries of a context changes when we remove attributes from its basis.

In the following definition and proposition, given a domain Σ of a context, we denote by $att(\Sigma)$ the set of all attributes occurring in Σ.

Definition 7 - Projection of a Context. *Let $\mathcal{C} = \langle R, B, \Sigma\rangle$ be a context, and let X be a set of attributes. If $att(\Sigma) \subseteq X \subseteq sch(B)$, then the* b-projection of $\mathcal{C}$ over X, *denoted by $\pi^b_X(\mathcal{C})$, is defined by:*
$$\pi^b_X(\mathcal{C}) = \langle R, \pi_X(B), \Sigma\rangle.$$

Proposition 4 *Let $\mathcal{C} = \langle R, B, \Sigma\rangle$ be a context, and let X be a set of attributes. If $sch(R) \subseteq X$ and $att(\Sigma) \subseteq X \subseteq sch(B)$, then for every support threshold α and every instance I, we have:*
$$freq_\alpha(\pi^b_X(\mathcal{C}), I) \sim \pi_X(freq_\alpha(\mathcal{C}, I)).$$

Example 4 *Consider again the database $DBSales$ of Example 1, and let $\mathcal{C}_1 = \langle AllCust, Cust \bowtie Sales \bowtie Prod, \Sigma_1\rangle$ where $AllCust = \pi_{Cid}(Cust)$ and Σ_1 is the set of all selection conditions of the form $(Cjob = job)$ or $(Ptype = type)$ where job and $type$ are constants in*

dom(Cjob) and dom(Ptype) respectively. In the following examples, we denote such a domain by: $\Sigma_1 = \{Cjob = *, Ptype = *\}$.
Then, let us consider the new context $\mathcal{C}_2 = \pi^b_{Cid,Cjob,Ptype}(\mathcal{C}_1)$. *We have:*

$$\mathcal{C}_2 = \langle AllCust, \pi_{Cid,Cjob,Ptype}(Cust \bowtie Sales \bowtie Prod), \Sigma_1\rangle.$$

Moreover, using Proposition 4, for every support threshold α *and every instance* I *of* $DBSales$, *we have:*

$$freq_\alpha(\mathcal{C}_2, I) \ \sim \ \pi_{Cid,Cjob,Ptype}(freq_\alpha(\mathcal{C}_1, I)).$$

Hence, applying Proposition 3, the set $freq_\alpha(\mathcal{C}_2, I)$ *can be obtained from the projection of the queries in* $freq_\alpha(\mathcal{C}_1, I)$.

3.3 Selection over the Basis of a Context

In this subsection, we study how the set of frequent queries of a context changes, when we apply a selection over its basis.

In the following definition and propositions, given a selection condition S, we denote by $att(S)$ the set of all attributes occurring in S.

Definition 8 - Selection over a Context. *Let* $\mathcal{C} = \langle R, B, \Sigma\rangle$ *be a context, and let* S *be a selection condition. If* $att(S) \subseteq sch(B)$, *then the* b-selection of $\mathcal{C}$ according to S, *denoted by* $\sigma^b_S(\mathcal{C})$, *is defined by:* $\sigma^b_S(\mathcal{C}) = \langle R, \sigma_S(B), \Sigma\rangle$.

Proposition 5 *Let* $\mathcal{C} = \langle R, B, \Sigma\rangle$ *be a context. If* S *is a selection condition in* Σ^*, *then for every support threshold* α *and every instance* I, *we have:*

$$freq_\alpha(\sigma^b_S(\mathcal{C}), I) \sim \tau_S(freq_\alpha(\mathcal{C}, I)).$$

In what follows, we say that a conjunction of atomic selection conditions $S = (A_1 = a_1) \wedge \ldots \wedge (A_n = a_n)$ is *independent* from a domain Σ if for every $i = 1 \ldots n$, the atomic selection condition $(A_i = a_i)$ is not in Σ.

Proposition 6 *Let* $\mathcal{C} = \langle R, B, \Sigma\rangle$ *be a context. If* S *is a selection condition independent from* Σ *such that* $att(S) \subseteq sch(B)$, *then for every support threshold* α *and every instance* I, *we have:*

$$freq_\alpha(\sigma^b_S(\mathcal{C}), I) \ \tilde{\subseteq}\ \sigma_S(freq_\alpha(\mathcal{C}, I)).$$

The following corollary is an immediate consequence of Propositions 5 and 6.

Corollary 2 *Let* $\mathcal{C} = \langle R, B, \Sigma\rangle$ *be a context, and* $S = S_1 \wedge S_2$ *be a selection condition such that* $att(S) \subseteq sch(B)$, $S_1 \in \Sigma^*$, *and* S_2 *is independent from* Σ. *For every support threshold* α *and every instance* I, *we have:*

$$freq_\alpha(\sigma^b_S(\mathcal{C}), I) \ \tilde{\subseteq}\ \tau_{S_1}(\sigma_{S_2}(freq_\alpha(\mathcal{C}, I))).$$

Using the commutativity of the selection operation, it is easy to see that $\tau_{S_1}(\sigma_{S_2}(freq_\alpha(\mathcal{C}, I))) \sim \sigma_{S_2}(\tau_{S_1}(freq_\alpha(\mathcal{C}, I)))$. Thus, Corollary 2 implies that: $freq_\alpha(\sigma^b_S(\mathcal{C}), I) \ \tilde{\subseteq}\ \sigma_{S_2}(\tau_{S_1}(freq_\alpha(\mathcal{C}, I)))$.

