Univesita' degli Studi di Milano Bicocca
Facolta' di Scienze Matematiche, Fisiche e Naturali
Dipartimento di Informatica, Sistemistica e Comunicazione
Dottorato di Ricerca in Informatica
XXIII Ciclo

Data Integration for Clinical Genomics

Doctorate Thesis

of

Andrea Calabria

Submitted in total fulfillment of the requirements of the degree of Doctor of Philosophy in Computer Science

Supervisors: **Prof. Luciano Milanesi**
Dr. Andrea Maurino

Tutor: **Prof. Carla Simone**

PhD Program Coordinator: **Prof. Stefania Bandini**

Academic Year 2009-2010

to Lorenzo and Francesca, my life

Abstract

Genetics and Molecular Biology are keys for the understanding the mechanisms of many of the human diseases that have strong harmful effects. The empirical mission of Genetics is to translate these mechanisms into Clinical benefits, thus bridging in-silico findings to patient bed side: approaching this goal means achieving what is commonly referred as *clinical genomics* or *personalized medicine*.

In this process, technologies are assuming an increasing role. With the introduction of new experimental platforms (microarrays, sequencing, etc), today's analyses are much more detailed and can cover a wide spectrum of applications, from gene expression to Copy Number Variants detection. The advantages of technological improvements are usually followed by data management drawbacks due to the explosion of data throughput that reflects on a real need for new systems of data rationalization and management, data access, query and extraction.

Our genetic laboratories partners encountered all those issues: what they need is a tool that allows *data-integration* and supports biological *data analysis* exploiting computational infrastructures on *distributed environment*. From such needs, we defined two main goals:

- *Computer Science* goal: to design and implement a framework that integrates and manages data and genetic analyses;
- *Genetics* and *Molecular Biology* goal (application domains): to solve biological problems through the framework and develop new methods.

Given these requirements and related specifications, we designed an extensible framework based on three inter-connected layers: (1) *Experimental data layer*, that provides data integration of data from high-throughput platforms (also called *horizontal data integration*); (2) *Knowledge data layer*, that provides data integration of knowledge data (also called *vertical integration*); (3) *Computational layer*, that provides access to distributed environments for data analysis, in our cases GRID and Cluster technologies.

Above the three design blocks, single biological problems can be supported and custom user interfaces are implemented. From our partner laboratories, two main relevant biological problems have been addressed:

- *Linkage Analysis*: given a large pedigree in which subjects were genotyped with chips of 1 million of SNPs, the linkage analysis problem presented real computational limits. We designed a heuristic method to overcome computational restrictions and implemented it within our framework, exploiting GRID and Cluster environments. Using our approach, we obtained genetic results, successfully validated by end-users. We also tested performances of the system, reporting compared results.
- *SNP selection and ranking*: given the problem of ranking SNPs based on a-priori information, we developed a novel method for biological data mining on genes' annotations. The method has been implemented as a web tool, *SNP Ranker*, that is under deep validation by our partners laboratories.

The framework here designed and implemented demonstrated that this approach is consistent and can have potential impacts on the scientific community.

Preface

My fascination for Genetics and Medicine has always driven me to deepen the mechanisms of Nature. Research is my highest way to understand the beauty of the Nature and Knowledge. Research also means climbing hard mountains where outcomes are not ensured, and a so challenging travel cannot be undertaken alone: travel-mates are one of the most important keys of success.

In many cases Philosophy highlighted the beauty and need for research, and the sense of knowledge that passes through wonder and amazement, like Plato and Aristotle said.

> (..) ο δὲ ανεξέταστοσς βίος ου βιωτὸς ανθρώπω (..)
> (..) the unexamined life is not worth living (..)
>
> Πλάτων, Απολογία Ζωκράτους, 38 α
> *Plato*, Apology Socrates, 38 a

> μάλα γὰρ φιλοσόφου τούτο τὸ πάθος, τὸ θαυμα΄ζειν: ου γὰρ άλλη αρχὴ φιλοσοφίας ἠ αύτη (..)
> For this feeling of wonder shows that you are a philosopher, since amazement is the only beginning of philosophy (..)
>
> Πλάτων, Θεαίτητος, 155 δ
> *Plato*, Theaetetus, 155 d

> (..) διὰ γὰρ τὸ θαυμάζειν οι άνθτρωποι καὶ νύν καὶ τὸ πρώτον ήρξαντο φιλοσοφείν (..)
> (..) it is through wonder that men now begin and originally began to philosophize (..)
>
> Αριστοτέλης, Μετά τα φυσικά, 1.982 β 12
> *Aristotle*, Metaphysics, 1.982 b 12

Preface

Given my computer science and engineering background, I thought that my support to genetic and clinical research would have been building new solutions in the field of Bioinformatics, both with new methods and algorithms, and with new tools and systems. This is what I am trying to pursuit with my experiences.

My research path started few years ago within the Bioinformatics group of Dr. Luciano Milanesi, my Ph.D. supervisor, at Institute for Biomedical Technologies, National Research Council[1]. He gave my many opportunities and supported me in every step of my professional growth. For this reason, my first thanks are for Luciano and the Bioinformatics group.

Due to the projects of ITB CNR, I collaborated with partner laboratories, both academic ones, such as the Universita' degli Studi di Milano (UniMI)[2], and private ones, like KOS Genetic[3] and Multimedica[4]. From the laboratories of UniMI and KOS Genetic I could put hands on genetic platforms and data, for example Genotyping technologies and Sequencing platforms, understanding the problems that these technologies introduced. None of these experiences would have taken place without Prof. Daniele Cusi, Prof. Fabio Macciardi, Dr. Pietro Conti, Eng. Leopoldo Frati and Dr. Cristina Barlassina, and I am really grateful to all of them. A similar experience has been done for the Multimedica laboratory of Dr. Annibale Puca.

I also thank my supervisors at Universita' degli Studi di Milano Bicocca[5]: Eng. Andrea Maurino, Prof. Carla Simone, Prof. Carlo Batini and Prof. Stefania Bandini.

A special thanks to my colleagues, for their enthusiasm and professional passion to research, always helpful everyone to each other. A circular hug to all of them.

[1]ITB CNR, Via Fratelli Cervi 93, 20090, Segrate (MI), Italy. Web site: itb.cnr.it.

[2]Department of Medicine, Surgery and Dentistry, Via Di Rudini' 8, 20142, Milano, Italy. Web site: unimi.it.

[3]Web site: kosgenetic.com.

[4]Web site: multimedica.it.

[5]DISCo, Department of Informatics, Systems and Communication, Viale Sarca 336, 20126, Milan, Italy. Web site: www.disco.unimib.it.

Given all collaborations with the different groups and institutions, my work has been supported by many projects, both international and national funding, coming from ITB CNR and UniMI. Here I listed some of them:

- BIOINFOGRID (EC FP6-026808), PI Dr. Luciano Milanesi;
- HYPERGENES European Network for Genetic-Epidemiological Studies (EC HEALTH-F4-2007-20150), PI Prof. Daniele Cusi;
- LITBIO Laboratory for Interdisciplinary Technologies in Bioinformatics, Italian MIUR FIRB 2003, PI Dr. Luciano Milanesi;
- BBMRI (EC 212111 1.2.2008-30.04.2010).

Contents

Chapter 1

Introduction

Chapter Summary

The introduction of the Ph.D. thesis presents the starting points of the work, describing which problems I will face within the application domain (Biomedicine). The discussion then focuses on two main goals:

1. a *Computer Science* goal: designing an integrated framework for data integration and management, approaching the final data mining goal;
2. a *Biological* goal: addressing some empirical biological problems through the implemented framework.

A final description of the thesis structure completes the chapter.

As much Nature is beauty, complex and enigmatic, as much epistemology Research greedy feeds every scientist and the amazement for each single finding fortifies this impulse.

Genetics and Molecular Biology are keys for the understanding the mechanisms of many of the human diseases that have strong harmful effects. We have many positive examples that encourage biomedical researches: the Thalassemia disease or the Sickle-cell disease, having the characteristic of heredity across generations, can really damage the life of affected individuals, also leading to death and severe complications. Through the early diagnosis before the disease onset, patients can have regular lives; the prevention is possible by using, for example, genetic tests. This is the final goal of the Epidemiology and the application of Genetics and Molecular Biology.

The studies of this Ph.D. project focus on applications of Computer Science and Engineering methods into Genetics and Molecular Biology, supporting both data and processes engineering and problem solving.

Technologies assume an increasing role in the fields of Genetics and Molecular Biology. With the introduction of new platforms, such as the Next Generation Sequencing (NGS) and microarrays, today's analyses are

much more detailed, and can cover a wide spectrum of applications, from gene expression to Single Nucleotide Polymorphism and Copy Number Variants detection. The advantages of technological improvements are usually followed by data management drawbacks, as it happened in other research fields like Astrobiology with the SETI projects and Physics with the LHC project. In these cases, the increase of analytical capacities corresponds to an explosion of data throughput and it generates a real need for new systems of data rationalization and management, data access, query and extraction. All these requirements weight on both hardware and software designs: we now need *architectures* with strong characteristics of reliability, stability, scalability and security, and *information systems* and *analytical tools* based on high performance environments and versatile to data types and users specifications.

Data analysis is the principal vector in Genetics. So far, many existing methods have been developed to solve genetic problems, from gene discovery to systems biology. Statistics is the most used discipline in such contexts: once a researcher creates a study design and a mathematical model, she/he tests hypotheses on data and analyze results. One of the main obstacles is represented by the analysis of big amount of integrated data so that empirical evidences show the inadequacy of many of the existing computational systems. For this reason, Computer Science methods helped Genetics on overcoming computational problems and Machine Learning [1, 2, 3] methods became very attractive and promising due to their flexibility: these techniques introduced the capability to learn data characteristics (features) from a sub-set of data and then to apply the model to all data. Machine Learning (ML) techniques can highly reduce the need for computational resources and grant a flexibility that is not usual in Statistics. Data Mining [4, 5, 6] is an approach that is very useful for ML and has the characteristics of extracting ("mining") new knowledge from existing data. Data must be engineered: consistent, harmonized, with proper schema and with high data quality level; all these requirements show us the importance of a very well structured data base. Without these data properties, data mining procedures could lead to inconsistent results[1].

What kind of data are used for genetic analyses? The first information is related to the Phenotype combined with patients' data: disease penetrance and inheritance model, clinical records and patient affection status. Genetic data are also important and usually come from biotechnological instrumentations or experiments. Many other external data can enrich and empower a genetic analysis; this is the case of public available information, both related to medical and biological knowledge, for example the disease de-

[1] In the context of databases with low data quality and accuracy, the adjective "inconsistent" is more precise than "wrong" because data mining are misled by data rather than the algorithm itself.

scription with causative markers, and experimental data, for example public SNP genotyping. Another important question arises after recognizing the needed types of data: what is the existing quality level of these data structures? How to access them all? The answer is that a standard for biological data does not exist and all available data are heterogeneous, often fragmented and not synchronized or obsolete, and the data access to multiple resources is very difficult. It is intuitive to understand the problem looking at the database evolution in latest years, as shown in Figure 1.1: the Nucleic Acids Research (NAR) Database Issue journal[2] is collecting all developed databases currently available in Molecular Biology since a decade and the graph shows a growing curve of existing databases that is not associated with a standardization of data schema, information access or data quality. Many of these data sources are very important for Biomedical researchers and this fact points the need for the integration of heterogeneous data: if the goal is mining new knowledge from data, we strongly need consolidated data and an integrated information access, useful for every analytical tool.

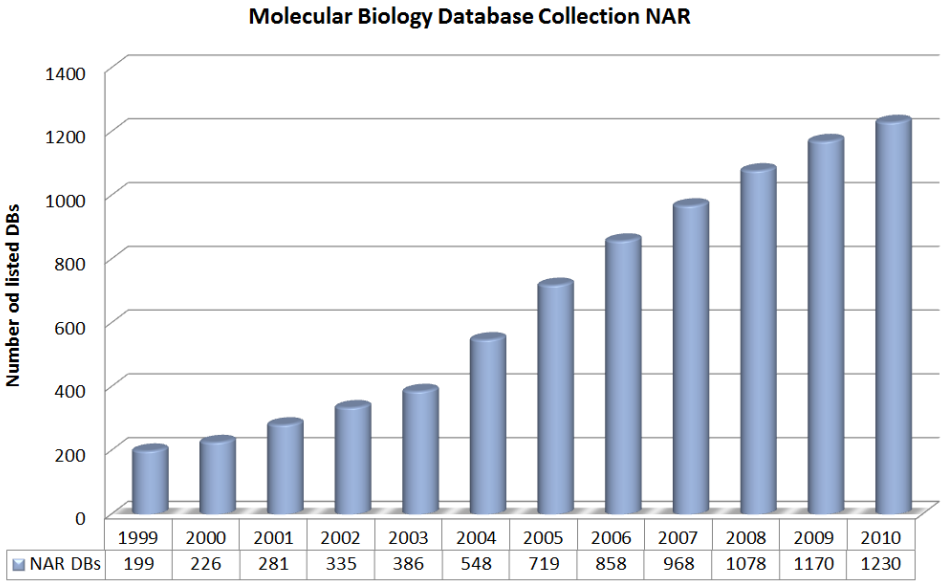


Figure 1.1: Growth in the number of Molecular Biology Databases, based on NAR listing.

All the concepts introduce us into the definition of the Ph.D. thesis goals:

- in the field of *Computer Science* the goal is to design and implement a framework to integrate and manage data and genetic analyses;
- in the application domain, *Genetics* and *Molecular Biology*, the goal

[2]NAR web site: http://www.oxfordjournals.org/our_journals/nar.

is to solve biological problems through the framework, developing new methods.

The title of thesis, "*Data Integration for Clinical Genomics*", directly comes from the combination of the two goals. One of the most relevant observations related to the title is the long term goal, the perspective that can be approached by this work: all our researchers moves to the idea of the Personalized Medicine [7, 8]. Personalized Medicine attempts to individual diagnoses through genetics and personal phenotype occurrence (clinical variations of the complex disease). An example of future real case application can be the following:

- given a lab-on chip technology [9] for a specific disease D to analyze genetic variations, given an information system that can retrieve many different patients' data (genotypes, phenotype and clinical records) and can support data analysis for the disease D,
- physicians can assist a patient on understanding the specific patient disease (both for preventive care and treatments) using both the lab-on-chip array and the subsequent patient data analysis compared with the available data in the information system;
- once the physicians exactly know what causes occur for this patient (often multi-factorial properties), they can administer the right medications and drugs.

This long term objective can be realized only exploiting integrated information systems over high performance infrastructures, covering both computational aspects and architectural ones.

Coming back to the Ph.D. goals, realizing the framework development means first of all understanding what efforts have been done in latest years in this context, analyzing the literature on data integration and what solutions are currently in use in the application domain. An accurate definition and analysis of the emerging requirements has to follow, defining the specific domain needs. Hardware requirements will help on the definition of the computational environment while software requirements will clarify what problems the systems has to address. Translating the requirements into specifications, the subsequent activity is the framework design and components implementation following the "*divide et impera*" approach. A final activity will be testing and validation of the components of the framework. These last steps have to be realized together with end users that have to exploit the framework for solving their biological problems applied to case studies.

The structure of this manuscript follows the same steps above mentioned. The first chapter, number 2, is focused on the state of the art about data

integration and existing systems. The next chapter 3 points the functional requirements and analyze how to satisfy them all, identifying the framework components and the computational environment. The complete framework design and implementation is described in the chapter 4. The next chapter 5 focuses on the system application and validation in which we will describe two new methods implemented on the framework that addressed two biological problems. A final description of future works and the conclusions will follow on the chapters 6 and 7 respectively.

Chapter 2

State of the Art

Chapter Summary

The state of the art focuses both on data integration, the core of our framework, and existing solutions applied to biomedical research. The schema of the presentation thus follows our two goals, previously mentioned in the introduction: first of all the computer science problem, analyzing data integration motivations, architectures and methods, and then the biology scope dealing with how data integration addressed some biological research problems.
The discussion introduces the general methodologies that researchers have pursued to overcome the problem of heterogeneous database integration and the specifications of several database-integration projects in biomedicine. Then a set of existing applications and relative properties will be analyzed.

The rapid expansion of biomedical knowledge combined with the reduction in computing costs and spread of Internet access have created an enormous number of electronic data. This is especially true in biomedicine for the decentralized nature of the scientific community leading to a patchwork of diverse and heterogeneous databases' implementation and making access to and aggregation of data across databases very difficult. Databases are highly heterogeneous with respect to the data models they employ, the data schemas they specify, the query languages they support, and the terminologies they recognize. Heterogeneous database systems (HDBS) attempt to unify disparate databases by providing uniform conceptual schemas that resolve representational heterogeneities, and by providing querying capabilities that aggregate and integrate distributed data. Research in this area has applied a variety of database and knowledge-based techniques, including semantic data modeling, ontology definition, query translation, query optimization, and terminology mapping. We will now review the most important techniques in heterogeneous database integration.

2.1 Heterogeneous database integration

The goal of an heterogeneous database system (HDBS) is to provide database transparency to users and application programmers. This means to provide a global and consistent database interface for applications as if the data were not distributed and all of the database management systems were of the same type. HDBS research emerges since the belief that heterogeneity at the level of constituent database systems will persist exists, despite standardization efforts.

The process of heterogeneous database integration may be defined as "the creation of a single, uniform query interface to data that are collected and stored in multiple, heterogeneous databases". Thus HDBSs are computational models and software implementations that grant heterogeneous database integration [10, 11, 12, 13, 14, 15].
Different kind of heterogeneous database integration methods can be distinguished and are also useful in biomedicine.

- *Vertical Integration.* The aggregation of semantically similar data from multiple heterogeneous sources. For example, a centralized database to functional magnetic resonances collected in a country.

- *Horizontal Integration.* The composition of semantically complementary data from multiple heterogeneous sources. An example can be a system that provides complex queries across genetics and clinical information sources.

