Proteus, a Grid based Problem Solving Environment for Bioinformatics: Architecture and Experiments

Mario Cannataro[1], Carmela Comito[2], Filippo Lo Schiavo[1], and Pierangelo Veltri[1]

***Abstract*— Bioinformatics can be considered as a bridge between life science and computer science. Biology requires high and large computing power to performance biological applications and to access huge number of distributed and (often) heterogeneous databases. Computer scientists and database communities have expertises in high performance algorithms computation and in data management. Considering bioinformatics requirements, in this paper we present** PROTEUS **, a Grid-based Problem Solving Environment for bioinformatics applications.** PROTEUS **uses ontology to enhance composition of bioinformatics applications. Architecture and preliminary experimental results are reported.**

***Index Terms*— Bioinformatics, Grid, Ontology, Problem Solving Environment (PSE).**

I. Introduction

RESEARCH in biological and medical areas (also known as *biomedicine*), requires high performance computing power and sophisticated software tools to treat the increasing amount of data derived by always more accurate experiments in biomedicine. The emerging bioinformatics area involves an increasing number of computer scientists studying new algorithms and designing powerful computational platforms to bring computer science in biomedical research. According to [5], Bioinformatics can thus be considered as *a bridge between life science and computer science*.

Biologists and computer scientists are working in designing data structure and in implementing software tools to support biomedicine in decoding the entire human genetic information sequencing (i.e. DNA), also known as *genome.* Even if many issues are still unsolved, (i.e., such as heterogeneous data sets integration and metadata definitions), the attention is now focused on new topics related to genomics. Today, the new challenge is studying the *proteome*, i.e. the set of *proteins* encoded by the genome, to define models representing and analyzing the structure of the proteins contained in each cell, and (eventually) to prevent and cure any possible cell-mutation generating human diseases such that producing cancer-hill cells [15].

Proteins characteristics can be simply represented by strings sequences encoding *amino acids* that are the basic building blocks composing proteins. Nevertheless, the high number of possible combinations of amino acids composing proteins, as well as the huge number of possible cell-mutation, require a huge effort in designing software and environments able to treat generic micro-biology problems. Moreover, proteins present spatial (i.e., three dimensional) structure that (partially) depends on amino acids composition: 3D protein structure predictions and folding are other important issues interesting medicine and drug discovery. Pattern matching algorithms and tools have to be combined with high performance multidimensional and imaging software tools to analyze and eventually prevent proteins behaviors.

Proteomics data sets in applications can be produced by experiments, or can be extracted from publicly available databases as those produced and maintained by research community: e.g. Protein Data Bank (PDB) [22], the SWISS-PROT protein database [29], the GenBank DNA sequences collections [21]. Optimized data models are required to represent protein structures as well as ”ad hoc” software tools are necessary to integrate and combine data obtained from experiments or from querying protein database and to extract information understandable by biomedical researchers. Nevertheless, heterogeneity both in data format and database access policy justify the interest of bioinformaticians for (biomedical-) data models, specialized software for protein searching and combinations, as well as data mining tools for information extraction from datasets. On the other hand, data and software distribution requires high performance computational platforms to execute distributed bioinformatics applications.

Computational Grids (or simply Grid) are geographically distributed environments for high performance computation [27]. In a Grid environment is possible to manage heterogeneous and independent computational resources offering powerful services able to manage huge volumes of data [28]. Grid community [14] recognized both bioinformatics and post-genomic as an opportunity for distributed high performance computing and collaboration applications. The Life Science Grid Research Group [24] established under the Global Grid Forum, believes bioinformatics requirements can be fitted and satisfied by Grid services and standards, and is interested in what new services should Grids provide to bioinformatics applications. In particular, given the number of applications requiring ability in reading large and heterogeneous datasets (e.g. protein databases) or in creating new datasets (e.g. mass spectrometry proteomic data [15]), a large number of biologist projects are investing in Grid environments as well as many computer scientists are investing in developing Bioinformatics applications on Grid (also known as *BioGrids*). E.g., the Asia Pacific BioGRID [4] is attempting to build a customized, self-installing version of the Globus Toolkit [32], a diffused environment for designing and managing Grid, comprising well tested installation scripts, avoiding dealing with Globus details. In the European Community Grid Project [31], whose

[1]University of Magna Graecia of Catanzaro, Italy `surname@unicz.it`
[2]University of Calabria, Italy `surname@si.deis.unical.it`

aim is funding Grid applications in selected scientific and industrial communities, the Bio-GRID work group is developing an access portal for biomolecular modeling resources [18]. The project develops various interfaces for biomolecular applications and databases that will allow chemists and biologists to submit work to high performance computing facilities, hiding Grid programming details. Finally, myGrid is a large United Kingdom e-Science project to develop open source data-intensive bioinformatics application on the Grid [30]. The emphasis is on data integration, workflow, personalization and provenance. Database integration is obtained both by dynamic distributed query processing, and by creating virtual databases through federations of local databases.

In this paper we consider a world where biomedical software modules and data can be detected and composed to define problem-dependent applications. We wish to provide an environment allowing biomedical researchers to search and compose bioinformatics software modules for solving biomedical problems. We focus on semantic modelling of the goals and requirements of bioinformatics applications using *ontologies*, and we employ tools for designing, scheduling and controlling bioinformatics applications. Such ideas are combined together using the Problem Solving Environment (PSE) software development approach [23]. A Problem Solving Environment is an integrated computing environment for composing, compiling, and running applications in a specific area [34], leaving the user free to work on application and not on software programming [9]. Grid-based PSEs are related to distributed and parallel computing and leverages basic Grid services and functionalities. E.g., the KNOWLEDGE GRID [13], based on the Globus Toolkit [32], is a Grid-based problem solving environment providing a visual environment (i.e., called VEGA) to design and execute distributed data mining applications on the Grid [12].

We present PROTEUS , a software architecture allowing to build and execute bioinformatics applications on Computational Grids [27]. The proposed system is a Grid-based Problem Solving Environment (PSE) for bioinformatics applications. We define an ontology-based methodology to describe bioinformatics applications as distributed workflows of software components. The architecture and first implementation of PROTEUS based on the KNOWLEDGE GRID [13], are presented. Also, we present use of PROTEUS to implement an application of human protein clustering. A preliminary version of this work can be found in [11].

The paper is organized as follows. Section II report biological data characterisics and environment requirements for bioinformatics applications. Section III presents a first implementation of PROTEUS based on KNOLEDGE GRID, reporting PROTEUS architecture and software modules. Section IV presents the ontology based processing to design bioinformatics applications with PROTEUS . Section V reports experiences on designing and running a simple case study of clustering human proteins using PROTEUS , and finally Section VI concludes the paper and outlines future works.

II. Bioinformatics Issues

Bioinformatics involves the design and development of advanced algorithms and computational platforms to solve problems in biomedicine. Applications deal with biological data obtained by experiments, or by querying heterogeneous and distributed databases. Methods for acquiring, storing, retrieving and analyzing such data are also necessary. In this section we sketch some characteristics of biological data, with particular emphasis to proteins data, and present some available biological databases. We then discuss about requirements of biological applications.

