Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 23 (2007) 485–496
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

From bioinformatic web portals to semantically integrated Data Grid networks☆

Adriana Budura*, Philippe Cudré-Mauroux, Karl Aberer

School Of Computer and Communication Sciences, EPFL – Switzerland

Received 31 January 2006; accepted 26 March 2006
Available online 4 May 2006

Abstract

We propose a semi-automated method for redeploying bioinformatic databases indexed in a Web portal as a decentralized, semantically integrated and service-oriented Data Grid. We generate peer-to-peer schema mappings leveraging on cross-referenced instances and instance-based schema matching algorithms. Analyzing real-world data extracted from an existing portal, we show how a rather trivial combination of lexicographical measures with set distance measures yields surprisingly good results in practice. Finally, we propose data models for redeploying all instances, schemas and schema mappings in the Data Grid, relying on standard Semantic Web technologies.

Keywords: Data sharing; Data mapping; Distributed databases; Semantics; Biology

1. Introduction

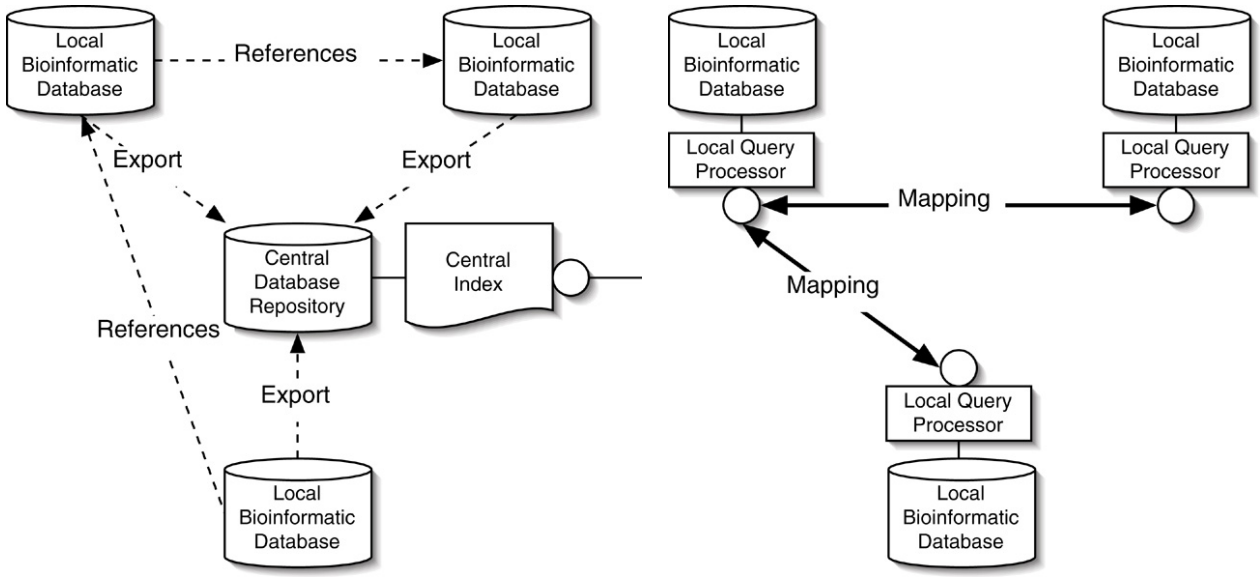
In the past, biologists used to collect and analyze data in isolation, creating proprietary schemas to annotate and store their information in various ways. Today, with the development of the Internet, complex experiments can be conducted by aggregating and interrelating data from multiple sources; this obviously requires the various data sources to be integrated beforehand, to allow the reformulation of a query posed against a local database to be propagated to related databases which store data using different schemas. Standard data integration techniques such as LAV (local-as-view) or GAV (global-as-view) have traditionally been used to achieve this goal. These techniques rely on a global schema to enable transparent access over heterogeneous databases and allow deterministic reformulations of queries using schema mappings (i.e., *views*). These centralized approaches, however, require the definition of an integrated (so-called *federated*) schema supposedly subsuming all local schemas. This requirement is particularly stringent in highly heterogeneous and decentralized environments such as Data Grid networks. In Data Grid settings, autonomous and heterogeneous data sources coexist with only loose coordination as resources comprising the grid typically come from different administrative domains. Such settings make it impractical to keep a global schema up to date, or even to define a schema general enough to encompass all information sources' needs.

Recently, new decentralized data integration techniques not requiring any central schema have been proposed. *Peer Data Management Systems* [1] (see also below Section 1.1) take advantage of local schema mappings between pairs of databases to propagate queries posed against a local schema to the rest of the network in an iterative and cooperative manner. No global semantic coordination is needed as peers (e.g., data sources) only need to define local mappings to a small set of related databases in order to become part of the global network of databases. Once a query is posed locally against a given schema, it can then be propagated and reformulated iteratively though the peer-to-peer mappings in order to be processed by all (or a specific portion) of the peers in the network. The local mappings needed to implement this approach are inferred either in a manual or a semi-automatic manner through the process of *schema matching*, by relating attributes from one database schema to semantically similar attributes from another schema.

☆ The work presented in this paper was carried out in the framework of the EPFL Center for Global Computing and was supported by the Swiss National Funding Agency OFES as part of the European project KnowlegeWeb No 507482.

* Corresponding address: EPFL-I&C, CH-1015 Lausanne, Switzerland. Tel.: +41 21 693 5260; fax: +41 21 693 8115.
E-mail address: adriana.budura@epfl.ch (A. Budura).

0167-739X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2006.03.002



(a) Centralized indexation.

(b) Service-oriented integration.

Fig. 1. Two models of integration: the *centralized indexing* providing keyword-search functionalities through a central index and the *service-oriented integration* supporting decentralized query reformulations through peer-to-peer schema mappings.

In this paper, we show how to automate the transition from a simple collection of centrally indexed but interrelated bioinformatic databases, as often found in bioinformatic Web portals, to a bioinformatic Data Grid supporting query reformulation mechanisms over distributed information sources. We take advantage of foreign key relationships defined between pairs of related instances from two different databases to bootstrap an automatic instance-based schema mapping process. The resulting mappings are in turn used by the various participants in the Data Grid to enable global reformulation of queries posed against a specific schema. We provide an extensive performance evaluation of our method on a set of real-world bioinformatic schemas and a concrete architecture based on Semantic Web standards for the redeployment of the databases in a Data Grid environment.

The rest of this paper is structured as follows: we start below with a general introduction to data integration in Web portals and bioinformatic grids. We then give a short overview of related works before giving a few details on the Sequence Retrieval System (SRS), whose databases will be used throughout the rest of the paper. Sections 4 and 5 respectively provide details on our instance-based schema matching algorithm and on its performance when applied to the SRS schemas. We describe the Data Grid system resulting from the creation of schema mappings and the export of SRS data and schema using Semantic Web technologies in Section 6 before concluding in Section 7.

1.1. Data integration in bioinformatic grids

Today, data integration in bioinformatics often revolves around *centralized indexing* as typically implemented by Web portals (see Fig. 1(a)). In such settings, all related databases are first exported to a central server and indexed. The central server can then provide a simple keyword-lookup service (e.g., "Find all entries containing the word *Theileriidae*") to the end-users. Complex, structured queries are not directly supported as these systems are mostly based on flat files and central indexes. Instead, complex queries have to be handled manually by the users, who need expert knowledge on all the databases they want to query and who have to navigate iteratively between different databases by mouse-clicking (this search process is usually referred to as "query by navigation" [15]) to resolve their query. The Sequence Retrieval System described below is an example of a system built on centralized indexing.