- *Integration for application portability.* The standardization of access to semantically similar information at disparate sources. For example, a universal database interface for decision-support applications that allows them to be shared across institutions with no modifications to their implementations

HDBSs distinguish from distributed database systems (DDBS) [16] even if they are frequently confused. The overlapping idea is the capability to provide a unified view and a common interface to data, physically stored in multiple locations. DDBS are more integrated and coordinated than HDBSs: DDBSs implement the same data model and query language and the core system uses the same distributed database management software. Moreover, in DDBSs, data fragmentation is designed to achieve efficiency and autonomy advantages of distributed computing. On the other hand, in HDBSs, the constituent database existed prior to the establishment of the HDBS and their coordination is much more weak.
The characteristics of HDBSs can be summarized as follows [17, 16]:

- *Heterogeneity on data representations.* The core database in an HDBS can use different query languages and data models terminologies to

represent the same real-world object and thus the same semantics. From this consideration follows that even if data are stored at multiple sites and could have identical semantics, the data representation and the data access methods at each site may be different.

- *Local autonomy.* HDBS grants rights to each constituent database on accessing, controlling and manipulating its own data independently from the central engine. Examples of such data manipulation or local database control can be changes of data representation or performance improvements.

- *Bottom-up integration.* A HDBS integrates data that were previously distributed, improving the interoperability aspects. On the other hand, a DDBS exploits the distribution of previously integrated data to obtain efficiency benefits. A bottom-up integration process requires that the HDBS provides interfaces to heterogeneous and preexisting information systems without needing of intensive local preexisting software modifications.

Providing a context to the challenges of heterogeneous database integration, we report a set of requirements [18]:

- Database heterogeneity is a fact and a single model, for example for biological databases, is hard to achieve although the presence of standards.

- Heterogeneous databases systems must provide general query capabilities supported by efficiency procedures. These queries retrieve all data referred to a single object or a set of objects that satisfy the search criteria. Query procedures have not to depend from any particular application of information need.

- HDBS has to allow transparency at data level and thus users and applications are not required to know the existence, access methods, physical location or the schema of the underlying local databases.

- Permissions on local databases, for example writing access, are not required by common users and applications but they are restricted to local administrators.

- Since local database are designed and maintained to meed local needs, the underlying local database schemas can quickly change. Changes are made independently of the integrated database structure.

- As well as schemas, also the content of the local databases could change frequently by updating, deleting or inserting data.

One of the most important problems in heterogeneous database integration deals with *query models*: independently developed and maintained databases are heterogeneous with respect to their model of data storage and information retrieval. A query model must be known to database users or applications when queries are encapsulated into the executable commands. A query model consists of four components:

1. The data representation abstract *model*, such as relational tables, flat files, and so on.

2. The *schema* of the represented data, for example in clinical context a schema of data can differ if starting from a set of patients and retrieving the set of relative physicians a system uses a single query or multiple queries.

3. The *language* for accessing data: syntax and semantics to query and retrieve data can be of high level (SQL for example) or low level (data fragmentation access).

4. The data *format*: it is common the use of abbreviations or codes for names or local formats, for example in the case of clinical phenotypes different systems may use different codes or abbreviations for the same object.

Heterogeneous database integration means also creating a single virtual query model that encapsulates the query models of underlying databases and allows users and programs to access data from the local databases using this virtual model.

Other important issues in heterogeneous database integration are represented by the variety with which similar data are represented in different databases; this multitude of data schemas is called *representational heterogeneity* (RH). The most general type of heterogeneity is that of the data models themselves: aggregating data from relational, hierarchical, object-oriented and flat file databases into a single representation is the first activity in schema integration. However, even if different database systems used the same model, for example a relational model, significant representational heterogeneity would remain such as structural differences, naming differences, semantic differences and content difference. In the following paragraphs these RH will be covered.

2.1.1 Structural Differences

We can enumerate three kind of structural differences: (1) alternative table decompositions (horizontal and vertical), (2) differences in data versus metadata representation, and (3) differences in structured versus free-text encodings.

Alternative table decomposition (TD) can be horizontal or vertical. In horizontal TD the same information is distributed across a varying number of tables. On the contrary, in vertical decomposition different rows are distributed among one or more tables, also partitioned across multiple databases for performance efficiency improvements. This structural differences can arise from different data design modeling during the Entity Relationship conversion in relational tables.

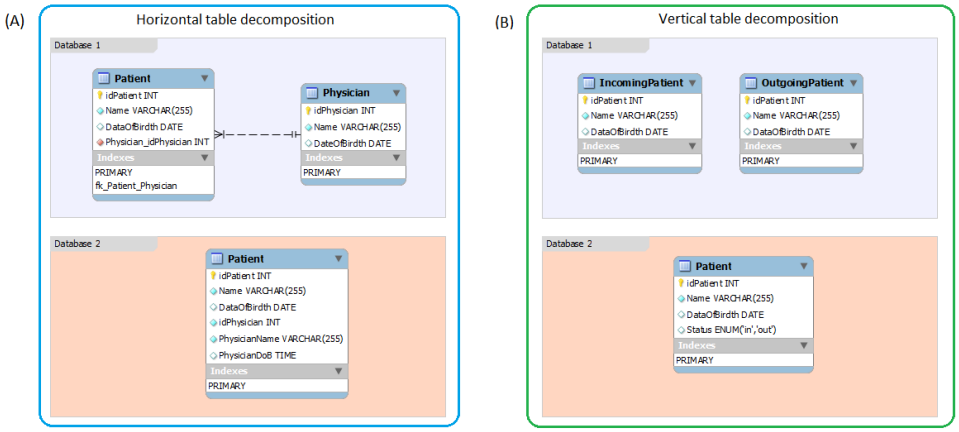


Figure 2.1: Examples of alternative table decompositions.

Examples of horizontal and vertical structural differences are shown in Figure 2.1. In the part (A) of the Figure we present a case of horizontal decomposition with the relation between patients and their physicians in database 1 (DB1) and database 2 (DB2). On the other hand, in the part (B) of the Figure we draw an example of vertical decomposition with a case of incoming and outgoing patients in a medical unit, in DB1 and DB2.

Because the relational model has no constructs for representing type hierarchies directly, such hierarchies may be encoded in a variety of ways in relational databases, and these encodings entail differences in the use of data and metadata. Figure 2.2 shows a simple example of differences between data and metadata representations using a case of two patient measurements of the blood pressure, systolic and diastolic blood sampling, in three different databases.

Structured versus free-text encodings are very common sources of heterogeneity. These differences convey on spreading data across multiple fields or concatenated in a single column. Common examples include the separation or concatenation of values and units, for example {“500 mmHg”} versus {“500”, “mmHg”}, or addresses and names, and so on.

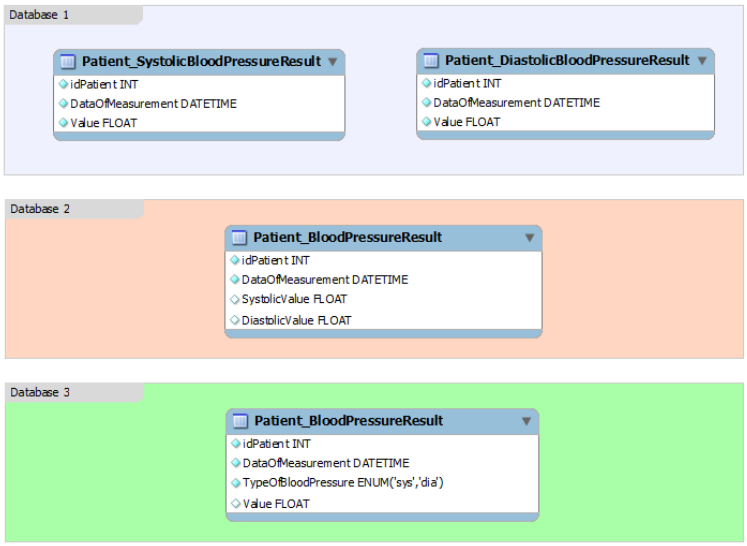


Figure 2.2: Example of structural differences with respect to data and meta-data representations.

2.1.2 Naming Differences

Naming differences occur using distinct lexical terms for the same semantic objects across database schemas. Naming differences may be referred to meta-data differences or to data differences. The first case deals with simple synonymous, for example the two names "Doctor" and "Physician" represent the same object. The second case refers to the similar name applied to different values. For example the "Patient_Code" could refer to an internal unique id of the database or to the country identification code. On the other hand, data-naming differences are often related to nomenclatures or abbreviations, for example "SystolycBooldPressure" or "SBP".

2.1.3 Semantic Differences

Semantic differences occur when the meanings of table names, field names, and data values across local databases are similar but not precisely equivalent. This problem is a big issue when the labels of tables, fields, and data values are identical across databases, but their meanings are different. Semantic differences may occur when there is no one-to-one correspondence among the concepts denoted by values within local databases.

For example, if two clinical studies defined a protocol and for the field "Smoke" of patient life style they collected data in two different ways these data could not be exactly mapped (one-to-many association). The first database could store information represented as "smoker" or "not smoker" while the second database contains data about the average number of cigarettes

per day. A reasonable integration strategy for these values could be keep the more general value set for the global schema and map the other database specific values to this set; in our example we would choose the first database method and then mapping numerical values to it. The drawback of this mapping strategy, from details to general information, is information loss.

When dealing with semantic differences that do not map one-to-many but many-to-many, the "categorical" approach cannot be used.

2.1.4 Content Differences

Content differences occur when data represented in one local database are not directly represented in another. The data may be implicit, derivable, or simply missing. Implicit data are usually constant, and therefore assumed within the environment of a local database, but cannot be assumed in the context of the global database. An example of derivable data is the representation of postal code versus countries or date-of-birth versus age. Each data may be derived from the other. The problem of missing data occurs when the global schema contains an information type that is simply not available in one or more local databases.

2.2 Strategies of Heterogeneous Db Integration

We summarize here some of the most important strategies to provide uniform interfaces (query models) for heterogeneous databases.

2.2.1 Data translation and query translation

One of the most relevant distinction among heterogeneous databases systems copes with the adoption of a *data translation* or *query translation* strategy.

Data translation (DT) involves the transformation of data from the various native formats in which they are collected and stored to a common shared format in which they can be uniformly accessed. The shared format directly implements all the elements of a query model; this allows users and applications to avoid the specifications of the constituent query models. Examples of such strategy are datawarehouses. Data translation can be automatic or manual [19].

Automated data translation is the widest approach since it translates data from the original stored formats in a common one. In the translation process, a gateway or a common interchange format can be adopted. An example of common format is HL7[1] standard. Even if automated data translation reduces the cost and delay of translating data, two problems affects this strategy. First, data translation duplicates the storage of data,

[1]Web site: http://www.hl7.org

in the original source and in the integrated database, with a possible data integrity damage. Second, the correspondence between source data and translated data is encoded in a procedural algorithm, difficult to inspect, validate and maintain. Manual data translation is directly curated by personnel. The task are the manual conversion of each data into a different format or standard; it is easy to understand that this approach is a high costly solution and it is not scalable.

The alternate strategy for providing a virtual query model is the translation of query instead of data. Query translation approaches (QT) deal with translation of queries formulated in a global data-manipulation language to equivalent queries in the specific language of the constituent databases. Data are stored only in the constituent heterogeneous databases and when a query is issued against the virtual model, the engine for query translation and execution decomposes and translates the query into an equivalent set of local queries. Local results are then transmitted, transformed and combined to be presented to users (or to application). Using this approach, we could reduce performances by adding overhead in the transformation and the remote execution of the queries. Given that many applications require timely access to real-time data and many heterogeneous databases cannot be forced to export and share their data, this strategy is valuable for a wide range of case studies.

Many software components exist for performing translation processes, from wrappers to mediators [20, 21, 22, 23, 24]. All query translation techniques can be categorized in two main classes: QT based on procedural mappings or on declarative mapping.

Query translation based on procedural mapping

The mappings between conceptual database schema and various underlying databases is performed through procedural functions that physically import objects from local databases into corresponding objects in the global environment. During this activity, mapped objects can be manipulated to improve query performances or access.

The advantages of this approach resides on simplicity of integrating legacy databases. Disadvantages are related to query optimization: the database engine cannot perform optimization procedures since mappings between query models are specified as procedural functions and cannot be decomposed or recombined for efficiency [25]. Another disadvantage is related to maintenance: on local database updates, all mappings must be retested and validated.

Query translation based on declarative mapping

This approach attempts to address the two main problems of the procedural mapping method earlier mentioned. A declarative representation of query model mapping specifies the correspondence between objects and operations at the level of global query model and objects and operations of the constituent query models.

The representation of correspondence is encoded such that a software process may inspect it, and it is stored independently of the software code that performs query translations. A query optimizer can inspect and handle the declarative mappings by understanding the mappings' semantics of local databases. For example, an optimizer can understand that a request for two types of objects at the global level can be processed by a single query at local level because they are stored in the same local database.

The query translation based on declarative mappings is less common than QT based on procedural mappings. The Object Protocol Model multidatabase query system [26] uses this approach storing mappings in a metadata file to translate queries. Other systems exploit description logics and query translation is performed via rule-based interfaces over mappings [27].

2.2.2 Global data model and query languages

Database interoperability requires to use a common data model sufficiently simple and abstract to represent the contents of various heterogeneous data models and a corresponding query language that can be used to formulate queries at an equally abstract level. Semantic data models grant abstract conceptualizations of domain data that can serve as common modeling for different implemented databases.

Usually Simple data models (SDM) are used for database design since they provide high abstractions for specifications of database schema. SDMs that represent atomic facts, such entities and relationships among entities, are easier to translate to more complex data models implemented by constituent database systems. One of the most common example of SDM is the Entity-Relationship (ER) [28] model.

The issue of global schema that encapsulate heterogeneous system implementations has been addressed by database researchers using knowledge base approaches. Ontologies are considered synonymous of global conceptualizations and define the object classes, relationships, functions and constants for the application domain. Ontologies are similar to query models since both include the following components: (1) a formal abstract model for representing properties of objects in a domain, (2) a definition of the object classes and of the relations and functions of the members of those classes, (3) a specification of the object constants. If an ontology also includes a query language, it is indeed equivalent to a query model. For deeper

considerations on ontologies and query model usage, refer to [29, 30, 31].

2.2.3 Summarizing integration aspects

The Table 2.1 summarizes integration techniques.

Approch	Methods
Instantiation	Materialized and Virtual integration
Global view	Global As View (GAV), Local As View (LAV) and Both As View (BAV)
Global Model	Relational-based, Tree-based, Graph-based
Query Model	Ad-Hoc, SQL, XPath, XQuery, SPARQL, etc.
Semantics	Dictionaries, Thesauri or Domain Ontologies

Table 2.1: Integration approaches and methods [12].

Instantiation refers to physical data location. In a virtual federation, data reside in the local data sources and the HDBS gives a unified view of them. On the other hand in a materialized federation, data are collected from the data sources, and after a process of cleansing these data are integrated and stored in a unique repository that physically exists. Although the materialized approach is computationally more efficient, in general the virtual approach is preferred for different reasons: it does not imply data redundancy, it is more flexible for further data sources updating and insertion, and it is easier to maintain [32].

In the Global View approach, Global As View (GAV) refers to creating a global model by merging local source schemas; this process is achieved unifying entities at schema level or instance level. On the contrary, Local As View (LAV) suggests that the global schema has been developed independently from local databases. For this reason, local data are adapted to the global model giving a homogeneous data representation. Both As View (BAV) is a hybrid approach that combines both GAV and LAV aspects: a loosely defined global schema is mapped to the set of reconciled local schemas (e.g. [33]). Figure 2.3 shows the GAV, LAV and BAV idea for the integrated global view development.

Global view approaches need the generation of mappings between local sources and the global view. In the literature, many methods for automating the *schema matching* have been proposed [34, 35]. The principal aim of a schema matcher is finding possible mappings between elements of two schemas. Mappings can be one-to-one, this is the usual configuration, and one-to-many. Schema matching has been developed for relational schemas but also XML and OWL formats can exploit such method and can be adopted for all the three approaches: LAV, GAV and BAV.

There are other strategies for data integration, such as direct interlinking between database records and cross-database indexing.

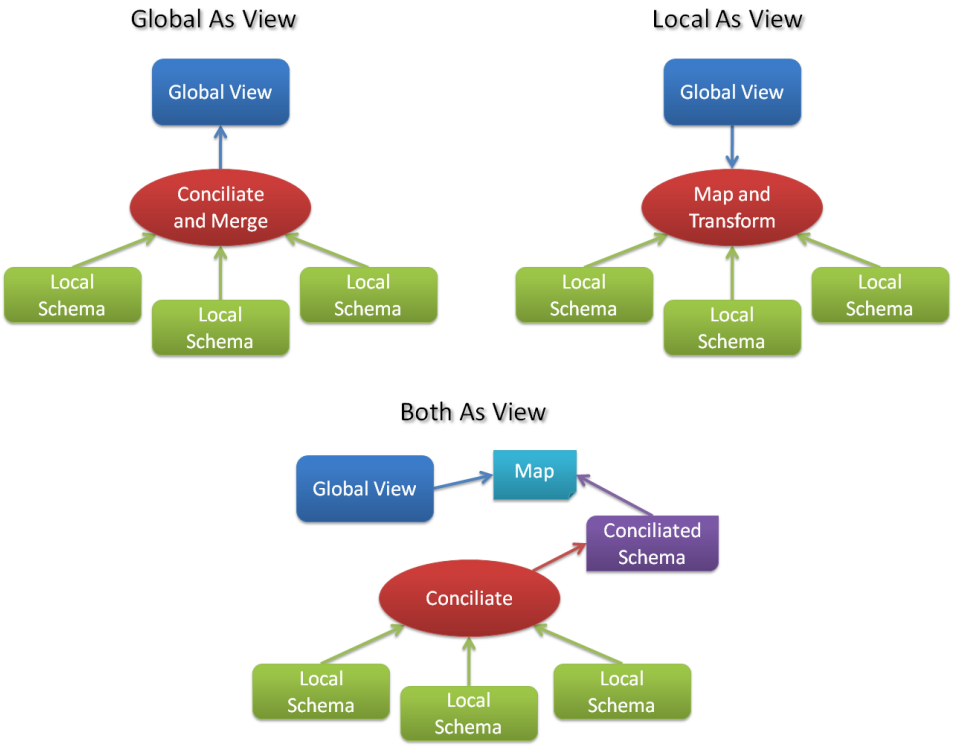


Figure 2.3: Global View approaches: GAV, LAV and BAV [12].

