Source Selection in Large Scale Data Contexts: An Optimization Approach*

Alexandra Pomares[1,2,3], Claudia Roncancio[1], Van-Dat Cung[1], José Abásolo[2], and María-del-Pilar Villamil[2]

[1] Grenoble INP, Grenoble, France
[2] Universidad de los Andes, Bogotá, Colombia
[3] Pontificia Universidad Javeriana, Bogotá, Colombia

Abstract. This paper presents OptiSource, a novel approach of source selection that reduces the number of data sources accessed during query evaluation in large scale distributed data contexts. These contexts are typical of large scale Virtual Organizations (VO) where autonomous organizations share data about a group of domain concepts (e.g. patient, gene). The instances of such concepts are constructed from non-disjointed fragments provided by several local data sources. Such sources overlap in a non mastered way making data location uncertain. This fact, in addition to the absence of reliable statistics on source contents and the large number of sources, make current proposals unsuitable in terms of response quality and/or response time. OptiSource optimizes source selection by taking advantage of organizational aspects of VOs to predict the benefit of using a source. It uses an optimization model to distinguish the sets of sources that maximize benefits and minimize the number of sources to contact to while satisfying resource constraints. The precision and recall of source selection is highly improved as demonstrated by the tests performed with the OptiSource prototype.

Keywords: Large Scale Data Mediation, Source Selection, Combinatorial Optimization.

1 Introduction

A Virtual Organization (VO) is a set of autonomous collaborating organizations, called VO units, working towards a common goal. It enables disparate groups to share competencies and resources such as data and computing resources [1]. VOs have evolved to a national and world-wide magnitude [2,3], introducing complex business processes and data sharing contexts.

This work deals with large scale VOs following co-alliance and value-alliance models, where participants provide data related to a group of agreed domain concepts (e.g. patient, client, gene). The VO data context involves a large number of autonomous and distributed sources (tens to thousands), provided by VO units. Sources contain structured data, typically fragmented horizontally

* This research was supported by the project Ecos-Colciencias C06M02.

and vertically w.r.t. the domain concepts of the VO. Therefore sources may be incomplete intentionally (schema) and extensionally (contents). Moreover, participant interactions produce logical relationships between sources generating uncontrolled situations of data overlapping and data replication. In most cases, query processing becomes difficult and heavily resource dependent because the mediation level does not have precise knowledge of the contents of each source. Considering the large scale context, a tradeoff between the number of answers to obtain and the number of sources (potential contributors) to access becomes crucial, especially while not compromising the accuracy of query processing.

Current source selection approaches are unsuited for this kind of contexts because they do not significantly reduce the number of sources to access when their schemas are similar [4,5,6,7,8] like is the case in VOs. Others do not take into account source overlapping leading to redundant answers when used in VOs [9,10,11,12]. Proposals such as [13,8] do consider overlapping, but assume the existence of some metadata that is difficult to obtain and maintain in large VOs.

Contribution. This paper presents OptiSource, a new approach that addresses the source selection problem[1] as a decision making problem within a complex environment. OptiSource predicts the benefit of using a source during query evaluation, and optimizes the assignment of query predicate evaluation to sources. It uses a combinatorial optimization model to distinguish the sources that maximize the benefit and minimize the number of sources to contact to while satisfying processing resource constraints. OptiSource is well suited when sources are numerous, when the instances that match the query are fragmented and distributed in several sources, and when not exhaustive[2] answers are acceptable. A prototype of OptiSource is operational. This paper reports test results, which demonstrate the improvement of the precision and recall of source selection.

Outline. In the following, Section 2 presents the context of the problem and related works. Section 3 presents OptiSource. Section 4 presents the combinatorial optimization model used in OptiSource. Section 5 presents the OptiSource prototype and its evaluation. Finally, Section 6 concludes this paper.

2 Data Source Selection: Context and Related Works

The selection of the right set of data sources to evaluate a query is one of the crucial steps to assure an efficient query evaluation on large scale distributed systems. Considering all the available sources is of course unsuitable because the number of possible query plans grows rapidly as the complexity of the query and the number of sources increase. This section first points out the main characteristics of VO's data context and then discusses related works.

2.1 Data Context in Large Scale Virtual Organizations

In large scale VOs, VO units (participants) are distributed in a wide area network. They share preexisting heterogeneous sources following a federated approach and

[1] Henceforth, source and data source are going to be used interchangeably.

[2] In these contexts a sub set of the available answers is enough.

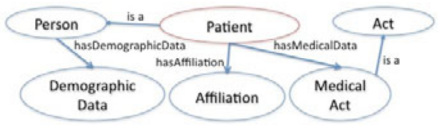

Fig. 1. VDO Patient

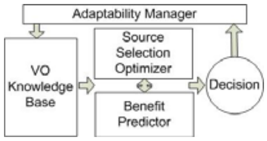

Fig. 2. OptiSource Components

according predefined cooperation agreements [14]. Shared data are related to these concepts (e.g. patient, client, gen), henceforth to be referred to as Virtual Data Objects (VDO). An instance of a VDO may not exist completely in one single source, but may be distributed across several sources. The creation of such an instance requires the integration of data issued by several sources.

