Ranking Approximate Query Rewritings Based on Views

Hélène Jaudoin, Pierre Colomb, Olivier Pivert

Ranking approximate query rewritings based on views

Hélène Jaudoin[1], Pierre Colomb[2], and Olivier Pivert[1]

[1] IRISA-ENSSAT, Université de Rennes 1, France
[2] LIMOS, Université Blaise Pascal, France
jaudoin@enssat.fr, colomb@isima.fr,pivert@enssat.fr

Abstract. This paper considers the problem of rewriting queries using views by means of a tolerant method. The approach proposed is based on an approximate matching between the constraints from the query and those from the views, in the case where both the query and the views contain arithmetical constraints expressed as intervals. In such a context, the answers obtained are not certain anymore but only more or less probable. An algorithm which retrieves the top-k rewritings of a given query is described. Experimentations are reported, which show that the extra cost induced by the approximate nature of the rewriting process is perfectly acceptable.

1 Introduction

An integration system provides a uniform query interface to distributed data sources on a network. The problem of answering queries in integration systems has been intensively studied during the last decade [10]. In particular, it has been shown that the semantics of queries in such a setting can be formalized in terms of certain answers [1]. Intuitively, a certain answer to a query Q over a global schema — i.e., a uniform interface — with respect to a set of source instances is an answer to Q in any database over that interface that is consistent with the source instances. The problem of answering queries in integration systems can be formalized as the problem of computing all the certain answers to the queries. A technique to effectively computing the certain answers to a query in Local-As-View based integration systems (i.e., where the data sources are defined as views over the global schema) is to reduce this problem into that of *rewriting queries using views*. Given a user query expressed over the global schema, the data sources that are relevant to answer the query are selected by means of a rewriting algorithm that allows to reformulate the user query into an equivalent or maximally *contained* query whose definition refers only to *views*. Any such query rewriting must satisfy all the constraints conveyed by a given query Q in order to return only correct answers to Q.

However, for integration systems in open environments — like the web where data sources are autonomous —, it can be very problematic to find views that totally satisfy the domain constraints imposed by the query. This raises the

question of relaxing the notion of a mapping between the value domains of the views and those from the query. As an example, let us consider a query Q that aims to retrieve *names* of *persons* whose *age* is in the interval $[28, 38]$ and two views V_1 and V_2 such that:
V_1 supplies *names* of *persons* whose *age* is in $[25, 35]$ and
V_2 supplies *names* of *persons* whose *age* is in $[36, 46]$.
The views V_1 and V_2 both have an interval constraint on attribute *age*. The two intervals are not included in that of the query, thus the mappings between the two intervals and that of the query are only partial (imperfect). Moreover, since V_1 and V_2 only supply names of persons, selection on attribute age is impossible, hence V_1 and V_2 cannot be used to get certain answers to Q. In such a context, rewriting algorithms based on the certain answer semantics fail to reformulate the query, thus to provide the user with any answer.

In this paper, we assume an integration system based on a SuperPeer architecture [3] where each SuperPeer acts as a mediation system based on a Local-As-View (LAV) approach. Each SuperPeer stores a global schema and a set of views describing queries that can be performed on data sources connected to the SuperPeer. Queries are submitted to a given SuperPeer that rewrites them in terms of its views. We consider that views and queries are expressed in the formal setting of Datalog and that they involve interval constraints. The idea we advocate is to exploit approximate mappings between the constraints from the views and those from the queries in order to compute *approximate* query rewritings. Any such rewriting Q' is associated with a score between 0 and 1 which reflects the probability for a tuple returned by Q' to satisfy the initial query Q. Consequently, the rewriting mechanism is not based on the notion of certain answers anymore, but on that of *probable answers*. Since approximate mappings between queries and views are obviously more frequent than strict mappings, the number of possible approximate rewritings may be huge. To cope with this situation, we propose an algorithm which generates only the top-k rewritings to a given query. This algorithm has been implemented and some experimentations on real data have been conducted. They show that the extra cost induced by the approximate nature of the approach (with respect to regular query rewriting algorithms) is perfectly acceptable.