2.3 Integration Architectures

In this section we focus on current integration architectures, describing main characteristics, as pointed by the Table 2.2 and Figure 2.4.

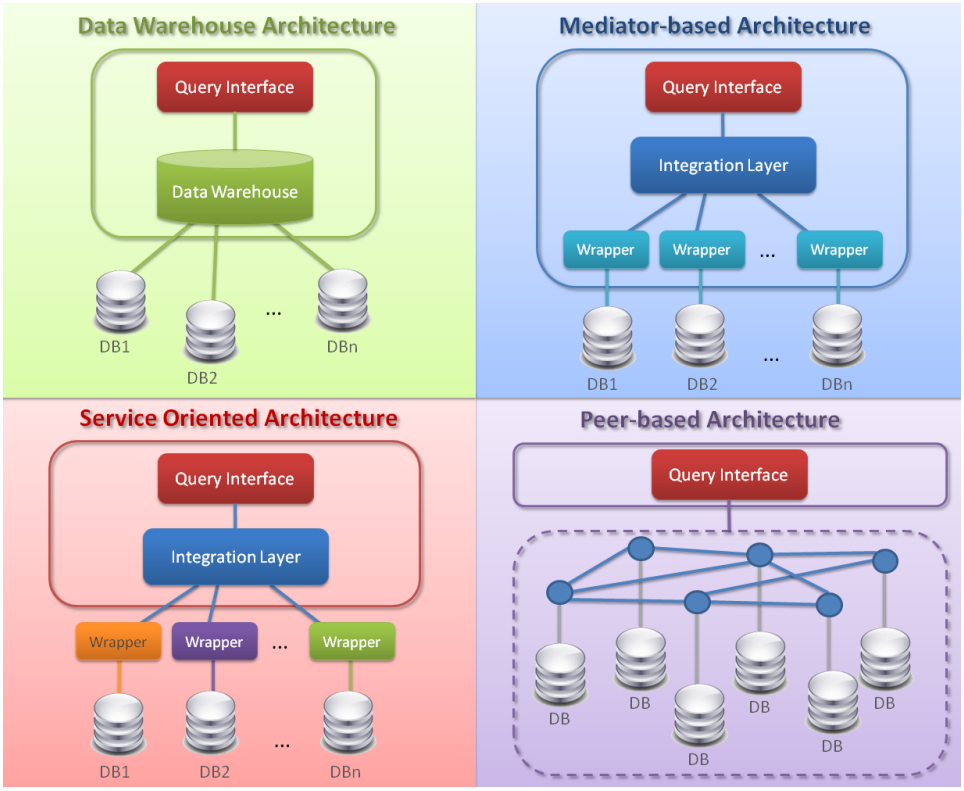


Figure 2.4: Data integration architectures

A data warehouse aggregates and integrates data of several different DBMSs into a single repository. A global database schema is developed to encompass the schemas of the local data sources. Usually an integrated database schema is developed from scratch and the updating occurs with low frequency and the reason is often a change of user requirements. Systems that rely on the data warehouse architecture are usually restricted to consider a few source databases, but can achieve a higher degree of integration of the data sources. The limitation of warehouse system is mainly due to the intrinsic rigidity of database schema and thus for each new local database could be needed to change the schema of the data warehouse. These are the reasons since data warehouse systems allow to obtain an high degree of instantiation.

In mediator-based systems [36] local data sources maintain their independence. Data integration is achieved by defining a global view, or integrated schema, which is shared by all sources. A middle-ware component,

Architecture	Advantages	Disadvantages
Data-Warehouse	Fast queries. Clean dat.a Control over query optimization.	Stale data. Complex schema. Redundant data.
Mediator-based	Current data. Flexible architecture. No duplication of data. Data autonomy.	Slower queries. Complex schema. Little or no data cleansing. Temporary unavailable services. cause incomplete data fetching.
Service oriented	Current data. Flexible architecture. Schema tailored to users. No duplicate data.	Requires source availability. Little or no data cleansing.
Peer-based	Current data. No global schema. Flexible and independent.	Slower queries. Experimental. Little or no data cleansing. Redundant data.

Table 2.2: Integration architectures.

the "mediator", translates queries from the global view to the local data. Typically, each local source needs a "wrapper" component on itself which exports a view of the local data in a common format for mediation. Query processing is achieved by translating and forwarding sub-queries to specific local sources and then combining all local results. As main advantages, the returned data are always up-to-date, since queries are performed dynamically, and data are not replicated since they reside in their source database. Adding new sources of information is not a hard task. On the contrary, mediator-based systems need to manually specify the mappings between local and global schemas.

Service-Oriented Architecture (SOA) provides a standard method to integrate data sources and grant programmatic access to information, and allows service interoperability [37]. Client applications can combine a set of services to accomplish a specific tasks, creating their own workflows and pipelines. In order to use web services, applications and users need to know the XML schema of input and output parameters, expressed with the Web Service Description Language (WSDL).

The three types of architectures early mentioned rely on the definition of a global schema to whom all local databases agree. Current efforts are devoted to the definition of peer networks where data can be locally organized and managed[38, 39, 40]. Each peer or group of peers shares the same database schema and local mapping among pairs of schemas can be

established; this mapping can be considered a semantic network. When a new peer wishes to join the semantic network, it simply needs to establish a mapping with a single peer or a group of peers in the network. When a query is submitted to one of the network peers, the query is routed to the peers that can contain possible answers. Pre-defined mappings among schemas allow translating a query to be executed in local schema and to obtain more precise results or allow translating the results making them homogeneous and comparable. The peer, that initially received the query, is in charge of collecting the answers and returning them to the requesting user or application. Peers can develop schemas that best fit main users needs and then establish a mapping with a small fraction of other peers. A peer can easily join and leave the network. Drawbacks of this architecture are the need for developing mappings and using them on the fly to evaluate queries; this characteristics can highly affect the performances of the retrieval process and thus current studies are trying to address this issue [41, 42].

With the introduction of the Web 2.0 idea [43], a new architecture is emerging: *Mashup* [44, 37]. This architecture is not exactly tailored for data integration but for aggregation. A mashup is transient; it lasts for as long as it is needed, and its interpretation is dictated by the mashup application rather than a model. Mashups allows getting data from more than one Web-based resource, also from different providers different, to make a new Web application. Mashup exploits RESTful APIs or RSS-based syndication feeds to fetch data and results and mashes them to provide new ways of combining and presenting information. The innovative idea is the role of the user in creating a specific, light touch, on-demand integration, following the mantra of "just in time, just enough" design [22]. An example of free mashup tool is Yahoo! Pipe[2].

2.4 Integration systems applied to Biology

In this section, we will just understand which heterogeneous database systems already exist applied to Biology and Biomedicine, our application domain. The description starts with a general and lite definition of the main biomedicine research fields, pointing what types of data do they use and thus what are their real needs for data integration, and finally understanding what integration architecture can they exploit. Once we defined the biomedicine research categories, we will be able to classify some of the main data integration systems and map them within the integration architecture space.

The main biomedicine research categories are six [14]: human genetics, genomics and clinical practice, microarrays, pharmacogenomics, rational drug design and biobanks.

[2]Yahoo! Pipe web site: http://pipes.yahoo.com

Human Genetics studies diseases caused by genes and often involve family-based or population-based studies, looking for inheritance paths of genes and diseases' traits. During the gene discovery process, geneticists need to use a wide range of data and gene annotation information (knowledge about genes and their biological components) that is often stored in different data sources, one for each data type. A common task is the comparison of multiple genes per time, requiring different queries to different databases. The queries posed by researchers can be highly complex and require a "join" of multiple databases [45]. Genetic data have also to be combined with clinical records or phenotypical data, granting better results and focusing analyses on relevant genes to specific pathologies.

Due to the introduction and development of molecular medicine, in next years diagnosis and treatment of diseases will be based on knowledge of the underlying molecular defects rather than evaluation of the symptoms [46]. Epidemiology aims to use molecular information on understanding predictive genes characteristics to assess susceptibility of a population to a disease. Physicians require access all patient information, from medical patient history and phenotype data to genetic data; then all patient data need to be contextualized and combined with public data. This approach will really read to the so called personalized medicine [47].

Microarrays are a recent technology and are used to measure multiple data types, from tissue specific chip to gene expression and nucleotide polymorphisms. Biological objects in microarrays are represented as "spots" on the chip and scanned through lasers. In order to make sense of the experiment (semantics), external annotation is needed for each biological object. The external annotation is usually imported from different public domain databases and researchers have to reconstruct the relationships between biological elements and their connected objects; for example paths between the datasets of nucleotide, relative genes and derived proteins. The integration of data and a simple data access are important needs while dealing with microarrays studies.

Pharmacogenomics is aimed no understanding how individual genetics plays a role in variation to drug treatment or how systems of genes are involved in modulating drug response [48]. The same considerations are valid for pharmacogenetics. The discovery process is based on a genotype-to-phenotype approach, starting from the identification of suspected genes or system/set of genes and relative variations, then searching for phenotypes associated with variation, and finally confirming the results' clinical relevance. Research in pharmacogenomics involves integration of heterogeneous types of data with high quality including genetic, genomic, phenotypic, and clinical.

In drug discovery, the pharmaceutical industry needs to make informed decisions about proceeding with the costly development of a drug [49, 50] due to the expensiveness of the overall process. The in-silico drug discovery

discovery process can address the problem, using high performance infrastructures. Rational drug design involves integration of diverse, heterogeneous data types which can include legacy and private data. The main goal of rational drug design is molecular modeling of disease, prediction of specific compounds which interact with identified proteins, identification of proper patient population through disease sub-classification, and predicting absorption, distribution, metabolism, and excretion of a drug.

Biobanks are mainly large databases which integrate both medical data, clinical records, patient life style information, demographics data (genealogy, family history data, etc) and genotypic data, with prov icy policies. Biobanks also aim to incorporate bio-repositories for physical sample storage. The role of Biobanks will be central in understanding complex diseases where many different factors interact, such as environment or life style in addition to genetics. What the scientific community is facing with is the unification of the multiple existing Biobanks, for example CaBIG[3] or BBMRI[4] projects.

If we combine the integration architectures (Section 2.3) and the biomedical research fields with relative requirements, we obtain the Table 2.3.

Table 2.3: The areas in genomic medicine mapped on the data integration architectures based on their specific requirements.

	Biomedicine Research Areas					
Int. Archs	Human Genetics	Genomics Clinical Pr.	Microarr.	Pharm. Genomics	Drug Design	Biobanks
Warehouse	√					√
Mediator			√	√	√	
SOA	√	√		√	√	√
Peer-based	√	√				

We can now map the biomedicine research fields in the "data integration architecture space", where on the x-axis we have the expressiveness, the knowledge representation, divided into four levels: relational, semi-structured, mediated schema and ontologies; in the y-axis we pointed the data storage characteristic, from federated data to warehoused data. Figure 2.5 shows the mapping between biomedicine areas and data integration architectures.

Now that we mapped the biomedicine areas to data integration architectures, we can identify what existing solutions exist as good examples for each area. Figure 2.6 shows some existing solutions for the different areas.

UCSC Genome Browser [51, 52] and EBI EnsEMBL [53] are some of the most important data warehouses that implement their databases with the

[3]Web site of CaBIG project: https://cabig.nci.nih.gov.
[4]Web site of BBMRI project: http://www.bbmri.eu.

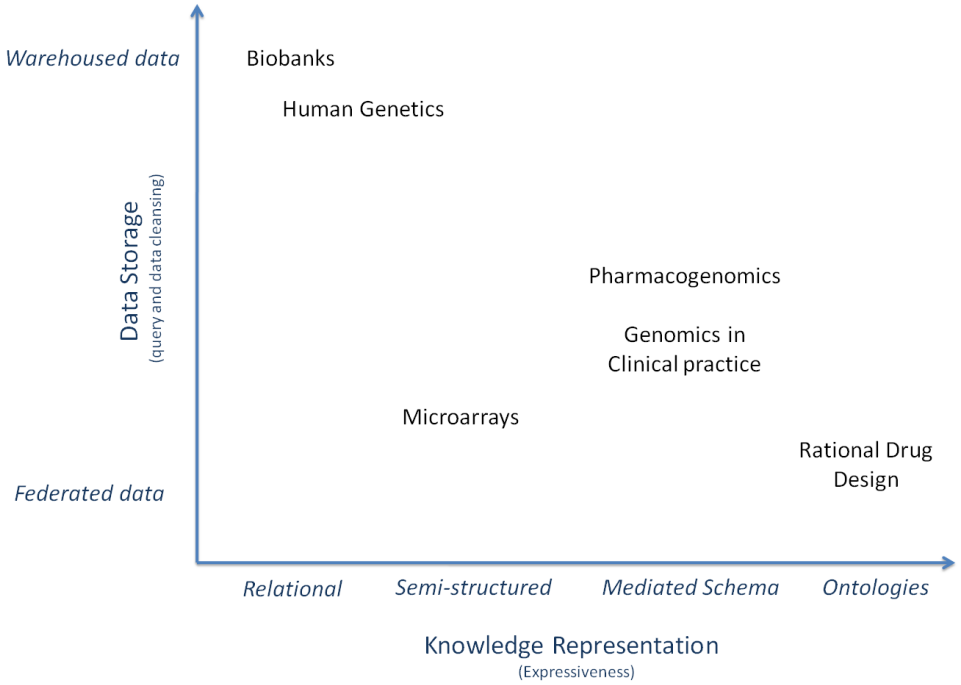


Figure 2.5: Mapping between biomedicine areas and data integration architectures [14].

relational approach. Both databases are open access and free for download and academic use, and they have data mart tools to access data and visualize them in a genome browser. PharmGKB [48] is a private datawarehouse with high quality level, implemented with XML representation.

Important federated data engine are for example Entrez cross-database search[5], one of the most used tool, and BioKleisli, a pioneer in applying data integration approaches to biological data [54].

TAMBIS [55] is a commercial tool that integrates data using mediators and grants transparent access to multiple biological resources; a parallel tool for data integration about genes is GeneCards[6]. Biomediator [56] has an XML representation and exploits a federated approach.

Piazza is a valid example of peer data management system [57], with a native high federation level.

There are many ontological resources in biomedicine research; here we mentioned some integrating applications such as GeneX[7] and Bio2RDF[8]. Both of them exploit RDF approach to map multiple data sources and inte-

[5]Entrez web site: http://www.ncbi.nlm.nih.gov/Entrez/.
[6]GeneCards web site: http://bioinfo.weizmann.ac.il/cards/index.shtml.
[7]GeneX web site: http://genex.sourceforge.net/.
[8]Bio2RDF web site: http://sourceforge.net/projects/bio2rdf/.

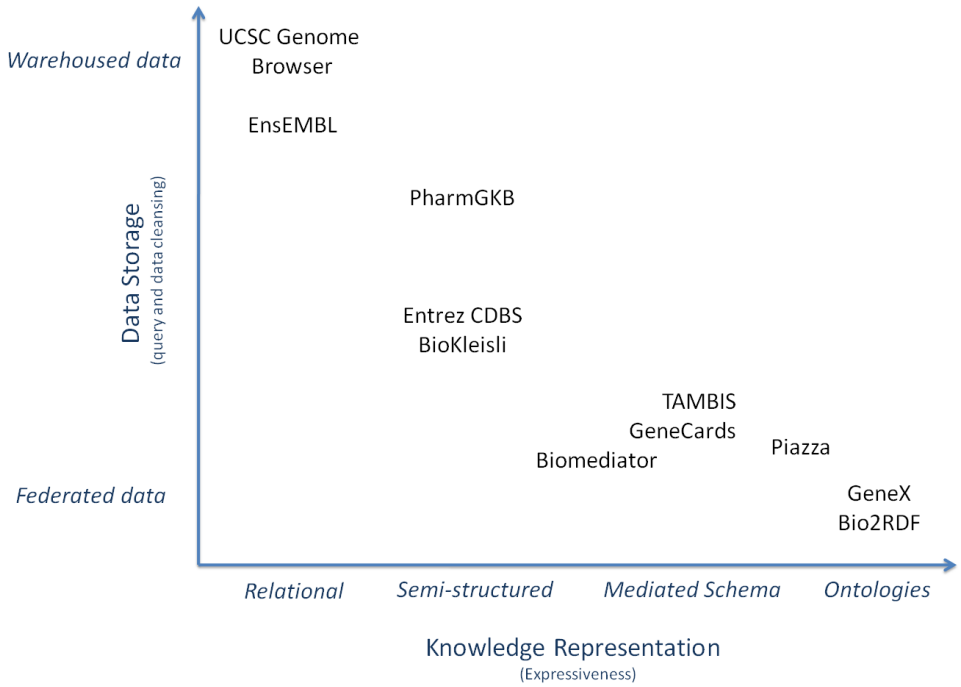


Figure 2.6: Existing solutions for each area, reflecting the mapping biomedicine fields - data integration architectures [14].

grate data. We also note that a standard definition from W3C is in progress: BioRDF[9].

[9]BioRDF web site: http://esw.w3.org/HCLSIG_BioRDF_Subgroup.