A VDO is defined by a set of classes and related properties. Figure 1 presents the definition of the VDO Patient. It inherits from the class *Person*, and is linked to classes *Medical Act* and *Affiliation*. *Person* is linked to the class *Demographic Data*. VDO Queries specify the set of required conditions of the instances.

Data characteristics. Business processes involving the VO units have a significant impact on VDO instances distribution. They introduce data source fragmentation and overlapping without providing a global description of this situation to the system. The following are the most important of these aspects, which affect source selection:

-Non disjoint vertical fragmentation. VDO properties are fragmented between different sources. However, fragments are not disjointed which means that a VDO's property may be provided by several sources.

-Non disjoint horizontal fragmentation. Several sources may provide the same fragment or instance of a VDO, but may also provide different groups of properties and values related to the same VDO.

-Uncertainty on data location. Instances of VDOs follow uncertain pattern distribution. Indeed, two instances may be fragmented and distributed in completely different ways. Such fragmentation is not intentionally designed, but results from preexisting data sources and various business processes.

-Fuzzy copies. A phenomenon of inaccurate copies[3] may even appear, particularly if there are integration efforts of VO unit's subgroups. In a health VO, instances of *Patient* may be replicated due to the mobility of people whereas batch replication may be introduced by reporting governmental politics.

Data distribution scenario. Figure 3 illustrates a typical scenario of data distribution in a health VO. Six sources (represented by ovals) provide instances of VDO Patient (represented by rectangles). None of the sources includes all the properties of the class *Patient* (vertical fragmentation) and no source contains all the instances of *Patient* (horizontal fragmentation). Additionally, vertical

[3] Fuzzy copies are copies that do not have an explicit protocol of consistency.

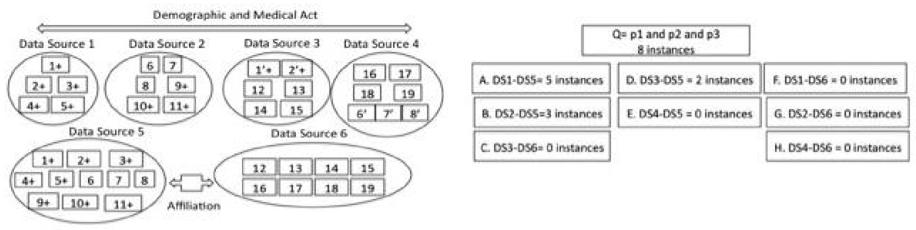

Fig. 3. Health VO Data Context

fragments are not disjointed because multiple sources (DS1 to DS4) provide the properties of the class *Demographic Data* (e.g. gender, age) and of the class *Medical Act* (e.g. diagnosis), and multiple sources (DS5 and DS6) provide the properties of the *Affiliation* class. Horizontal fragments are not disjointed neither. Instances 1 and 2 and instances 6 to 8 are contained in two sources that provide the same properties. Although instances 1 and 2 provided by DS1 and DS3 have the same *id*, it is not possible to establish in advance whether or not they are replicas due to the autonomy of sources. This occurs in a similar way for instances 6,7 and 8 in DS2 and DS4.

Equation (1) presents the logical query plan that provides a complete answer for a query Q asking for instances of *Patient* that match a predicate *p* with three conditions *p1* , *p2* and *p3*. These conditions are related to the properties of classes *Demographic Data*, *Medical Act*, and *Affiliation* respectively (e.g hasGender=F and hasDiagnosis=Cancer and isAffiliatedtoSS=true). This plan does not discard any data source because the query planner does not know how the distribution of instances that match the query is.

$$\begin{aligned} Plan(VDO_{patient}, Q(p1,p2,p3)) = DS1 \bowtie DS5 \sqcup DS2 \bowtie DS5 \sqcup DS3 \bowtie DS5 \sqcup \\ DS4 \bowtie DS5 \sqcup DS1 \bowtie DS6 \sqcup DS2 \bowtie DS6 \sqcup DS3 \bowtie DS6 \sqcup DS4 \bowtie DS6 \end{aligned} \quad (1)$$

However, assuming instances that match these conditions are 1,2,3,4,5,9,10 and 11, it can be noticed that $DS1 \bowtie DS5$ provides more than 60% of the answers (instances 1 to 5). If a complete answer is required, adding $DS2 \bowtie DS5$ will be enough. The problem is how to identify during query planning the sources that best answer the query without knowing exactly which instances are contained by each source. This is the main issue addressed by this paper.

2.2 Related Works

Even though multi-sources querying has been intensively studied for more than 15 years, none of the available strategies for query planning are well suited for large scale VO data contexts. In the first generation of integration systems for structured sources like Information Manifold [4], TSIMMIS [5], DISCO[6] and Minicon [7], source selection is performed according to the capabilities of sources (e.g. the number of required conditions, the attributes that can be restricted). Although such proposals are efficient in several contexts, they have little effect on VOs where the processing capabilities of sources are alike.

Another group of algorithms like iDrips and Streamer [8] could be used in VO contexts. They propose to put in order query plans according to an utility function. The coverage plan is an example of this function that measures the number of instances provided by each query plan. Even though the coverage considers the extensional overlapping between sources, the algorithms do not specify how this information is obtained before the query execution; hence, the utility function directly affects the behavior of the algorithms (see Section 5).