Even though the notion of gradedness is central to the approach proposed here, it is important to emphasize that the problem tackled is essentially different from the issue of rewriting top-k queries using views dealt with, e.g., in [6]. In the work presented here, only regular (i.e., Boolean) queries are considered and the goal is not to compute a set of graded tuples, as in [6], but rather to compute the most satisfactory sets of answers, corresponding to the best rewritings of the query. In other words, we are not interested in rewriting top-k queries, but rather in computing the top-k rewritings of regular queries. Moreover, although the problem of rewriting queries using views in the presence of interval constraints is well known and the corresponding rewriting language has been shown to be a union of conjunctive queries with semi-interval constraints [13, 2], our contribution is to tackle this problem by means of a *tolerant* method. The

purpose of our approach also differs from [7] which deals with the problem of rewriting queries in the presence of imprecise view descriptions. In our approach, imprecision results from the approximative nature of the mappings considered, it does not concern the description of the data.

The remainder of the paper is organized as follows. Section 2 introduces the general principle of our approach and defines the notion of an approximate rewriting while Section 3 describes the algorithm aimed to compute the top-k approximate rewritings of a user query. Section 4 presents some experimentations and notably deals with the performance aspect, whereas Section 5 concludes the paper.

2 Approximate rewriting using views

2.1 General objective

We study the problem of ranking query rewritings based on views in the setting of conjunctive queries with simple interval constraints $X_i \in [a, b]$, where X_i is an attribute and a and b are constants. Such a query is an expression of the form:

$$Q(Y) : -r_1(Y_1), \ldots, r_n(Y_n), C_1, \ldots, C_m$$

where Q, r_1, ..., r_n are predicate names, and Y, Y_1,..., Y_n are sets of variables, i.e., sets of attributes names, and $Y \subseteq \cup_{j=1}^{n} Y_j$. The atoms $r_1(Y_1), \ldots, r_n(Y_n)$ are the subgoals of the query where $r_1, \ldots, r_n$ denote relations of the global schema. The atom $Q(Y)$ is called the head of the query and represents its result. Each C_i, $i \in [1, m]$ is an interval constraint represented hereafter by $X_i \in I_{X_i}$ where X_i belongs to $\cup_{j=1}^{n} Y_j$ and I_{X_i} denotes an interval $[a, b]$.

As an example, let us consider a global schema $\mathcal{S}$ made of the relations $Person(ssn, name, firstname)$, $Child(ssn, Cfirstname, Cage)$ and $Emp(ssn, job, sal)$, and a query Q on $\mathcal{S}$ aimed to retrieve the social security number, name and first name of every person who has a child between 22 and 35 years old and a salary between 1200 and 2300 euros. Q can be expressed as: $Q(nss, name) : -Person(nss, name, firstname)$, $Child(nss, Cfirstname, Cage)$, $Emp(nss, job, sal)$, $Cage \in A_Q$, $sal \in S_Q$ with $A_Q = [22, 35]$ and $S_Q = [1200, 2300]$.

A *view* is a named query which uses relations from the global schema $\mathcal{S}$. The idea we advocate in this paper is to reformulate a given query Q into expressions based on views which satisfy the interval constraints attached to Q *as well as possible*. Every such *approximate rewriting* is associated with a degree $\alpha \in]0, 1]$ which expresses the *probability* for a tuple that it returns to be a certain answer to Q.

Definition 1 (Approximate rewriting). *Let Q be a query, $\overline{Q}$ the query Q without its interval constraints and $\mathcal{V}$ a set of view definitions. Query Q' is an approximate rewriting of Q using $\mathcal{V}$ if:*

- *Q' is a conjunction of views from $\mathcal{V}$,*

– $Q' \sqsubseteq \overline{Q}$, *and*
– $Q' \sqsubseteq_\alpha Q$, *with* $\alpha \in]0, 1]$.