Chapter 3

Analysis of requirements

Chapter Summary

Given the starting motivations, this chapter discusses how we designed a generic methodology to address the visual mining problem on data integration. This approach defines a method that can be decoupled into three design blocks: "experimental data", "knowledge data" and "computational layer". Each of these blocks will be deeper described in this section with the perspective of being system's requirements.
A discussion on the infrastructures on which we developed the system will be presented. In this context, requirements needed a distributed environment with strong properties and for this reason we chose the Grid environment and the Cluster solution. Finally, we will provide reasons for which Grid is a suitable solution and how the three design blocks fit Grid specifications.

Achieving personalized medicine for health care through integrated support of genetics and clinical practice implies dealing with a structured huge amount of data that must be consistent and reliable. Above data is also crucial to provide to users analytical methods for empirical data analysis. What genetic laboratories need is a tool that allows *data-integration* and supports biological *data analysis* exploiting computational infrastructures on *distributed environment*.

Translating this general sentence in general requirements, we can define three macro-categories:

1. *data integration*: aspects on data (*what* to analyze), how to unify and integrate all needed data;
2. *infrastructure*: *where* to run programs and methods;
3. *analytical methods*: *how* to analyze data.

Once we defined the three macro-categories, next step will be a refinement of each category of requirements: a fine granularity of requirements will allow us to design the right tool with the right features, fitting objectives and scopes of the end users, biomedical researchers. Following the schema of the three categories, we will discuss in deeper details all requirements and their analysis will lead us to design our framework.

3.1 Data Integration

Many times we refer to biological data as "huge" amount of data. The size of the data is not the only one problem: biological data are often heterogeneous and collected with different quality ratio. Dealing with biological data means coping with different *data types*, *data structures* (with *data quality* problems) and *data size*.

For what concerns *data types*, we identified three main categories:

- *Experimental data.* Biological experiments exploit specific protocols to produce quantitative results, such as a sample sequence. In many laboratories, technicians use biotechnological platforms to obtain final results; this is the case of chip technologies for example Single Nucleotide Polymorphism (SNP [58]) analysis by arrays, or gene expression experiments. In this category we include genotype data, sequencing data, and so on.

- *Biological knowledge and annotations.* Biomedical research produced a set of annotations for known biological elements, for example coding genes, proteins and transcripts. Annotations represent what we actually know about these elements: a gene, for example, is defined by a set of transcripts and has got coding and non-coding regions depending on the specific proteins that the gene coded for. Annotations of biological elements are computed through custom pipelines by main research centers (i.e. NCBI, EBI; for deeper details of an example annotation pipeline see [59]). Together with annotations, we have also functional information or other meta-data; for example, a gene can be responsible for a single disease (monogenic or mendelian disease with known disease causing gene) and the same gene is involved in a biochemical pathway related to a specific tissue and it also connected in a gene regulatory network. All these information came from experiments and once they have been confirmed and validated, results became "biological knowledge".

- *Clinical phenotypes.* Phenotype is a week definition in literature because it can include a wide range of data; anyway phenotypes are always related to patients. When clinicians deal with a disease or a

disorder, they can define a patient's "phenotype" as a brief representation, affected or not-affected, or as a detailed set of variables, collecting life style data, clinical parameters, health history, and so on. Moreover, phenotypes can be also combined with clinical investigations, such as functional magnetic resonances (FMRI), radiographies, and so forth.

In latest years, a new important category of data type is emerging: *Results of Analyses*. So far, many experiments have been conduced and we now have many data. One of the recent attempt is to reuse such data by standardizing information and meta-data. Meta-data here are very important since they represent outputs and results of the past analyses. In this direction goes dbVar[1] (Database of Genomic Variants), and dbGaP [60]. We did not include in the main categories this set of data because literature did not achieve a standard definition for this data type. We also believe that together with results of analyses (general outputs and data sets), authors should provide description of methods, significance values, and other useful information: only improving these meta-data, the "category" results of analyses will be really used and useful. This is the reason why we are studying how to satisfy this emerging need and include it in the "knowledge" standardization (in the final chapter 6 we discuss what we are realizing now and our perspective, including the definition of "results of analyses").

For what concerns *data structures*, we faced all kind of data representations while analyzing the three data types above mentioned, from unstructured data to semi-structured and structured data; we earlier discussed about data structures in the Chapter 2. Example of unstructured data are scientific articles for which text mining techniques can extract data with a degree of reliability (given by accuracy, precision and recall parameters). On the other hand, a large set of data for the three data types has a structured representation. For example, annotation data from NCBI and EBI are collected in relational databases, fully and freely accessible. Also biological knowledge has a structured representation through the use of ontologies, for example Gene Ontology [61].

Data structures directly affect how to access data. This is an important consideration for our requirements because we will have first to select the most reliable data sources for each of the three categories and then we will have to study custom procedures for data access and extraction. The definition of "reliable" is joint to many aspects, from data quality (data curation, data representation, updating, and so on) to data availability and data access. Moreover, reliable means with no conflicts on data. Conflicts on data occur when different data sources are not in agreement on the same data; in this case we will have to decide how to solve the conflict. Our framework

[1] Web site of dbvar: www.ncbi.nlm.nih.gov/dbvar/.

has to define the data sources and to consider these aspects of data quality inspection.

The *data size* problem is directly related to efficiency, speed-up, computational cost in time and space, and infrastructure set up, and can highly affect the system flexibility, scalability, availability and robustness. Data size varies for each data type; the experimental data are usually larger than other data types. Actual technologies can produce data with size of Tera-Bytes, for example next generation sequencing platforms (NGS). Even if the overall experiment size is not so huge, we have to manage data of millions of markers (for example SNPs) for each subject in a project, where in a project we could have thousands of subjects, and for each marker we have many values to store. It is thus easy understand that the system must be designed on a strong computational infrastructure with servers having high specifications. Moreover, mechanisms for speed up queries and fast data access must be evaluated, such as binary data representation, materialized views and caching methods.

Once all data types and relative characteristics have been considered, we have to deal with data integration problems, the core of the system. The requirements for data integration are trivial: collect the most important data sets for each of the three categories of data types (so far the "Phenotypes" data type is considered in the simple format for the data available). What we have to do now is translate this goal in specifications and for this purpose we engineered data in a data mining point of view. In the data mining discipline, data marts are sub-sets of structured data that fill all needed data space for a goal: this means that for the specific queries and mining procedures that satisfy a specific goal, a pre-defined sub-set of data can be extracted and represented as a multi-dimensional mart. The data representation is viewed in a multi-dimensional space in which each table is a dimension of the "mining space". We exploited the same basic concept of multi-dimensional data representation in our context where each dimension is related to one of the data types. Once we defined our mining space in the biological context, we can use data mining procedures to extract new knowledge from data, both supported by statistical tools and machine learning algorithms. The idea of the mining space is explained in Figure 3.1.

Figure 3.1 shows the three main categories on the three dimensions (solid lines), while the next category of the "results of analyses" will be added in future developments (dashed line).

3.2 Infrastructures

In the Data Integration section, 3.1, we briefly mentioned some of the features of the systems describing the data size problem. All those features are requirements for both the system and the infrastructure. We now will

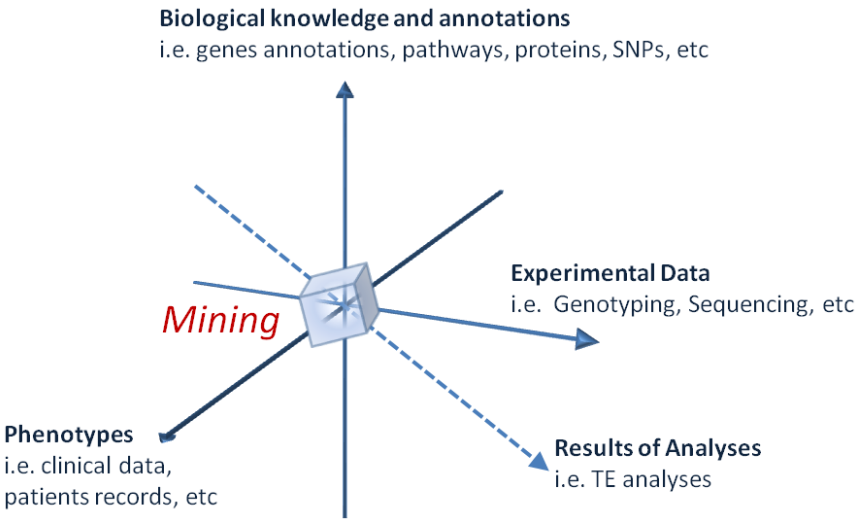


Figure 3.1: A general representation of the *Mining Space*.

deeper point main features that will led us to choose which infrastructure best satisfies our requirements.

We identified three areas of requirements:

1. system's requirements;
2. infrastructure's requirements;
3. data management requirements.

For what concerns the system, the main features are *efficiency* and *accessibility*. Efficiency deals with computational costs in time (the space one is mostly related to infrastructure requirements even this boundary is lazy): end users of our tool should wait for results in reasonable time, depending on the specific task. In Bioinformatics we often exploit very computational intensive applications and most of them are massively parallelized. For this reason, when we will design applications to run within the system we have to implement parallel version of the methods or to port a single thread program in a distributed one. Similar considerations must be taken while dealing with relational database engines and database structures, designing an efficient database with custom solutions to speed up performances on data queries and data handling. Examples of such approaches are binary data representations, data indexing, materialized views on data, caching of most frequent queries, and so on. Accessibility is related to the way of access and use the tool. We have to design simple methods for accessing system's contents over Internet for example using user friendly web interfaces or providing standard application protocol interfaces (API), for example with web services ports.

The framework has to stand on a strong infrastructure with these main properties: *scalability*, *flexibility*, *robustness* and *reliability*. We already pointed that we need high computational resources in time; here we also add requirements for costs in terms of space. Scalability means easy way of extending middle-ware as much as the system grows and needs more resources. This feature is important, for example, to grant space to new data integration or to add computing elements to support data analysis. Flexibility is a requirement between the system and the infrastructure: as system needs to add new features (in our idea in a plug-in perspective), as the infrastructure has to modify and adapt the structure of the middle-ware on changes without re-designing the overall platform. Robustness and reliability concerns features for adaptive mechanisms on critical events and fault tolerance and for load balancing, with performance speed-up. An example of methods to achieve such requirements is the use of redundant middle-ware, resources monitoring systems, hot swapping procedures for servers and disks, load balancing with brokers as middle-ware, and so on. Moreover, we need to cope with low cost infrastructure solutions. This requirement lead us to fit two other features: *resource sharing* and *reuse*, preventing also wasting of energy with benefits to the environment (electrical power saving, reduction of CO_2 emissions, etc).

Resource sharing and reuse well also fit the last type of requirements: data management requirements. Sharing resources is one of the steps for creating a research network. A research network is a key point in biological research since only a common effort can really produce effective scientific results. In a research network each partner can share data, resources and knowledge, without loosing the ownership of them nor the intellectual property of possible results. Participating on a research network means having both rights and dues. Rights are mostly related on scientific results and ownership of data and resources, but also dues are related on the responsibility on shared data and resources. For example, each research center can decide to share a sub-set of data but then the data management belongs to the data owner that decides data policies and is also responsible for data quality and maintenance. Similar considerations in case of resource sharing: if a partner shares a server as computational middle-ware, it is also responsible for the server status. On the other hand, other partners that use other data must follow data owner policies, and, if they run data analyses on other servers, they must be responsible for their actions. Traceability is an important requirement to fit the last need; also security must be carefully considered as requirement.

Pointed all requirements, we found two main infrastructure solutions that best address all problems:

1. Grid technology;

2. Cluster environment.

The Grid solution satisfies all requirements, from data management and privacy issues to resource sharing and computational need. In the Grid paradigm, different partners share resources and use a common middle-were and architecture to interact and communicate. Sub-sets of partners form Virtual Organizations (VO), a network of people that usually exploit common applications or data, for example the Bioinformatics applied to medical research. All Grid users are identified by their own Certification Authority, usually the same partner research center; with this requirement, all users have a private certificate. The user certificate is the key for accessing Grid and the shared resources: in this way all users are authenticated, authorized and can be monitored. The private certificate takes trace of the user's activities and thus induces responsibility using data and server's usage. On the other hand, each research center maintains and is responsible for its own shared data and hardware components. The Grid infrastructure is based on many different middle-ware components that together natively grant all our infrastructure's requirements, such as scalability, robustness, redundancy, fault tolerance, and so on.

Beside the Grid technology, Cluster solution is also appropriate. Similar conditions can be applied in a cluster environment because usually a very large cluster is used by a set of groups or research centers. Each user is identified by the cluster administrator and gets a set of privileges for some data (if available) or computational resources, both space and time. Since users share the same resource, they have to take care or their actions and can be tracked by system administrators.

3.3 Analytical methods

So far, we focused only on the properties of the system to grant the best efficiency, performances, and all other infrastructure's requirements. What we still need is understand which analytical methods we need to provide to users and how can they solve their problems.

Biomedical research studied a wide range of problems related to medicine and health care and Bioinformatics got empirical solutions to problems combining computer science algorithms and engineering methods. Since many software were implemented in Bioinformatics, we could not think to integrate at the starting point all software in our tool. We decided to release on demand new software import. The simple idea can be summarized as follows: we define a general framework that is the most flexible as possible and when users have to solve a problem we create a custom plug-in to the system and we integrate it to the framework. With the plug-in approach, our system has the maximum flexibility and scalability.

3.4 A unified approach

In the introduction, Chapter, 1, we identified two main scopes related to distinct goals: a biological scope and a computer science one. The three macro-categories that we mentioned at the beginning of this chapter can be linked to the two scopes: the categories of data integration and infrastructures are related to computer science aspects whereas the category of analytical methods to the biological scope. For this reason we primary focus on the computer science scope to design our framework and thus on the first two categories. Once the system is designed, we can address emerging biological problems.

The analysis of the requirements of the categories data integration and infrastructure led us to generate the core model in Figure 3.2. As we previously saw in this chapter, data integration category handles three different data types; for our purposes we will need only two of them: the "experimental data" and the "knowledge data", visualized as different boxes, blue one and red one respectively. The infrastructure category adds the third box in the picture, the green one, allowing the computational layer access. Each box represents distinct conceptual user needs and motivations to use our framework. It means that, if a user is interested on running only his own data and applications in the framework, he will start from the box "computational layer"; if he decides to use also our data, then he will virtually jump to the two data boxes, this is the meaning of bi-directional arrows between two boxes. Similar considerations if a user needs only to access data. The data boxes are two distinct ones since, in this way, we can model the user need to access separately experimental data or knowledge data.

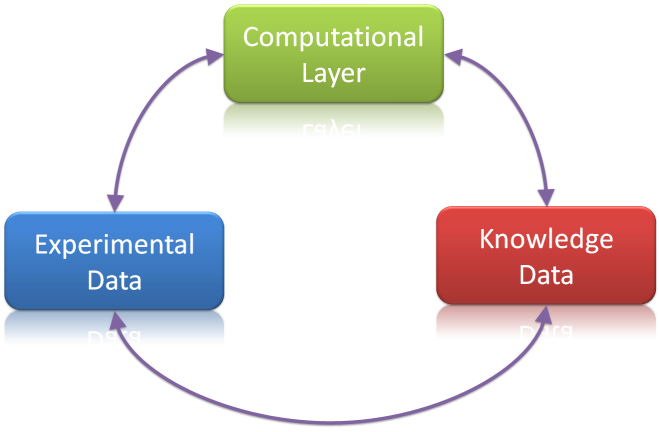


Figure 3.2: The core system model using three design blocks.

We have to keep in mind that these three boxes are conceptual blocks and users can access them only by interfaces, both application and web user

interfaces. For this reason, we have to extend the core system of Figure 3.2 in a more general framework model. Figure 3.3 shows the core system boxes in the computer science scope, the "framework space", that exposes an interface to the "biological space". In the biological scope we find both users and general biological problems, that is the analytical method category. In a general design perspective, users can access the framework through a biological problem, the light blue circle, or directly through system interfaces to data and computational layer boxes, the orange arrows. The orange circle represents the general interface between the framework space and the biological space.

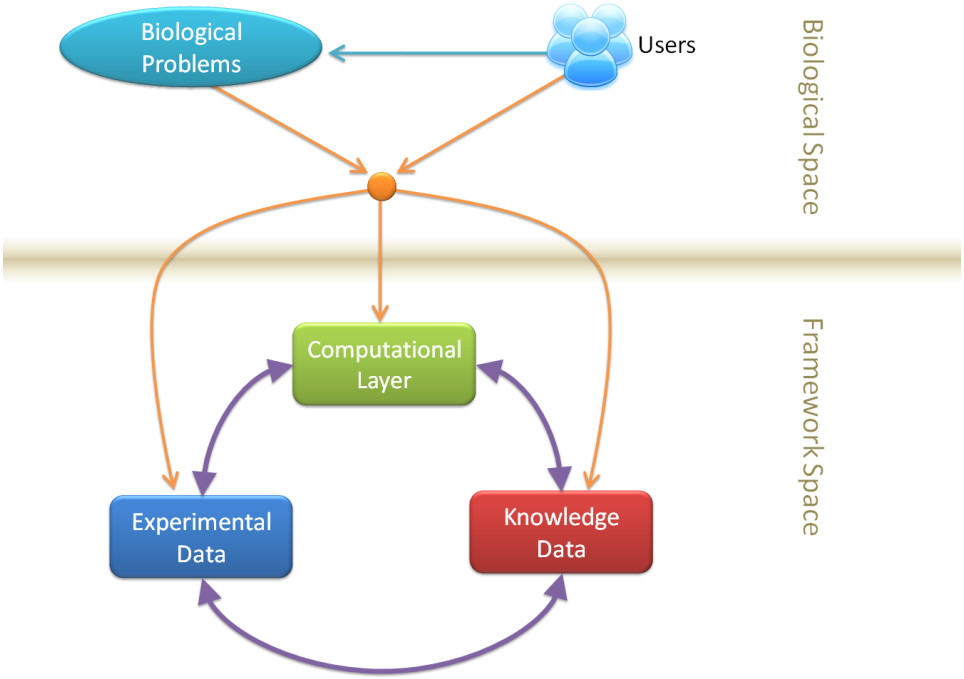


Figure 3.3: The general framework model.