Other proposals like QPIAD [15] and Quality Oriented [16] use detailed statistics on data source quality to reduce the number of possible query plans and therefore sources. Unfortunately, such statistics are difficult to obtain in VOs due to the large number of sources and the high volume of contents. Similarly, the proposal to improve the selection of navigational paths between sources presented in [13] assumes a previous knowledge of the instances in each source and how they are linked together. These detailed metadata is not available in VOs. The proposal of Balancing Providers [17] is well suited for environments where finding one possible query plan is enough to obtain the required set of answers. Nevertheless, it is not appropriate for VO where data sources are incomplete, and several query plans are therefore necessary to obtain a complete answer.

Proposals using a P2P approach (PIER [9], PIAZZA [10], EDUTEL-LA [11], SomeWhere [12]) manage numerous sources. However, as their rewriting and source selection strategies do not take into account source overlapping, they would lead to high redundancy in the answers when used in large scale VOs.

3 OptiSource: A Decision System for Source Selection

This section presents OptiSource, our proposal to handle source selection in large scale VOs. OptiSource reduces the queried sources to the most profitable w.r.t. the query. It takes advantage of the organizational notion of VOs to reduce the uncertainty on data location, and to differentiate sources, even if they overlap.

3.1 Queries and Joining Sets

The source selection process must decide which sources will evaluate each predicate condition (or simply condition) and which source answers are going to be integrated for creating the required instances. Knowing that sources can have common instances, the process must decide for each condition the minimal set of sources required to obtain the maximum number of instances that match the condition, while avoiding the contact of unnecessary sources. VO query processors require an appropriate source selection strategy able to identify joining sets (see Definition 1) that will provide higher benefits to the query.

Definition 1. ***Joining Set.*** *Given a set of data sources and a query, a joining set is a group of data sources able to evaluate all the conditions of the query predicate and whose join operation between the data sources it comprises will not produce an empty set of instances.*

The input of the source selection process are n available data sources DS_i, their mapping (or view) M_i to the VDO (e.g. *Person, MedicalAct*) and the conditions p_j involved in the query predicate:

$$Input = [(DS_1, M_1), (DS_2, M_2), ..., (DS_n, M_n)], [p_1, p_2, ..., p_m].$$

The output is the collection of joining sets: $Output = (JS_1, JS_2, ..., JSp)$,

where $JS_i = (DS_k, p_1), ..., (DS_l, p_m), 1 \leqslant k, l \leqslant n$. A couple (DS_k, p_l) means that the data source DS_k will evaluate the condition p_l. A source can evaluate several conditions p_j and a condition p_j can be evaluated by several sources.

OptiSource will produce the appropriate joining sets and optimize query processing by using a dynamically updated knowledge base to predict the benefit of using particular sources. This knowledge will act as another input for the source selection process.

3.2 Components of OptiSource

The main components of OptiSource (Figure 2) are the VO knowledge base, the benefit predictor, the source selection optimizer, and the adaptability manager. The VO knowledge base maintains knowledge facts and events related to the data context of the VO. This includes intentional (schema) and extensional information (contents) of sources. The knowledge is represented as an ontology in OWL [18] with three initial classes: VOUnit, VOResource and VODomainConcept. These classes are specialized and related to better reflect the evolution of the VO data context. **VOUnit** class represents the participants of the VO. An instance of VOUnit can be atomic or composite. In the latter case, it represents a temporal or a permanent group of atomic VO units working together around a specific collaboration process. **VOResource** class represents the physical or logical resources (like data sources) provided by VO units. Finally, **VODomain-Concept** includes the subclasses that describe the domain of the VO. For a health VO for instance, initial classes are *Patient, Medical Act*, and so on.

The Benefit Predictor uses VO knowledge base to predict the benefit of using a source to evaluate the conditions of a query predicate. The Source Selection Optimizer determines the best assignment of data sources for each condition and produces the joining set that maximizes the benefit, in terms of the number of resulting instances. The Adaptability Manager is in charge of updating the VO knowledge base according to the findings after the execution of a query.

3.3 Benefit Prediction Using Data Source Roles

OptiSource estimates the benefit (see Definition 2) of using a source in the evaluation of a query to identify the level of relevance it will have in the query. *DSa* is more relevant than *DSb* if it provides more instances that match the query predicate, even if both of them have the same schema. In order to predict the benefit we use the available knowledge about the extension (contents) of sources. To obtain this knowledge we work under the assumption that the organizational dimension of VOs allows to relate the role that VO units play in the VO with the contents of the sources they provide. The more complete and reliable the extensional knowledge, the more accurate the measurement of the benefit.

Definition 2. ***Data Source Benefit.*** *Given a Query Q with a query predicate P with conditions* $[p_1, p_2, ..., p_m]$ *the benefit of using a data source* DS_i *to evaluate Q is a variable that measures the degree of contribution of* DS_i *in terms of instances that match one or more* $p_j \in P, 1 \leqslant j \leqslant m$.