A. Biological Data and Databases

Handling biological data has to deal with exponentially growing sets of highly inter-related data rapidly evolving in type and contents. Designers of biological databases and querying engines have to consider some data management issues well known to database community. Biological data are often obtained combining data produced by experiments, or extracted by common databases. Data are thus often heterogeneous both in structure and content. Combining data coming from different sources requires human expertise to interact with different data format and query engines: e.g., data can be reported in text files or in relational tables or in HTML documents, while query interfaces may be textual or graphical (e.g., SQL-like, or query by example). Moreover, databases need to react to frequent data update: new data emerge regularly from new experimental results, thus databases must be updated and refreshed accordingly.

Biological data are often represented as string sequences and described using natural language. Most of the existing biological data represent data as flat file structured as a set of field/value pairs, weakly interconnected with indexing systems such as the Sequence Retrieval System (SRS) [7] (see below). Even 3D protein structures are often represented as raster images which content cannot be captured by any automatic query engine (e.g., based on similarity image matching), and need human interaction.

Biological data in bioinformatics comprise sequences of nucleotides (i.e., DNA) and sequences of amino acids (i.e., proteins). There are four different type of nucleotides, distinguished by the four bases: adenine (A), cytosine (C), guanine (G) and thymine (T), thus a single strand of DNA can be represented as a string composed of the four letters: A, C, G, T. A *triple* of nucleotides encodes an amino acid, while amino acids form proteins. Although there are $4^3 = 64$ different triples of nucleotides, in nature there exists only 20 different amino acids that can compose a protein. Each protein can be thus represented as a string composed by a 20-character alphabet, where each character represents an amino acid (e.g., G for glycine, A for alanine, V for valine, etc.). Since nucleotides and amino acids are represented with alphabet letters, the natural representation of a biological element (genes sequence or proteins sequence) is a string of characters. Data models are then based on string structures. To represent both nucleotides and amino acid chains, flat non-structured files as well as files enriched by field/value pairs structures can be used.

Structured data models (e.g., object oriented or relational [33]) are useful for data retrieval. Nevertheless, most of the useful biological databases are populated gathering data from different and often heterogeneous sources each providing its own database structure and query search engine. The data integration topic and the effort of defining uniform data model and query engine is another important issue that has been interesting computer scientists, for all kind of data. E.g., XML (eXtensible Mark up Language), the language for data exchange on the Web, has been attracting bioinformaticians. Thanks to its semi-structured nature [1], in XML it is possible to represent both data and (when present) structure in a single paradigm. XML query engine can filter data using their structure (if presents) and finally extract data using key-word based queries. Where still documents exists in different databases, XML ”abstract” documents [2] can be used to integrate heterogeneous data sources or as exchange mechanism (data mediator) between different databases. Moreover, *ontologies* can also be used for data integration. An Ontology is a system to share standard and unambiguous information about an observed domain. Ontologies are used to realize semantic tools to retrieve and analyze biological data coming from different data sources, using a given set of similar terminology. As we will see, PROTEUS utilizes ontologies to leverage users from knowing exactly all applications specifications and data locations and structures.

The existent biological databases contain protein and DNA sequences, 3D structures of protein sequences (i.e., images and description) and relationships between different sequences. They are mainly public available through the Web and offer database query interfaces and information retrieval tool to catch data coming from different databases. Most of them are produced and maintained by the research community; e.g., European Molecular Biology Laboratory (EMBL) [29] and American National Center for Biotechnology Information (NCBI) [21] give access to nucleotide and protein sequence databases. The former gives access to SWISS-PROT, a database of protein sequences obtained from translations of DNA sequences or collected from the scientific literature or applications. The latter maintains GenBank, a collection of all known DNA sequences. Moreover, a useful protein database is the Protein Data Bank (PDB) [22], that is a database of 3D-coordinates of macromolecular structures. Moreover two Web publicly available databases are the *Sequence Retrieval System (SRS)* and the *Entrez system*. SRS [7] is a Web-based retrieval system for biological data. It accesses to different available web databases and builds an index of URLs to integrate them. The index is used as a database view on different databases, providing a single interface allowing users to formulate queries on different databases. SRS provides the user with transparency from communication with sources (i.e. location, connection protocols and query language), but it does not provide guidance about source relevance for a given query, and no data integration is provided in the query results. Entrez [20] is the NCBI text-based search interface on the major biological databases (e.g., nucleotide database, protein sequence databases, structure databases, etc). Query results are obtained by combining data coming from different databases, using a proximity score grouping sequences and references based on similarity characteristics. Queries can be built using a ”query by example” based interface.

B. Biological Application Requirements

Novel Bioinformatics applications and in particular Proteomics ones, involve different data sets either produced in a given experiment, or available as public databases or different software tools and algorithms. Applications deal with (i) data sources, i.e. local and/or remote databases, and (ii) specialized services, algorithms and software components: e.g., pattern matching algorithms to match protein sequences in protein databases. From a computational point of view, it is necessary consider that Bioinformatics applications:

- are naturally distributed, due to the high number of involved data sets;
- require high computing power, due to the large size of data sets and the complexity of basic computations;
- access heterogeneous and distributed data, e.g. answering queries may require accessing several databases;
- need secure software infrastructures to manage private data.

Computational requirements have to deal with the sharing of computational resources, the integrated access to biological databases, as well as an efficient, large-scale data movement and replication. High performance requirements and distribution of software and data in Bioinformatics created a great interests in the Grid community.

Finally, software tools, data sources and Grid computational nodes, can be glued by using knowledge representation and management techniques. Defining semantic representation of data is one of the last challenge of the computer science community [26]. A possibility is using *ontologies* to build Knowledge Bases modeling knowledge about bioinformatics resources and processes. Basic retrieval techniques, as well as querying tools, can be used to extract knowledge by ontology databases.

III. PROTEUS: ARCHITECTURE AND SOFTWARE MODULES

This Section presents PROTEUS , a Grid-based Problem Solving Environment for composing, compiling, and running Bioinformatics applications on the Grid. To fulfill bioinformatics application requirements and to help biologists in their applications, PROTEUS introduces semantic modeling of Bioinformatics processes and resources, following an emergent trend in Semantic Grids and Knowledge Grids.

To fulfill bioinformatics application requirements, we propose a framework based on:

- Grids, with their security, distribution, service orientation, and computational power;
- Problem Solving Environment approach, useful to define, describe and execute (i.e. control) such applications;
- Ontologies, Web (Grid) Services, and Workflows technologies, at an inner level, to describe, respectively, the semantics of data sources, software components with their interfaces, and performances and bioinformatics tasks.

With the first item PROTEUS satisfies the high powerful computational requirements of bioinformatics applications. Moreover Grid environment is composed of distributed computational nodes, and fulfill the distributed nature of bioinformatics applications and data management.