Example 5 *Consider again the database DBSales of Example 1 and the query AllCust defined by:* $AllCust = \pi_{Cid}(Cust)$.

Assume first that we are interested in the relationships between the jobs of customers and the types of products they buy. Then we consider the context $\mathcal{C}_1 = \langle AllCust, Cust \bowtie Sales \bowtie Prod, \Sigma\rangle$ *where* $\Sigma = \{Cjob = *, Ptype = *\}$.

Assume now that we are interested only in the rules concerning lawyers. If we consider the context $\mathcal{C}_2 = \sigma^b_{Cjob=Lawyer}(\mathcal{C}_1)$, *then, according to Proposition 5, for every support threshold* α *and every instance* I *of DBSales, we have:* $freq_\alpha(\mathcal{C}_2, I) \sim \tau_{Cjob=Lawyer}(freq_\alpha(\mathcal{C}_1, I))$.
Thus, applying Proposition 3, it turns out that the frequent queries of $\mathcal{C}_1$ *that deal with lawyers are exactly the frequent queries of* $\mathcal{C}_2$.

Finally, assume that we are interested in the rules involving only customers living in Paris. The corresponding context $\mathcal{C}_3$ *can be defined by* $\mathcal{C}_3 = \sigma^b_{Caddr=Paris}(\mathcal{C}_1)$, *and, according to Proposition 6, for every instance* I *of DBSales, we have:* $freq_\alpha(\mathcal{C}_3, I) \,\tilde{\subseteq}\, \sigma_{Caddr=Paris}(freq_\alpha(\mathcal{C}_1, I))$. *In this case, applying Proposition 3, if a query* q_1 *in* $\mathcal{Q}(\mathcal{C}_1)$ *is not frequent, the corresponding query* $q_3 = \sigma_{Caddr=Paris}(Q_1)$ *in* $\mathcal{Q}(\mathcal{C}_3)$ *is not frequent either, and this conclusion is reached without any access to the database.*

3.4 Join of Contexts

In this subsection, we consider the join of two contexts.

Definition 9 - Joining Contexts. *Let* $\mathcal{C}_1 = \langle R, B_1, \Sigma_1\rangle$ *and* $\mathcal{C}_2 = \langle R, B_2, \Sigma_2\rangle$ *be two contexts having the same reference* R. *The* join *of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$, *denoted by* $\mathcal{C}_1 \bowtie_b \mathcal{C}_2$, *is defined by:* $\mathcal{C}_1 \bowtie_b \mathcal{C}_2 = \langle R, B_1 \bowtie B_2, \Sigma_1 \cup \Sigma_2\rangle$.

Proposition 7 *Let* $\mathcal{C}_1 = \langle R, B_1, \Sigma_1\rangle$ *and* $\mathcal{C}_2 = \langle R, B_2, \Sigma_2\rangle$ *be two contexts having the same reference* R. *For every support threshold* α *and every instance* I, *we have:*

$$freq_\alpha(\mathcal{C}_1 \bowtie_b \mathcal{C}_2, I) \,\tilde{\subseteq}\, freq_\alpha(\mathcal{C}_1, I) \bowtie freq_\alpha(\mathcal{C}_2, I).$$

Example 6 *Consider again the database DBSales of Example 1 and the query* $AllCust = \pi_{Cid}(Cust)$.

Assume first that we are interested in the relationships between the jobs of customers and the types of products they buy. Then, as already mentioned, we specify the context $\mathcal{C}_1 = \langle AllCust, Cust \bowtie Sales \bowtie Prod, \Sigma_1\rangle$ *where* $\Sigma_1 = \{Cjob = *, Ptype = *\}$.

Then assume that we are interested in the relationships between the names of stores where customers issue their transactions and the types of products they buy. The corresponding context $\mathcal{C}_2$ *is defined by* $\mathcal{C}_2 = \langle AllCust, Store \bowtie Sales \bowtie Prod, \Sigma_2\rangle$ *where* $\Sigma_2 = \{Sname = *, Ptype = *\}$.

Now, if we consider the relationships between the jobs of customers, the names of stores where they issue their transactions and the types of products they buy, then the corresponding context $\mathcal{C}_{12}$ *can be defined by:* $\mathcal{C}_{12} = \mathcal{C}_1 \bowtie_b \mathcal{C}_2$.

According to Proposition 7, for every support threshold α *and every instance* I *of DBSales, we have:* $freq_\alpha(\mathcal{C}_{12}, I) \,\tilde{\subseteq}\, freq_\alpha(\mathcal{C}_1, I) \bowtie freq_\alpha(\mathcal{C}_2, I)$. *Thus, by Definition 6, for any query* q_{12} *in* $\mathcal{Q}(\mathcal{C}_{12})$, *there exist two queries* q_1 *and* q_2 *of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *respectively, such that* $q_{12} \equiv q_1 \bowtie q_2$. *Moreover, by Corollary 1,* $Sup(q_1 \bowtie q_2 \mid \mathcal{C}_{12}, I) \leq Sup(q_i \mid \mathcal{C}_i, I)$, *for* $i = 1, 2$. *Therefore, if* q_{12} *is frequent, then so are* q_1 *and* q_2.

3.5 Union, Intersection and Difference of Contexts

In this subsection, we construct new contexts from old by applying set theoretical operations over the basis of the input contexts.