Data source's roles. In a health VO, for instance, two data sources *DSa* and *DSb* provide information about procedures performed on patients. Let's assume that *DSa* is provided by a VO unit specialized on cancer whereas *DSb* is provided by a hospital specialized on pediatric procedures. In this case, the *DSa* may be considered specialist of patients with cancer (vo:Patient, hasDiagnosis vo:Cancer) whereas the *DSb* is children specialist (vo:Patient, hasAge $\leq$ 15). A source can therefore play different **roles** as a contributor of instances of a VDO (e.g, vo:Patient) verifying some predicate conditions (e.g, hasAge $\leq$ 15). Roles reflect the ability of sources to resolve conditions. Given the analysis of the roles played by VO units in the business processes of the VO, we propose the following three roles for their data sources: *authority*, *specialist* and *container*.

The definition of each source role is described in Definition 3, 4 and 5. In these definitions all the instances of *VDOj* stored in data source *DSi* are noted $ext(DSi, VDOj)$[4]. U designates the data sources participating in the VO. All the instances of *VDOj* available in the VO are noted $ext(U, VDOj)$. The subset of $ext(DSk, VDOj)$ corresponding to the instance that verifies a condition p is denoted $ext(DSk, VDOj)^p$ and $card()$ is the cardinality function.

Definition 3. ***Authority Role.*** *A data source DSi plays an authority role w.r.t. a condition p in a query on VDOj iff it stores all the instances of VDOj available in U that match p.* $IsAuthority(DSi, VDOj, p) \implies ext(U, VDOj)^p \subset ext(DSi, VDOj)$

Definition 4. ***Specialist Role.*** *A data source DSi plays a specialist role w.r.t. a condition p in a query on VDOj iff most instances of VDOj stored in DSi match p.* $IsSpecialist(DSi, VDOj, p) \implies card(ext(DSi, VDOj)^p) \geq card(ext(DSi, VDOj)^{\neg p})$

Definition 5. ***Container Role.*** *A data source DSi plays a container role w.r.t. a condition p in a query on VDOj iff DSi contains at least one instance of VDOj that matches p.* $IsContainer(DSi, VDOj, p) \implies ext(U, VDOj)^p \cap ext(DSi, VDOj)^p \neq \varnothing$

Data source roles are registered as extensional facts in the VO knowledge base and can be acquired using three approaches: (a) Manually (e.g. expert's or DBA's definition of knowledge), (b) Interpreting the execution of processes, (c) Automatically extracting it from sources of knowledge. In [19] we present strategies for the last two approaches.

[4] This extension contains the object identifiers.

Benefit model. In order to predict the benefit of a source *DSi* w.r.t. a predicate condition *p* the model uses Formula 2. This formula relates the role of the data source and the relative cardinality of the data source. The intention is to take into account the expected relevance of the source, given by the role factor, with the maximum number of instances that a data source may provide (in the best case). This combination of knowledge was necessary because sources that play a specialist role could be less important than sources that play container role if the number of instances that the first group may provide is considerably lesser.

The *RoleFactor()* (Formula 3), returns a value between [0,1] indicating the most important role that a source may play in the query. *ContainerFactor, SpecialistFactor and AuthorityFactor* are constants reflecting the importance of the role.

$$Benefit(DSi, VDOj, p) = \frac{RoleFactor(DSi, VDOj, p) * card(ext(DSi, VDOj))}{\max\{card(ext(Dk, VDOj)), \forall\, Dk \; in \; U \; where \; RoleFactor(DSk, VDOj, p) > 0\}} \tag{2}$$

$$\begin{aligned} RoleFactor(DSi, VDOj, p) = \max(&(IsContainer(DSi, VDOj, p) * ContainerFactor),\\ &(IsSpecialist(DSi, VDOj, p) * SpecialistFactor),\\ &(IsAuthority(DSi, VDOj, p) * AuthorityFactor)) \end{aligned} \tag{3}$$

Although Formula 2 would be more accurate if it uses the cardinality taking into account the query predicate -$card(ext(DSi, VDOj)^p)$-, the exact number of instances that satisfy a predicate is not available in VOs.

Creating joining sets. To predict the set of sources that will not produce empty joins, the VO knowledge base is queried for obtaining the individuals of the composite VO units. This decision was made under the assumption that atomic VO units belonging to the same composite VO unit have more probability of containing the same group of instances. However, the creation of joining sets can use other type of rules to identify when two or more sources may share instances. For example, if two VO units are located in the same region, the probability that their data sources share instances may increase. Similarly, the fact that two VO units participate in the same VO process increases this probability.

Algorithm 1 uses the rule of composite units to create joining sets. The objective is to group together sources whose join will not produce an empty set and that are able to evaluate all the conditions of the query. It first `[1]` obtains the data sources of the composite units and creates one set for each of them. If there are redundant sets, the algorithm removes the set with fewer atomic units. Then, in `[2]`, it validates the completeness of a set. A set is complete if the data sources it contains are able to evaluate all the conditions of the query. In `[3]` it extracts data sources from complete sets and `[4]` removes those incomplete sets whose data sources exist in the complete sets. If it is not the case, the algorithm gets the query conditions that are not already evaluated on each incomplete set `[5]` and completes these sets finding `[6]` the data source with higher role able to evaluate the missing conditions, from the complete sets.