In this definition, $Q' \sqsubseteq_\alpha Q$ means that Q' is approximately contained in Q at a degree α, according to the *tolerant inclusion* which will be defined in the next subsection. An approximate rewriting is then a regular rewriting in terms of views of a query when interval constraints are omitted. However when interval constraints are considered, the approximate rewriting approximates the interval constraints of the query. In addition to certain answers, such approximate rewritings provide answers that are likely to satisfy the initial query. Since a degree is associated with every approximate rewriting, it is possible to order them. Therefore, instead of computing all the possible approximate rewritings of Q — which can be costly and not very useful for the user —, we focus in the following on the problem of computing only a meaningful subset of them: either the top-k ones (k being an integer) or those whose associated degree is over a given threshold $\alpha_{min} \in]0, 1]$. The computation of such approximate rewritings requires two main steps: i) the research of candidate approximate rewritings, i.e., the regular query rewritings of the query when interval constraints are omitted, and ii) the assessment of the candidate approximate rewritings. As to the first step, we can adapt regular algorithms aimed at rewriting queries using views such as, for example, the `MiniCon` [13]. However, one must define a new algorithm in order to assess and rank approximate rewritings of a given query.

Concretely, the assessment of an approximate rewriting is based on the proportion of answers that it returns which satisfy the constraints from the user's query. Hereafter, we give a more formal definition of the notion of *tolerant inclusion* between two intervals that founds the semantics of the approximate rewriting approach.

2.2 Tolerant inclusion

Let $I_{Q'}$ and I_Q be two intervals. We define the tolerant inclusion of $I_{Q'}$ in I_Q in the following way:

$$deg(I_{Q'} \subseteq_{tol} I_Q) = \alpha = \frac{\mid I_{Q'} \cap I_Q \mid}{\mid I_{Q'} \mid} \tag{1}$$

where $|I|$ denotes the cardinality of I. Let $I_Q = [a, b]$, $I_{Q'} = [c, d]$, and $I_Q \cap I_{Q'} = [e, f]$, and let us assume that the attribute considered is encoded by numbers with n decimals ($n \geq 0$), we get:

$$deg(I_{Q'} \subseteq_{tol} I_Q) = \frac{(f - e) \times 10^n + 1}{(d - c) \times 10^n + 1}.$$

Obviously, this degree corresponds to the proportion of elements from $I_{Q'}$ which are in $I_{Q'} \cap I_Q$ when the distribution of the values over the domain is uniform.

Equation 1 can be straightforwardly adapted to the case where one has available some *histogram* or *distribution function* describing the value distribution

over the domain. One uses the distribution function F the following way. Assume again that $I_Q = [a, b]$, $I_{Q'} = [c, d]$, and $I_Q \cap I_{Q'} = [e, f]$. The degree $\deg(I_{Q'} \subseteq_{tol} I_Q)$ is given by:

$$\alpha = \frac{\sum_{x \in [e, f]} F(x)}{\sum_{x \in [c, d]} F(x)}. \tag{2}$$

2.3 From tolerant inclusion to approximate rewritings

In this section, we exploit the notion of a tolerant inclusion to make query rewriting using views a *graded* process. We consider a user query Q expressed in terms of a global schema $\mathcal{S}$ and a candidate approximate rewriting Q' based on views, also expressed in terms of $\mathcal{S}$, such that $Q' \sqsubseteq \overline{Q}$. Taking the interval constraints into account leads to computing tolerant inclusion degrees between intervals involved in Q and Q' according to the principle described above, then to aggregating these degrees if several intervals are involved. The final degree obtained can be seen as the satisfaction degree attached to the rewriting as a whole. Due to the semantics of Equations 1 and 2, it expresses the *probability* for an answer returned by the approximate rewriting Q' to be an answer to the initial query Q.

First, let us consider the situation where Q has a constraint on a single attribute X. It is worth noting that when Q' has no interval constraint on X, the active domain of X is used as the interval constraint on X in Q'. When Q has only one interval constraint, two cases must be considered: a) the constraint is attached to a *distinguished variable* in Q', i.e., X appears in the head of at least one of the views defining Q' and, b) the constraint is attached to an *existential variable* in Q', i.e., X does not appear in the head of the views from Q'. The following proposition states how the satisfaction degree attached to Q' can be obtained in each case.