Definition 10 - Union, Intersection and Difference of Contexts. *Let $\mathcal{C}_1 = \langle R, B_1, \Sigma\rangle$ and $\mathcal{C}_2 = \langle R, B_2, \Sigma\rangle$ be two contexts having the same reference R and the same domain Σ, such that $sch(B_1) = sch(B_2)$.*

1. *The* union *of $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted by $\mathcal{C}_1 \cup_b \mathcal{C}_2$, is defined by: $\mathcal{C}_1 \cup_b \mathcal{C}_2 = \langle R, B_1 \cup B_2, \Sigma\rangle$.*
2. *The* intersection *of $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted by $\mathcal{C}_1 \cap_b \mathcal{C}_2$, is defined by: $\mathcal{C}_1 \cap_b \mathcal{C}_2 = \langle R, B_1 \cap B_2, \Sigma\rangle$.*
3. *The* difference *of $\mathcal{C}_2$ and $\mathcal{C}_1$, denoted by $\mathcal{C}_2 \setminus_b \mathcal{C}_2$, is defined by: $\mathcal{C}_2 \setminus_b \mathcal{C}_1 = \langle R, B_2 - B_1, \Sigma\rangle$.*

According to items 3, 4 and 5 of Corollary 1, it is easily seen that the following proposition holds.

Proposition 8 *Let $\mathcal{C}_1 = \langle R, B_1, \Sigma\rangle$ and $\mathcal{C}_2 = \langle R, B_2, \Sigma\rangle$ be two contexts having the same reference R and the same domain Σ, such that $sch(B_1) = sch(B_2)$. For every support threshold α and every instance I, we have:*

1. $freq_\alpha(\mathcal{C}_i, I) \sqsubseteq freq_\alpha(\mathcal{C}_1 \cup_b \mathcal{C}_2, I)$, *for* $i = 1, 2$.
2. $freq_\alpha(\mathcal{C}_1 \cap_b \mathcal{C}_2, I) \sqsubseteq freq_\alpha(\mathcal{C}_i, I)$, *for* $i = 1, 2$.
3. $freq_\alpha(\mathcal{C}_2 \setminus_b \mathcal{C}_1, I) \sqsubseteq freq_\alpha(\mathcal{C}_2, I)$.

Example 7 *Consider again the database $DBSales$ of Example 1 and the query $AllCust = \pi_{Cid}(Cust)$.*

Assume first that we are interested in the relationships between the customers issuing transactions in stores named $Carrefour$ and the types of products they buy. The corresponding context $\mathcal{C}_1$ is defined by $\mathcal{C}_1 = \langle AllCust, CCarrefour \bowtie Sales \bowtie Prod, \Sigma\rangle$ where $CCarrefour = \pi_{Cid,Cjob}(Cust \bowtie Sales \bowtie \sigma_{Sname=Carrefour}(Store))$ and $\Sigma = \{Cjob = *, Ptype = *\}$. Similarly, if we are interested by the relationships between the jobs of customers issuing transactions in stores named $Fnac$ and the types of products they buy, the corresponding context $\mathcal{C}_2$ is defined by $\mathcal{C}_2 = \langle AllCust, CFnac \bowtie Sales \bowtie Prod, \Sigma\rangle$ where $CFnac = \pi_{Cid,Cjob}(\sigma_{Sname=Fnac}(Store) \bowtie Sales \bowtie Cust)$ and $\Sigma = \{Cjob = *, Ptype = *\}$. Then the contexts $\mathcal{C}_1 \cup_b \mathcal{C}_2$, $\mathcal{C}_1 \cap_b \mathcal{C}_2$ and $\mathcal{C}_1 \setminus_b \mathcal{C}_2$ refer respectively to the relationships between jobs of customers issuing transactions:

- *in stores named $Carrefour$* or *in stores named $Fnac$,*
- *in stores named $Carrefour$* and *in stores named $Fnac$,*
- *in stores named $Carrefour$* but not *in stores named $Fnac$,*

and the types of products they buy. Moreover, by Proposition 8, we have:

- $freq_\alpha(\mathcal{C}_i, I) \sqsubseteq freq_\alpha(\mathcal{C}_1 \cup_b \mathcal{C}_2, I)$, *for* $i = 1, 2$,
- $freq_\alpha(\mathcal{C}_1 \cap_b \mathcal{C}_2, I) \sqsubseteq freq_\alpha(\mathcal{C}_i, I)$, *for* $i = 1, 2$, *and*
- $freq_\alpha(\mathcal{C}_1 \setminus_b \mathcal{C}_2, I) \sqsubseteq freq_\alpha(\mathcal{C}_1, I)$.

Therefore, Proposition 3 implies that the computations of $freq_\alpha(\mathcal{C}_1 \cup_b \mathcal{C}_2, I)$, $freq_\alpha(\mathcal{C}_1 \cap_b \mathcal{C}_2, I)$ and $freq_\alpha(\mathcal{C}_1 \setminus_b \mathcal{C}_2, I)$ can take advantage of the fact that the sets $freq_\alpha(\mathcal{C}_1, I)$ and $freq_\alpha(\mathcal{C}_2, I)$ have already been computed.

3.6 Restriction of a Context

In this section, we consider an operation that changes the reference of a context. More precisely, given a context $\mathcal{C} = \langle R, B, \Sigma \rangle$, this operation restricts the reference to those objects that satisfy a selection criterion specified through a query of $\mathcal{C}$. It follows that the new reference is more specific than R.