While providing fast and robust mechanisms for searching collections of heterogeneous data, we argue that such integration techniques are inadequate in Grid environments for obvious reasons. Grid computing typically involves sharing heterogeneous resources from different physical locations and belonging to different administrative domains. Nodes in a Grid should act autonomously and provide higher-level services through standard interfaces. Unfortunately, central indexing applied to a Grid setting implies the loss of autonomy for all nodes maintaining a local database, as changes performed locally are not reflected in the central repository until the local database is re-imported and re-indexed by the central server. Moreover, central indexing only provides keyword-search functionalities through centralized, custom interfaces where distributed services and standard interfaces should be supported in a Grid. Centralized indexing does not support transparent access over several databases; on the contrary, it requires in-depth knowledge of all indexed data as one has to navigate manually through the databases in order to retrieve relevant information.

In the rest of this paper, we propose a semi-automated method to migrate centrally indexed Web portals to Data Grid environments supporting service-oriented, decentralized integration. Fig. 1(b)) depicts a service-oriented integrated system. Note that in this setting, the central repository has disappeared. Individual nodes do not rely on a central

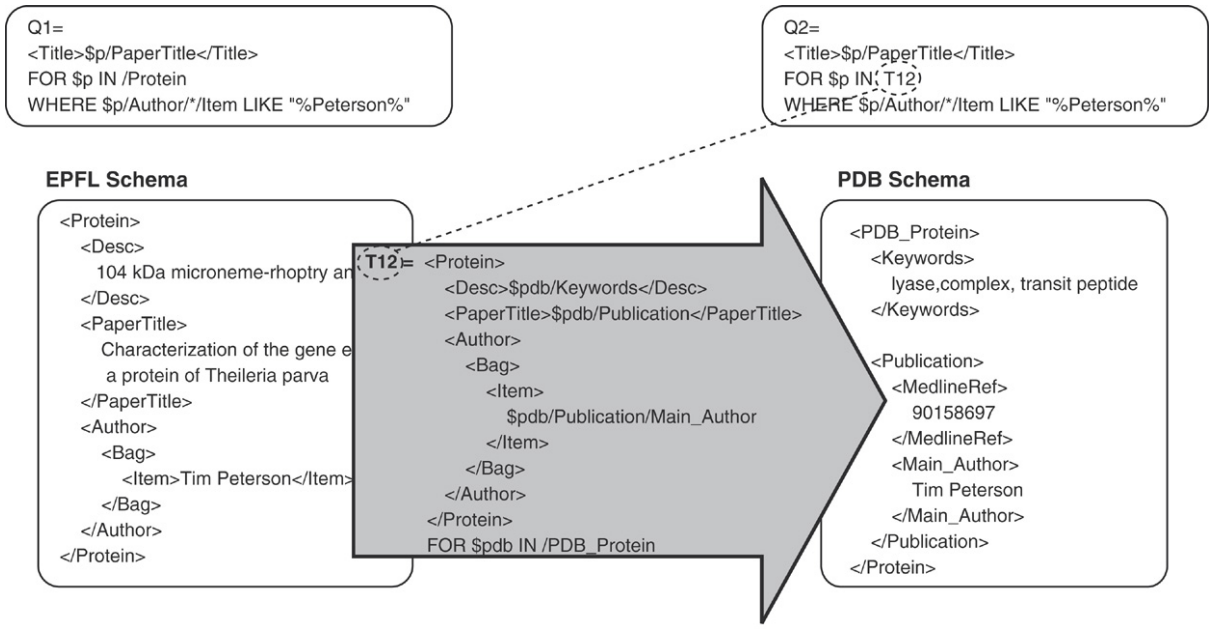


Fig. 2. An example of peer-to-peer mappings used to propagate a structured query from one database to the other.

index anymore, but connect to each other through peer-to-peer mappings. Each node supports higher-level data services through local query processors. Any node can thus pose complex, structured queries locally and propagate them to the rest of the network by reformulating the queries thanks to the peer-to-peer schema mappings. Most of the time, query reformulation in this context is based on query rewriting using views: mappings are expressed as queries relating sets of attributes from one database to sets of attributes from another database. Fig. 2 gives a simple example of query reformulation in a semi-structured (i.e., XML and XQuery) context. The query propagation can stop after a couple of steps, or can go on until the query reaches all possible nodes in the network (see [2] for a discussion on decentralized query routing strategies).

We see this new kind of integration techniques as particularly suited to Data Grid environments as data sources keep total control over their data and can provide higher-level services thanks to (semi) structured data processing support. Furthermore, these techniques are scalable (new data sources simply have to connect locally to a couple of existing databases to be part of the network) and robust (contrary to the centralized indexing, there is no single point of failure), which are two highly desirable properties for dealing with large and heterogeneous data in distributed Grids.

2. Related work

Automatic invocation or composition of services have received a lot of attention in Grid environments. There is, however, little focus on data integration. In [11], Frishman et al. motivate the need for standard languages and integrated database information systems for molecular biology in the Web era. The SIMDAT Pharma Grid [18] is an industry-oriented Grid environment aiming at assisting biologists in the discovery, selection or composition of services. SIMDAT provides a semantic broker whose role is to reduce the user's interaction with Web Services but the broker is focused on service discovery and composition and not on data integration. MyGrid [20] is an e-Science Grid project aiming at helping biologists and bioinformaticians to perform workflow-based experiments. Workflows in MyGrid comprise interconnected sets of input, output and processors, which can either be local routines or distant Web Services. MyGrid also supports the use of autonomous agents for personalizing workflow plans or negotiating on behalf of the end-users [15]. MyGrid supports distributed semi-structured data repositories but does not directly address the issue of transparent access over the repositories. Bio-GRID [17], part of the EUROGRID project, is an access portal for biomolecular modeling resources. Bio-GRID provides the users with standard interfaces to various databases but does not focus on data integration over these sources. More recently, Milena and Bartosz [14] proposed a model for integrating Grid middle-ware with stateful access of resources using Web Services and publish/subscribe notifications.

Other approaches, such as the one proposed by Arshad et al. [8] rely on central data warehousing techniques to provide homogeneous access to life science data stored in distributed, heterogeneous databases. Views on the warehouse data are then typically defined and materialized as a set of databases, which are made locally available to different applications. GeneGrid [12] supports seamless integration of heterogeneous applications and datasets that span over multiple administrative domains and locations across the globe. Data integration is based on a centralized data manager responsible for the integration and access of a number of disparate and heterogeneous biological datasets. Mougin et al. [16] proposed an automatic method for generating schemas of biomedical sources. Their integration approach, however, is centralized using a mediator and a centralized ontology. Closer to our concerns, Comito et al. [6] provide a framework for