Proposition 1. *Let Q be a query and Q' a conjunction of views such that $Q' \sqsubseteq \overline{Q}$. Let X be the single attribute on which Q has an interval constraint. Let I_Q and $I_{Q'}$ be the associated intervals in Q and Q' respectively (if $I_{Q'}$ does not explicitly exist, the active domain of the corresponding attribute is used). Let α be the degree of tolerant containment of Q' in Q ($Q' \sqsubseteq_\alpha Q$).*

a) if X is a distinguished variable of a view from Q' then I_Q is added to Q' and
- *if $I_Q \cap I_{Q'} = \emptyset$ then $\alpha = 0$*
- *else $\alpha = 1$.*

b) if X is an existential variable in the views from Q' then $\alpha = deg(I_{Q'} \subseteq_{tol} I_Q)$.

Consequently, the degree attached to an approximate rewriting is either 0, 1 or the inclusion degree between I_Q and $I_{Q'}$. When α is 0, Q' does not provide any probable answer and is not considered an approximate rewriting. The proof of this proposition is given hereafter.

- Case a): As X is a distinguished variable of at least one view from Q', there may exist a join over X between some views from Q'. Moreover, one or several views from Q' may have an interval constraint on X. Let $\{I_1, \ldots, I_k\}$ be the set of those intervals and t an answer returned by Q'. Then, the projection of t on X belongs to the interval $I_{Q'} = I_1 \cap \ldots \cap I_k$.
  As X is a distinguished variable of at least one view from Q', one can apply a condition on X and notably, the interval constraint I_Q, as done in [13], in addition to $I_{Q'} = I_1 \cap \ldots \cap I_k$. Therefore, two subcases must be considered. Either $I_Q \cap I_{Q'}$ is empty and Q' does not provide any probable answer to Q: then degree 0 is attached to Q'; or $I_Q \cap I_{Q'}$ is not empty and Q' with the constraint $I_Q \cap I_{Q'}$, returns only certain answers to Q since only the "compatible part" of the constraints was kept: then degree 1 is attached to Q'.
- Case b): X is an existential variable in all the views from Q'. As no selection and no join on X are possible ([13]), only one view of Q' covers the subgoals of Q involving X. Therefore, only this view has an interval constraint $I_{Q'}$ on X.
  Let t be an answer returned by Q'. Then the projection of t on X belongs to $I_{Q'}$. However $I_{Q'}$ may not be contained in I_Q. In this case, the satisfaction degree of Q' with respect to Q must be based on a tolerant inclusion between I_Q and $I_{Q'}$ (cf. Equations 1 and 2).

In accordance with Equations 1 and 2, the degree obtained denotes the *probability* that the projection on X of an answer to Q' be in I_Q, knowing that the projection of answers to Q' on attribute X is in the interval $I_{Q'}$. More generally, the degree obtained with a such procedure specifies the probability for answers to Q' to be certain answers to Q. When constraints are associated with distinguished variables, query constraints are added to the rewritings. Therefore, either we get no correct answer and α is 0 or we get only certain answers and α is 1.

The following example illustrates the former case.

Example 1. Let us consider the query:
$Q(N, A, S) : -Man(N, A), Salary(A, S),\ A \in I_{A_Q} = [28, 38]$
and the two views V_6 and V_7 defined as:
$V_6(N_1, A_1) : -Man(N_1, A_1),\ A_1 \in I_{A_1} = [25, 40]$
$V_7(A_2, S_2) : -Salary(A_2, S_2),\ A_2 \in I_{A_2} = [27, 45]$.
A possible rewriting of $Q(N, A, S)$ is $Q'(N, A, S) : -V_6(N, A), V_7(A, S)$ where attribute A appears in the head of V_6 and V_7, i.e., A is a distinguished variable in V_6 and V_7. The final constraint applying to Q' over attribute A would then be the interval $[27, 40]$. However, since A is a distinguished variable, it can be restricted by a selection and the only possible rewriting is $Q''(N, A, S) : -V_6(N, A),\ V_7(A, S),\ A \in [28, 38]$. The interval constraint added to Q' is that from Q. The degree attached to $Q''(N, A, S)$ is then 1 since the intersection between $[27, 40]$ and $[28, 38]$ is not empty: $Q'' \sqsubseteq_1 Q$.$\diamond$

The example below illustrates Case b) of Proposition 1.