Definition 11 *Let $\mathcal{C} = \langle R, B, \Sigma \rangle$ be a context with $K = sch(R)$, and let S be a selection condition in Σ^*. The restriction of $\mathcal{C}$ over S is defined by:*

$$\rho_S(\mathcal{C}) = \langle \pi_K(R \bowtie \sigma_S(B)), \sigma_S(B), \Sigma \rangle.$$

Proposition 9 *Let $\mathcal{C} = \langle R, B, \Sigma \rangle$ be a context, and let S be a selection condition in Σ^*. Then, for every threshold α and every instance I, we have:*

$$\tau_S(freq_\alpha(\mathcal{C})) \,\tilde{\subseteq}\, freq_\alpha(\rho_S(\mathcal{C})).$$

Example 8 *Consider again the database $DBSales$ of Example 1 and assume now that we are interested in the relationships between the jobs of customers, the town where they live and the types of products they buy. Then we specify the context $\mathcal{C}_1 = \langle AllCust, Cust \bowtie Sales \bowtie Prod, \Sigma \rangle$ where $\Sigma = \{Cjob = *, Caddr = *, Ptype = *\}$.*

Suppose that we are interested in focussing on customers living in $Paris$. The corresponding context $\mathcal{C}_2$ can be defined by $\mathcal{C}_2 = \rho_{Caddr=Paris}(\mathcal{C}_1)$. Note that the reference of $\mathcal{C}_2$ is defined by the query $\pi_{Cid}(AllCust \bowtie \sigma_{Caddr=Paris}(Cust \bowtie Sales \bowtie Prod))$, meaning that the support of queries is computed with respect to the number of customers living in $Paris$ (and not with respect to the total number of customers). Moreover, by Proposition 9, for every support threshold α and every instance I of $DBSales$, we have: $\tau_{Caddr=Paris}(freq_\alpha(\mathcal{C}_1, I)) \,\tilde{\subseteq}\, freq_\alpha(\mathcal{C}_2, I)$.

As a consequence, every query in $\tau_{Caddr=Paris}(freq_\alpha(\mathcal{C}_1, I))$ is equivalent to a query in $freq_\alpha(\mathcal{C}_2, I)$. Thus, applying Proposition 3, if q_1 is a frequent query in $\tau_{Caddr=Paris}(freq_\alpha(\mathcal{C}_1, I))$, then there exists a query in $freq_\alpha(\mathcal{C}_2, I)$ which is equivalent to q_1, and this conclusion can be reached without any access to the database.

4 Implementation Issues: Incremental Extraction of Frequent Queries

We have seen so far operations for combining contexts in order to define new contexts. We have also given properties that show how the set of frequent queries of a context changes when we apply operations over its reference or basis. In this section, we review all cases where these properties allow for optimization in the computation of frequent queries of combined contexts.

Of course, in order to be able to use the frequent queries of the component contexts, we must store them. For the storage of frequent queries of a given context, we propose two ways: either store all queries, or only their positive boundaries ([13]). We note that, in the second case, one can compute the whole set of frequent queries from the positive boundary through only one pass over the data ([13]).

4.1 Possible Optimizations

Let $\mathcal{C}_{new}$ be a new context defined by combining other contexts. Given a support threshold α and an instance I of DB, we consider all cases where the computation of $freq_\alpha(\mathcal{C}_{new}, I)$ can be optimized using the sets of frequent queries in I that have already been computed. We distinguish three such cases:

1. $freq_\alpha(\mathcal{C}_{new}, I)$ is strongly equivalent to a set of frequent queries already computed;
2. $freq_\alpha(\mathcal{C}_{new}, I)$ is included or strongly included in a set of frequent queries already computed;
3. $freq_\alpha(\mathcal{C}_{new}, I)$ contains, strongly or not, a set of frequent queries already computed.

These different cases are discussed separately below.

4.1.1 Strongly Equivalent Sets of Frequent Queries

Cases where $freq_\alpha(\mathcal{C}_{new},\, I)$ is strongly equivalent to a set of frequent queries of a context $\mathcal{C}_{old}$, occur when $\mathcal{C}_{new}$ is defined from $\mathcal{C}_{old}$ in one of the following ways:

- by projection of $\mathcal{C}_{old}$ (see Corollary 4),
- by selection of $\mathcal{C}_{old}$, with a selection condition involving only conditions in the domain of $\mathcal{C}_{old}$ (see Proposition 5).

In both cases, if all queries in $freq_\alpha(\mathcal{C}_{old}, I)$ and their supports are stored, then no new computation or access to data is needed to obtain $freq_\alpha(\mathcal{C}_{new}, I)$. One has just to apply the corresponding operation to the queries of $freq_\alpha(\mathcal{C}_{old}, I)$ to get the set $freq_\alpha(\mathcal{C}_{new}, I)$.

However, if only the positive boundary of $freq_\alpha(\mathcal{C}_{old}, I)$ is stored, then it is necessary to rebuild the set $freq_\alpha(\mathcal{C}_{old}, I)$, and this needs one pass over the data.

4.1.2 Subset of a Set of Frequent Queries

Cases where $freq_\alpha({}_{new},\, I)$ is included, strongly or not, in the set of frequent queries of a task $\mathcal{C}_{old}$ occur when $\mathcal{C}_{new}$ is defined in one of the following ways:

- by selection of $\mathcal{C}_{old}$ with a selection condition involving conditions not in the domain of $\mathcal{C}_{old}$ (see Proposition 6 and Corollary 2),
- by join or intersection of two contexts $\mathcal{C}_{old_1}$ and $\mathcal{C}_{old_2}$ (see Proposition 7 and Proposition 8).