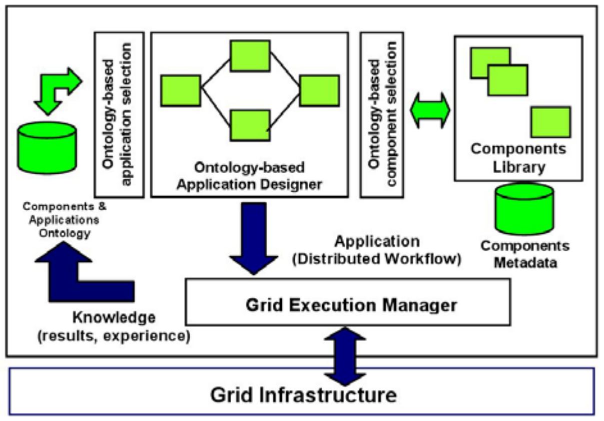

Ontology-based application selection
Ontology-based Application Designer
Ontology-based component selection
Components Library
Components & Applications Ontology
Application (Distributed Workflow)
Components Metadata
Grid Execution Manager
Knowledge (results, experience)
Grid Infrastructure

Fig. 1. PROTEUS General Architecture

PSE provide a dictionary of data and tools locations allowing users to build their applications disposing of all necessary tools. We imagine a world where biologists want to access a single tools and data virtual store where they may compose their applications. In particular, PROTEUS modules uses and combines open source bioinformatics software, and public-available biological databases. Private databases (i.e. databases accessible with registration via Web) can be also considered. Drawback in using open source packages (i.e., often defined in research environments) and in providing software tools, is that users have to know the nature of their data (i.e. their semantic) and details of software components, while they have to concentrate on biological domain and attended results. Moreover, the access to such components is often available by command line only. To overcome such problems, PROTEUS simplifies the use of software tools by adding metadata to available software and modelling applications through *ontology*. Ontologies are used to build PROTEUS Knowledge Base, modeling knowledge about bioinformatics resources and processes.

PROTEUS can be used to assist users in:

- formulating problems, allowing to compare different available applications (and choosing among them) to solve a given problem, or to define a new application as composition of available software components;
- running an application on the Grid, using the resources available in a given moment thus leveraging the Grid scheduling and load balancing services;
- viewing and analyzing results, by using high level graphic libraries, steering interfaces (that allow to interactively change the way a computation is conducted), and accessing the past history of executions, i.e. the past results, that form a knowledge base.

In the following, we present the PROTEUS overall architecture, while the next subsection describes a first implementation of the system and its main software modules.

A. Architecture

A main goal of PROTEUS is to leverage existing software easing the user work by: (i) adding metadata to software, (ii) modeling application through ontology, (iii) offering pre-packaged bioinformatics applications in different fields (e.g. proteomics), (iv) using the computational power of Grids. PROTEUS extends the basic PSE architecture and is based on the KNOWLEDGE GRID approach [13]. Main components of PROTEUS (see Figure 1) are:

- **Metadata repository** about software components and data sources (i.e. software tools, databases and data sources). It contains information about specific installed resources.
- **Ontologies**. We have two kinds of ontology in our system: a domain ontology and an application ontology. The domain ontology describes and classifies biological concepts and their use in bioinformatics as well as bioinformatics resources spanning from software tools (e.g. EMBOSS) to data sources (biological databases such as SWISS-PROT). The application ontology describes and classifies main bioinformatics applications, represented as workflows. Moreover it contains information about application's results and comments about user experience. Both ontologies contain references to data in metadata repository.
- **Ontology-based application designer**. An ontology-based assistant will either suggest the user the available applications for a given bioinformatics problem/task, or will guide the application design through a concept-based search of basic components (software and databases) into the knowledge base. Selected software components will be composed as workflows through graphic facilities.
- **Workflow-based Grid execution manager**. Graphic representations of applications are translated into Grid execution scripts for Grid submission, execution and management.

Ontologies and metadata are organized in a hierarchical schema: at the top layer ontologies are used to model the rationale of bioinformatics applications and software components, whereas at the bottom layer specific metadata about available (i.e. installed) bioinformatics software and data sources are provided. Ontology guides the user in the choice of the available software components or complete applications on the basis of her/his requirements (ontology-based application design) [8], whereas the low layer metadata will be used to really access software tools and databases, providing information like installed version, format of input and output data, parameters, constraints on execution, etc. When the application requires an installed tool, i.e. the ontology-based application design module issues a (resource) request, an ontology-based matchmaking algorithm finds the best match between the request and the available resources.

The ontology will be updated whenever new software tools or data sources are added to the system, or new applications are developed (i.e. designed through composition of software components). This enables the realization of a Knowledge Base of applications/results, which is enriched whenever new applications are developed or new results are obtained. Thus, new users may gain knowledge about pre-existing experiments.

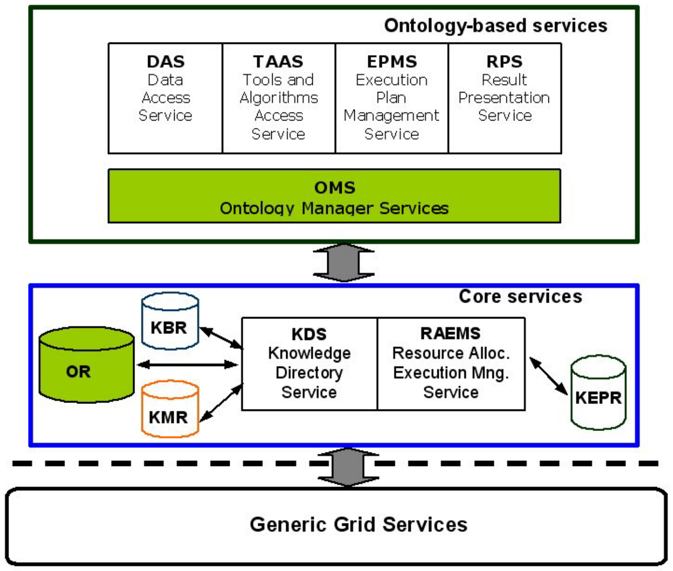

Ontology-based services
DAS Data Access Service
TAAS Tools and Algorithms Access Service
EPMS Execution Plan Management Service
RPS Result Presentation Service
OMS Ontology Manager Services
Core services
KBR
OR
KMR
KDS Knowledge Directory Service
RAEMS Resource Alloc. Execution Mng. Service
KEPR
Generic Grid Services

Fig. 2. Software Modules of PROTEUS

B. A First Implementation

The current implementation of PROTEUS is based on the KNOWLEDGE GRID, a joint research project of ICAR-CNR, University of Calabria, and University of Catanzaro, aiming at the development of an environment for geographically distributed high-performance knowledge discovery applications [13]. PROTEUS system modules are described in Figure 2. The ontology modules represent the main innovation with respect to the KNOWLEDGE GRID. It allows to describe bioinformatics resources (i.e. the Ontology Repository) offering new ontology-based services (i.e. the Ontology Management Services) to search and find the most appropriate software components needed to solve a bioinformatics task. We are working on PROTEUS implementation based on a new architecture specialized to support the complex workflows of bioinformatics applications on Grid [10].

Similarly to the KNOWLEDGE GRID, PROTEUS is built as a bag of services divided in two layers: the Core services that interface the basic Grid middleware and the Ontology-based services that interface the user by offering a set of services for the design and execution of bioinformatics applications.

The Core services allow the submission, execution, and control of a distributed computation over the Grid. Main services include the management of ontologies and metadata describing features of software components, applications and data sources. Moreover, this layer coordinates the application execution by attempting to fulfill the application requirements and the available grid resources. The Core services comprise:

- The Knowledge Directory Service (KDS) offers a uniform access to ontologies and metadata stored in the following repositories: resource ontology (OR), resource metadata (KMR), execution plans, i.e., application workflows (KEPR), and results of bioinformatics applications (KBR). The ontology is represented by a DAML+OIL [16] document stored in the Ontology Repository (OR), whereas metadata are represented as XML documents.
- The Resource Allocation and Execution Management Service (RAEMS) is used to find the best mapping between an execution plan and available Grid resources, with the goal of satisfying the application requirements and Grid constraints.