Example 2. Let us consider the query Q:
$Q(ssn, name) :- Person(ssn, name, fname),\ Child(ssn, Cfirstname, Cage),$
$Cage \in [22, 35]$
and Q' the candidate rewriting of Q:
$Q'(ssn, name) : -V_8(ssn, name)$
where
$V_8(ssn, name) :- Person(ssn, name, fname),\ Child(ssn, Cfirstname, Cage),$
$Emp(ssn, job, sal),\ Cage \in [20, 40]$.
The attribute $Cage$ does not apear in the head of V_8 and consequently is existential in V_8. If we ignore the interval constraints, Q' is contained in Q. On the other hand, the presence of interval constraints leads to discard Q' if a Boolean matching is performed. Using a tolerant matching, the "containment degree" of Q' in Q is based on Equation 1. It is the extent to which interval $[20, 40]$ is included in $[22, 35]$. If one assumes that attribute *age* is encoded by integers, one gets the inclusion degree $\alpha = \frac{(35-22)*10^0+1}{(40-20)*10^0+1} = \frac{14}{21}$. Hence: $Q' \sqsubseteq_{0.67} Q$.$\diamond$

Let us now consider the general case where Q involves constraints on *several* attributes. In this case, the problem is to compute the probability for an answer returned by Q' to be an answer to Q.

Proposition 2. *Let Q be a query and Q' be a view-based expression such that $Q' \sqsubseteq \overline{Q}$. Let $\mathcal{C}_\mathcal{Q}$ be the set of interval constraints — assumed to be independent — in Q such that $|\mathcal{C}_Q| = n$. Let α_i be the degree attached to Q' for any constraint $c_i \in \mathcal{C}_\mathcal{Q}$, obtained with the criteria of Proposition 1. The degree α of tolerant containment of Q' in Q ($Q' \sqsubseteq_\alpha Q$) is given by:*

$$\alpha = \prod_{i=1}^{n} \alpha_i$$

Proof. Under the independence assumption of the interval constraints, if α_i denotes the probability that an answer to Q' satisfies constraint c_i from Q, then the probability α that it satisfies all of the constraints from Q is the product of the α_i's.

The following example illustrates the computation of the overall degree attached to a given rewriting in such a complex case.

Example 3. Let us consider the following views:
$V_1(ssn_1, n_1)\ : - Person(ssn_1, n_1, f_1),$
$V_2(ssn_2)\ : - Child(ssn_2, cf_2, a_2),\ a_2 \in [20, 40],$
$V_3(ssn_3)\ : - Emp(ssn_3, j_3, s_3),\ s_3 \in [2100, 3400],$
$V_5(ssn_5)\ : - Child(ssn_5, cf_5, a_5),\ Emp(ssn_5, j_5, s_5),\ a_5 \in [18, 38],$
$s_5 \in [1000, 2400]$.
None of these views has any constraint on distinguished variables. Let us now consider the query:
$Q(ssn, name) : -Person(ssn, name, fname),\ Child(ssn, Cfirstname, Cage),$
$Emp(ssn, job, sal),\ Cage \in [22, 35],\ sal \in [1200, 2300]$
and the approximate rewritings:

$Q_1(nss, nom) : -V_1(nss, nom), V_2(nss), V_3(nss)$ and
$Q_2(nss, nom) : -V_1(nss, nom), V_5(nss)$.
Assuming that the attributes are encoded by integers, the degree attached to:

- the rewriting V_1 of the first subgoal of Q is $\alpha_1 = 1$ as there is no interval constraints for this subgoal.
- the approximate rewriting V_2 of the second subgoal of Q is computed from the interval constraints on age. It is $\alpha_2 = \frac{(35-22)+1}{(40-20)+1} = 14/21 = 0.67$.
- the approximate rewriting V_3 of the third subgoal is computed from the interval constraints on salary. It is $\alpha_3 = \frac{(2300-2100)+1}{(3400-2100)+1} = 201/1301 = 0.15$.
- the approximate rewriting V_5 of the second and the third subgoals of Q is computed from interval constraints on age and salary. It is:
  $\alpha_4 = \frac{(35-22)+1}{(38-18)+1} * \frac{(2300-1200)+1}{(2400-1000)+1} = 14/21 * 1101/1401 = 0.52$.