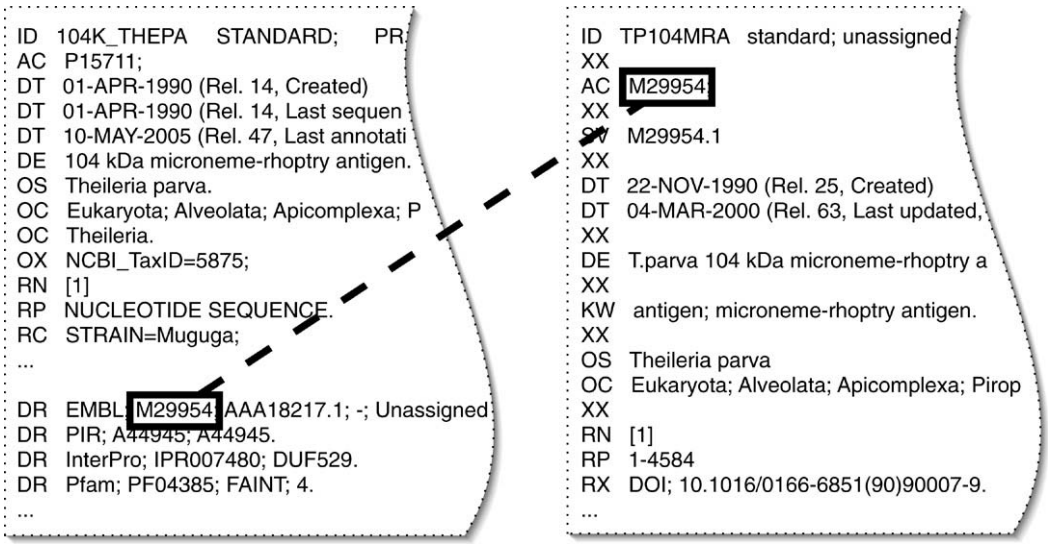


Fig. 3. A cross-reference link relating two instances with a foreign-key relationship in SRS; the left-hand side entry is taken from the SwissProt database while the right-hand side is from EMBL.

integrating heterogeneous data sources distributed over a Grid, centered around an XML-based query reformulation algorithm. OGSA-DAI [9] provides distributed query processing and data federation in Grid environments. OGSA-DQP [3] is a distributed query processor for OGSA-compliant Grids. It supports the evaluation of queries that combine data from multiple Grid services. Compared to the systems presented above, our approach deals with the problem of database heterogeneity from a totally decentralized point of view, as it does not rely on any predefined, global semantic model. Moreover, we allow the automated transition from a system based on central indexing to a completely decentralized system, by making use of schema matching algorithms based on functional dependencies between attributes. Semantic interoperability emerges gradually as peers use local semantic agreements to reformulate and propagate queries through the network.

3. The Sequence Retrieval System

We give below a succinct overview of the Sequence Retrieval System (SRS) [13], which will be used subsequently to test our semi-automated integration techniques. SRS is a commercial indexing and retrieval system designed for bioinformatic libraries such as the *EMBL* nucleotide sequence databank, the *SwissProt* protein sequence databank or the *Prosite* library of protein subsequence consensus patterns. It started as a data management system initially developed at the European Molecular Biology Laboratory in Heidelberg. As such, it allows the querying of one or several databases simultaneously, regardless of their format. SRS repositories typically contain indices for one hundred or more databases, whose data is saved mainly as flat files.

Administrators wishing to register new databases with an SRS repository first have to define *structure files* which detail on a syntactic level the schema according to which data has been organized in the flat files. Once their schemas have been defined, administrators can export schema instances (i.e., flat text files) whose data will be centrally parsed, indexed and processed thanks to the corresponding schema definitions. Entries in bioinformatic databases often contain explicit or implicit references to each other; for example, information about elements related to a nucleic acid segment can be available in a protein databank. Taking advantage of this fact, SRS lets administrators manually define relationships between their database schema and similar schemas. In SRS, these relationships are represented as links (i.e., foreign key relationships) relating instances from two different schemas (see Fig. 3).

SRS allows users to issue queries over the centralized index using a custom query language. Thanks to the structure of links between the databases, users can pose queries over related databases by explicitly referencing foreign key relationships. However, the system does not permit automatic reformulation of queries over related databases as a standard federated database system would: to solve complex queries, users have to manually specify intricate queries spanning over multiple databases, or have to navigate by mouse-clicking through series of databases using a Web interface. Thus, users need in-depth knowledge of all databases present in the central registry to fully exploit the system. Our goal in the rest of this paper will be to take advantage of the links defined in SRS between some of the databases to automatically create pairwise schema mappings. Once created, these mappings will be used to foster the transition of the whole system towards a Data Grid environment providing transparent access over distributed and autonomous databases.

4. Instance-based schema matching

We describe below the heuristics used to generate schema mappings from the cross-reference links of the databases using instance-based schema matching algorithms. The general goal of a schema matching operation is to find semantic correspondences between elements of two different schemas [7]. In our case, elements are represented by attributes (such as *Protein Description* or *Sequence*) defined in the

schema of a biologic database. The result of a schema matching operation is a mapping consisting of a set of mapping elements, each of which indicates that certain elements of a given schema are semantically related to certain elements of another schema (see also [19] for a survey on schema matching techniques).

Our approach aims at implementing an instance-level schema matcher using the databases registered with SRS to infer mappings between their schemas. Our matcher fundamentally differs from previous matchers in the sense that it is based on cross-reference links defined between the databases. Links built in the SRS system indicate which instances of two databanks reference each other. Taking this into account, we came to the idea of analyzing the content of the two linked instances (i.e., the two schema instances) in order to infer additional mappings for the other attributes contained in those instances. By further generalizing this result, we can infer mappings between the schemas of the two databanks, in a bottom-up manner, by aggregating the local results obtained for the pairs of cross-referenced instances. Our fundamental assumption is that the cross-reference links, which have been extracted from the structure files in SRS, relate semantically similar instances; comparing the content of these instances at the data level, we can then infer attribute correspondences at the schema level.

Schema definitions are typically quite limited in SRS: attributes are often written as two letter acronyms without any additional information on their semantics, their relationships to other attributes or on the constraints they might be subject to. Thus, using other matching methods based for instance on constraint analysis or taxonomical techniques would prove to be difficult. Therefore, we decided to implement a pure instance-based matcher, which analyzes the content of related instances without using any other auxiliary information to generate the mappings. We show in Section 5 that a rather trivial combination of lexicographical measures with set distance measures yields surprisingly good results in our case.

4.1. Finding correspondences at the data layer

Our schema matching algorithm starts with a content-based analysis of a pair of instances I_1 and I_2, each belonging to a different database but related through a cross-reference link in SRS. In order to detect such pairs of instances, we take advantage of a particular operator in SRS which allows us to query for pairs or related instances given two databases. Each instance is constituted of series of *attribute–value* pairs A_i–V_i (e.g., *RP–NUCLEOTIDE SEQUENCE*). Let us call $A_0^1, \ldots, A_m^1$ the set of attributes appearing in the first instance I_1 and $A_0^2, \ldots, A_n^2$ the set of attributes appearing in the second instance I_2. All attributes appearing in the instances are defined in their corresponding schema. The values corresponding to those attributes will be referred to as $V_0^1, \ldots, V_m^1$ and $V_0^2, \ldots, V_n^2$ respectively.

Two attributes A_0^1 and A_0^2 are already related through a cross-reference (foreign key relationship) in the system. In this first step, our goal is to discover other pairs of semantically related attributes A_i^1 and A_j^2 from the two selected instances, by analyzing the degree of similarity between their values V_i^1 and V_j^2. The instance-level similarity metric we use is terminological and as such, based on string similarity. The complete algorithm which computes the similarity between two attribute values V^1 and V^2 is reproduced below (Algorithm 1).