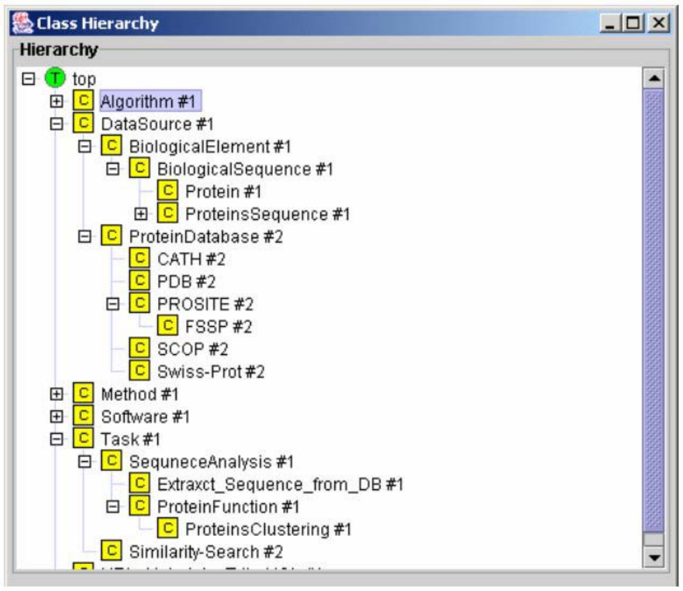
The Ontology-based services allow to compose, validate, and execute a parallel and distributed computation, and to store and analyze its results. The Ontology-based services comprise:

- The Ontology Management Services (OMS) offer a graphical tool for the ontology browsing, a set of utilities for the updating of the ontology, and a set of APIs for accessing and querying the ontology by means of a set of object-oriented abstractions of ontology elements. These services are used to enhance the following services.
- The Data Access Service (DAS) allows to search, select, extract, transform and delivery data to be analyzed.
- The Tools and Algorithms Access Service (TAAS) allows to search and select bioinformatics tools and algorithms.
- The Execution Plan Management Service (EPMS) is a semi-automatic tool that takes data and programs selected by the user and generates a set of different, possible execution plans (workflows) that meet user, data and algorithms requirements and constraints. Execution plans are stored into the KEPR.
- The Results Presentation Service (RPS) allows to visualize the results produced by a bioinformatics applications. The result metadata are stored in the KMR and managed by the KDS.

The design and execution of an application using PROTEUS run through the following steps:

1) Ontology-based resources selection. The search, location and selection of the resources to be used in the applications are executed by using the DAS and TAAS tools that invoke the OMS. Using the OMS the design process is composed of two phases:
    - Software tools and data sources selection. Browsing and searching the ontology allow a user to locate the more appropriate component to be used in a certain phase of the application.
    - XML metadata access. The ontology gives the URLs of all instances of the selected resources available on the grid nodes, i.e. the URLs of the relevant metadata files stored in the KMRs.
2) Visual application composition, through a graphical model that represents the involved resources and their relations.
3) Abstract execution plan generation, corresponding to the graphical model of the application. The plan is generated by using the EPMS services and then is stored into the KEPR.
4) Application execution on the Grid. The abstract execution plan is translated into a source Globus RSL (Resource Specification Language) script by the RAEMS module, then this script is submitted to the GRAM (Globus Resource Allocation Manager) service.

5) Results visualization and storing, by using the RPS services.



Class Hierarchy

Hierarchy

```
⊟ top
  ⊞ Algorithm #1
  ⊟ DataSource #1
    ⊟ BiologicalElement #1
      ⊟ BiologicalSequence #1
          Protein #1
        ⊞ ProteinsSequence #1
    ⊟ ProteinDatabase #2
        CATH #2
        PDB #2
      ⊟ PROSITE #2
          FSSP #2
        SCOP #2
        Swiss-Prot #2
  ⊞ Method #1
  ⊞ Software #1
  ⊟ Task #1
    ⊟ SequneceAnalysis #1
        Extraxct_Sequence_from_DB #1
      ⊟ ProteinFunction #1
          ProteinsClustering #1
      Similarity-Search #2
```

Fig. 3. Some Taxonomies of the Bioinformatics Ontology

IV. Ontologies in PROTEUS

Ontologies are used in PROTEUS to describe the semantics of the components and data resources involved in applications. In this section we describe a first Bioinformatics Ontology, and its management using the Ontology Management Services.

A. An Ontolgy on Bioinformatics Domain

Currently PROTEUS presents an ontology on bioinformatics domain that tries to integrate different aspects of bioinformatics, including computational biology, molecular biology and computer science. In such ontology we classify the following bioinformatics resources:

1) biological data sources, such as protein databases (e.g., SwissProt, PDB);
2) bioinformatics software components, such as tools for retrieving and managing biological data (e.g., SRS, Entrez, BLAST, EMBOSS);
3) bioinformatics processes/tasks (e.g. sequence alignment, similarity search, etc.).

The modelling of the above cited bioinformatics resources, has been made on the basis of classification parameters that will guide users in the composition of the application and in the choosing of the most suitable resources to use.

Biological data sources have been classified on the basis of the following features:

- the kind of biological data (e.g., proteins, genes, DNA);
- the format in which the data is stored (e.g., sequence, BLAST proteins sequence);
- the type of data source (e.g., flat file, relational database, etc);
- the annotations specifying the biological attributes of a database element.

Bioinformatics processes and software components have been organized in the ontological model on the basis of the following parameters:

- the *task* performed by the software components; that is the typology of the bioinformatics process (e.g., sequence analysis, secondary structure prediction, etc);
- the steps composing the task and the order in which the steps should be executed;
- the methodology (*method*) that the software uses to perform a bioinformatics task;
- the *algorithm* implemented by the software;
- the *data source* on which the software works on;
- the kind of *output* produced by the software;
- the *software* components used to perform a task (e.g. BLAST, EMBOSS, etc.).

Taxonomies that specialize each of those classification parameters have been partially implemented. Every taxonomy specializes the concept of interest using two kinds of relationships through which simple/multiple inheritance could be applied: the first kind of relationship is the *specialisation/generalisation* ("is-a") relationship that specialises/generalises general/specific concepts in more specific/general ones; and the *part of/has part* relationship that defines a partition as subclass of a class. Figure 3 shows some taxonomies of the ontology by using the OilEd ontology editor [6].

Thus we have organized our ontological model in such a way to have a large number of small local taxonomies that may be linked together via non-taxonomic relations. As an example, since every software performs a task, the *Software* taxonomy is linked to the *Task* taxonomy through the *PerformsTask* relation. The ontology can be explored by choosing one of the previous classification parameters. For example, exploring the *Task* taxonomy it is possible to determine for a given task what are the available algorithms performing it and then which software implements the chosen algorithm. Moreover it is possible to find the data sources and the biological elements involved in that task. On the other hand, exploring the *Algorithm* taxonomy it is possible to find out the biological function behind an algorithm, the software implementing it, the kind of data source on which it works.