Therefore, $Q_1(nss, nom)$ gets the degree $1 * 0.67 * 0.15 = 0.1$ while $Q_2(nss, nom)$ gets $1 * 0.52$ ($Q_1 \sqsubseteq_{0.1} Q$ and $Q_2 \sqsubseteq_{0.52} Q$). The tuples issued from Q_1 are not likely to satisfy the query while those from Q_2 have a probability over 50%.$\diamond$

3 An algorithm for computing the top-k rewritings

Instead of computing all the approximate rewritings of a query and then ranking them, we propose to adapt a well-known regular query rewriting algorithm, namely `Minicon` [13], in order to directly generate the k best rewritings ranked in decreasing order of their degree. `MiniCon` computes the maximally-contained rewriting of a given query Q, i.e., that which provides all the certain answers to Q, in the case where queries and views involve semi-interval constraints. The first step of this algorithm consists in enumerating all the so-called *MiniCon descriptions* (denoted by *MCD's*) of Q. Intuitively, the existence of an MCD associated with a view denotes that the view covers a subset of the query subgoals. The second step is devoted to the computation of MCD combinations which cover every subgoal in Q and which are pairwise disjoint w.r.t. the covered subgoals.

When tolerance comes into play, the first step of the `MiniCon` algorithm must be slightly modified in order to associate a degree α_M with every MCD M of a given view V. First, the regular `MiniCon` procedure $formMCD(V, M)$ is used to check if an MCD can be created with V. If it is the case, the degree to be associated with M is computed on the basis of Proposition 1 or 2.

As to the second step of the `MiniCon` algorithm, it can be modeled by a hypergraph [4] where query subgoals correspond to vertices and each set of query subgoals covered by an MCD corresponds to an edge. Any satisfactory combination of MCDs corresponds to an *exact cover* of this hypergraph. This type of object is strongly related with the notion of a *minimal transversal* [8, 9] in hypergraph theory. Even though determining whether a given hypergraph possesses at least one exact cover is an NP-Complete problem [11], some scalable algorithms exist for computing all the exact covers of a hypergraph [12]. Such algorithms can be used to get an efficient implementation of `MiniCon`. Algorithm 1

presented below belongs to this family; it generates either the k best rewritings of a query, or all the rewritings whose degree is over a given threshold α_{min}. The rewritings are generated in descending order of their degree. Therefore, the k best rewritings of a query are those k first produced.

Our algorithm maintains an ordered list of *partial rewriting* of the considered query. A partial rewriting is a set of MCDs, pairwise disjoint, that does not cover all the subgoals of the query.

Algorithm 1 ComputeRW(X, $\mathcal{M}$)

Require: $\mathcal{M}$ the set of weighted MCDs
Ensure: The set $RW(Q, \mathcal{V})$ of rewritings of Q using $\mathcal{V}$
1: $RW(Q, \mathcal{V}) = \emptyset$
2: $PartialRW = \mathcal{M}$
3: **while** $PartialRW \neq \emptyset$ **do**
4: $M = PartialRW.max()$;
5: $PartialRW = PartialRW \setminus M$;
6: **if** M covers Q **then**
7: $RW(Q, \mathcal{V}) = RW(Q, \mathcal{V}) \cup M$;
8: **else**
9: $\mathcal{M}$.cover(M);
10: **for** $m \in \mathcal{M}$ **do**
11: **if** $M.\alpha * m.\alpha \geq \alpha_{min}$ **then**
12: $PartialRW.Add(M + m)$;
13: **end if**
14: **end for**
15: $\mathcal{M}$.uncover(M);
16: **end if**
17: **end while**

Given a partial rewriting M and an MCD m covering others subgoals, we denote by $(M+m)$ the (partial) rewriting corresponding to the concatenation of M and m. Moreover, we denote by $M.\alpha$ the degree of a partial rewriting M. Let us remind than according to previous propositions, the degree of a concatenation $(M + m).\alpha$ is given by the product of $M.\alpha$ and $m.\alpha$. Algorithm 1 takes as an input the set $\mathcal{M}$ of weighted MCDs returned by the first step, stored in a data structure which offers the following operations:

- cover(M) deletes all the MCDs sharing a subgoal with M,
- uncover(M) re-inserts the MCDs sharing a subgoal with M previously deleted by cover(M).