Algorithm 1. Joining Sets Creation

```
Input:  Q, CompositeUnits{ID1(DSi,..,DSj),...,IDm(DSk,...DSm)},
        QRoles(DS1:role,...,DSn:role)
Output: JoiningSets{JS1(DSp,...,DSq),...,JSn(DSt,...,DSv)}
Begin
[1]initalSets{} = CreateInitialSets(CompositeUnits,QRoles)
   incompleteSets{} = {}      completeSets{} = {}
   ForAll (set in initialSets)
[2]   If (isComplete(set,Q)){ add(completSets,set)}
      Else{ add(incompleteSets,set)}
   If (size(incompleteSets) > 0){
[3]  com{} = getDataSources(completeSets)//The data sources of complete sets
     ForAll (set in incompleteSets){
       inc{} = getDataSources(set)//The data sources of one incomplete set
       If ( contains(com, inc))  // com contains inc
[4]      remove(set, incompleteSets)
[5]    Else{ conditions{} = getMissingEvaluation(set, Q)
             ForAll (cond in Conditions){
[6]          dataSource = getHighestRole(com,cond)
             addDataSource(set,dataSource)}
             add(completeSets,set)}
    }}
   Return(completeSets)
End
```

The output of the benefit predictor is the set of joining sets. According to Definition 1 a joining set must contain at least one data source to evaluate each condition of the query predicate. However, if more than one data source may evaluate the same condition, should this condition be evaluated at all data sources or it would be possible to choose only one of them. The optimizer component, described in the next section, helps to make the right decision.

4 Optimizing Source Selection

A joining set may have several sources able to evaluate a query condition with different values of benefit w.r.t. the condition. Querying all of them is neither necessary nor convenient, considering the cost of querying a source. The proposal is to see the problem of deciding which conditions are evaluated by which sources as an assignment problem [20] subject to resource constraints that assigns a condition to a source to maximize the general benefit. We propose a mathematical model that receives the results of the benefit predictor component and the processing capacities of sources, if they are known, and determines the best assignment. Although this model has been created for source selection in VOs, it can be used in any distributed query processing system.

4.1 Source Selection as an Assignment Problem

In the source selection problem, there are a number of conditions to assign to a number of sources. Each assignment brings some benefits, in terms of response quality, and consumes some resources of the sources. The objective is to maximize the benefits using the minimum number of sources while satisfying the resource constraints. From the query point of view, one condition is assigned to one main source as *main asignment*, but this condition could also be evaluated as *secondary assignment* in parallel on other sources which have been selected by other conditions. The reason is that once a source is queried, it is better to evaluate there a maximum possible number of conditions, expecting to reduce the number of sources and the cost of transmitting instances that are not completely evaluated. Indeed, we have to deal with a bi-objective combinatorial optimization problem subject to semi-assignment and resource contraints. In practice, we choose to control the objective of minimizing the number of selected sources by converting it into a constraint. By default, the number of selected sources is limited to the number of conditions.

4.2 Mathematical Model

Given the set of sources $I = \{1, ..., m\}$ and the set of predicate conditions $P = \{1, ..., n\}$ of a query over a VDO, the input data are as follows:
-$Ben_{i,p}$: benefit of assigning condition p to source i, $\forall i \in I$,$\forall p \in P$, as explained in formula 2 in Section 3.3, $Ben_{i,p} = Benefit(DSi, VDOj, p)$ for a given $VDOj$;
-$MaxRes_i$: processing resource capacity of source i, $\forall i \in I$;
-$Res_{i,p}$: processing resources consumed in assigning condition p to source i, $\forall i \in I$,$\forall p \in P$;
-$MaxAssig_i$: maximal number of condition assignments for source i, $\forall i \in I$.

The decision variables are:
-$x_{i,p}$ are 0-1 variables that determine whether source i has (=1) or not (=0) been selected as a main source to evaluate the condition p, $\forall i \in I$,$\forall p \in P$.
-y_i are 0-1 variables that turn to 1 when the source i is selected, $\forall i \in I$.
-$assig_{i,p}$ are 0-1 variables that determine whether a condition p is assigned to source i (=1) or not (=0), $\forall i \in I$. These variables represent the final assignment of conditions to sources. The $x_{i,p}$ variables indicate the main assignments while the $assig_{i,p}$ variables indicate all the main and secondary assignments.

The mathematical program can be formulated as follows:

$$max \sum_{p=1}^{n} \sum_{i=1}^{m} Ben_{i,p} * (x_{i,p} + assig_{i,p}), \qquad (4)$$

subject to:

$$\sum_{i=1}^{m} x_{i,p} = 1, \forall p \in P\,; \quad (5) \qquad \sum_{i=1}^{m} y_i \leq k\,; \quad (6) \qquad \sum_{p=1}^{n} x_{i,p} \geq y_i, \forall i \in I\,; \quad (7)$$

$$\sum_{p=1}^{n} Res_{i,p} * assig_{i,p} \leq MaxRes_i, \forall i \in I; \quad (8) \qquad \sum_{p=1}^{n} assig_{i,p} \leq MaxAssig_i, \forall i \in I\,; \quad (9)$$

$$x_{i,p} \leq assig_{i,p}, \forall i \in I, \forall p \in P; \quad (10) \qquad assig_{i,p} \leq y_i, \forall i \in I, \forall p \in P. \quad (11)$$

Constraint (5) ensures that any condition p is assigned to one main source. Constraint (6) limits the total amount of queried sources to k, we take k=n by default to start the optimization process. This constraint is somehow redundant with constraint (7) which prevents to select a source if no condition is assigned to it, but in practice one could reduce k in constraint (6) to control the minimization of the number of selected sources. Constraints (8) and (9) ensure that all the main and secondary assignments of conditions do not exceed neither the processing resource capacities nor the maximum number of possible assignments per source. Finally, coupling constraints (10) and (11) indicate respectively that the main assignments should be in all main and secondary assignments as well, and that a source i is selected when at least one condition p is assigned to it.