With the model in Figure 3.3 we can design our general framework and then apply it to specific problems. Not all biological problems needs to use all the elements of the framework: some of them could need only a sub-set of core system boxes, for example only data boxes, and the custom pipeline to access data will also exclude some of the directions of the arrows. Examples of two biological problems will be presented in Chapter 5.

Chapter 4

Framework Design and Implementation

Chapter Summary

In this section, the discussion is related on what has been done, focusing on three design blocks:

1. *Horizontal Integration*: design and support for implementation of a database to collect experimental data: here we developed the schema of a system, called SNPLims, that models genotype and phenotype data;
2. *Vertical Integration*: design and implementation of a database for biological knowledge integration with quality controls; our approach unifies multiple level knowledge and this has been implemented by importing data from external data sources such as NCBI, EmsEMBL, UCSC, Gene ontology, etc. We also studied a method for conflict inspection among imported data sources;
3. *Computational Layer*: this section focuses on the computational architecture and which solutions we exploited. The framework can access *EGEE Grid* environment (VO Biomed) and the *Michelangelo* cluster (LITBIO project) to run analyses on distributed architectures.

In latest years, a huge amount of biological data have been produced in different bioinformatics areas, the "omics" fields. Big projects and efforts have been promoted on the integration and the fusion of these data and recent fundings went to *bio-banks* related projects whose objective is the integration of the data of different heterogeneous/homogeneous sources widely distributed to improve statistical analysis and diagnoses' discovery (for example the EU project BBMRI [62], the US project CaBIG [63], etc). What is actually the most important challenge is the integration of the biological knowledge and many efforts have been done in this way [45, 64, 65]. Based

on this integration, we could use the biological data, for example genotypes, to infer or discover other knowledge (disease genes for example). This new kind of integration is exactly perpendicular to the first one and for this reason is sometimes called *vertical integration* whereas the former one is called *horizontal*[1]. In other words what is missing in the bioinformatics scenario is a comprehensive system for global and ontological integration, starting from genes (or other measures) and combining gene's networks, pathways, systems biology models, and so forth, also able to support a wide data mining analysis.

To achieve our mining goal, we designed a data warehouse to integrate annotation data in a vertical perspective while experimental data in a horizontal one. Vertical and horizontal notation refer to the specific way in which we integrate data. The term horizontal integration is related to experimental data, based on the idea that many different research institutes generate same data and thus they can create a network for common data exchange and sharing. Such research centers often have different projects in which usually they exploit similar methodological approaches and platforms, producing experimental data in the similar format. For example, if two different centres are addressing GWAS studies for a specific disease, the first one with Illumina array of 1M Single Nucleotide Polimorphisms (SNPs) while the second one with Affymetrix platform of 600K SNPs, they both have fluorescences of SNPs. The difference in their data is related to the sample population and the specific set of SNPs present in each array, but since they are studying the same disease and also with the same statistical approach, they can share their own private data set and increase the sample power. In such a perspective, horizontal integration means to join similar datasets of different centres.

On the other hand, vertical integration is related to knowledge data. In fact biological data is naturally structured in overlapped layers, from tissues to genes and finally to nucleotides. Each layer groups similar biological entities, for example genes, and within each layer many different relations can occur, for example gene networks, coming from a semantic knowledge representation. Each ontological representation creates a map. We designed the database of annotation using a vertical integration approach and thus relating every neighboring layer. Since the notion of gene is a central role in genetics and in all the "omics" disciplines, the vertical integration points the gene layer in the center realizing a gene-centric integration approach: from the gene layer we can move upward to tissues and downward until single nucleotides.

Data are not all that a biologist needs. A very important task is to

[1]NB: this definition of vertical and horizontal integration is different from the same adjectives used in the Section 2.1. In the previous section we refer to vertical and horizontal integration as a mere database operation, strictly related to tables. Here we redefine the terms as conceptual data representation in a virtual mining space.

provide a set of analytical tools to researchers. The set of analytical tools is strictly related to specific biological problems, for example the Alignment problem or a statistical method for analyzing data. All tools that we can release, adapt or install have to be related to data and thus to the integrated database. Two of these biological problems will be discussed in the next Chapter (5) as use cases of the framework. An important feature that a framework for bioinformatics has to cope with is related to computational aspects. For this reason, we designed the framework able to run processes on both Grid environment and Cluster technologies.

Next sections will discuss in deeper details the three "blocks" that form the framework: horizontal integration, vertical integration and computational layer. Since the framework needed a validation for each of the three blocks, we separately designed and implemented them. The implementation process is ending and we am actually focusing on the integration of the three blocks in a new perspective, as we will describe in the final section 6.

4.1 Horizontal Integration

The first step for a coherent horizontal integration is the understanding of the data and processes for different laboratories. During the period of the doctorate project, we have been in touch with two laboratories and, based on their needs and procedures, we wrote specifications and requirements. The two laboratories belong to different institutions: the first one is a laboratory of genetics from the *Universita' degli Studi di Milano* (University of Milan) and the needs for this laboratory are related both to genetics and clinical aspects since the researchers of the institute are also medical doctors and clinicians. The second institute is *Multimedica*, a private research center that combines genetic research and biotechnology; Multimedica is also a laboratory for biochemical analyses.

The two research institutes needed a datawarehouse for data analysis and, based on their use cases, we designed a system for horizontal data integration. The experimental data that the two laboratories deal with mostly come from genotyping projects. Other data types are related to tissue expression analyses and sequencing; only for a small subset of projects they use the latter platforms. Each technology has got multiple platforms, for example Illumina or Affymetrix; in our case both laboratories use the BeadArray genotyping technology from Illumina, a chip for SNPs analysis.

A research project is structures in multiple work-packages and deliverables and may have got more phases. For each phase, a single platform can be employed. For example, in the project *Hypergenes*[2] we have got two phases: the first one is the genotyping phase while the second one is the validation step and this activity exploits sequencing activity and custom

[2]European grant HEALTH-F4-2007-20150, web site www.hypergenes.eu.

genotyping process. For each genotyping array, a set of different markers is defined and markers are strictly related to a unique annotation human genome build[3].

Each project enrolls a set of patients, depending on the specific phenotype under investigation. A set of patients defines a cohort: the term cohort is adopted since a project can include multiple sets of already available patients and each cohort could use different protocols for patient monitoring even though the phenotype is the same for two. On the other hand, a set of new patients can constitute a cohort with its own protocols and clinical variables.

A genetic or clinical project usually is build on a phenotype. For this reason, all patients are related to at least one phenotype, for example Hypertesion. A phenotype is represented by a set of clinical variables, like "systolic blood pressure" or "body mass index" and each single variable has got a type and a valid range of values. Each patient can have the value for each phenotype variable and these instances can be collected during the patient life (temporal variable).

Moreover, we have also to manage analyses and relative results. A study is defined on a specific project phase with a subset of patients and a phenotype. Dealing with genotyping technologies and statistics, the first project phase of *Hypergenes* for example analyzes data with Genome Wide Association Studies (GWAS) and thus a researcher needs to set up two sets of patients, cases and controls related on a phenotype, and for each SNP GWAS methods gets a statistical significance score (association value of SNP to phenotype). Other studies are related to Copy Number Variants (CNV), tissue expression, ans so on.

Given all these requirements, we designed a database; the structure of the database is described by Figure 4.1.

The database schema also contains a representation for researchers, users of the system and owner of studies. A user can process a study and store results and for each study it is possible to set users' permissions. Moreover, we designed a general tracking system for all tables and custom procedures and triggers to monitor user behavior and data editing.

Based on the first database schema, we implemented the database and we apply strategies to enhance database query engine, for example adopting binary data representation and so on. We exploited PostgreSQL[4] as relational database server.

So far, our solution is centralized but we designed the system with flexible and extensible technologies so that we are also able to release a distributed version of the system.

[3] A human genome release is periodically build by a specific annotation pipeline; for a deeper discussion and an example of EnsEMBL genome build, see [59].

[4] Link to PostgreSQL web site: http://www.postgresql.org.

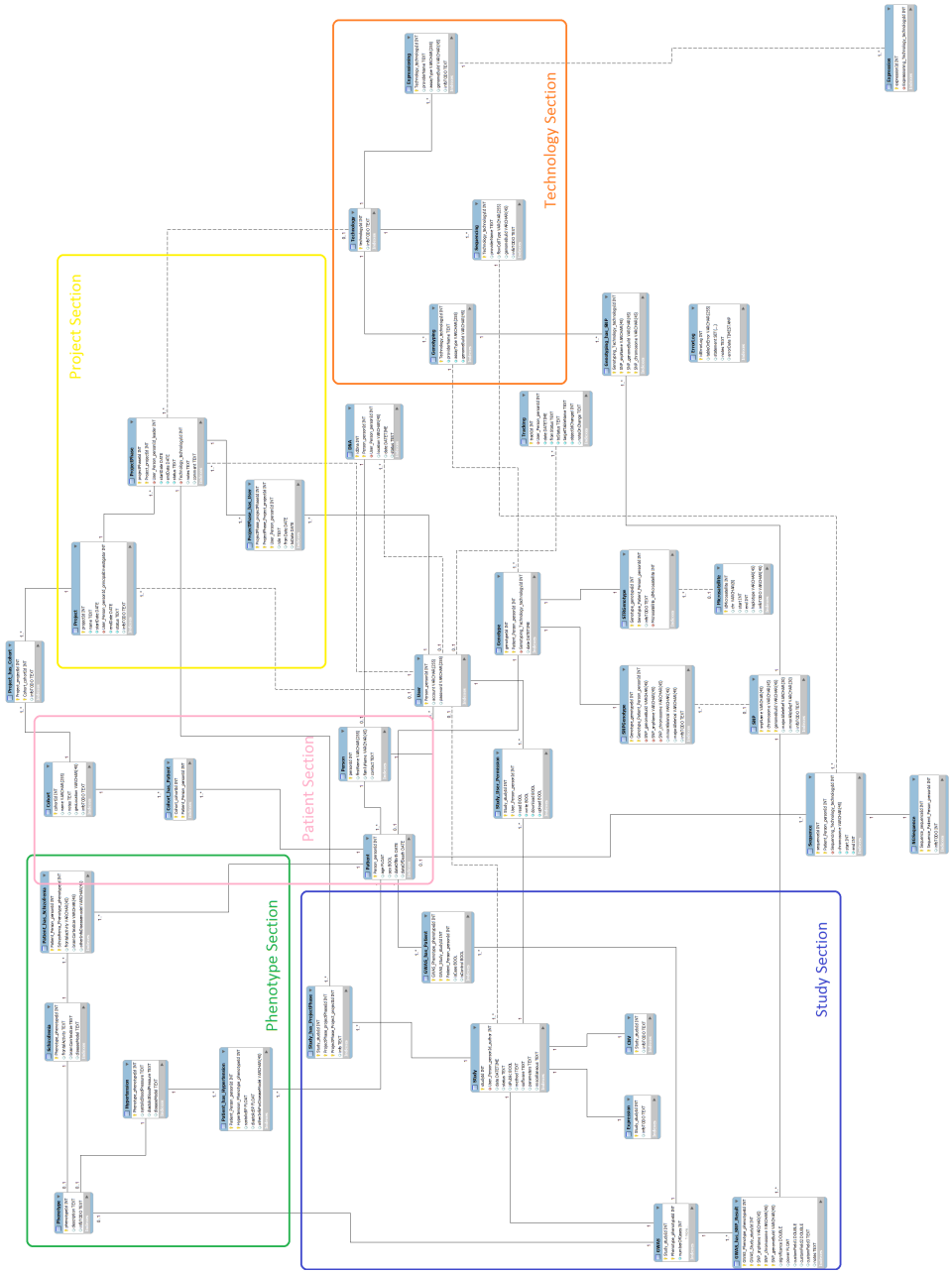


Figure 4.1: Database Schema.

4.2 Vertical Intagration

Approaching the vertical integration needs a completely reliable mechanism to ensure quality of data, their provenance and accuracy.
The most important public databases have an accurate data-quality control and in some emerging cases the operation of quality assurance is performed by hand. Considering a specific biological domain, for example gene area, some of the most accessed data sources are NCBI Entrez GeneBank [66], UCSC [52] and EnsEMBL [53].
For genome data, the annotation process that identifies new genes from different transcripts is based on different pipelines and algorithms [67, 68, 69]. Such differences sometimes lead to collect different information for the same biological object, in this case the gene: the differences occur as conflict in four forms: (1) missing values in some sources for the same gene, (2) completely different values stored in the data sources (not overlapping and not matching data) for the gene selected, (3) partially overlapping information for the biological object, and (4) absence of the entire genetic element in a database.

So far, systems that integrates genomic data in the same web page with a long list of identifiers, for example HGNC [70] and GeneCards [71, 72], can implicitly show to a scientist only such a behavior and present the different cases but do not explain any conflict, the reason for its occurrence, in which value and what sources are in agreement (consistent to the same object).
In case of different information for the same gene within the major considered databases, a biologist has to choose from what source to believe. Actually this choice is taken only based on the user's experience and practice but sometimes without a real awareness of the information quality for the queried genes and no explicit metrics are given to lead scientist through its choice. The importance of this choice is a crucial point since genetic data are sometimes blindly used from the databases and these information became the bases on which statistical analysis or clustering and classification are performed: if underlying data are unreliable then all activities standing on them are weak.

In literature, few studies exist on the problem of data-fusion of genetic data; one of these examples treat the data-fusion process as part of classification activity oriented to the so called *gene prioritization* [73, 74, 75, 76]. The gene prioritization approach is based on machine learning techniques and proposes a valid statistical framework to create a good classifier. The problem is that the gene prioritization model generates a classifier without performing a quality control on the data adopted and the authors refer to the fusion activity as the process to only collect data from the different databases and integrate the gene knowledge. They do not explain if they perform any conflict resolution, a basic procedure for the data-fusion, and they do not cite any index nor metrics to estimate the reliability of the data

source. In this way, even if data conflicts could be few or not significant, the classifier should be biased by a partial knowledge retrieving.

An example of conflict for discordant data is depicted in Figure 4.2 (example retrieved on June 2009; during the evolution and update of database part of the framework, most of these examples could change). The case illustrated is extracted from a real situation where a scientist wants to know something more about genes on the region of chromosome 6 at 136 mega-bases. Querying the region in the chromosome 6 at 136 mega-bases, the UCSC Genome Browser shows three transcripts: two of them referred to EnsEMBL (codes `ENST00000392356`, `ENST00000403495`) and the last one to Refseq (`NR_026805.1`). Next step is discovering the gene for the transcripts and relative proteins but each database refers the symbol linked from UCSC to different biological objects, and there is no suggestion to user for such discordance: navigating the links of these transcripts in the original databases we obtain three different genes, one for each transcript, but the single gene retrieved by RefSeq does not match any of the two EnsEMBL genes. Moreover the RefSeq gene id linked to a protein, thus is a coding gene; otherwise the EnsEMBL genes are not related to any protein, "No protein product" retrieved.

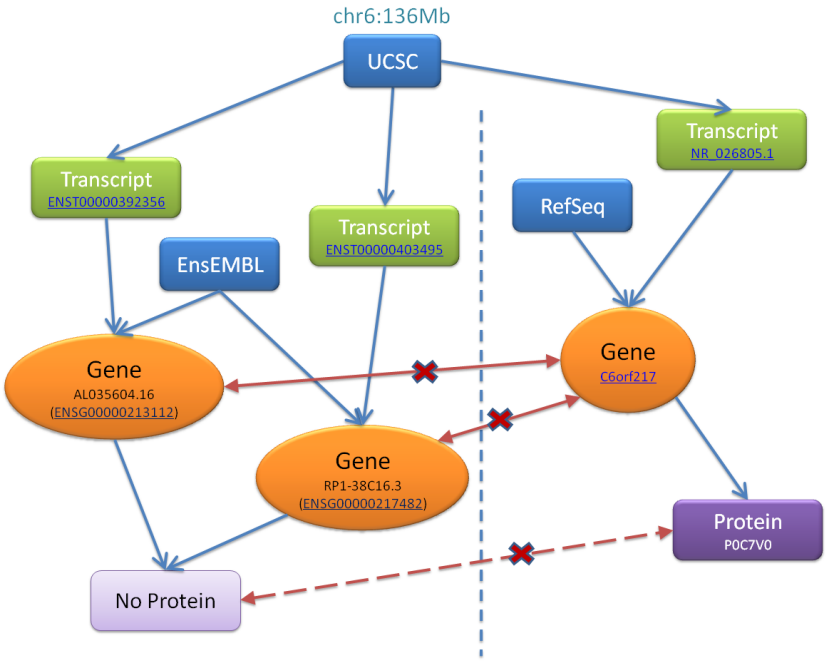


Figure 4.2: An example of discordant data on genes among UCSC, Entrez Gene and EnsEMBL.

Motivated by these reasons we found the need for a method that helps biologist to rationalize knowledge and makes explicit the differences among data sources based on their own information content, even if the percentage of data conflicts among different data sources is only relative. Based on the query of the user that chooses the data sources from which to retrieve data, the final objective of the model is to generate a synthetic and aggregated view of the data flow among sources through an integrated multi-query approach and to present results in two perspectives:

1. a *database-centric* view which analyzes the data flow among sources based on properties like depth of links and number of hops, type of database, and so forth, and returns a graph where each data source is a node and the arches are the integration connections on a gene field: objective of this view is to draw the data flow shared and look for annotation conflicts (as showed in Figure 4.2) directly from the graph;

2. a *gene-centric* view which presents the information added by each source to the gene annotation: objective of this view is to present what is each source contribution to the gene annotation and estimate what is the information gain of the source integrated.