Algorithm 1 Attribute value similarity metric

tokenize attribute values: $V^1 = \{v_1^1, v_2^1, \ldots, v_{n1}^1\}$ and $V^2 = \{v_1^2, v_2^2, \ldots, v_{n2}^2\}$;
for each pair of tokens (v_x^1, v_y^2) **do**
  compute the distance $\gamma(v_x^1, v_y^2) = \frac{\delta(v_x^1, v_y^2)}{length(v_x^1)+length(v_y^2)}$;
  // δ = *Levenshtein distance;*
end for
select n pairs (v_x^1, v_y^2) with the lowest distance values (top-n matches) ;
// n = *min(n1, n2);*
return the overall similarity value for the corresponding pair (A^1, A^2) of attributes as:
$\Delta(A^1, A^2) = 1 - \frac{\sum_1^n(\gamma(v_x^1, v_y^2))}{n}$

We start by tokenizing the attribute values ($V^1 = \{v_1^1, v_2^1, \ldots, v_{n1}^1\}$ and $V^2 = \{v_1^2, v_2^2, \ldots, v_{n2}^2\}$) using white spaces and special characters, such as semi-colons or dashes, as separators. We then compute distances ($\gamma(v_x^1, v_y^2)$) between pairs of tokens belonging to the two attribute values by making use of the well-known Levenshtein distance (Edit distance). Generally speaking, the Levenshtein distance between two objects is the minimal cost of operations to apply to one of the objects in order to obtain the other. For strings, it is the minimum number of insertions, deletions, and substitutions of characters required to transform one string into the other. The greater the Levenshtein distance, the greater the dissimilarity between the strings. The result, $\delta(v_x^1, v_y^2)$, is normalized by the sum of the lengths of the two tokens. After computing the similarity between all pairs of tokens, we aggregate those values in order to asses the overall similarity between the two attribute values (i.e. the two sets of tokens). In data analysis, the so-called *linkage aggregation* [19] methods allow us to assess the distance between two similar sets. In our case, we chose the *average linkage* to measure the mean Levenshtein distance between the tokens: $\sum_1^n(\gamma(v_x^1, v_y^2))/n$.

When developing our metric, we also took into consideration the recurring case when tokens of one attribute exactly represent a subset of the tokens of the other attribute. As a concrete example, let us consider two cross-referenced instances belonging to SwissProt and EMBL respectively and the values contained in their *DE (Description)* attributes:

```
SwissProt: DE   104 kDa microneme-rhoptry antigen.
EMBL:      DE   T.parva 104 kDa microneme-rhoptry
                antigen gene, complete cds.
```

After having analyzed several such occurrences in SRS, we decided to maximize the similarity value between the two attributes in this case as follows: our metric considers only the best n matching tokens, where n is the cardinality (the number of tokens) of the smallest set ($n = \min(n1, n2)$). In other words, for all the tokens in the smallest set, we identify the

best matches (lowest distance values) in the second set. We then quantify the set distance as the sum of those n distance values, normalized by the cardinality of the smallest set. As a result, the distance successfully measures the similarity between the two sets, by selecting the best matching candidates (tokens) out of both sets. The normalization serves two purposes: first, it allows us to return a value in the interval [0, 1] and second, it lowers the computed distance value (the sum of the best n distance values) proportionally to the number of tokens in the smaller set. In other words if, for example, the smallest set contains a single token, then the distance returned will be exactly the distance computed for the best match of that token to any other one from the second set. In this way, we keep relatively low distances for attribute values with many tokens, thus encouraging *imperfect* matches. In the example presented above, the distance between the *DE* attributes will be equal to zero, since all the tokens from the SwissProt attribute are contained in the EMBL attribute.

4.2. Aggregating attribute similarities at the schema layer

In the preceding section, we described the algorithm that computes instance level similarity values for pairs of attributes belonging to cross-referenced instances of two different databases. In order to generate high-quality schema mappings, we need now to properly aggregate those locally computed similarity values. We do it as follows: after having selected two databases linked with at least one foreign key relationship in SRS, we query for P pairs of linked instances. We compute attribute similarities at the instance level, $\Delta_p(A_i^1, A_j^2)$ (see above Algorithm 1), but discard low-quality mappings (typically, those whose similarity values are below a threshold η) to filter out accidental matchings. An aggregated attribute similarity value is computed by averaging the results returned from different pairs of instances. This average is only retained for mappings which occur frequently enough at the instance level, for instance for pairs of attributes with similarity values above η discovered in at least κ of the considered cases. The exact algorithm is reproduced (Algorithm 2).

Algorithm 2 Schema matching algorithm

```
for each pair of linked databases do
  query for P pairs of linked instances;
  for each pair p of linked instances do
    compute similarity between each pair of attributes
    Δ_p(A_i^1, A_j^2);
    return similarity if and only if Δ_p(A_i^1, A_j^2) ≥ η;
    //η = threshold on the similarity value
  end for
  for all pairs attributes (A_i^1, A_j^2) where at least κ similarity
  values were returned do
    compute aggregated attribute similarity values as:
    Δ(A_i^1, A_j^2) = (Σ_p Δ_p(A_i^1, A_j^2)) p^{-1};
    //κ = threshold on the number of similarity values
    returned
  end for
end for
```

5. Schema matching results

5.1. Reference alignments and metrics

In order to evaluate our approach, we decided to run our schema matching algorithm on several databases registered with SRS whose instances cross-reference each other. When matching the database schemas, we disregarded the protein sequence fields, which contain data of a very peculiar nature and for which specific tools already exist (e.g., BLAST [4], FASTA [10]).

To provide a basis for evaluating the quality of our results, the matching task had to be handled manually first. As this process is rather subjective, we invited an external domain expert and asked her to provide us with reference alignments between different schemas. We concentrated on three well-known databases: SwissProt, PDB and EMBL. In this process, all possible attribute mappings were given a grade of 1, 2 or 3. A grade of 1 indicates an incorrect mapping, in other words a mapping between semantically different attributes. A value of 2 indicates attributes whose contents are semantically related, while a grade of 3 designates semantically identical attributes. As a concrete example, a mapping between SwissProt's *RT* (Reference Title) and PDB's *JRNL* would be graded as 3, since both attributes contain titles of papers related to the instance. A mapping graded as 2 could relate SwissProt's *DE* attribute, which holds a keyword description of the instance, to EMBL's *RT* attribute, which represents the title of a paper constituted of a similar series of keywords and *de facto* representing another description for the instance. As such, grade 2 represents imperfect semantic matches (the two concepts are different, one being a description and the other one a title), which however return pertinent results from an end-user's point of view.

For every pair of schemas, we chose to define the set of *correct* mappings as the set of mappings graded as 2 or 3 by the external expert. We compared then the standard alignments to the alignments produced automatically by our instance-based schema matcher taking advantage of two common information retrieval metrics: *precision* and *recall*. Intuitively, precision indicates the fraction of correct results. In our case, it is defined as the fraction of automatically generated and correct mappings over the total number of generated mappings for a pair of schemas. Recall indicates what fraction of the total number of correct results is returned. For us, this corresponds to the fraction of automatically generated mappings which are correct over the total number of correct mappings for the pair of schemas considered.