The resolution of this model provides the selected sources in the variables y_i and all the main and secondary condition assignments in the variables $assig_{i,p}$. The joining of the results provided by each source are the instances required by the user. If the number of instances obtained are not enough to satisfy user requirements, the query planner will use the model to generate a new assignment with the remaining sources (i.e. those that were not in y_i).

5 Implementation and Validation

The objective is to measure the recall and precision provided by OptiSource. Section 5.1 presents the evaluation using different data contexts and levels of knowledge about sources. Section 5.2 compares OptiSource to related works.

5.1 OptiSource Experiments

Our prototype of OptiSource is written in Java. The knowledge base was implemented in OWL [18]. The optimization model was written in GNU MathProg modeling language and is processed by the GNU Linear Programming Kit (GLPK) [21]. We also validated our optimization model performance in CPLEX 10.2. Queries of VDOs are accepted in SPARQL [22].

We define the knowledge base of a VO in the health sector. Metadata are generated at different levels of the knowledge base using a developed generator of metadata. This generator describes the intentional knowledge of sources, the known roles of each source and possible replication of instances.

Experiment scenarios vary in five dimensions: the number of sources, the level of knowledge of sources, the number of composite VO units, the relationships between sources and the query. The experiments are focused on measuring three metrics: Extensional Precision, Extensional Recall, and Extensional Fall-Out. To obtain these metrics, we measure the following variables:

S: set of sources available on the VO	***R:*** set of relevant sources for Q
-*R:* set of not relevant sources for Q	***A:*** set of sources selected by OptiSource.
Extensional Precision: $\|A \cap R\|/A$	**Extensional Recall**: $\|A \cap R\|/R$
Extensional Fall-Out: $\|A \cap -R\|/-R$	

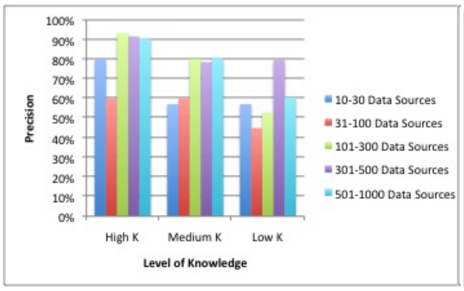

Fig. 4. Precision evaluation changing the data context and the level of knowledge

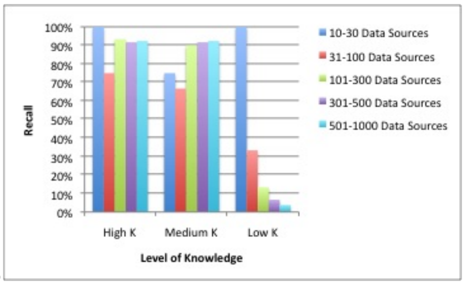

Fig. 5. Recall evaluation changing the data context and the level of knowledge

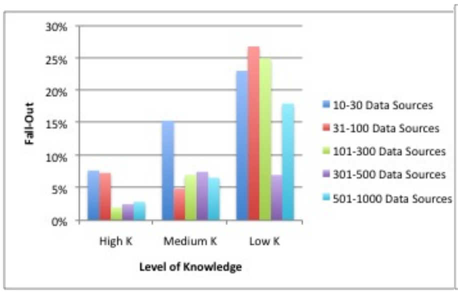

Fig. 6. Fall-Out evaluation changing the data context and the level of knowledge

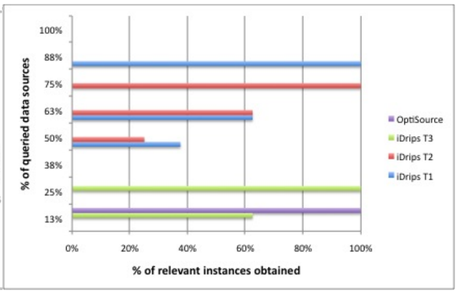

Fig. 7. OptiSource vs. iDrips

Experiment results. Figures 4, 5, 6 present respectively the precision, recall and fall-out, varying the level of knowledge (K) and the number of sources. Low level means the knowledge base only contains the intentional knowledge of sources (e.g. can evaluate the diagnosis). The medium level means it also knows a group of roles of sources, but they are related to higher classes of the knowledge base. For instance, sources have related roles to *cancer* and queries ask for *kidney cancer*. High level indicates that roles are related to more specific classes (e.g. roles related to kidney cancer).