We design the method decoupled in two steps: *data fusion* and *data flow inspection* activity. While the former manages data integration procedures, the latter helps the data fusion through data flow analysis by drawing a schema of data sources integrated and interconnected; the aim of this section is to describe such steps.

4.2.1 Data Fusion Activity

A general model for data fusion is composed by three main activities, summarized as follows:

1. *data quality inspection*: this step allows to inspect the data sources chosen with the objective of studying how the maintainers of the data collections produce their data, which updating procedure they use and with which frequency, what kind of pipelines they adopt and then which is level of reliability for each database;

2. *data integration*: once the data sources have been chosen, the data integration activity concerns how to unify the access to the data among the collections;

3. *data fusion*: finally the procedure of data fusion attempts to resolve the conflicts on discordant fields of the data among the different sources and present a unified view of the biological object with the level of reliability on each data source quantitatively explicit.

All these steps can be viewed as part of a data-warehouse and data-mining procedure.

Data Quality Inspection

There are many public databases that store genomic data [77] and all sources can be classified based on biological information collected. If we are interested in databases for genes and that produce their data from specific sequences and annotation pipelines, we mainly refer to NCBI Gene [66] and EBI EnsEMBL [53].

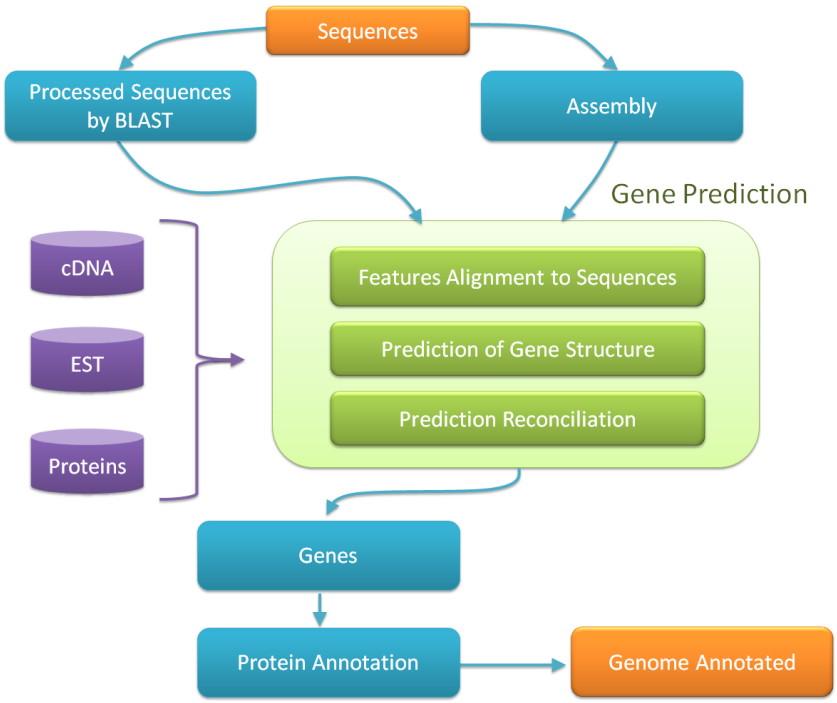


Figure 4.3: A general annotation pipeline adopted to predict genes based on sequences, extracted from EBI model. Ref. to [78].

These databases perform automated annotation pipelines to predict genes [78] from transcripts, protein sequences, as showed in the general Figure 4.3 where, starting from single sequences, creating the assembly, the gene prediction exploits more data related to different biological elements, such as proteins, EST, coding DNA. Specific algorithms and pipelines process all data and produce the gene prediction and relative annotation. Since the annotation pipelines use different algorithms and biological elements to get results, data collected in these sources could be different for predicted genes.

For Entrez Gene, the annotation process identifies sequence features

on the contigs such as variation, sequence tagged sites, transcript alignments, known and predicted genes, and gene models. Transcript models are generated via a Hidden Markov Model (HMM) using transcript alignment constraints and, if available, protein hit information. The curators of the database adopted manually reviewed annotations from RefSeq regions instead of corresponding gene models generated by automated processing.

Identification of genes within the genome assembly reveals the functional significance of particular stretches of genomic sequence. Genes are found using three complementary approaches: (1) known genes are placed primarily by aligning mRNAs to the assembled genomic contigs; (2) additional genes are located based on alignment of ESTs to the assembled genomic contigs; and (3) previously unknown genes are predicted using hints provided by protein homologies.

In EnsEMBL, for genome projects without existing high-quality annotation, Ensembl provides an automatic annotation. Ensembl is biased to producing a set with high specificity (few predictions being incorrect) potentially at the expense of sensitivity: the choice is to prefer missing a few features than heavily overpredict. Even if the automatic annotation cannot replace human annotation, the best automatic methods is retrieving all the evidence as best as possible on the reference to a human annotator then create custom heuristics to resolve the set of systematic errors. A disadvantage of this annotation procedure is that it becomes more detailed and therefore less generalized. Manual annotation is also used and EnsEMBL efforts are oriented to putative transcripts for the gene prediction.

Each of the database cited shows very accurate procedures for the annotation pipeline and their policies to accept a gene as predicted have a high level of accuracy and are widely significant. Thus in case of discordant values between them, a scientist cannot easily know if there are conflicts and moreover how to solve them or which source rely on and take the information since each of them are reliable in their point of view.

What we propose is a method that gets explicit existing conflicts on the biological queried objects and in next phases attempt to provide custom indexes for the user's choice.

Data Integration

The integration of biological data is a key feature in our approach since it is the layer which does the data retrieval. Since each public database exposes a different way to access data from each other, we design the integration layer with different procedures for data extraction and import, a single one for each database, and then all data are collected in a support database, designed using a data-warehouse approach. This internal integrated database can be adopted as data source for annotation analysis and conflict detection.

Based on the idea that biological knowledge is structured in different layers, from genes down to nucleotides and up to cells and systems, the vertical integration approach led us to design our data-warehouse using snow-flake schema, combining all identifiers of each single layer in a unique representation. Given a biological element, for example a gene, we created an object *Gene* which includes all key fields coming from the imported gene tables. Then we created an instance of gene schema for each gene and thus for each gene we are able to associate different annotations. Moreover, since we had to implement the vertical integration between neighboring layers, we combined the general snow-flake schema representations, that is the objects, of neighboring layers, for example the gene object is connected with the *Transcript* object and the *Protein* object. These observations are drawn in Figure 4.4 and 4.5, where the first picture represents the vertical data integration idea and the second one the snow-flake approach.

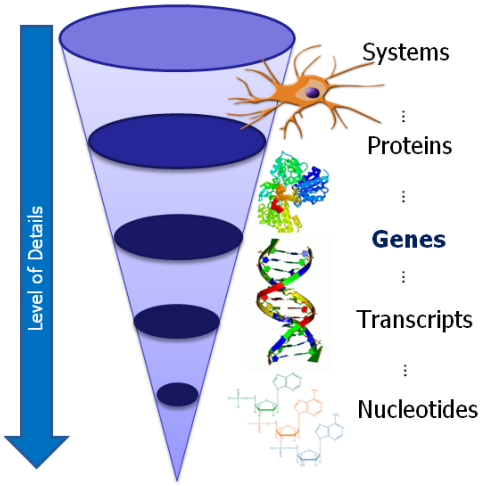


Figure 4.4: The vertical integration meaning on biological annotations.

The database is oriented to systems biology and is also ontology-based. It exploits a MySQL server and the data warehouse approach consists in collecting and transforming heterogeneous data from different sources, allowing their integration and accessibility. This approach is typical of data integration models and differs from data integrity models, often used for normalized databases which are widely used to maintain primary resources. The database is gene-centric and considers at the moment only human genes which are annotated, among other features, by symbol, description, aliases and sequences. Data about SNP are downloaded from dbSNP [79], which allows to integrate data about chromosomal and *contig* position, heterozygosity, alleles and function of the related DNA portion. Moreover, gene products have been collected as list of mRNAs sequences and related protein isoforms according to the NCBI RefSeq annotations (NCBI Nucleotide [80]).

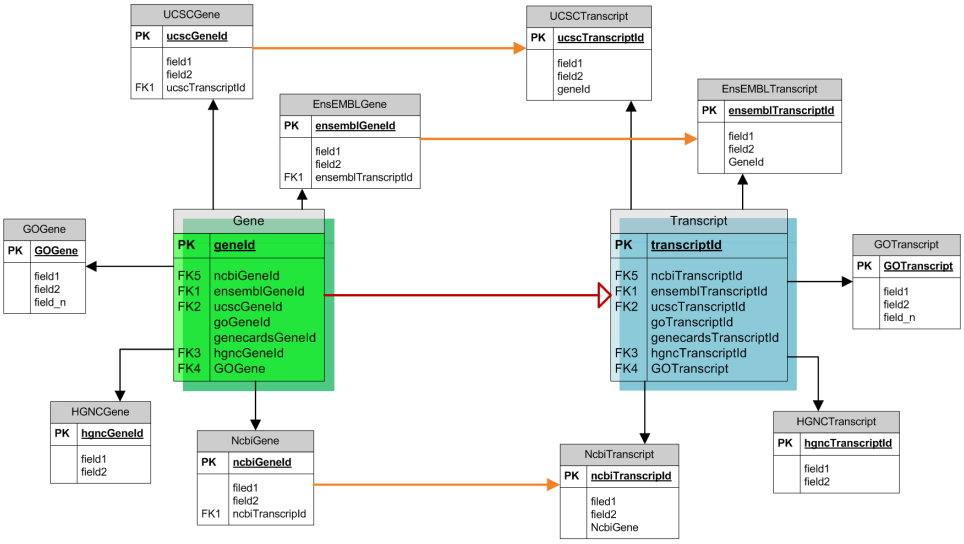


Figure 4.5: Snow-flake schema representation for two simple nodes: genes and transcripts.

Data about proteins include all the identifiers suitable to download the related sequences, functional domains [81, 82] and structural models from the Protein Data Bank [83].
The systems biology perspective leads to consider data such as the list of the biochemical pathways (KEGG [84] and Reactome [85]) where human gene products are involved, and information about protein-protein interactions (PPIs), collected from BioGRID [86], which complement knowledge about pathways and enable crucial network based analysis.

Finally, in order to provide a standard framework for data integration and a reliable engine for SNPs selection, the database has been built on a strong ontology layer. Whenever available, data have been annotated using ontological terms: Gene Ontology [87] for genes and KEGG Pathway ontology (derived from the hierarchical organization of KEGG pathways) for pathways are just some of the hierarchically structured vocabularies that underlie the infrastructure. Additionally, ontology structures allow to improve the performance of statistical and analytical evaluations by means of the graphs that undergo the hierarchically structured vocabularies and that shed light on the relationships between biological components.

To access and import external sources, we adopted different methods for each data source. The UCSC Genome Browser grants access to the database via FTP (to files and tables) and directly by MySql ODBC. NCBI created access to data via API, the *eUtils*, with more programming languages such as *Perl* and *Java*. Moreover EnsEMBL exposes a good system of web-services,

very useful also for work-flows. Other public collections don't have programming interfaces or accession systems and thus they need custom wrappers, for example exploiting community tools and libraries already created for this facility or implementing new ones.

Our system is periodically synchronized with original data-sources and thus our data are always up to date.

Data Fusion

The final data fusion process provides a way to recognize and make explicit conflicts among the same biological objects and it presents a complete and consistent representation as a whole. While *schematic conflicts* (dealing with different attribute names or differently structured data sources) and *identity conflicts* (related to different way of identifying a real world object) are considered in the integration process, the data fusion activity focuses on explaining multiple representation of the same biological object. Even if the semantics of the relative attributes is equal, data conflicts refer to the different attribute values. In such cases there are two kind of conflicts: the *uncertainty* about the attribute values caused by missing information, and the *contradiction* caused by different attribute values.

In our examples from the biological database related to genes, one of them presented in Figure 4.2, the most troublesome conflict regards the contradiction and a domain specific technique has to be evaluated: the ontological derivation and the suggestions coming from other biological objects related are the key point to solve the contradiction. Missing values in some sources will be ignored and the data present in the other databases will be considered as effective values. Tracking of the missing values has to be reported in the database, in particular if in the presentation layer we planned to realize also link integration or DAS.

To address the problem of conflict analysis we design the data flow inspection activity, a way to discover conflicts using graph and multi-query approach to our internal integrated database.

4.2.2 Data Flow Inspection

The data flow activity is based on the integrated database that has been created in the data fusion phase and replicates the graph of the existing data flow of integration among databases and reveal whether conflicts on annotations subsist: all the process is based on a multi-query approach that checks for consistency and congruence of the data and annotations. Moreover, it helps scientists on understanding what is the current integration status on the biological element considered among the data sources, what is the contribution of each database to the overall gene information, and how much does each source increases the knowledge level on the element

queried. While the conflict inspection activity is related to a database-centric perspective, since it is based on databases and their data properties, the information content analysis phase is linked to a gene-centric perspective, since it exploits data sources to estimate the contribution of each added annotation which increases the gene knowledge.

Database Perspective

We propose to extend the user behavior in Figure 4.2 to draw the schema of data integration and understand if conflicts occur and where they are located. To address the problem we chose a multi-query approach and the algorithm can be synthesized as follows:

- select some data sources: to reflect the previous case for example UCSC, NCBI and EnsEMBL;
- choose a gene and the root database: this collection will be the first source to query (the reference source);
- query the root source in the internal database (integrated and updated) and fetch all the objects related to the specific gene (transcripts, proteins, etc) and their identifiers for the other sources;
- `while`: all identifiers are related,
  `do`: query the internal database with all the identifiers fetched and retrieving the results for all the data sources (the root and the other remaining ones) check for inconsistencies between identifiers.

If the internal database is consistent and coherent, that means no conflicts among the data sources considered, then the method converges to a unique solution quickly; otherwise the computational cost to solve the relations is exponential, proportional to the amount of conflicts since the while loop runs a query for each added conflict using all identifiers.

The key point in the `while` loop statement is when all identifiers of the biological objects can be related each other. Drawing these relations we discover that we are plotting a graph where nodes are the data sources and arches are the integrated identifiers (on a specific gene field). In this way we can consider the data flow as a path among nodes, and if the final graph, resulting after run the method of multi-query, is connected and has cycles then we have no conflicts. Otherwise the plot will show how much conflicts propagate the errors and generate information entropy.

Gene Perspective

The last word in the paragraph 4.2.2 suggests that we can also study how much the information added by each source improve the annotation for the

gene, and thus estimate the information gain. This is still a work in progress which is in analysis in the gene perspective.

The main idea of this other view is to point out what benefits are there by using a set of sources instead of others and by creating a plot of both what information is shared among the sources and what is unique. The starting drawing presents all the elements that the first query fetched (based on identifiers) and how they are connected in a graph. Since this view is symmetric to the database one, we handle the same problems, limits, and computational bounds: if conflicts exist, the graph would have a increasing number of nodes.

This method for data quality control has been presented to the conference CIBB 2009, Computational Intelligence methods for Bioinformatics and Biostatistics [88].

4.3 Computational Layer

The computational platforms that we enabled to the framework are two:

1. Grid EGEE infrastructure;
2. Michelangelo LITBIO cluster.

The Grid computational infrastructure is operated by the Eurpean Project EGEE (Enabling Grids for E-sciencE [89], Phase III) for the European scientific and research communities with more than 240 sites worldwide, providing access to more than 50.000 CPUs.

It consists of a collection of computers, storages, special devices and services that are heterogeneous in every aspect, geographically distributed and dynamically linked by a wide-area network which can be accessed on-demand by a set of users with appropriate authentication and authorization.

The Grid was originally invented as a practical solution to the problems of storing and processing the large quantities of data that are going to be produced by CERN"s Large Hadron Collider (LHC). This computational facility can be seen as a service for sharing computer power and data storage capacity over the Internet, going beyond simple communication between computers, and aiming ultimately to turn the global network of computers into one vast computational resource. In this scenario, this infrastructure enables data sharing between thousands of scientists with multiple interests, tries to ensures that all data is accessible anywhere, anytime and copes with different computer centers access policies, ensuring data security.

The EGEE Grid infrastructure runs upon a set of middleware services called *gLite*[5], which is integrated, certified and distributed by the project

[5]Link: https://edms.cern.ch/file/722398/gLite-3-UserGuide.html.

itself and world-widely deployed on the computational resources. The services available in the gLite distribution can be broadly classified into two categories: the Grid Foundation Middleware, covering the security of the infrastructure, the information, monitoring and accounting systems and the access to computing and storage resources; the other is a higher-level Grid Middleware, including services for job management, data catalogs and data replication, providing applications with end-to-end solutions. In order to use this infrastructure a personal authentication certificate is required. The certificate is released by a Certification Authority and associated to a Virtual Organization, a geographically independent group of collaborating scientists; the access to computational resources is executed through dedicated User Interfaces running the gLite middleware and its CLI commands.

On the other hand we could use also a cluster, called *Michelangelo*, provided by the LITBIO project[6] and is composed by 70 nodes, and 18.5TB of redundant storage. Each node is composed of two 275 dual-core AMD Opteron CPUs (total of 4 cores) and 8GB ram; all the nodes are connected via 10 Gbit/sec Infiniband, granting maximum scalability.

The Grid-enabling part of the framework (VNAS [90]) has not been all implemented by the author of this thesis, but he contributed on the design of the second version (the current stable release). A short description of VNAS software follows.