In both cases, we only know an upper bound of the supports of the candidate queries in $\mathcal{C}_{new}$. Thus, these supports have to be computed. On the other hand, it is important to note that considering the positive boundaries of the sets of frequent queries already computed, allows for some additional pruning (with respect to standard levelwise algorithms) when generating the candidate queries of level $k+1$ from the frequent queries of level k.

To see this, consider the case of a b-selection with selection condition S independent from the domain of $\mathcal{C}_{old}$. If a candidate query q_{new} belongs to $freq_\alpha(\mathcal{C}_{new}, I)$, then by Proposition 6, there exists q_{old} in $freq_\alpha(\mathcal{C}_{old}, I)$ such that $q_{new} \equiv \sigma_S(q_{old})$. Moreover, since q_{old} is in $freq_\alpha(\mathcal{C}_{old}, I)$, there exists a query q^+_{old} in the positive boundary of $freq_\alpha(\mathcal{C}_{old}, I)$ such that $q^+_{old} \sqsubseteq q_{old}$. By monotonicity of the selection operation, we have that $\sigma_S(q^+_{old}) \sqsubseteq \sigma_S(q_{old})$. Thus we obtain $\sigma_S(q^+_{old}) \sqsubseteq q_{new}$.

Therefore, if there is no query q^+_{old} as above, then q_{new} can be removed from the set of candidate queries of $\mathcal{C}_{new}$. We show in Section 4.3 how to efficiently test the existence of q^+_{old}.

Similarly, in the case of a join, if a candidate query q_{new} belongs to $freq_\alpha(\mathcal{C}_{new},\, I)$, then by Proposition 7, there exist two frequent queries q_{old_1} and q_{old_2} such that $q_{new} \equiv q_{old_1} \bowtie q_{old_2}$. Thus, there exist $q^+_{old_1}$ and $q^+_{old_2}$ in the positive boundaries of $freq_\alpha(\mathcal{C}_1, I)$ and $freq_\alpha(\mathcal{C}_2, I)$ such that $q^+_{old_1} \bowtie q^+_{old_2} \sqsubseteq q_{new}$. Thus, we have the same case as above, and we show in Section 4.3 how to efficiently test the existence of $q^+_{old_1}$ and $q^+_{old_2}$.

4.1.3 Superset of a Set of Frequent Queries

The cases where $freq_\alpha(\mathcal{C}_{new},\, I)$ contains, strongly or not, a set of already computed frequent queries occur when $\mathcal{C}_{new}$ is defined by:

- restriction of $\mathcal{C}_{old}$ (see Proposition 9),
- union of $\mathcal{C}_{old_1}$ and $\mathcal{C}_{old_2}$ (see Proposition 8).

In all cases, a subset $\mathcal{F}$ of $freq_\alpha(\mathcal{C}_{new},\, I)$ is known without any further computation. Indeed, by Proposition 3, for every query q_{old} in $freq_\alpha(\mathcal{C}_{old}, I)$ (or in $freq_\alpha(\mathcal{C}_{old_1} \cup_b \mathcal{C}_{old_2}, I)$) there exists a query q_{new} in $freq_\alpha(\mathcal{C}_{new})$ whose support is greater than or equal to that of q_{old}. Thus $\mathcal{F}$ is precisely the composition of $freq_\alpha(\mathcal{C}_{old}, I)$ (or of $freq_\alpha(\mathcal{C}_{old_1})$ and $freq_\alpha(\mathcal{C}_{old_2}, I)$) by the corresponding operator.

Moreover, in the case of restriction, the supports of the queries in $\mathcal{F}$ can be easily obtained, since only the support denominator of the corresponding queries in $freq_\alpha(\mathcal{C}_{old}, I)$ is changed. However, in the case of union, the supports of the queries in $\mathcal{F}$ have to be computed, and this can be achieved through one pass over the data.

Finally, for queries that are not in $\mathcal{F}$, no further optimization is possible. For such queries one will have to use the standard approach [1].

4.2 Storage of the Sets of Frequent Queries

In this section, we outline how the sets of frequent queries can be stored in a database that we call *DMining*. This database contains the following four tables:

- $Freq(CId, RQuery, BQuery, DId, Minsup, Border)$, where:
  – CId stores the context identifier,
  – $RQuery$ and $BQuery$ store expressions of the relational algebra defining respectively the reference and the basis of the context,
  – DId stores the identifier of the domain,
  – $Minsup$ stores the support threshold,
  – $Border$ stores a boolean value indicating whether the boundary is stored or not.

- $Dom(DId, Name)$, where DId has the same meaning as above and $Name$ stores the attribute occurring in the domain. We implicitly assume here that if attribute A is in the domain of a context, then the corresponding selection conditions are of the form $A = a$, for $a \in dom(A)$.

- $Query(CId, QId, Sup, Border)$, where CId stores a context identifier, QId stores a query identifier, Sup stores the support of this query, and $Border$ stores a boolean value indicating whether the query belongs or not to the positive boundary.

- $Cond(CId, QId, Name, Const)$, where CId and QId are as above and where $Name$ stores an attribute name and $Const$ stores a constant of the domain of that attribute.