B. The Ontology Management Services

PROTEUS offers ontology-based services and as such it needs a means through which manipulate and access ontologies stored in the *Ontology Repository* (see Figure 2). To this aim we introduced in the architecture shown in Figure 2 the *Ontology Management Services* (OMS). The OMS provides a set of high-level services for managing ontologies such as utilities for browsing and querying them. These utilities are supplied both as graphical tools as well as a set of Java APIs.

The API implementation is realized for accessing and querying the ontology: the API will provide a set of object-oriented abstractions of ontology elements such as Concept, Relation, Properties, and Instance objects providing query facilities.

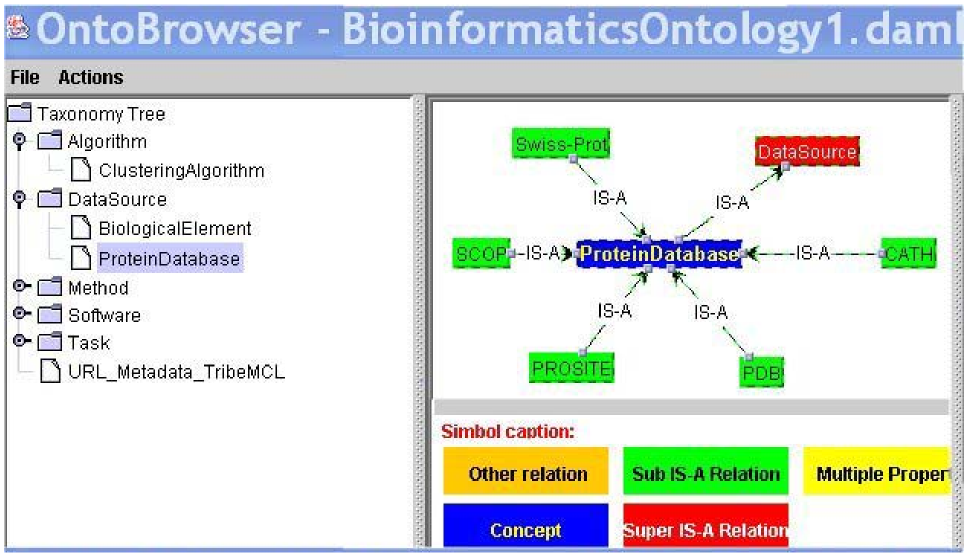


Fig. 5. Snapshot of the Ontology Browser

The graphical tool provides a combined search and browse facility over the ontology:

- *Ontology querying*. Through the ontology-based search engine offered by the OMS, user can find detailed information about domain resources modeled in the ontology. The result set is accurate, because the semantic of the target terms is indicated by concepts from the underlying ontology. Our ontology-based search engine supports several kinds of simple inference that can serve to broaden queries including equivalence (to restate queries that differ only in form), inversion, generalization, and specialization to find matches or more general or more specific classes and relations. If the result set of a query is empty, the user can at least find objects that partially satisfy the query: some classes can be replaced by their superclasses or subclasses. Both narrowing and broadening the scope of the query are possible due to the ontological nature of the domain description.
- *Ontology browsing*. The ontology browser is a navigation facility that presents an overview of the whole data set: it shows the classes, their relations and instances. The browser gradually presents deeper levels of the ontology: the user starts at the top of the ontology and can navigate towards more specific topics by clicking the classes of interest (diving into the information).

Since we have implemented the ontology in the DAML+OIL ontology language, the services offered by the OMS allow support only for DAML+OIL [16] encoded ontologies. At this time we have implemented a graphical tool for the browsing of ontologies (see Figure 5); using such tool the user browses the ontology choosing one of the input point (left panel of the frame) representing the taxonomies of the ontology and navigates visiting the sub tree topics until reaching a concept of interest. The concept of interest is shown in the middle of the right panel of the frame and related concepts are displayed around it. The ontology may be browsed by promoting any of the related concepts to be the central concept. The new central concept is then linked to all its related concepts.

V. A CASE STUDY: CLUSTERING OF HUMAN PROTEINS

This Section presents some first experimental results obtained implementing a simple bioinformatics application. We first present the overall application workflow, and then we discuss the design of such application. Currently, the application is first designed by using the Ontology Management Services described in the previous section, and then the selected resources are composed into a Data Flow Diagram by using VEGA (*Visual Environment for Grid Applications*) [12], the KNOWLEDGE GRID user interface.

Protein function prediction uses database searches to find proteins similar to a new protein, thus inferring the protein function. This method is generalized by protein clustering, where databases of proteins are organized into homogeneous families to capture protein similarity. We implemented a simple application for the clustering of human proteins sequences using the TribeMCL method [3]. TribeMCL is a clustering method through which it is possible to cluster correlated proteins into groups termed ”protein family”. This clustering is achieved by analysing similarity patterns between proteins in a given dataset, and using these patterns to assign proteins into related groups. In many cases, proteins in the same protein family will have similar functional properties. TribeMCL uses the Markov Clustering (MCL) algorithm [17].

We organized the application (see Figure 4) into four phases: the *Data Selection phase* extracts sequences from the database, the *Data Preprocessing phase* prepares the selected data to the clustering operation, the *Clustering phase* performs the Markov Clustering algorithm to obtain a set of protein clusters, and finally the *Results Visualization phase* displays the obtained results.

In the Data Selection phase all the human protein sequences are extracted from the Swiss-Prot database using the `seqret` program of the EMBOSS suite. EMBOSS is a package