The experiments show that even if a high level of knowledge is desired, OptiSource precision and recall have a good behavior with a medium level. This is probably due to the fact that a medium level is enough to discard a large group of sources and to direct the queries to sources with higher probabilities of having matching instances. In contrast, the fall-out metric improves (i.e. is lower) when the level of knowledge increases. However, as it can be noticed in Figure 6 even with a lower level of knowledge the fall-out can be low due to the existence of sources that have a considerably higher benefit in comparison with the majority of sources, as is the case in the group of 301 and 500 sources. Tests also show that for contexts with fewer sources, the improvement in precision according to the level of knowledge is not significant. This validates our assumption that in small data contexts great efforts in query planning for reducing queried sources are not necessary. On the other hand, the improvement observed when there is a large number of sources led us to conclude that OptiSource is especially useful under these circumstances. Finally, the experiments show that in the worst case OptiSource selects the same set of sources that traditional strategies of query rewriting select because it only considers the intentional views.

5.2 Comparison with Related Works

Although OptiSource can not be directly compared with any of the proposals in the literature because they worked under different assumptions, we evaluated the implications of using three available strategies (QPIAD [15] and Navigational Paths [13], iDrips [8]) in VOs contexts.

The strategy of rewriting of QPIAD [15] allows the query processor to return certain and "possible" certain answers from autonomous databases. The first type are answers that match all the query predicate; the second type are the answers that probably match the query, but have one condition that can not be evaluated because the required value is *null*. QPIAD generates and uses detailed statistics on source's quality to reformulate a query according to the characteristics of each source that is able to evaluate the predicate. Even though the principle of QPIAD will be very attractive for use in VOs whose sources have quality problems, the strategy could not scale up when the number of sources is large. To prove this we will consider a VO with 20 sources, which are able to evaluate a query Q in terms of the schema. We supposed that in average each source provides 500 certain answers. For each source 50 query reformulations are required to obtain the possible answers. This means that 1020 queries are executed including the original Q to obtain certain answers from each source.

Another related strategy is the proposal to improve the selection of navigational paths between biological sources [13]. It assumes the availability of a graph describing the objects contained in the sources and their relationships. The objective is to find the path with best benefit and lower cost given an initial and a final object class. Applying this proposal to VOs we found that it is possible to organize sources in a graph. Each node could contain sources with equivalent intentional fragments of VDO, and the relationships between sources that contain complementary fragments could be the links between nodes. Using this graph it will be possible to select the most "profitable" sources for a query; however, it is not clear how the benefit of each path is obtained and how the overlapping between each path can be calculated. This lack could be combined with our strategy using the roles of sources to compute the potential benefit of a path. Thus, the prediction model of OptiSource can be used to compute the ratios between paths used in this proposal.

Finally, we compared OptiSource with the iDrips algorithm. iDrips works under the assumption of source similarity that allows the query processor to retrieve the most relevant plans efficiently. It proceeds in two phases. The first phase groups similar sources and creates abstract plans that are able to solve all the predicates. Each abstract plan represents a set of concrete plans with an utility. This utility is an interval that contains the utility of the concrete plans of the set. The second phase obtains query plans in decreasing order of utility selecting only the dominant plan in each iteration. A dominant abstract plan has at least one concrete plan whose utility is not less than the utility of all concrete plans in other abstract plans.

The comparison with iDrips is focused on measuring the recall in terms of relevant instances obtained vs. the number of sources queried. We did not measure precision because iDrips and OptiSource always return correct instances. In order to make comparable the measurements of recall in OptiSource and iDrips, we evaluate in each iteration of iDrips the number of relevant instances until it finds the complete set of relevant instances. We also measure the number of queried sources in each iteration. In the case of OptiSource, we measure the relevant instances obtained after the execution of subqueries of the joining sets.

We used the plan coverage as the utility measure for iDrips. Given a query $Q\{p(p_1, ..., p_m)\}$ with m conditions, and a plan $P\{DS_1, ..., DS_n\}$ where DS_i are the sources that will be queried on the plan, the formulas used to compute plan coverage are the following:

$$Cov(P) = min(ext(DSi, VDOj)) \; where \; DS_i \in P, \tag{12}$$

$$Cov(P) = min(overlap(DS_i, DS_k)) \; where \; DS_i \; andDS_k \in P, \tag{13}$$

$$Cov(P) = \frac{ext(DSi, VDOj)^p - (ext(DSk, VDOj)^p ... ext(DSl, VDOj)^p)}{ext(U, VDOj)^p} \tag{14}$$

where $DS_i, DS_k \; and \; DS_l \in P \; and \; DSk...DSl$ have been previously queried.

The first group of experiments are run in a context where sources are extensionally and intentionally overlapped, and a few number of sources contain a large number of relevant instances; even though instances are distributed largely along the VO sources. In other words, there are sources specialized in one or more of the conditions. Consequently, querying them is enough to obtain almost all the relevant instances. Using the aforementioned utility measures for iDrips. Figure 7 illustrates the differences on recall between iDrips and OptiSource. The y axis shows the % of queried sources from the total number of relevant data sources. On the other hand, the x axis illustrates the % of instances obtained.