Example 9 *Figure 1 shows how the positive boundaries of two contexts are stored. In our example, the query identified by* 3 *is a frequent query of the context identified by* 1*. This query is defined by the following expression:*

$$q_3 = AllCust \bowtie \sigma_{Cjob=Lawyer \wedge Ptype=Bread}(Cust \bowtie Sales \bowtie Prod).$$

4.3 Effective Use of the Stored Frequent Queries

In Subsection 4.1.2, we have seen that the positive boundary of sets of frequent queries can be used in order to optimize the computation of sets of frequent queries when new contexts are defined from old using b-selection or join. However, as mentioned above, this optimization is based on the comparison of a candidate query with all queries of the positive boundary.

We propose an efficient way to process this test, using an *inverse representation* of the positive boundary. Roughly speaking, the inverse representation of a given boundary consists in considering separately, for each atomic selection condition S, the set of identifiers of the queries involving S. Then, given a conjunction S' of atomic selection conditions, the set of queries involving S' is simply the intersection of the sets associated with each atomic selection condition of S'. We illustrate our method next.

Let $\mathcal{C}_1 = \langle R, B_1, \Sigma_1 \rangle$ and $\mathcal{C}_2 = \langle R, B_2, \Sigma_2 \rangle$ be two contexts, and let $\mathcal{C}_{new}$ be the context defined by $\mathcal{C}_{new} = \mathcal{C}_1 \bowtie_b \mathcal{C}_2$. Given a support threshold α and an instance I, let Bd_1^+ and Bd_2^+ be the positive boundaries of $freq_\alpha(\mathcal{C}_1, I)$ and $freq_\alpha(\mathcal{C}_2, I)$, respectively.

For $i = 1, 2$, let $QId(i, A = a)$ be the set of identifiers of all queries in Bd_i^+ whose selection condition involves $A = a$. As seen in Subsection 4.1.2, a candidate query $q_{new} = R \bowtie \sigma_S(B_1 \bowtie B_2)$ of $\mathcal{C}_{new}$ is frequent if and only if there exist q_1^+ in Bd_1^+ and q_2^+ in Bd_2^+ such that $q_1^+ \bowtie q_2^+ \sqsubseteq q_{new}$.

On the other hand, the selection condition $S = (A_1 = a_1) \wedge \ldots \wedge (A_n = a_n)$ of q_{new} can be written as $S = S_1 \wedge S_2$ where $S_1 = (A_{11} = a_{11}) \wedge \ldots \wedge (A_{1p} = a_{1p})$ is in Σ_1^* and $S_2 = (A_{21} = a_{21}) \wedge \ldots \wedge (A_{2q} = a_{2q})$ is in Σ_2^*.

Then, it can easily be seen that there exist q_1^+ in Bd_1^+ and q_2^+ in Bd_2^+ such that $q_1^+ \bowtie q_2^+ \sqsubseteq q_{new}$ if and only if:

$$\begin{array}{l} QId(1, A_{11} = a_{11}) \cap \ldots \cap QId(1, A_{1p} = a_{1p}) \neq \emptyset \quad \text{and} \\ QId(2, A_{21} = a_{21}) \cap \ldots \cap QId(2, A_{2q} = a_{2q}) \neq \emptyset. \end{array}$$

As shown in the following example, the two statements above can be tested efficiently.

Example 10 *In the database of Example 1, consider the positive boundaries Bd_1^+ and Bd_2^+ as shown in Figure 1. Let $\mathcal{C}_{new} = \mathcal{C}_1 \bowtie_b \mathcal{C}_2$, and let $q_{new} = AllCust \bowtie \sigma_S(Cust \bowtie Sales \bowtie Prod \bowtie Store)$ with $S = (Cjob = Lawyer) \wedge (Caddr = Paris)$.*

With the notation just introduced, we have $S = S_1 = S_2$, and since:

$$QId(1, Cjob = Lawyer) \cap QId(1, Caddr = Paris) = \{2, 3\} \cap \{1\} = \emptyset,$$

it turns out that q_{new} is not *frequent. Therefore, this query is removed from the set of candidates of level* 2.

Consider now $q_{new} = AllCust \bowtie \sigma_S(Cust \bowtie Sales \bowtie Prod \bowtie Store)$ with $S = (Cjob = Lawyer) \wedge (Ptype = Milk) \wedge (Sname = Ikea)$ as a candidate query of $\mathcal{C}_{new}$. We have: $S_1 = (Cjob = Lawyer) \wedge (Ptype = Milk)$ and $S_2 = (Cjob = Lawyer) \wedge (Sname = Ikea)$, and thus:

$$\begin{array}{l} QId(1, Cjob = Lawyer) \cap QId(1, Ptype = Milk) = \{2, 3\} \cap \{2\} \neq \emptyset \quad \textit{and} \\ QId(2, Cjob = Lawyer) \cap QId(2, Sname = Ikea) = \{5, 6\} \cap \{6\} \neq \emptyset. \end{array}$$

As a consequence, q_{new} can be frequent and must be kept among the candidate queries of level 3.

4.4 First Experiments

In this section, we present the results of our first experiments using synthetic data that have been generated in a similar way as in [1]. However, the generator used in [1] has been modified so as to comply with the case where the data is stored in more than one table.

More precisely, the tests have been conducted on a database, called $DBTest$ and containing three tables over the schemas $Cust(X1, X2, \ldots, X10)$, $Sales(X1, Y1, Z1, D1)$ and $Prod(Y1, Y2, \ldots, Y10)$, as in our running example (but with additional attributes).