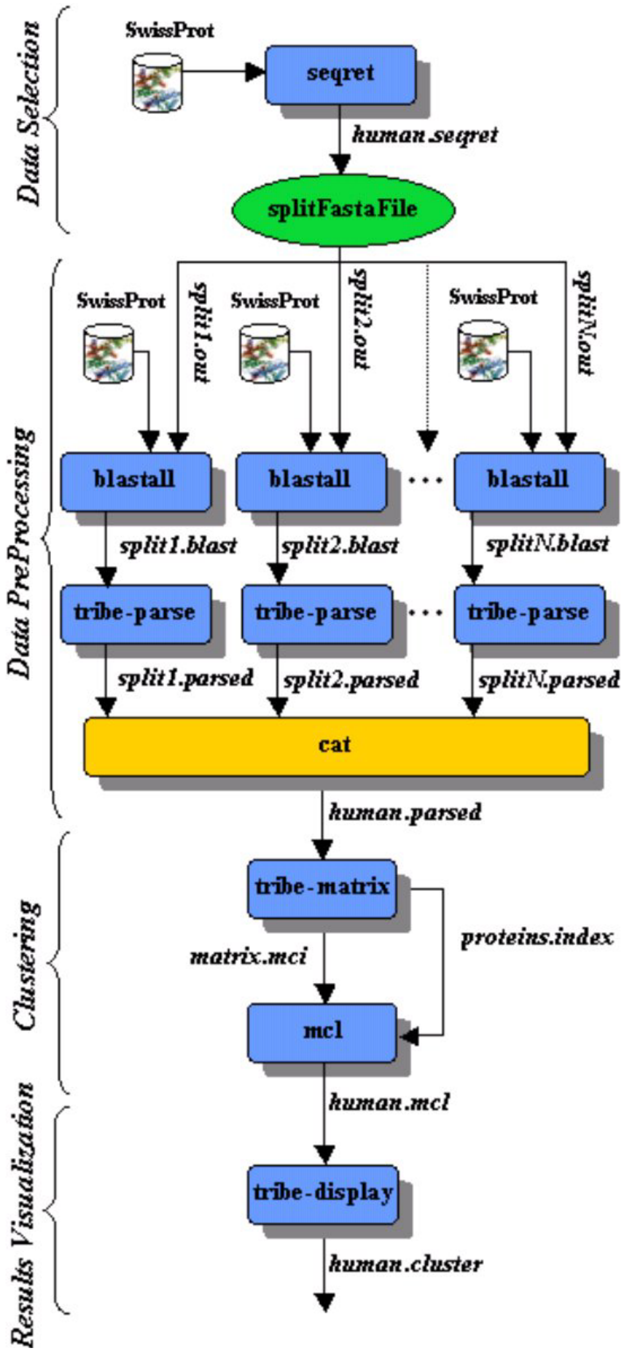


Fig. 4. Human Protein Clustering Workflow

of high-quality Open Source software for sequence analysis [25]. `seqret` is a program for extracting sequences from databases: in our application this program reads sequences from the database and then write them to a file.

TribeMCL needs a BLAST comparison on its input data. BLAST is a *similarity search* tool based on string matching algorithm [19]. Given a string it finds string sequences or sub-sequences matching with some of the proteins in a given database (*alignment*). BLAST carries out *local alignments* between sequences or between a sequence and protein database. Local alignment algorithms look for protein string matching between protein subsequences. It ranks the subsequence results using an expectation value (e-value). Given a sequence, it is able to return the probability of a particular alignment to occur. E.g., an e-value equal to zero means that the probability for a given alignment to occur by chance is zero. In particular, TribeMCL uses an *all against all* BLAST comparison as input to the clustering process, thus once the protein sequences have been extracted from the database, a BLAST computation has to be performed.

The Data Preprocessing phase comprises the following steps. To speed up the similarity search activity we partitioned the `seqret` output in three smaller files; in this way three BLAST computations can be run in parallel. The obtained raw NCBI BLAST outputs are converted in the format required to create the Markov Matrix used in the clustering phase by TribeMCL. The parsing has been executed by using `tribe-parse` program. Finally, the files obtained in the `tribe-parse` steps are concatenated by using the `cat` program.

In the Clustering phase, the Markov Matrix is built by using the `tribe-matrix` program that produces the `matrix.mci` and `proteins.index` files. Then the clustering program `mcl` is executed using the file `matrix.mci`.

Finally, in the Results Visualization phase the clustered data are arranged in an opportune visualization format.

A. Application Development on PROTEUS

In VEGA the resources are just described by basic metadata about technical details, and it does not provide any semantic modelling. Moreover, users have to browse metadata on each Grid node to search and select the resources needed in an application.

In order to overcome these limitations, we have supplied the VEGA environment with an ontological modelling of the bioinformatics resources and an ontologies mananging tool.

The proposed Ontology Management Services can be used both to enhance the application formulation and design, and to help users to select and configure available resources (software components and data sources).

The first step in the development of bioinformatics applications on PROTEUS is the *Ontology-based resource selection* in which the user browses the ontology locating the more appropriate components to use in the application. Next, the selected resources are composed through the graphical model of VEGA (*Visual application composition*).

The application workflow shown in Figure 4 has been modelled as a set of VEGA workspaces [12]. We briefly remind that a computation in VEGA is organized in *workspaces*. The jobs of a given workspace are executed concurrently; whereas workspaces are executed sequentially. The implementation of our application required the development of 13 workspaces grouped into the four different phases of the application: *Data Selection*, *Data Preprocessing*, *Clustering* and *Results Visualization*.

Consider the following scenario: a PROTEUS user logged on the host `minos` wants to define and execute the clustering of human proteins. He/she only knows that needs a protein sequences database from which to retrieve the sequences and a software tool performing the clustering process. Moreover,

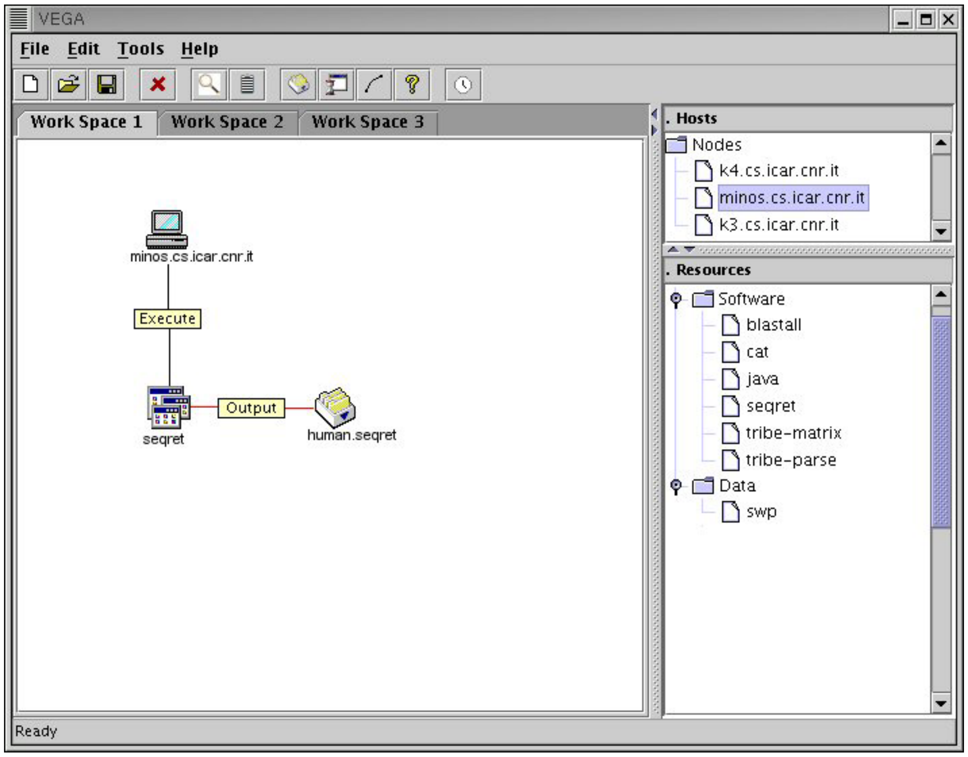


Fig. 6. Snapshot of VEGA: Workspace 1 of the Data Selection Phase

let suppose that Grid nodes are configured as shown in Table I and the `Swiss-Prot` database is replicated on each of them.

As a first step of the application formulation, the user browses the *Data Source* taxonomy (see Figure 5) of the domain ontology to locate the `Swiss-Prot` database. After that he/she searches software for extracting sequences from the database. Thus the user starts the ontology browsing from the *Task* taxonomy and identifies the `Extracting-sequences-from-DB` concept. From there following the `performed-by` label the user finds the seqret program (see Figure 7) and through its metadata file he/she locates the software on the `minos` node.