A common strategy for enhancing the success rate for the execution of jobs is to develop a system to monitor each launch and to manage the resubmission of failed jobs. GMarte, for example, an infrastructure for studying myocardial ischemia [91], has check-pointing capabilities that allow execution recovery in case of remote resource failure. Another example is the WISDOM initiative [92], the objective of which is to develop a virtual screening pipeline for drug discovery: during the challenge execution, its system is able to track failed jobs, to ban the faulty computing elements from the requirements section of the job definitions and to resubmit the failed jobs. Submission engines may also provide strategies to address jobs to the best available nodes: this task, known as resource mapping, is generally executed by the Grid middleware. The general problem of optimally mapping tasks to machines in a heterogeneous computing environment has been shown to be NP-complete [93, 94], thus, analytical solutions give way to heuristics. Defining a fixed strategy applicable to different scenarios like data intensive or computing intensive applications is a problem and different heuristics apply to different situations [95]. Static mapping is therefore unlikely to be optimal in a time-evolving scenario like the available Grid resources and dynamic load balancing strategies are in continuous development, both at mid-

[6]Laboratory of Interdisciplinary Technologies in Bioinformatics (LITBIO) is funded by Ministero dell'Istruzione, dell'Universita' e della Ricerca (MIUR - Italy) through a FIRB 2003 grant. For more information visit the project web site http://www.litbio.org.

dleware level (an interesting comparison of late binding versus early binding in gLite 3.1 can be found in [96]) and at higher abstraction levels. In the latter case, an interesting solution may be provided by submission with pilot jobs (see DIANE, PANDA, glideInWMS, gLExec [97, 98, 99, 100, 101]), in which tasks are no longer pushed through the grid scheduler but are put in a master pool located on the User Interface and then pulled by pilots running on computing nodes. Other approaches make use of meta-schedulers which compute metrics through resource discovery and matchmaking, in order to optimize job submission [102, 103] and adopt job replication techniques for those jobs that are most likely to fail [104].
The pilot job technique permits the exploitation of grid resources very efficiently since pilots running on faster resources can pull more tasks than those on slow resources. However, this implies that only the pilot jobs are managed by the grid scheduler (the Workload Management System) while all the effective jobs bypass grid queues and are consecutively executed until the maximal wall-clock time allowed for the (pilot) job, controlled by the local resource management system, is reached. This could potentially keep an arbitrarily high number of grid computing nodes occupied for days and this aspect rendered the pilot approach to be considered unfair with respect to the grid computing philosophy [105].
Another common problem of existing solutions is the excessive network traffic between the master pool and remote jobs, during the connection for data fetching and results pushing. This is due to the lack of an optimized data-management such as basic data identification and deduplication algorithms. Usually the problem is partially worked around by placing the necessary files (especially input files) on Storage Elements. However in this case the uploading has to be performed by hand at the beginning of the computation, and also algorithms for garbage collection of data on Storage Elements at the end of challenge computation are in general not provided by the main grid submission softwares. In addition, this solution implies monitoring the files availability from the worker nodes, creating replicas in order to reduce network latency and recovering data if one replica at one location is lost or unavailable. This task can be handled with different strategies, as reviewed in [106], and is a critical aspect as in case of a file failure, the pilot job like any other job has to be marked failed and resubmitted.
The potentially unfair [105] aspects, implicit in the techniques that bypass the Grid WMS and its queues along with some security issues (defined as serious in [96], deep in [107] and as potential security holes [105]) implied in many implementations and the need for a better data management made us prefer the implementation of a different approach, based upon the enhancement of an existing framework called VNAS.

We developed a software layer, built on top of the grid middleware, for launching and monitoring grid jobs and managing the success of the overall computation by resubmitting problematic jobs automatically. This

layer was developed by enhancing an earlier framework (VNAS [90]), adding important features aimed at increasing the success rate of the submission. These improvements include, among general code optimizations, the implementation of a multi-threaded submission engine, the use of a dedicated FTP for file transfer and a management system for Grid replicas of needed files. These are described below in greater detail. The VNAS framework also features data deduplication and management capabilities, integrating strategies for the detection of failed, hanged, endlessly queued, or otherwise problematic grid jobs, allowing an automatic canceling and resubmission for jobs detected in such conditions. This results in an abstraction layer with reliability over the Grid platform, accessible with a set of API for grid operation which significantly eases the task of developing new applications for the grid. Currently VNAS features pilot jobs only in the very limited implementation of one task per pilot, meant as a job-wrapper that prepares data on the Worker Node (WN), runs the job and prepare the outputs for reclaim. We considered it was not appropriate to significantly bypass the grid WMS and queues and were concerned about the excessive exploitation of the grid that can be caused by a full implementation of pilot jobs. This can result in unfairly low grid availability for other grid users. However, we recognize that a full pilot job implementation could greatly ameliorate some of the problems we have faced and which are described in this paper, such as challenge completion times being impacted by late jobs on slow nodes, so we are in fact considering implementing a full-fledged pilot job feature (i.e. multiple tasks per pilot) for a future version. On the other hand, VNAS data management and deduplication capabilities as described below is a distinctive feature and makes it unique among grid submission frameworks.

During the design phase of VNAS framework we introduced the requirement for this system to be capable of handling jobs which required huge amounts of data. Even if the current version of gLite allows to specify the max amount of data which can be submitted by the physical input-sandbox, it has been shown [106] that it is better not to use a large sandbox and instead put big files or databases on the Storage Elements (SE), in order to facilitate the single job submission. Currently DIANE [97], for example, cannot handle this aspect, and so the user has to handle the replica management strategy by manually putting files on storage elements, managing replication and monitoring of data availability and take actions in case files are not reachable on the SEs any longer (which unfortunately does happen). Finally, they have to remember to clear data off the storage elements upon end of the challenge. This means that for every challenge/analysis, the user has to perform essentially the same operations over again.

For this reason we designed our VNAS to handle the replication of data on storage elements. This is achieved via a "virtual sandbox" feature: an emulated sandbox which transparently uses the Grid's Storage Elements for

temporarily storing and retrieving files. During the submission of one job, if the files required for job's operation are not detected to be duplicates (see below), they are immediately uploaded on a VNAS-controlled home FTP server and are immediately available to grid jobs. Further replications of such files on Storage Elements are performed at job execution time directly from the Computing Elements by VNAS job-wrapper agents: an additional replica is added in normal conditions onto the SE nearest to the CE (this is LAN distance in most cases) just prior to file downloading, if not existing already: this comes free in terms of time and bandwidth, compared to simple download, and the number of replicas for virtual sandbox files becomes in this way proportional to the files' usage. All replicas are garbage collected after a period of inactivity as explained below. This solution greatly reduces the time needed for submission. A copy of the data is still maintained at the home FTP server, so to be able to proceed with the computation even in case of a simultaneous failure of all the replicas (this is possible when they are still few, or in case of temporary failure of Logical File Catalog). This solution also limits the impact of overall data transfer during big challenges, since during execution of jobs common data gets heavily replicated, eventually ending up in most SEs, and at that point the inbound data transfer to the Grid sites (WN+SE) performing the computation substantially decreases. The virtual sandbox also features integrated optimizations such as data deduplication: this is performed by automatically detecting equality of virtual sandbox files based on content, across different submissions, even when performed by different users. This avoids uploading again files which were recently uploaded through the virtual sandbox system, hence saving Grid's bandwidth and storage space. After N days of no-use, automatic garbage collection removes virtual sandbox files, hence guaranteeing cleanup of Grid resources while relieving the user from the burden of continuously thinking whether they might or might not still need a certain file loaded on the Grid for their future Grid launches.
VNAS also simplifies the development of complex grid pipelines by providing a callback system for performing automatic actions upon events, such as single jobs or groups of jobs having reached global completion or irrecoverable failures (max number of attempts exceeded). This can ease the creation of complex applications such as this one and up to arbitrarily complex pipelines and workflows with multiple computational stages on the Grid. The VNAS system is divided into three main sub-applications: the launcher, the submitter, and the receiver/monitor application; such applications allow to independently manage the creation and submission of the grid jobs as well as the retrieval of the results. The launcher application is directly called by the user/application, and is used to "prepare" the job execution environment (substantially a virtual sandbox directory with the executable to be run and input data for a single job execution, plus some management parameters). The job payload is split into an SE part (virtual

sandbox) which is directly uploaded at this stage, and a stub part containing a job-wrapper (substantially a 1:1 pilot) and some management parameters for the wrapper, including instructions needed to reconstruct this specific sandbox at CE side and run the computation with proper parameters. It is at launcher's stage that data deduplication kicks in: current virtual sandbox files are matched against old ones (MD5 hash based on content, detecting equality even upon file name changes) and this prevents useless resubmission of existing files, saving bandwidth, storage space, and job submission time. Also, at this stage the garbage collector days-counter gets reset, so to extend the life of existing matched (deduplicated) files. The launcher finally delivers the stub containing the job-wrapper and management/execution parameters to the spool for the submitter application. Each job stub prepared by the launcher application is then submitted to the grid environment by the submitter component, running as a multi-threaded server application, polling for data in the spool from the launcher application. The receiver/monitor application periodically checks the different status of each job, identifying failed jobs that have to be re-submitted, and fetching the results for completed jobs. In our experience the retrieval of results can be a long process and the risk of losing communication between the Grid user interface (UI) and the Grid environment in this time window does subsist, resulting in erroneous truncated results files at the receiver's side. In order to address this problem, at the end of the computation the MD5 checksum of the results file is computed by the VNAS's job-wrapper agent at the worker node side and this is sent to the VNAS's User-Interface side separately. After this, the job output itself is transferred to the home FTP server. At the UI-side VNAS checks the MD5 information against the results file: a mismatch in the hash re-computation flags the job as generically "failed" and job resubmission is performed.

The submitter and the receiver/monitor applications run as multi-threaded processes on the UI, which means that more grid operations can run concurrently: this decreases the time needed for jobs submission and monitoring while enabling the system to be fault tolerant in case of grid failures. A higher performance in job submissions is also achieved by using several WMS servers, in a round-robin or randomly-chosen fashion so to split the workload for job submission onto several servers. This leads to a more even load on WMS servers and also limits the number of failures during submissions due to WMS server failures (a submission failure would be retried on a different server).

The VNAS framework has been used in various earlier and current projects in order to raise the reliability and reduce the development time for new Grid applications [108, 90]. A scheme of VNAS design is shown in Figure 4.6.

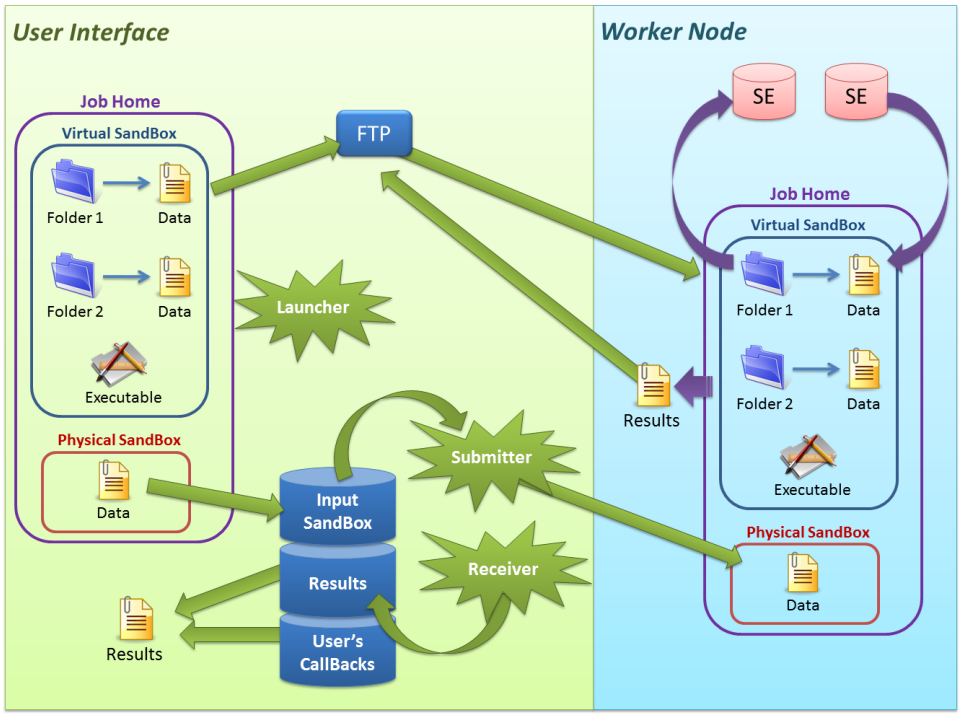


Figure 4.6: The VNAS framework scheme.

Chapter 5

Framework Application

Chapter Summary

In this chapter, we are presenting two biological problems and their application within the framework. The discussion will follow how the system (the framework) supported the solutions to the problems following the three design blocks description. The two applications are:

1. *Linkage Analysis*, a very well known statistics in human genetic epidemiology with high computational limits; we focused on developing and implementing a new solution to the problem. We also tested system's performances in Grid and Cluster environments, comparing results;
2. *SNP Ranker*, a new method for biological data mining on gene knowledge that we developed and implemented. We will present a brief introduction and the biological context description, and then the design and implementation steps, based on the three blocks of the system, will follow.

Given the requirements and the design of the overall framework, we are presenting here two case studies in which we exploited the integrated clinical genomics framework approach. The two examples arose from two real needs in our laboratory and, by using the different tools we developed, we could solve both problems. The first problem is related to Human Linkage Analysis (section 5.1) and the second one to a new approach for SNP data mining (section 5.2).

5.1 Linkage Analysis

5.1.1 Introduction

Finding the relationships between the occurrence of human diseases and the expression of a particular gene or genes among the estimated thirty to

thirty-five thousand protein-coding genes that form the human genome is a theoretically and practically hard challenge. A common approach to this task begins with Genetic Linkage Analysis, which is a statistical method exploiting the presence of known genetic markers in order to identify the location on a chromosome of a given gene involved in a disease. This is obtained by analyzing genetic data matched with information on the family structure, following the inheritance of phenotypic alterations in families through the generations while measuring the propensity for genetic loci to be inherited together due to their proximity on a specific chromosome.

Among the individuals of a population, phenotypes or traits are said to be independently assorted whenever the responsible genes are subject to recombination with a probability of at least 0.5. When genes, or more generically loci, appear near one another on the same chromosome they are usually inherited as a single unit and are said to be linked. In many cases, even linked genes produce offspring with novel allele combinations due to crossing over during meiosis. Genetic Linkage Analysis is a statistical method used to identify the location of a given gene or locus involved in a disease by finding a known genetic marker close enough to be considered linked and by following its transmission through generations. Such analysis is based on the evaluation of a test parameter called the LOD score, defined as the Logarithm Of Odds (LOD) ratio between hypothesis of linkage (H_1) versus the null (H_0) one. For more detailed explanations about the biological domain for linkage analysis we suggest specific literature [109, 110].
Two major algorithms [111, 112], with several implementations [113, 114, 115] have been proposed for the assessment of linkage between the disease and the commonly used markers, such as the Quantitative Trait Loci (QTLs: regions of DNA associated with a particular phenotypic trait), Microsatellites (polymorphic loci that consist of repeating units of 1-6 base pairs in length) and, recently, also Single Nucleotide Polymorphisms (SNPs: DNA sequence variations involving a single nucleotide). From the computational point of view, Linkage Analysis and pedigree analysis is ascribed to the NP-hard problems [116] and the computational cost of all the mentioned algorithms grows exponentially for one of the two variables of the LOD Score equation, pedigree size and number of markers.

The computational effort required by a medium size problem can be up to several CPU hours and huge RAM allocation, therefore the role of high performance computing is highly relevant in this field. The use of distributed architecture environments can be an appropriate solution both from the computational point of view and for data management, even if, submitting jobs, monitoring their status and retrieving the results can be challenging when working on huge amounts of data. Since we are dealing with SNP chips, and consequently with a very large number of markers, we adopted the algorithm proposed by Lander and Green [112]. This allows us to handle sets of about 100 markers at a time (depending on other variables

of analysis), but we have to monitor the pedigree size variable. In addition, we need to validate the porting of this algorithm, or more precisely, the data splitting procedure adopted to overcome its computational limits (see Methods section).

Related Works

Few implementations of linkage analysis systems on the Grid platform have been described in the literature. Grid-Allegro [117] is a Grid porting for one of the linkage analysis programs, Allegro, which implements the Lander-Green algorithm. Grid-Allegro offers a series of scripts for managing linkage processes in Grid and supports parallel analyses for the same input data. However, for our project we needed a method to compute a whole genome analysis for thousands of SNPs and this appeared not possible with Grid-Allegro. Moreover, we wanted to exploit a wider portfolio of linkage analysis software, for example adding Merlin [109]. Grid-QTL [118] implements a web version of linkage analysis in Grid for Quantitative Traits Loci (QTL). As the name suggests, Grid-QTL is mostly oriented to quantitative traits instead of SNPs.

Since we wanted to compute a general linkage analysis, not only oriented to QTL, and we also wanted to provide an easy interface allowing users to configure a linkage analysis pipeline including marker filtering, error detection and other features, we first designed a new method for genome-wide (GW) linkage studies, then we implemented such method in the Grid environment and finally we created a web-based user interface so to ease user access and to hide the grid layer complexity. From the infrastructural point of view, numerous works in literature describe the techniques and efforts needed for handling and optimizing the submission of a large number of jobs in the Grid environment. Having to cope with the great heterogeneity in the available resources and the very variable response times typical of the Grid environment, along with some possibly peculiar requirement specific to each Grid porting, there is often need for a specific framework developed on top of the Grid middleware.

Objectives

Current technologies for SNP chips (for example Illumina) provide SNP genotyping arrays ranging from 10,000 to more than 1 million SNPs. Pedigree files, which collect the information about the family structure, are also often large, containing more than 30 individuals. Computational time and space on a single CPU is unreasonable with these data volumes and therefore there is a need for a distributed high performance computational infrastructure and a system capable of performing linkage analysis with large datasets.