Two instances of this database, called $I10$ and $I40$, have been considered, with respective cardinalities of 2.000, 6.000 and 54.000 for the tables $Cust$, $Sales$ and $Prod$. In instance $I10$ (respectively $I40$), all domains of attributes A in the domains of contexts are such that $|dom(A)| = 10$ (respectively $|dom(A)| = 40$). Moreover, the data are such that when the size of the domain increases, then the ratios of frequent queries in positive boundaries decrease.

For example, let $\alpha = 0.5\%$ and $\mathcal{C} = \langle AllCust, Cust \bowtie Sales \bowtie Prod, \Sigma \rangle$ where $AllCust = \pi_{X1}(Cust)$ and $\Sigma = \{X_i = *, Y_i = * \mid i = 2, \ldots, 6\}$. In instance $I10$, 45% out of the 10.086 frequent queries in $\mathcal{Q}(\mathcal{C})$ belong to the positive boundary of $freq_\alpha(\mathcal{C}, I10)$, whereas, in instance $I40$, 18% out of the 2.678 frequent queries in $\mathcal{Q}(\mathcal{C})$ belong to the positive boundary of $freq_\alpha(\mathcal{C}, I40)$.

We consider now the case of a b-selection with an empty selection condition S, and the context $\mathcal{C}$ given just above. In each of the instances $I10$ and $I40$, we compare:

- the time for the computation of the frequent queries of $\mathcal{C}$, without any optimization, and
- the time for the incremental computation of the frequent queries in $\sigma^b_S(\mathcal{C})$, assuming that *the positive boundaries* of $freq_\alpha(\mathcal{C}, I10)$ and $freq_\alpha(\mathcal{C}, I40)$ are stored.

In this case, for every candidate query of $\sigma^b_S(\mathcal{C})$, we search for the corresponding query in the positive boundary, as explained in the previous subsection. Since S is always satisfied, this search is performed with respect to *every* query in the positive boundary. This shows that considering an empty selection condition is the worst case in the number of searches. Moreover, since only the positive boundaries are stored, we have to recompute the supports of *all* queries in $freq_\alpha(\mathcal{C}, I10)$ and $freq_\alpha(\mathcal{C}, I40)$, which is also the worst case in this respect.

The results of this comparison are shown in Figure 2. We observe that the gain is generally more important for the instance $I40$ than for the instance $I10$. This is due to the fact that the relative size of the positive boundary is smaller for $I40$ than for $I10$ (18% and 45%, respectively). Indeed, the smaller the size of the boundary, the more efficient the pruning phase in the incremental approach.

These tests clearly show that our approach is more efficient than a non incremental approach. Moreover, it is important to note that the sizes of the positive boundaries that are stored have a significant impact on the gain in computation time obtained by our approach. We are currently running tests using all operations introduced in this paper. The corresponding results are reported in the full paper [8].

5 Conclusion and Further Work

We have introduced a new formalism for the incremental learning of association rules through composition of mining contexts. Assuming that the frequent queries of the component contexts are stored, we have shown that our incremental approach results in a significant reduction of computation time for the computation of frequent queries of composed contexts.

Based on this work, we are currently considering the following lines for further research:

- More experimental results have to be considered in order to better quantify the gain that our approach can provide. We are currently working on this issue.
- Our incremental approach can be generalized as follows: even if the new context is not expressed as the composition of other contexts, find a relationship between this new context and the contexts that are currently stored. We note that some results have already been obtained in the case of conjunctive queries [7].
- The influence of redundancies in the storage of frequent queries of the component contexts is crucial in our model. We think that the approach of [15, 16] provides some interesting directions in this respect.
- Given a set of contexts, determine a subset such that the storage of their frequent queries can optimize the computation time of the frequent queries for the whole set.
- So far, in this work, we assume that the database is static, which is not the case in practice. Therefore, an important issue is to consider updates. This problem is clearly similar to the problem of maintenance of association rules, already considered in [5, 19].

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A.I. Verkamo (1996). *Fast Discovery of Association Rules.* In Advances in Knowledge Discovery and Data Mining, pp 309-328, AAAI-MIT Press.

[2] E. Baralis and G. Psaila (1999). *Incremental Refinement of Mining Queries.* In Proc. of DAWAK'99, pp. 173-182, Florence.

[3] S. Brin, R. Motwani, J.D. Ullman and S. Tsur (1997), *Dynamic Itemset Counting and Implication Rules for Market Basket Data.* SIGMOD Conference, pp. 255-264.

[4] A. K. Chandra and P. M. Merlin (1977). *Optimal implementation of conjunctive queries in relational databases.* In Proc. of the Ninth ACM Symposium on Theory of Computing, pp. 77-90.

[5] D. Cheung, J. Han, V. Ng, and C.Y. Wong (1996) *Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique.* Proc. of International Conference on Data Engineering (ICDE'96), pp. 106-114, New Orleans, USA.

[6] L. Dehaspe and H. Toivonen (1999). *Discovery of Frequent Datalog Patterns.* Data Mining and Knowledge Discovery, 3, pp. 7-36, Kluwer Academic Publishers.

[7] C.T. Diop, A. Giacometti, D. Laurent and N. Spyratos (2000). *Extraction de règles d'association entre vues : vers une approche incrémentale* (French). 16èmes journées Bases de Données Avancées (BDA'2000), pp. 323-344, Blois, France. English version available as Tech. Rep. LI/E3I No. 241, University of Tours, France.

[8] C.T. Diop, A. Giacometti, D. Laurent and N. Spyratos (2001). *Combining Mining Tasks through Relational Algebra.* Techn. Rep. LI/E3I, Tours, France (full paper, to appear).