Software Components	Grid Nodes		
	minos	k3	k4
seqret	•		
splitFasta	•		
blastall	•	•	•
cat	•	•	•
tribe-parse	•	•	•
tribe-matrix	•		
mcl		•	
tribe-families			•

TABLE I
SOFTWARE INSTALLED ON THE EXAMPLE GRID

At this point the user is ready to design the *Data Selection* phase through VEGA constructing the following three work spaces:

1) Workspace 1. The human protein sequences are extracted from the `SwissProt` database using the `seqret` program on `minos` (see Figure 6).
2) Workspace 2. The file obtained as result of the `seqret` execution is partitioned in three smaller files using the `splitFasta` java utility class available on `minos` producing the files `split1.out`, `split2.out` and `split3.out`.
3) Workspace 3. `split2.out` and `split3.out` files are transferred respectively on `k3` and `k4` nodes.

The next step in the application design is to identify the tool performing the clustering process. To this aim the user starts the ontology browsing from the *Task* taxonomy (see Figure 7) and identifies the `proteins-clustering concept` (see Figure 8). From this point following the `performed-BySoftware` property, the user finds out that `TribeMCL Tool` is a software used for the clustering of proteins (see Figures 8, 9). The *HasInput* property specifies that TribeMCL takes as input the results of a BLAST computation, and the *producesOutput* property states that output is a clustering of protein families.

Following the `HasMetadata` link the user finds the URL of the software metadata file. This file other than locating on which Grid nodes the tool is installed, contains information about how to access and use the tool, e.g. TribeMCL tool uses an *all against all* BLAST comparison as input to the clustering computation. Once again the user traverses the ontology to search the opportune version of the BLAST software needed in the process. This time the user explores the *Software Tool* taxonomy in the direction of the `similarity-search-sw` concept and from here identifies the BLAST tool and thus the `blastp` program needed.

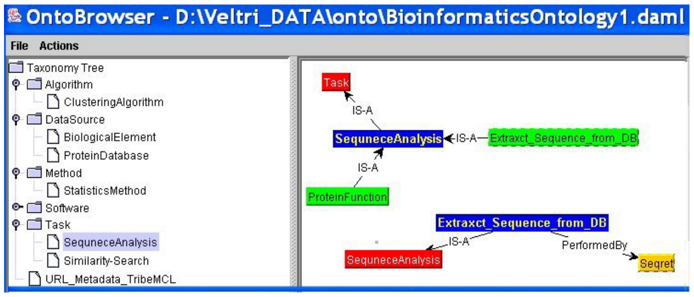


Fig. 7. Snapshot of the ontology browser

The Data Preprocessing phase consists of four VEGA work spaces:

1) Workspace 1. The BLAST computation is performed on the three nodes involved in the application containing the output files of the first phase (see Figure 10).
2) Workspace 2. The sequence similarity search output files are parsed using the `tribe-parse` software installed on three nodes.
3) Workspace 3. The files created on the nodes `k3` and `k4` in the Workspace 2 are transferred to the `minos` node where the software necessary to construct the Markov matrix is available.
4) Workspace 4. `cat` execution to concatenate the files.

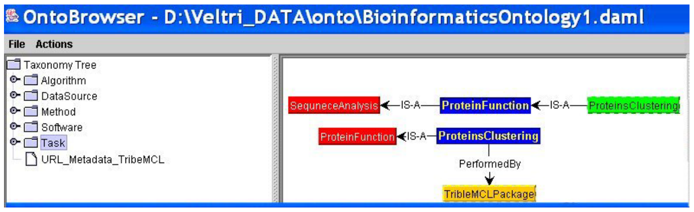


Fig. 8. Snapshot of the Ontology Browser

Once the files have been parsed using `tribe-parse`, it is possible to build the Markov matrix using the `tribe-matrix` program and perform the clustering operation. To this aim we have organized the Clustering phase into three VEGA workspaces:

1) Workspace 1. The Markov matrix is built using the `tribe-matrix` program installed on `minos`
2) Workspace 2. The `matrix.mci` file is transferred to `k3` where the clustering program `mcl` is available.
3) Workspace 3. `mcl` execution producing the `human.mcl` file.

Finally the Result Visualization phase has been organized in three VEGA workspaces:

1) Workspace 1. The `human.mcl` and the `protein.index` files are transferred on `k4` node
2) Workspace 2. The `tribe-families` program is executed on `k4` producing the file `human.cluster`.
3) Workspace 3. The final result, `human.cluster`, is transferred on `minos` to make it available to the user.

B. Experimental Results

The measurement of the execution times has been done in two different cases: a) we considered only 30 human proteins, and b) all the human proteins in the Swiss-Prot database (see Table II). Comparing the execution times shown in Table II we note that:

- The Data Selection and Results Visualization phases take the same time for the two cases, meaning that sequences extraction, file transfers and results displaying do not depend on the proteins number to be analyzed.
- In the Pre-processing phase there is a huge difference between the execution times of the two cases: the BLAST computations considering all the proteins are computationally intensive, so we have *8h50'13"* in the all proteins case compared to *2'50"* of the 30 proteins case.
- The execution of the mcl clustering program in the Clustering phase is a computationally intensive operation and consequently takes much more time when all the proteins have to be analyzed (*2h50'28"* versus *1'40"*). Note that the matrix file transferring time is the same for both applications.

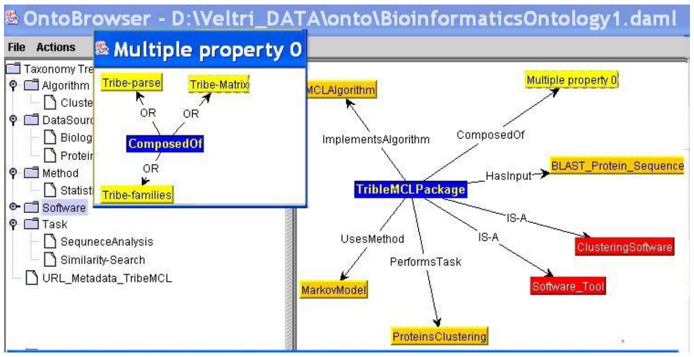


Fig. 9. Snapshot of the Ontology Browser

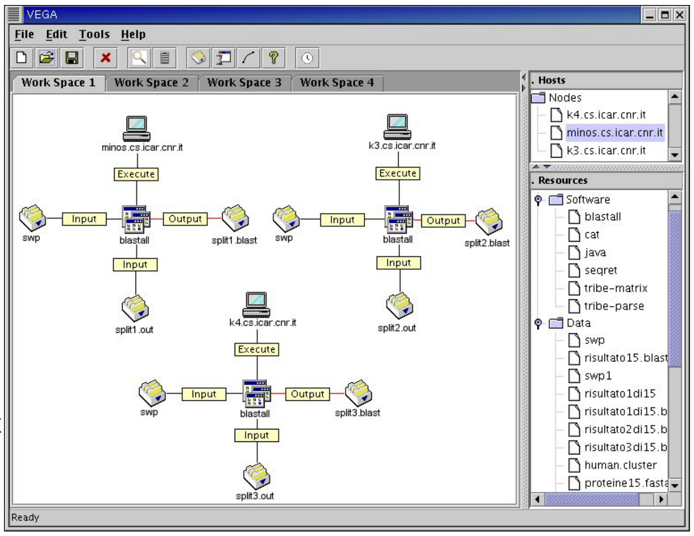


Fig. 10. Snapshot of VEGA: Workspace 1 of the Pre-processing Phase

Finally, a sequential version of the application, all human proteins case, has been executed on the minos host. This computation has taken a total execution time of *26h48'26"* compared to the *11h50'53"* of the parallel version. Moreover, some problems occurred in the management of the BLAST output file by the tribe-parsing program due to the high dimension of the file (about 2GB).