5.2. Sensitivity to the number of instances analyzed

Figs. 4 and 5 present the results of our experiments for two pairs of databases and for different numbers of instances analyzed at the data layer. Except when specified otherwise, we chose 80% for the threshold on the similarity value η and 10% for the threshold on the number of similarity values returned κ (see previous section). The results are quite encouraging for a totally automated method, with recall values close to 100%

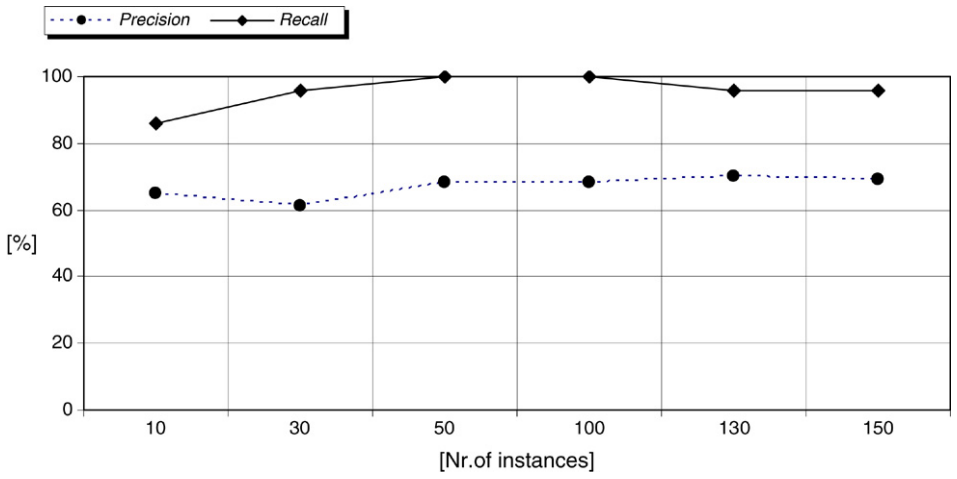


Fig. 4. Precision and Recall values for the SwissProt–EMBL mappings derived from sets ranging from 10 to 150 instances, $\eta = 80\%$, $\kappa = 10\%$.

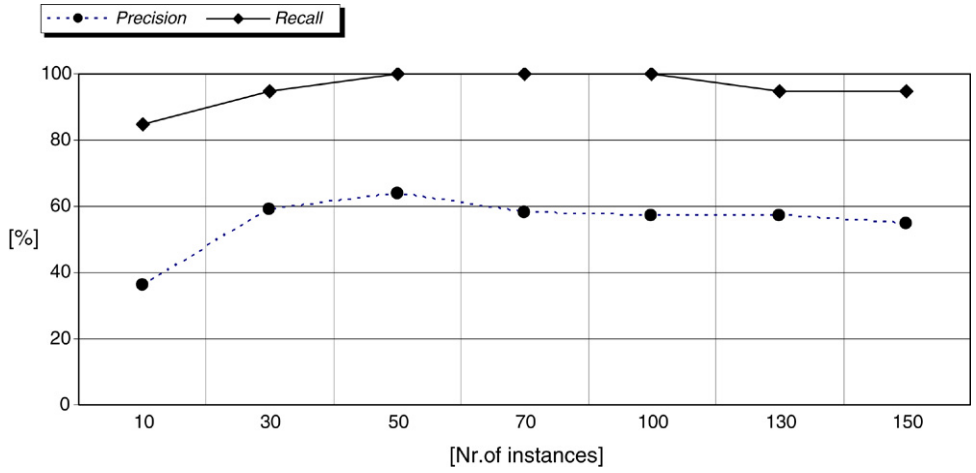


Fig. 5. Precision and Recall values for the SwissProt–PDB mappings derived from sets ranging from 10 to 150 instances, $\eta = 80\%$, $\kappa = 10\%$.

(i.e., practically all correct mappings have been discovered) and precision values averaging 65%. Note that precision values are higher for syntactically closer databases (e.g., SwissProt and EMBL, which share many similarities at the instance level). All figures show roughly convex functions for the precision and recall results with respect to the number of instances analyzed. Obviously, analyzing too few instances results in a biased view over the possible mappings, as some data dependencies might not appear clearly from a small arbitrary subset of the instances. Analyzing too many instances might result in slightly sub-optimal mappings as well, as the threshold κ causes infrequent but correct data-level dependencies to get lost when analyzing a bigger set of instances (e.g., when the correct mappings appear in less than 10% of the considered cases and as such are not included in the final result).

5.3. Sensitivity to the threshold on similarity values

The tradeoff between precision and recall results is well-known: by relaxing constraints on information retrieval techniques for example, one might retrieve a larger set of correct results (better recall) while retrieving at the same time many false positives (worse precision). In our case, this tradeoff is observable by taking various values for the threshold on the similarity (η). Fig. 6 shows some results for the SwissProt–EMBL case with a varying threshold.

5.4. On bootstrap attribute alignments

Our algorithm heavily relies on cross-referenced instances (materialized as predefined inter-schema relationships between pairs of attributes), which allow us to considerably narrow the search space by restricting ourselves to semantically related pairs of instances only. We call the pair of initially related attributes the *bootstrap* attributes as they are used to foster the rest of the mappings between the two schemas. As a next experiment, we looked into what other kinds of bootstrap alignments we could base our algorithm on, provided that we do not take advantage of cross-reference links. In other words, we wanted to verify whether minimal alignments, other than the existing foreign-key relationships, could be used to generate full-fledged schema mappings.

Thus, we ran several batteries of tests to determine whether a single mapping between two semantically related attributes (e.g. A_i^1 and A_j^2) at the schema level – *RT* from SwissProt and *JRNL* from PDB, for example – would be sufficient

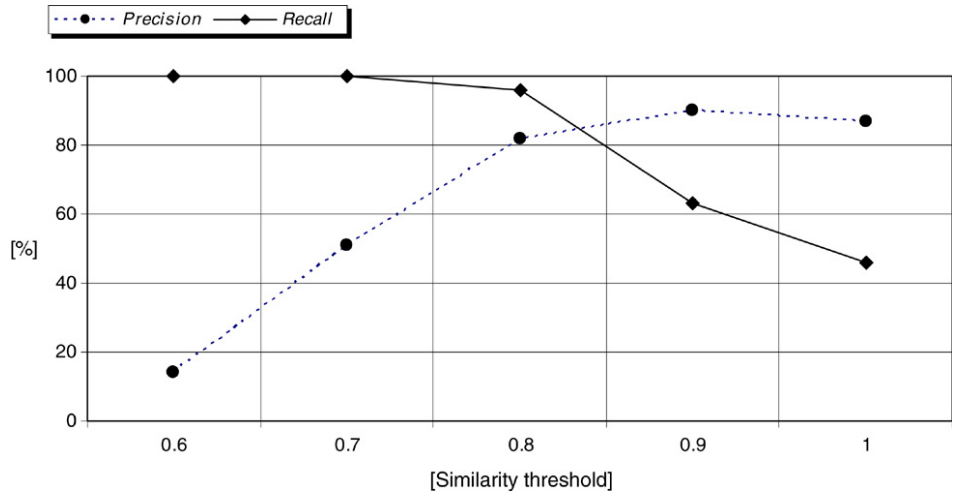


Fig. 6. Tradeoff between Precision and Recall of the SwissProt–EMBL mappings for similarity threshold values (η) ranging from 0.6 to 1.