[9] J. Han, Y. Fu, K. Kopersky and O. Zaiane (1996). *DMQL: A Data Mining Query Language for Relational Databases.* Proc. DMKD'96 (SIGMOD'96 Workshop on KDD), Montreal, Canada.

[10] T. Imielinski and A. Virmani (1999). *MSQL: A Query Language for Database Mining.* Data Mining and Knowledge Discovery, Vol. 3, No. 4, pp. 373-408, Kluwer Academic Publishers.

[11] M. Kamber, J. Han, J. Chiang (1997). *Metarule-Guided Mining of Multi-Dimensional Association Rules Using Data Cubes.* Proc. of International Conference on Data Mining and Knowledge Discovery (KDD'97), pp. 207-210, Newport Beach, USA.

[12] N. Lavrač, P. Flach et B. Zupan (1999). *Rule Evaluation Measures: a Unifying View.* Proc. of ILP'99.

[13] H. Mannila, H. Toivonen (1997). *Levelwise Search and Borders of Theories in Knowledge Discovery.* Techn. Rep. C-1997-8, University of Helsinki.

[14] T.Morzy, M. Wojciechowski and M. Zakrzewicz (2000). *Materialized Data Mining Views.* Proc. of European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'2000), Springer Verlag, pp. 65-74, France.

[15] B. Nag, P. Deshpande and D.J. DeWitt (1999). *Using a Knowledge Cache for Interactive Discovery of Association Rules.* Proc. of International Conference on Data Mining and Knowledge Discovery (KDD'99), pp. 244-253, San Diego, USA.

[16] B. Nag, P. Deshpande and D.J. DeWitt (2001). *Caching for Multi-dimensional Data Mining Queries.* Proc. of Int. Conf. on Systemics, Cybernetics and Informatics (SCI'2001), Orlando, Florida.

[17] J.S. Park, M. Chen, P.S. Yu (1995). *An Effective Hash Based Algorithm for Mining Association Rules.* SIGMOD Conference, pp. 175-186.

[18] A. Savasere, E. Omiecinski and S.B. Navathe (1995). *An Efficient Algorithm for Mining Association Rules in Large Databases.* VLDB Conference, pp. 432-444.

[19] S. Thomas, S. Bodagala, K. Alsabti and Sanjay Ranka (1997). *An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases.* Proc. of International Conference on Data Mining and Knowledge Discovery (KDD'97), pp. 263-266, Newport Beach, USA.

[20] S. Tsur and al. (1998), *Query Flocks: A Generalization of Association-Rule Mining.* SIGMOD Conference, pp. 1-12.

[21] J.D. Ullman (1989). *Principles of Databases and Knowledge-Base Systems.* Vol. 1-2, Computer Science Press.

Freq

CId	RQuery	BQuery	DId	Minsup	Border
1	AllCust	Join(Cust,Sales,Prod)	1	0,1	true
2	AllCust	Join(Cust,Sales,Store)	2	0,1	true

Query

CId	QId	Sup	Border
1	1	0,3	true
1	2	0,2	true
1	3	0,4	true
1	4	0,2	true
2	5	0,3	true
2	6	0,2	true
2	7	0,3	true

Dom

DId	Name
1	Cjob
1	Caddr
1	Ptype
2	Cjob
2	Caddr
2	Sname

Cond

CId	QId	Name	Const
1	1	Cjob	Professor
1	1	Caddr	Blois
1	2	Cjob	Lawyer
1	2	Caddr	Paris
1	2	Ptype	Milk
1	3	Cjob	Lawyer
1	3	Ptype	Bread
1	4	Cjob	Professor
1	4	Ptype	Bread
2	5	Sname	Carrefour
2	5	Caddr	Paris
2	5	Cjob	Lawyer
2	6	Sname	Ikea
2	6	Caddr	Paris
2	6	Cjob	Lawyer
2	7	Caddr	Paris
2	7	Sname	Fnac

Figure 1: Storage of the frequent queries of two contexts with identifiers 1 and 2.

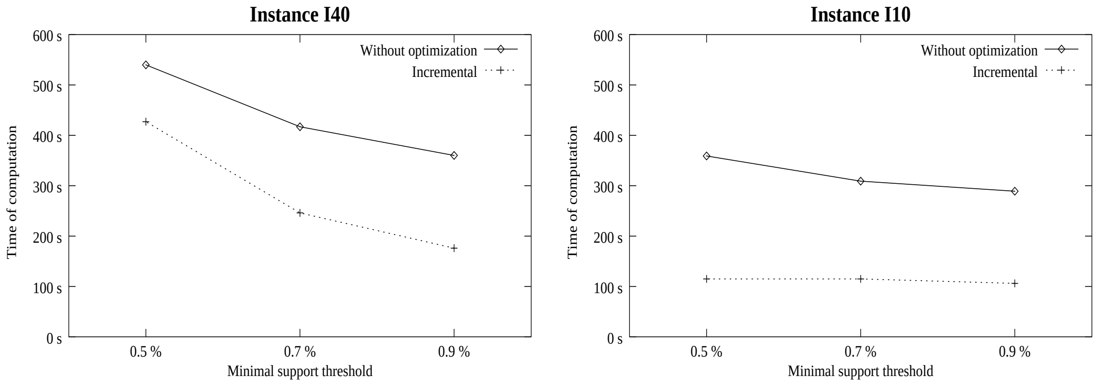

Figure 2: Time of computation without optimization and using our incremental approach