VI. Conclusion and Future Work

Novel Bioinformatics applications, and in particular Proteomics applications, will involve different software tools and various data sets, either produced in a given experiment, or available as public databases. Such applications will need a lot of semantic modeling of their basic components and will require large computational power.

In this paper we presented the design and implementation of PROTEUS , a Grid-based Problem Solving Environment for Bioinformatics applications. PROTEUS uses an ontology-based methodology to model semantics of bioinformatics applications. The current implementation of PROTEUS , based on the KNOWLEDGE GRID, has been successfully used to implement an application of human protein clustering.

We are improving PROTEUS architecture and functionalities by adding workflows methodologies for designing and monitoring applications [10]. Future works will regard the full implementation of PROTEUS and its use for the advanced analysis of proteomic data produced by mass spectrometry, for the early detection of inherited cancer [15].

TribeMCL Application	**Execution Time**	
Data Selection	30 proteins	*1'44"*
	All proteins	*1'41"*
Pre-processing	30 proteins	*2'50"*
	All proteins	*8h50'13"*
Clustering	30 proteins	*1'40"*
	All proteins	*2h50'28"*
Results Visualization	30 proteins	*1'14"*
	All proteins	*1'42"*
Total Execution Time	30 proteins	*7'28"*
	All proteins	*11h50'53"*

TABLE II
EXECUTION TIMES OF THE APPLICATION

Acknowledgment

This work has been partially supported by Project "FIRB GRID.IT" funded by MIUR. Authors are grateful to Domenico Talia for several suggestions on the main topic of this paper: we owe him many ideas on Grid use and applications. Authors also thank Antonio Massara, for support on the DAM+OIL ontolgy browser. Finally, authors are particularly grateful to Antonio Congiusta for discussion and contributions on the first implementation of PROTEUS on Vega System.

References

[1] S. Abiteboul and P. Buneman D. Suciu. *Data on the Web*. Morgan Kauffman, 2000.
[2] Vincent Aguilra, Sophie Cluet, Tova Milo, Pierangelo Veltri, and Dan Vodislav. Views in a Large Scale XML Repository. *VLDB Journal*, 11(3), November 2002.
[3] Enright A.J., Van Dongen S., and Ouzounis C.A. Tribemcl: An efficient algorithm for large scale detection of protein families. http://www.ebi.ac.uk/research/cgg/tribe/.
[4] ApBIONet.org. Asia pacific biogrid initiative. http://www.ncbi.nlm.nih.gov/.
[5] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. MIT Press, 1998.
[6] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reasonable ontology Editor for the Semantic Web. In *Artificial Intelligence Conference*. Springer Verlag, September 2001.
[7] LION bioscience AG. Srs search data bank system. http://srs.ebi.ac.uk/.
[8] M. Cannataro and C. Comito. A DataMining Ontology for Grid Programming. In *Workshop on Semantics in Peer-to-Peer and Grid Computing (in conj. with WWW2003)*, Budapest-Hungary, 2003.
[9] M. Cannataro, C. Comito, A. Congiusta, G. Folino, C. Mastroianni, A. Pugliese, G. Spezzano, D. Talia, and P. Veltri. Grid-based PSE Toolkits for Multidisciplinary Applications. FIRB "Grid.it" WP8 Working Paper 2003/10, ICAR-CNR, December 2003.
[10] M. Cannataro, C. Comito, A. Guzzo, and P. Veltri. Integrating Ontology and Workflow in PROTEUS, a Grid-Based Problem Solving Environment for Bioinformatics. Technical report, Univ. of Catanzaro, 2003.
[11] M. Cannataro, C. Comito, F. Lo Schiavo, and P. Veltri. PROTEUS: a Grid Based Problem Solving Environment for Bionformatics. In ISBN 0-9734039-0-X, editor, *Workshop on DataMining Ontology for Grid Programming (KGGI 03)*, Halifax-canada, 2003.
[12] M. Cannataro, A. Congiusta, D. Talia, and P. Trunfio. A Data Mining Toolset for Distributed High-performance Platforms. In Wessex Inst. Press, editor, *Data Mining Conference*, 2002. Bologna, Italy.
[13] M. Cannataro and D. Talia. KNOWLEDGE GRID An Architecture for Distributed Knowledge Discovery. *Communication of ACM*, 46(1), 2003.
[14] Grid Community. Global grid forum. http://www.gridforum.org/.
[15] G. Cuda, M.Cannataro, B. Quaresima, F. Baudi, R. Casadonte, M.C. Faniello, P. Tagliaferri, P. Veltri, F.Costanzo, and S. Venuta. Proteomic Profiling of Inherited Breast Cancer: Identification of Molecular Targets for Early Detection, Prognosis and Treatment, and Related Bioinformatics Tools. In *WIRN 2003, LNCS*, volume 2859 of *Neural Nets*, Vietri sul Mare, 2003. Springer Verlag.
[16] Daml.org. Daml+oil language. http://www.daml.org/2001/03/daml+oil-index.html.
[17] A.J. Enright, S. Van Dongen, and C.A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids*, 30(7), 2002.
[18] EUROGRID. Biogrid. http://biogrid.icm.edu.pl/.
[19] NCBI-National Cancer for Biotechnology Information. Blast database. http://www.ncbi.nih.gov/BLAST/.
[20] NCBI-National Cancer for Biotechnology Information. Entrez, the life science search engine. http://www.ncbi.nlm.nih.gov/Entrez/Index.html.
[21] NCBI-National Cancer for Biotechnology Information. Genbank dna sequences. http://www.ncbi.nlm.nih.gov/.
[22] Research Collaboratory for Structural Bioinformatics (RCSB). Protein data bank (pdb). http://www.rcsb.org/pdb/.
[23] S. Gallopoulos, E.N. Houstis, and J. Rice. Computer as Thinker/Doer: Problem-Solving Environments for Computational Science. In *Computational Science and Engineering*. IEEE, 1994.
[24] Grid.org. Grid life science group. http://forge.gridforum.org/projects/lsg-rg.
[25] EMBOSS Group. The european molecular biology open software suite. http://www.emboss.org.
[26] WWW Semantic Group. World wide web semantic group. http://www.w3c.org/semantic/.
[27] Foster I. and Kesselman C. *The Grid: Blueprint for a Future ComputingInfrastructure*. Morgan Kaufmann Publishers, 1999.

[28] W. E. Johnston. Computational and Data Grids in Large-Scale Science and Engineering. *Future Generation Computer Systems*, 18, 2002.
[29] EMBL-European Molecular Biology Laboratory. The swiss-prot protein database. http://www.embl-heidelberg.de/.
[30] University of Manchester. mygrid. http://mygrid.man.ac.uk/.
[31] Research and Technology Development project (RTD)-granted by the European Commission. Eurogrid- application testbed for european grid computing. http://www.eurogrid.org/.
[32] The globus project. http://www.globus.org/.
[33] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, 1988.
[34] D. Walker, O. F. Rana, M. Li, M. S. Shields, and Y. Huang. The Software Architecture of a Distributed Problem-Solving Environment. *Concurrency: Practice and Experience*, 12(15), December 2000.