When we use a MCD M in a partial rewriting, we must forbid the use of all MCDs intersecting M. Indeed, every subgoal of the query must be covered only once. The operation *cover*() is used to do so. On the other hand, the purpose of the function *uncover*() is to re-allow the use of an MCD in the rest of process. Notice that $\mathcal{M}.cover(X).uncover(X) = \mathcal{M}$ and the use of $uncover(X)$ always

follows the use of an operation *cover*(). This data structure can be implemented very efficiently using *Dancing Links* [12], which are a structure based on circular doubly-linked lists to implement binary matrix. It makes it possible to implement the operations *cover* and *uncover* with a complexity linear in the number of deleted/inserted MCDs.

The ordered generation of the rewritings is based on the property that the degree of a partial rewriting M can only decrease when it is combined with another MCD m. This property ensues from the fact that the degrees handled are probabilities, therefore real numbers between 0 and 1.

Algorithm 1 considers the partial rewritings in descending order of their degree and tries to complete them so as to generate the set $RW(Q, \mathcal{V})$ of rewritings of the query Q considered. It first considers the partial rewritings made of a single MCD (line 2). Then, as long as there are still partial rewritings which may be completed, it considers the partial rewriting X of maximal degree (line 4), checks if it is a rewriting of Q (line 6), and tries to complete it if necessary (lines $9 - 15$). For doing so, one first removes from $\mathcal{M}$ all the MCDs which share a subgoal with X, then one tries to form a (partial) rewriting with each of the remaining MCDs. Finally, the MCDs previously removed are reinserted before the next iteration.

4 Experimentations

The approach has been implemented in a prototype using Java SE 5, and PostgreSQL 8.3 databases. We performed experimentations using sources of agricultural data consisting of 310 tables containing about 600 Mb of data. Twenty queries provided by real users were processed. In this experimental context, all data sources are related to the same application area. Such a context is convenient to find exact rewritings, and consequently certain answers (which explains the large number of certain answers retrieved). The following results show the evolution of the average number of tuples corresponding to certain and probable answers to queries.

The first experimentation shows the evolution of the number of answers when the threshold α_{min} changes. Here the parameter k is fixed to infinity, i.e., all the rewritings are generated. Notice that the number of certain answers remains constant. Indeed, all exact rewritings are computed for any value of α_{min}. On the other hand, the number of probable answers decreases when the threshold increases. When α_{min} equals 1, only certain answers are provided.

In the second experimentation, the threshold is set to 0 but k, the number of best rewritings sought for, must be specified. When k decreases, the number of probable answers decreases too, which shows that only the best rewritings are generated. When k is less than 10, the number of certain answers decreases too. Indeed, for some queries, a number of rewritings less than 10 is not always enough to compute all the certain answers.

The prototype shows very good performances: the execution time is about 2 seconds in the worst case, and the performances of the approximate rewriting

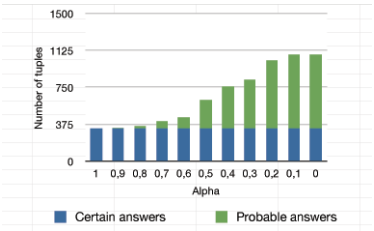

Fig. 1. Impact of α_{min} on the number of answers

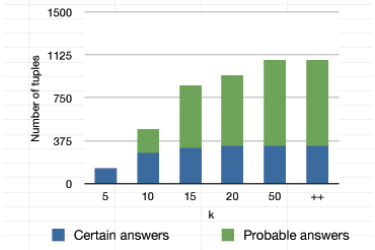

Fig. 2. Impact of k on the number of answers

approach are comparable to that of the classical `MiniCon` algorithm (in the worst case, the overhead is only 10%). As mentioned above, the use of appropriate data structures makes our algorithm highly scalable: it can efficiently manage the extra MCDs induced by the approximate nature of the rewriting process.