The results demonstrate the effectiveness of OptiSource in this type of contexts where it only queried 13% of relevant sources to obtain 100% of relevant instances. On the contrary, iDrips required to query 25% of relevant sources to obtain the 100% of relevant instances in the best case (using (14)). Figure 7 shows that the number of sources queried by OptiSource is considerable lower than those queried using (12) and (13) of the plan coverage. Although this could be inferred considering the difference on the level of knowledge, it allows us to conclude that solutions for ranking queries cannot be applied to VOs without using an utility function able to represent source relationships and source probability of having query relevant instances. Nonetheless, even with strong utility functions, the absence of replication knowledge prevents discarding sources that do not contribute with new instances. This lack affects iDrips when we use (14).

6 Conclusions and Future Work

Source selection in large scale VO data contexts is a complex decision problem. It requires finding the minimal sets of sources for each condition and performing a final intersection operation to avoid empty joins. Current strategies of query planning are not focused on producing query plans under these circumstances, generating redundant and unnecessary plans when they are applied to VOs. Our proposal to solve this problem is a decision system called OptiSource. It uses extensional characteristics of sources to predict the benefit of using them during query evaluation. OptiSource produces joining sets using a combinatorial optimization model that establishes which sources will evaluate each condition of the query. Although OptiSource was created for VOs, it can be used in any distributed query processing system during the planning phase. A prototype of OptiSource has been developed to evaluate its precision. Experiments have demonstrated that OptiSource is effective to reduce the contact of unnecessary sources when data contexts involve extensional and intentional overlapping. They have also showed that OptiSource is an evolving system whose precision grows according to the degree of knowledge of sources, but does not require a high level of knowledge to have good levels of precision and recall.

Future work will be focused on performance evaluation of the approach. We are also interested on applying data mining techniques on logs of query executions to identify the role of sources. Additionally, we are going to improve the prediction of the benefit using live statistics of sources that have a higher role.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. 15, 200–222 (2001)
2. NEESGrid: Nees consortium (2008), `http://neesgrid.ncsa.uiuc.edu/`
3. BIRN: Bioinformatics research network (2008), `http://www.loni.ucla.edu/birn/`
4. Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: VLDB 1996, Bombay, India, pp. 251–262 (1996)
5. Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J.D., Vassalos, V., Widom, J.: The tsimmis approach to mediation: Data models and languages. Journal of Intelligent Information Systems 8, 117–132 (1997)
6. Tomasic, A., Raschid, L., Valduriez, P.: Scaling access to heterogeneous data sources with DISCO. Knowledge and Data Engineering 10, 808–823 (1998)
7. Pottinger, R., Halevy, A.Y.: Minicon: A scalable algorithm for answering queries using views. VLDB J. 10, 182–198 (2001)
8. Doan, A., Halevy, A.Y.: Efficiently ordering query plans for data integration. In: ICDE '02, Washington, DC, USA, p. 393. IEEE Computer Society, Los Alamitos (2002)

9. Huebsch, R., Hellerstein, J.M., Lanham, N., Loo, B.T., Shenker, S., Stoica, I.: Querying the internet with pier. In: VLDB 2003, pp. 321–332 (2003)
10. Tatarinov, I., Ives, Z., Madhavan, J., Halevy, A., Suciu, D., Dalvi, N., Dong, X.L., Kadiyska, Y., Miklau, G., Mork, P.: The piazza peer data management project. SIGMOD Rec. 32, 47–52 (2003)
11. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., Risch, T.: Edutella: a p2p networking infrastructure based on rdf. In: WWW '02, pp. 604–615. ACM, New York (2002)
12. Adjiman, P., Goasdoué, F., Rousset, M.C.: Somerdfs in the semantic web. J. Data Semantics 8, 158–181 (2007)
13. Bleiholder, J., Khuller, S., Naumann, F., Raschid, L., Wu, Y.: Query planning in the presence of overlapping sources. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 811–828. Springer, Heidelberg (2006)
14. Venugopal, S., Buyya, R., Ramamohanarao, K.: A taxonomy of data grids for distributed data sharing, management, and processing. ACM Comput. Surv. 38, 3 (2006)
15. Wolf, G., Khatri, H., Chokshi, B., Fan, J., Chen, Y., Kambhampati, S.: Query processing over incomplete autonomous databases. In: VLDB, pp. 651–662 (2007)
16. Naumann, F., Freytag, J.C., Leser, U.: Completeness of integrated information sources. Inf. Syst. 29, 583–615 (2004)
17. Quiané-Ruiz, J.A., Lamarre, P., Valduriez, P.: Sqlb: A query allocation framework for autonomous consumers and providers. In: VLDB, pp. 974–985 (2007)
18. Horrocks, I.: Owl: A description logic based ontology language. In: CP, pp. 5–8 (2005)
19. Pomares, A., Roncancio, C., Abasolo, J., del Pilar Villamil, M.: Knowledge based query processing. In: Filipe, J., Cordeiro, J. (eds.) ICEIS. LNBIP, vol. 24, pp. 208–219. Springer, Heidelberg (2009)
20. Hillier, F.S., Lieberman, G.J.: Introduction to Operations Research, 8th edn. McGraw-Hill, New York (2005)
21. Makhorin, A.: Gnu project, gnu linear programming kit (2009), `http://www.gnu.org/software/glpk/`
22. Eric Prud, A.S.: Sparql query language for rdf (2007), `http://www.w3.org/tr/rdf-sparql-query/`