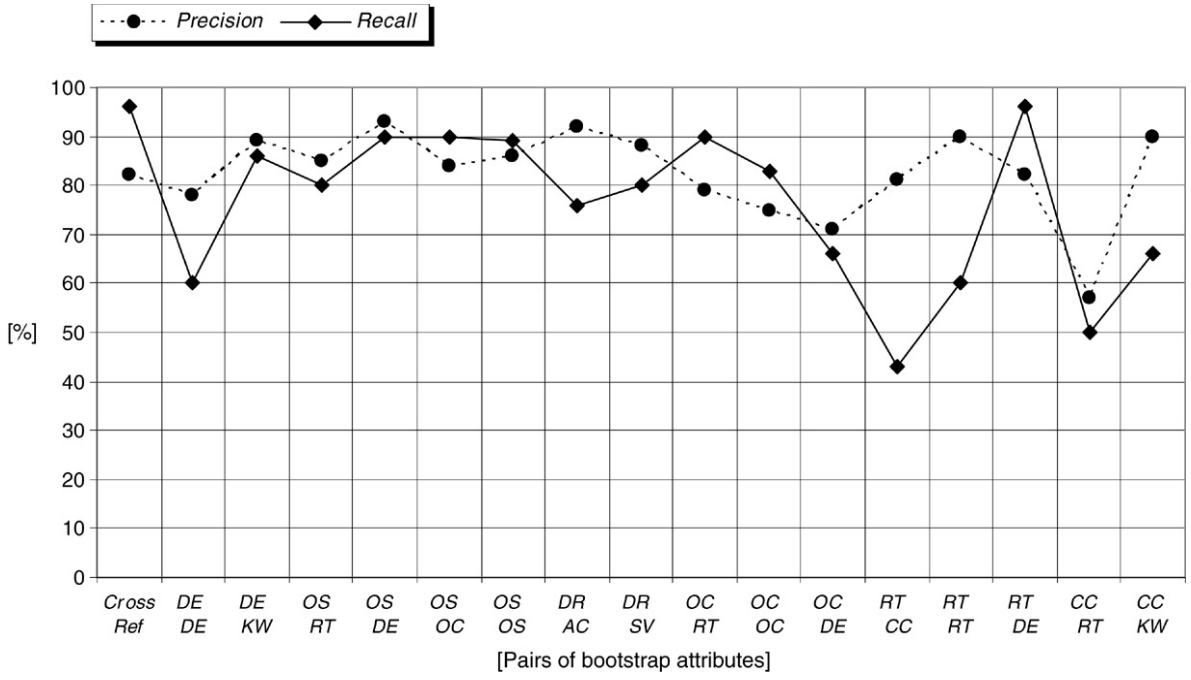


Fig. 7. Precision and Recall values for the SwissProt–EMBL mappings derived from different bootstrap attribute alignments on sets of 50 instances, $\eta = 80\%$, $\kappa = 10\%$.

for producing meaningful results using our instance-based methods. The matching process in this case is based on the assumption that if the values V_i^1 and V_j^2 contained in the two bootstrap attributes (at the data level) have a big degree of similarity, then the two instances are most probably semantically related. We start by choosing two databases (which contain cross-referencing instances) and by selecting a pair of bootstrap attributes whose values are semantically related. The algorithm works in two steps: first, it projects both the identifier and the value of the bootstrap attribute from the set of instances considered. This results in two sets containing relations of the form *(Instance identifier, Attribute value)* – $\{(ID_0^1, V_0^1), \ldots, (ID_n^1, V_n^1)\}$ – each corresponding to a database. So far, we have no knowledge of the pairs of instances which are semantically related. Next, we use the attribute value similarity metrics described above in order to infer pairs of potentially related instances. As a result, we return a set consisting of pairs of identifiers (ID_i^1, ID_j^2) indicating possibly related instances from the two databases. From this point on, we run the matching algorithm as described in Section 4 using as input the aforementioned pairs of instances.

The results obtained are shown in Figs. 7 and 8, based on an set of 50 pairs of instances analyzed. The first column indicates the precision and recall values corresponding to the analysis based on the cross-referenced attributes. The rest of the graph represents the precision and recall values when considering other pairs of attributes as bootstrap alignment, as described above. A first observation is that in both cases

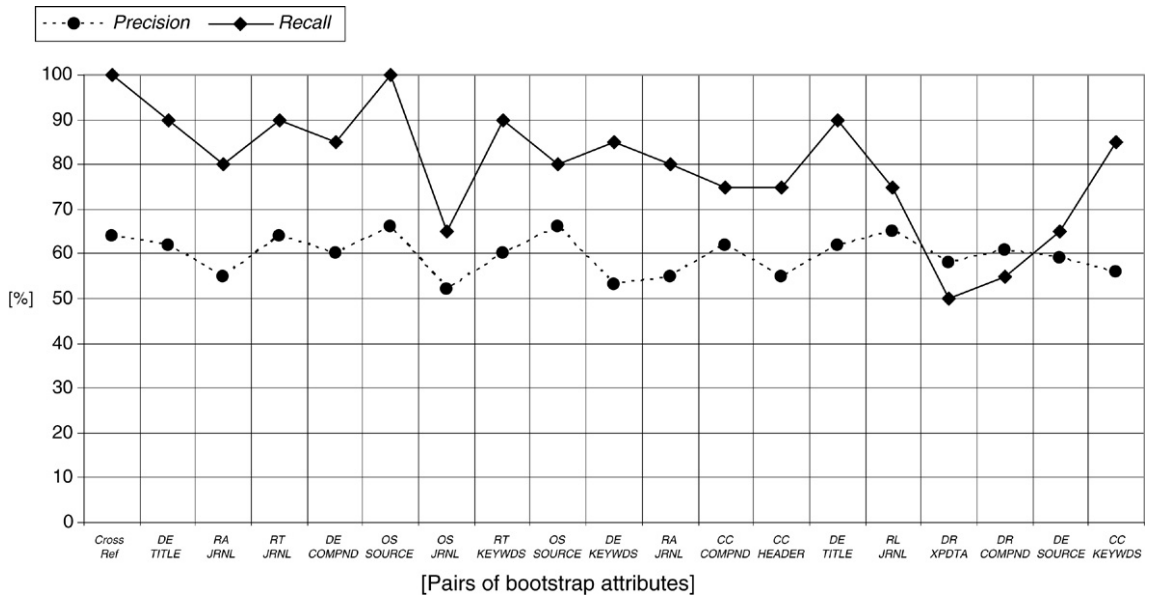


Fig. 8. Precision and Recall values for the SwissProt–PDB mappings derived from different bootstrap attribute alignments on sets of 50 instances, $\eta = 80\%$, $\kappa = 10\%$.

(cross-references and bootstrap-attribute matching), the results are quite satisfactory, with a precision of 82% and 64% in the case of the cross-reference mappings and an average precision of 82% and 60% in case of the bootstrap-attribute matching. The corresponding recalls are 96% and 100% for the cross-reference cases and averages of 75% and 76% for the bootstrap-attributes. Thus, the method based on bootstrap-attributes yields mappings almost as good as those generated from foreign-key relationships. The main reason behind this somewhat surprising result lies in the peculiar nature of the attributes in bioinformatic databases. Contrary to many other databases, attribute values in SRS are often distinct for every instance, thus implying numerous functional dependencies at the schema layer and representing *de facto* pseudo-identifiers for the instances.

In all cases considered, the recall value is highest for the matching based on cross-references. This is easy to explain, as we start our algorithm with pairs of instances that are always semantically related (the pairs of instances are selected using the SRS linking option), as opposed to the second case, where some of the pairs of the bootstrap mappings might have been wrongly selected. Similarly, recall is high when bootstrap attributes can act as pseudo-identifiers for the instances. As soon as too many collusions occur at the data level for the considered bootstrap attributes (i.e., when too many instances of one database share similar values for the bootstrap attributes), it starts to be difficult to identify related instances and recall values decrease. One typical example of this is the low quality of the results yielded by the bootstrap pairs containing the attribute CC, which supposedly contains free text but in practice often contains standard copyright statements and from which it is then very difficult to produce meaningful pairs of related instances: as the copyright statement is the same for most instances, it becomes quite hard to discover pairs of related instances based on this information only.