5 Conclusion

This paper deals with the problem of getting ranked approximate rewritings in order to process only the best ones, thus reducing the processing cost of a query in a decentralized database context. We have proposed an approach to obtain such rewritings on the basis of interval constraints involved in the views and the query. This approach attaches a degree $\in [0, 1]$ to each rewriting Q' of a query Q, which corresponds to the probability for a tuple returned by Q' to be an answer to Q. We have adapted the well-known `MiniCon` algorithm to get only the best rewritings, reducing thereby the combinatorics relative to the potentially high number of MCDs. Experiments show that the extra cost induced by the approximate nature of the matching is perfectly acceptable, while a significant number of additional answers is obtained.

These experiments also made it possible to estimate — in a specific context — the amount of additional answers provided by this approach with respect to the number of certain answers. However, it is worth noticing that the answers produced may not satisfy all of the criteria from the initial user query. So as to make the user aware of this state of fact, a simple solution consists in associating a degree with each answer: that attached to the approximate rewriting which produced the answer in question. Concerning this aspect, it would be interesting to complete the experimentations by some tests involving real users so as to assess their overall satisfaction regarding the non-certain answers that they are provided with.

Generally speaking, the approach described here applies to a particular context, namely the Web where exact mappings between the descriptions of data sources and a given user query are not always possible. In such a context, the answers to a user query can only be computed on the basis of the information

which is available, even if it is pervaded with uncertainty. Flexible rewriting approaches allow to provide users with answers where classical rewriting algorithms based on the semantics of certain answers fail. The idea of computing *probable* answers is not exactly new since it has been used for open integration contexts in the works by Dalvi and Suciu [5] and by Das Sarma *et al.* [14]. The nature of the probability degree associated with the answers returned in our approach is different from that considered in these works, though. Indeed, in [5] it indicates the probability of *existence* of the answers produced, while in [14] it ensues from the uncertainty pervading the mappings between several data sources.

In terms of perspectives, we plan to investigate other, non-probabilistic, semantics to approximate query rewriting using views, based on different visions of query relaxation.

Acknowledgement. Work partially funded by the ANR Research Grant ANR-05-MMSA-0007.

References

1. S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *Proc. of PODS 1998*, pages 254–263, 1998.
2. F. Afrati, C. Li, and P. Mitra. Rewriting queries using views in the presence of arithmetic comparisons. *Theoretical Computer Science*, 368:88–123, 2006.
3. Z. Bellahsene and M. Roantree. Querying distributed data in a super-peer based architecture. In *Proc. of the 15th Int. Workshop on Database and Expert Systems Applications (DEXA'04)*, pages 296–305, 2004.
4. C. Berge. *Hypergraphs*. North-Holland, 1989.
5. N. Dalvi and D. Suciu. Answering queries from statistics and probabilistic views. In *Proc. of VLDB 2005*, pages 805–816. VLDB Endowment, 2005.
6. G. Das, D. Gunopulos, N. Koudas, and D. Tsirogiannis. Answering top-k queries using views. In *Proc. of VLDB 2006*, pages 451–462. VLDB Endowment, 2006.
7. X. Dong, A. Halevy, and C. Yu. Data integration with uncertainty. In *Proc. of VLDB 2007*, pages 687–698. VLDB Endowment, 2007.
8. T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J.*, 24(6), 1995.
9. T. Eiter, K. Makino, and G. Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Appl. Math. DOI*, 10, 2007.
10. A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
11. R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 43:85–103, 1972.
12. D.E. Knuth. Dancing links. *Arxiv preprint cs.DS/0011047*, 2000.
13. R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *Proc. of VLDB 2000*, pages 484–495, San Francisco, CA, USA, 2000.
14. A. Das Sarma, X. Dong, and A. Halevy. Bootstrapping pay-as-you-go data integration systems. In *Proc. of SIGMOD 2008*, pages 861–874. ACM, 2008.