5.5. Bootstrapping alignments on several attributes

In this last scenario, we tried to increase the number of bootstrap-attributes to determine whether having a higher number of initial alignments would result in better mappings in the end. We considered two distinct cases: Fig. 9 shows the results when considering *two* pairs of bootstrap-attributes and taking the intersection of the two sets of related instances generated. Fig. 10 depicts results for a similar situation but taking this time the union of the two sets of related instances generated by the two bootstrap-attribute pairs. In case of the intersection, the results deteriorate because of the small number of instances examined (only the instances appearing in both result sets generated by the bootstrap attributes are considered). Precision and recall values of zero are even observed in one case (involving, again, the CC attribute, and highlighting once more the importance of selecting proper bootstrap attributes). On the other hand, the union of the two sets yields the same average precision as the identity mapping, but a slightly higher recall, due to the higher number of related instances considered in this case.

6. Establishing a bioinformatic data grid

Once schema mappings between the databases are generated – and possibly corrected by domain experts – we are theoretically ready to redeploy the whole network of databases as a decentralized, service-oriented Data Grid. To do so, we first have to export all schemas, instances and mappings in a standard way in order to enable the various Grid participants to meaningfully share and process data.

We chose to export all information using semi-structured, Semantic Web data models proposed by the W3C. Hence, instances and schemas are exported as RDF instances and schemas while mappings are encoded using the KnowledgeWeb ontology alignment standard [5]. Taking advantage of information included in the SRS structure files, we could

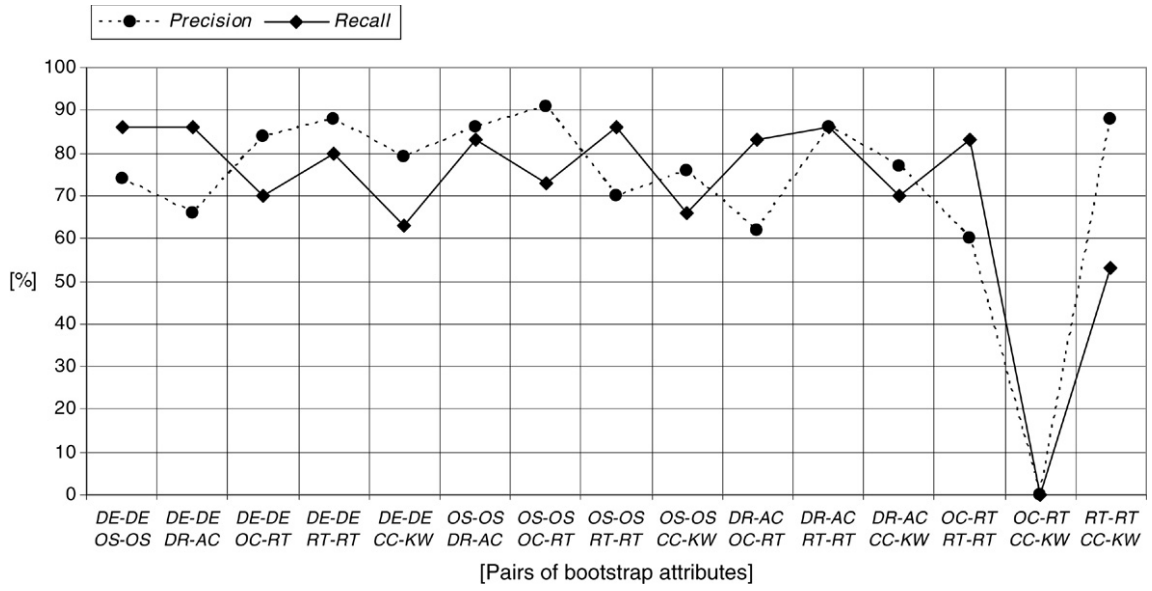


Fig. 9. Precision and Recall values for the SwissProt–EMBL mappings derived from two bootstrap-attribute pairs, by computing the Intersection of the result sets, on sets of 50 instances, $\eta = 80\%$, $\kappa = 10\%$.

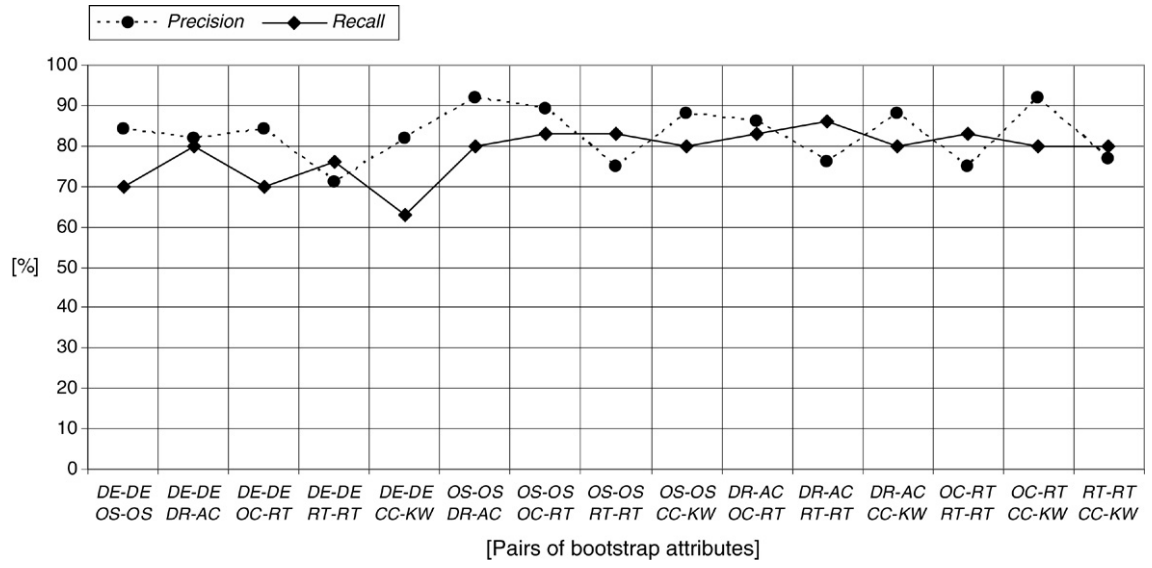


Fig. 10. Precision and Recall values for the SwissProt–EMBL mappings derived from two bootstrap-attribute pairs, by computing the Union of the result sets, on sets of 50 instances, $\eta = 80\%$, $\kappa = 10\%$.

implement a custom parser for the three aforementioned bioinformatic databases and automate the export of semi-structured data from the flat-files. Figs. 11 and 12 below show snippets of an exported schema and mapping respectively.

Each information source can export its data autonomously using these formats. In our prototype, we successfully used Jena (*jena.sourceforge.net*) as query processor running locally on every node. Once mappings to a couple of other participants in the Data Grid have been generated, a node can start posing queries globally by reformulating local queries thanks to the mappings as discussed in Section 1.1. The whole process results in a decentralized, scalable and service-oriented Data Grid where new participants can query information globally while providing new data in an autonomous way. Note that such a system can then be integrated in current Grid environments (see Section 2): local databases can for example take advantage of Grid directories to propagate queries or can serve as back-end infrastructure for operations triggered by standard Web-Service calls in the Grid.

7. Conclusions

Deploying a bioinformatic grid infrastructure implies tackling an old and difficult problem, namely the integration of heterogeneous databases. As Data Grids typically deal with large volumes of data, automated methods fostering semantic interoperability in distributed settings have to be proposed. In this paper, we introduced a completely decentralized approach to this problem, which enables an automatic transition from

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

    <rdfs:Class rdf:about="lsirwww.epfl.ch/BioGridData#BiologicalData"/>

    <rdfs:Class rdf:about="lsirwww.epfl.ch/BioGridData#Swissprot">
        <rdfs:subClassOf rdf:resource=
               "lsirwww.epfl.ch/BioGridData#BiologicalData"/>
    </rdfs:Class>

    <rdf:Property rdf:about="lsirwww.epfl.ch/BioGridData#Swissprot_RT">
        <rdfs:domain rdf:resource="lsirwww.epfl.ch/BioGridData#Swissprot"/>
    </rdf:Property>
</rdf:RDF>
```

Fig. 11. A snippet of an SRS schema exported as RDFS.

```
<?xml version=1.0 encoding=utf-8 standalone=no?>
 <rdf:RDF xmlns=http://knowledgeweb.semanticweb.org/heterogeneity/alignment
        xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
        xmlns:xsd=http://www.w3.org/2001/XMLSchema#>

 <Alignment>
 <xml>yes</xml>
 <level>0</level>
 <type>**</type>
 <onto1>lsirwww.epfl.ch/BioGrid/Swissprot_Onto</onto1>
 <onto2>lsirwww.epfl.ch/BioGrid/PDB_Onto</onto2>
 <map>
  <Cell>
   <entity1 rdf:resource=lsirwww.epfl.ch/BioGridData#Swissprot_DE/>
   <entity2 rdf:resource=lsirwww.epfl.ch/BioGridData#PDB_COMPND/>
   <measure rdf:datatype=http://www.w3.org/2001/XMLSchema#float>0.9</measure>
   <relation>=</relation>
  </Cell>
 </map>
 </Alignment>
</rdf:RDF>
```

Fig. 12. A snippet of a schema alignment exported as RDF.

a collection of centrally indexed but interrelated bioinformatic databases to a bioinformatic Data Grid. The interoperability is gradually established through the process of generating pairwise mappings between related schemas.

Our method takes advantage of foreign-key relationships often present in bioinformatic databases to run an instance-based schema matching algorithm which relies on lexicographic techniques for finding semantic correspondences between different schemas. In order to evaluate our method, we ran it on a set of databases found in an SRS repository and achieved surprisingly good results. Such methods are highly suitable to Grid environments, as they allow the establishment of integrated communications while maintaining local autonomy for the various information sources.

References

[1] K. Aberer, P. Cudré-Mauroux, Semantic overlay networks, in: International Conference on Very Large Data Bases, VLDB, 2005.

[2] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, The chatty web: emergent semantics through gossiping, in: International World Wide Web Conference, WWW, 2003.

[3] M.N. Alpdemir, A. Mukherjee, A. Gounaris, N.W. Paton, P. Watson, A.A.A. Fernandes, D. Fitzgerald, Ogsa-dqp: A service for distributed querying on the grid, in: International Conference on Extending Database Technology, EDBT, 2004, pp. 858–861.

[4] BLAST — Basic Local Alignment Search Tool, http://www.ebi.ac.uk/blast/.

[5] P. Bouquet, et al., Specification of a common framework for characterizing alignment, KnowledgeWeb Deliverable 2.2.1, http://knowledgeweb.semanticweb.org.

[6] C. Comito, A. Gounaris, R. Sakellariou, D. Talia, Data integration and query reformulation in service-based grids: Architecture and roadmap, in: 1st CoreGrid Workshop on Knowledge and Data Management, 2005.

[7] H. Do, S. Melnik, E. Rahm, Comparison of schema matching evaluations, 2002.

[8] A. Arshad et al., Heterogeneous relational databases for a grid-enabled analysis environment, in: Workshop on Web and Grid Services for Scientific Data Analysis, WAGSSDA, 2005.

[9] M. Antonioletti et al., Ogsa-dai: Two years on, in: The Future of Grid Data Environments Workshop, 2004.

[10] FASTA, http://www.ebi.ac.uk/fasta/.

[11] D. Frishman, K. Heumann, A. Lesk, H.W. Mewes, Comprehensive, comprehensible, distributed and intelligent databases: current status, Bioinformatics 14 (7) (1998) 551–561.

[12] P.V. Jithesh, N. Kelly, S. Wasnik, P. Donachy, T. Harmer, R. Perrott, M. McCurley, M. Townsley, J. Johnston, S. McKee, Bioinformatics application integration in genegrid, in: UK e-Science All Hands Meeting, 2005.

[13] Lion Bioscience, http://www.lionbioscience.com/.

[14] R. Milena, W. Bartosz, Life science grid middleware in a more dynamic environment, in: OTM Workshops, 2005, pp. 264–273.

[15] L. Moreau, S. Miles, C. Goble, M Greenwood, V. Dialani, M. Addis, N. Alpdemir, R. Cawley, D. De Roure, J. Ferris, R. Gaizauskas, K. Glover, C. Greenhalgh, P. Li, L. Xiaojian Liu an, P. Lord, M. Luck, D. Marvin,

T. Oinn, N. Paton, S. Pettifer, M.V. Radenkovic, A. Roberts, A. Robinson, T. Rodden, M. Senger, N. Sharman, R. Stevens, B. Warboys, A. Wipat, C. Wroe, On the use of agents in a bioinformatics grid, in: International Symposium on Cluster Computing and the Grid, CCGrid, 2003.

[16] F. Mougin, A. Burgun, O. Loreal, P. Le Beux, Towards the automatic generation of biomedical sources schema, Medinfo 11 (Pt 2) (2004) 783–787.

[17] BIO-GRID applications Overview and ICM configuration. P. bala. in: 1st Cracow Grid Workshop, 2001.

[18] C. Qu, F. Zimmermann, Application of standard semantic web services and workflow technologies in the simdat pharma grid, in: W3C Workshop on Frameworks for Semantics in Web Services, 2005.

[19] E. Rahm, P. Bernstein, A survey of approaches to automatic schema matching, 2001.

[20] R.D. Stevens, A.J. Robinson, C.A. Goble, mygrid: personalized bioinformatics on the information grid, Bioinformatics 19 (2003) i302–i304.

Adriana Budura is an assistant and a Ph.D. student in the Distributed Information Systems Laboratory at EPFL, Lausanne, Switzerland. Her research interests include peer data management systems and semantic interoperability in large-scale systems. Adriana obtained a M.Sc. in Computer Science (2005) from EPFL after graduating with a B.Sc. degree in Computer Science (2004) from the Politechnical University of Timisoara, Romania.

Philippe Cudre-Mauroux is a lecturer and Ph.D. student at EPFL. His research interests lie in the fields of decentralized information systems and peer data management systems. Philippe holds a B.Sc. in Communication Systems from EPFL (1999), a M.Sc. in Multimedia Communications from EPFL - Eurecom Institute (2001) and a M.Sc. in Networks and Distributed Systems from INRIA-U. Sophia Antipolis (2001). He worked on information distribution and management for HP, IBM T.J. Watson Research and Microsoft Research Asia. He is a member of the IFIP Working Group 2.6 on Databases.

Karl Aberer is a Professor for Distributed Information Systems at EPFL Lausanne, Switzerland, and director of the Swiss National Centre for Mobile Information and Communication Systems (NCCR-MICS). His research interests are on decentralization and self-organization in information systems with applications in peer-to-peer search, overlay networks, trust management and mobile and sensor networks. Before joining EPFL in 2000 he was leading the research division on open adaptive information systems at the Integrated Publication and Information Systems Institute (IPSI) of GMD in Germany, which he joined in 1992. He studied mathematics at ETH Zÿrich where he also completed his Ph.D. in theoretical computer science in 1991. From 1991 to 1992 he was postdoctoral fellow at the International Computer Science Institute (ICSI) at the University of California, Berkeley. He is member of several journal editorial boards, including VLDB Journal, and conference steering committees.