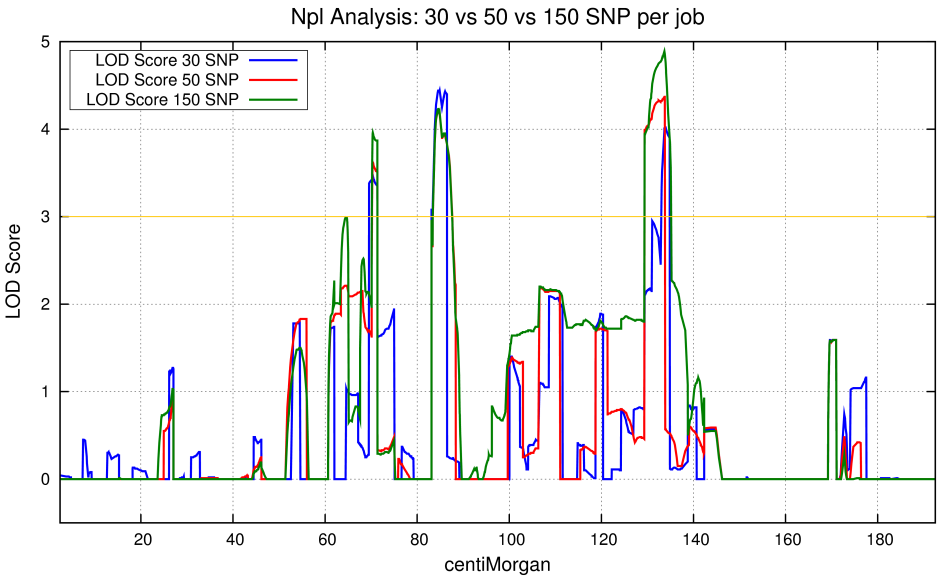


Figure 5.1: Methodology test comparisons and results plotting.

The aim is to enable the use of the EGEE Grid Infrastructure and Michelangelo LITBIO cluster for the execution of linkage analysis on very large SNPs data sets, creating a pipeline for the linkage process. Computational jobs are then launched over the grid infrastructure, distributing the needed computation among different computing elements. In this context, a user friendly web based access to grid resources is provided. The proposed facility will be tested with challenges performed with marker data collected using the densest chips currently available (up to 1 million SNPs).

5.1.2 Methods

Due to the computational problem of the linkage analysis and the need to analyze up to 1 million markers for a general pedigree, we designed a heuristic approach to overcome computational limits of available applications, such as Merlin, and compute a whole genome analysis in a single execution. Our heuristic method is based on the reduction of the input set of markers by splitting the total set of 1 million SNPs into smaller sets on which independent analyses are run.

The correctness of the proposed approach was tested by setting up a benchmark analysis with a given dataset: we extracted 2230 SNPs from a chromosome and created different input files from these SNP data for the linkage analysis. The formatted input files were created using a different subset of SNPs for each run: 30, 50 and 150 markers. After running our pipeline, the results revealed that the most significant peaks were identified in all the three independent executions, and this proved that our method-

ology can be a valid heuristic for this NP-hard problem. The analyses were run both with the proposed system, i.e. splitting the input files and merging results after separate parallel processing, and on the Michelangelo Cluster, capable of processing the whole dataset in a single run. Test results are shown in Figure 5.1. By comparing the outputs of each analysis, it can be seen that the major signal peaks are well represented in our system's results, showing that the approximation used, introduced by splitting the data and merging the results, is acceptable.

5.1.3 Interaction with Framework

Figure 5.2 summarizes how the Linkage Analysis problem interacts with the framework.

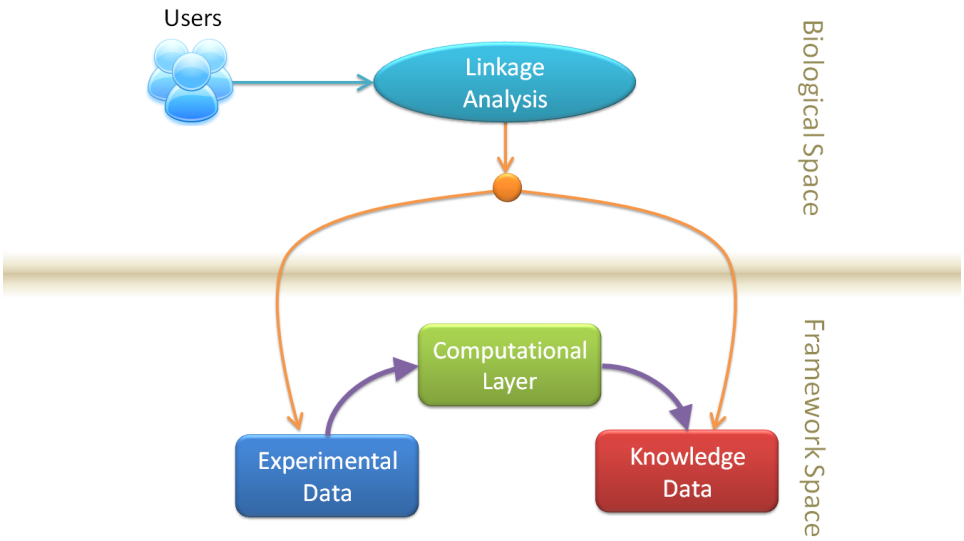


Figure 5.2: Linkage interaction with the Framework.

The idea is to release the Linkage Analysis support as web application. In the *biological space*, users can directly connect to the interface of the Linkage Analysis platform. One of the goal of this platform is to hide the computational layer complexity and for this reason users can interact with only two of the three design blocks within the *framework space*: the Experimental Data and the Knowledge data. The first one is the entry point, where users parametrize their own challenge for a linkage study. The last one is the final step: once results are ready, users can retrieve plots and

SNP/markers data. The knowledge block exploits annotation information and append all biological annotation to each considered marker, from gene symbol to position and allele coding.

In the framework space the three design blocks are sequentially connected like a pitfall and arrows between two consecutive blocks have a single direction. The meaning of this behavior is that in the linkage analysis use case we have to configure a pipeline and hide the computational layer and its complexity to users.

Since the framework has been designed by three conceptual steps, we summarize here how the use case exploits each layer:

- *Vertical Integration*, related to the *Knowledge data* block, is realized at the end of linkage analysis pipeline execution: each job produces a result file containing the list of markers and their LOD Score; once each single job is ready and has been retrieved from the computational layer, the system looks for each SNP in the annotation database and append in the result file a set of biological annotations related to genomic context, from SNP position and gene symbol to allele frequencies and so forth;

- *Horizontal Integration*, related to *Experimental data* block, focuses on genotype information and family structure. In our database, we collected demographic data, phenotype information and genotypes data. Since a linkage analysis is intended to family-based studies to understand a genotype that causes a disease, the system can extract all information from our internal database: the family structure from demographics tables, the phenotype information (affection status) from specific phenotype table and finally the genotype data of each individual from genotype tables.

- *Computational Layer*, related to *Computational layer*, is exploited through both computational infrastructures, Grid and cluster technology. The computational complexity is hidden to users that have only to interact with the web interface. For the Grid EGEE interaction, users must have a valid certificate, whereas for the Cluster Michelangelo LITBIO a specific user has been created with custom resource allocation.

5.1.4 Application Implementation

After solving the heuristic problem we shifted to designing the application implementing our heuristic approach in the Grid environment. Since our aim was to ease the use of linkage analysis, we needed a web application hiding the grid complexity and allowing users to interact via graphical tools. GUI components lead users at each single linkage analysis step, from input

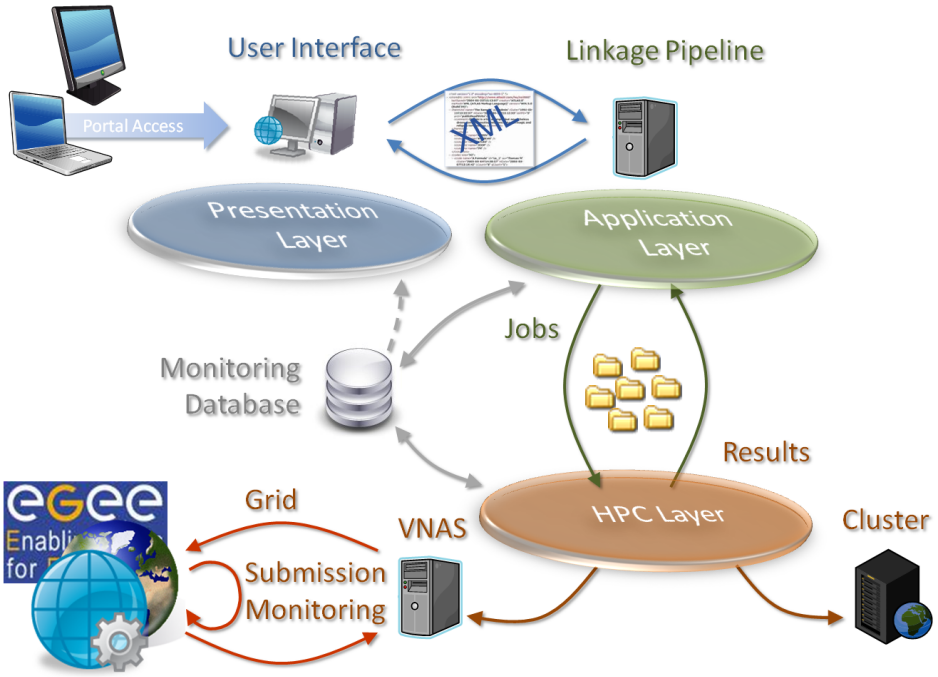


Figure 5.3: Diagram of system"s design.

checking to output formatting. Our system was designed as a web application with domain oriented features and computational facilities for pedigree plotting, linkage analysis and pipeline configuration.

At the lowest layer, our system implements the heuristic method by splitting inputs and running parallel jobs (a "challenge") on the Grid environment, transparently to users. During linkage processing, the system monitors grid jobs; when all jobs have reached completion, it retrieves outputs and merges the results of each job into a single file. Finally a unified plot is presented to the user, representing the results of the whole genome wide linkage analysis. Based on this design, the system is configured as a 3-layer application, as summarized in Figure 5.3: presentation, application and HPC layers. In the next paragraphs we will discuss each layer. At the presentation layer users interact with the application which produces an XML file. The application layer parses the XML file to extract the execution logic and linkage parameters. The HPC layer interacts with the Grid middleware submitting jobs and monitoring their execution, and with Cluster environment.

The Presentation Layer

The presentation layer delivers the web based user interface to the user. This was created to aid users to setup linkage analyses while masking the complexity of low level interactions with the Grid middleware. The interface was designed according to the directions of the human-computer interaction discipline (HCI [119]), being focused on the intuitiveness of the controls adopted to lower the users' learning curve. Final users of this application are identified mainly as biologists, not necessarily skilled in informatics, hence, visual techniques are adopted, where possible, to set up the system parameters and options. For example, the creation of the path for data processing from the input files to the retrieving of results is obtained with the on-screen arrangement of visual blocks in a flow-chart fashion. To increase interactivity and user friendliness, the web page makes use of JavaScript as the client-side scripting language, using asynchronous communications with a back end on PHP server pages, in order to manage information exchange with the application layer and handle the file uploads. According to the HCI principle of "User Centered Design", the web interface has been developed with the active participation of a set of effective end users, trying to optimize the interface around their needs and usage praxis. The features of the web interface can be summarized as follow:

- creation of the data pipeline using draggable components representing each operative linkage analysis step: the arrangement of logical blocks, managed by client-side scripting, makes building the data flow a visual matter from the input to the output files;
- customization of each linkage analysis step: choice and upload of the input files, definition of optional data pre-processing steps to adapt inputs to different algorithms requirements, selection of the available algorithms and setting up of the proper parameters;
- launch of the analysis: with the click of a button the system takes charge of the customized challenge and passes the required information to the application Layer;
- status visualization: the challenge completion percentage along with the details about each job status are shown in real time. The information presented in the end user's browser are gathered from the monitoring database, managed at HPC Layer level;
- results delivery to the end user: when the challenge has been completed (i.e. the HPC Layer has retrieved the jobs outputs from the Grid to our server and the application Layer has elaborated the outputs merging the results, as explained in the next paragraphs), unified results are made available to the end user for display in the browser and for downloading to his own computer.

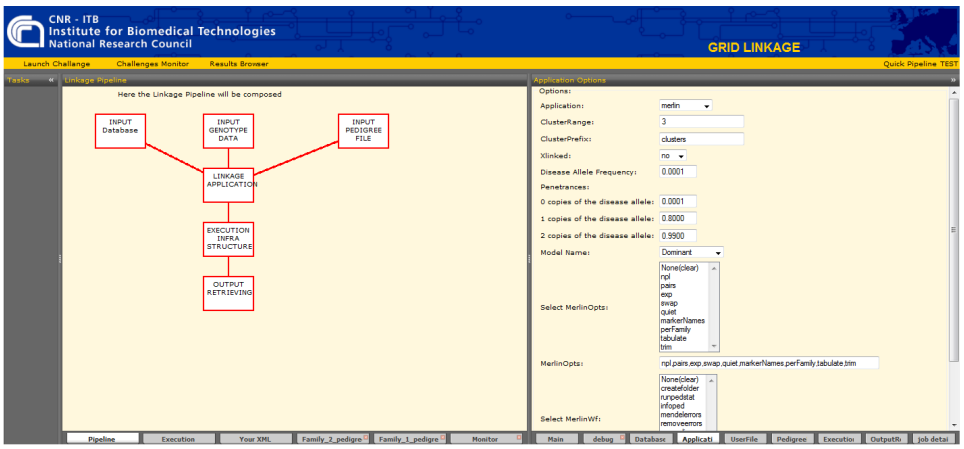


Figure 5.4: Application Web Interface, submission and the challenge pipeline configuration facility screenshot.

The communication between the presentation and the application layer is implemented by the creation of an XML file describing the data pipeline and all of its parameters: this file provides a complete description of the challenge characteristics making successive quick resubmissions possible; the choice of XML language reflects the future possibility for the system to migrate to web services technology, adapting the application Layer to this protocol with a minor effort. The communication in the opposite direction, i.e. between the application and the presentation layer, is obtained with asynchronous data exchanges polled by client scripts: these fetch information from the server back-end and display available information without the need for page reloads. An example of the presentation layer and job status visualization is shown in Figures 5.4 and 5.5.

The Application Layer

The application and the HPC layers provide the grid execution and hide infrastructural complexity to the users. At the application layer the system first parses the XML file created by the presentation layer and extracts all parameters for linkage analysis configuration and grid challenge set up. After that the input data uploading and splitting happens. The uploaded files are managed by the web application server, and this is where the splitting procedure starts. The splitting procedure is based on the workload. Since data analysis requires a wide set of tools and quality filter controls, the splitting procedure is supported as a two-steps procedure: in the first step users set the linkage application parameters and customize filter methods for SNPs selection and linkage disequilibrium modeling; in the second step the algorithm gathers the SNP set the user wants to analyze and splits

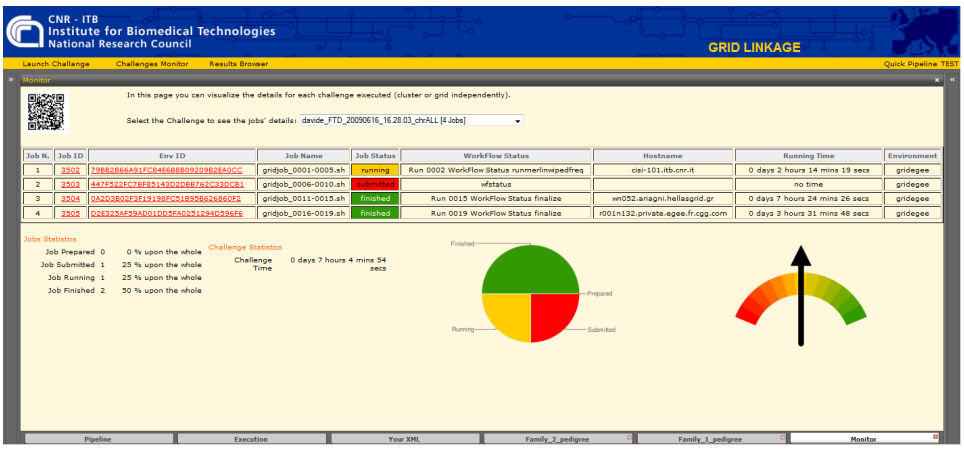


Figure 5.5: Application Web Interface, monitoring screenshot: the challenge status display.

such input set in smaller sets according to the computational cost (adopting preconfigured rules for the best strategy) and the SNP set size.

During the splitting procedure the system creates Grid jobs: one job for each SNP set. When the splitting procedure starts, the application layer records in the monitoring database one entry for the challenge; a similar initialization on the database occurs when each single job is created. Since during each job execution the job is programmed to update its entry in the monitoring database by means of a compiled-in MySQL client, we can efficiently derive the status of all jobs and, from that, the overall challenge status.

Once the splitting procedure has been completed, the application layer leaves ground to the HPC layer for performing the submission and monitoring of jobs to the grid platform, and finally the retrieval of results. When all jobs' results are eventually gathered, they are passed to the application layer for the final procedure for merging results and creating unified plots. These are eventually passed back to the presentation layer.

The Execution Layer

The HPC layer is in charge of submitting the jobs to the Grid or Cluster and actively monitors their execution on the available resources, dealing with jobs failures and resubmissions. When all jobs related to a challenge have been computed, single job outputs are retrieved and passed to the above layer for merging, and eventually made available for downloading through the web interface.

5.1.5 Testing and Performances

To evaluate the effective performance of the proposed system, a series of tests were carried out, including real data analysis and test computations of different durations performed to obtain homogeneous indications about efficiency and reliability. A total number of more than 25,000 jobs have been launched through the resources of the Biomed Virtual Organization onto the EGEE Grid. These have been submitted as a set of challenges, or collections of jobs, each representing a complete analysis on a number of chromosomes, as well as on the whole genome, obtaining an estimate of the Grid infrastructure performance compared with a single 2 GHz CPU workstation and with a Cluster composed by 280 CPU cores. Consistency of results was also verified in the framework of the project "EDGeS - Enabling Desktop Grids for e-Science"[1]; a detailed comparison of these environments is beyond the scope of this paper. As an example of a real case study, an analysis on Atrial Flutter has been performed and the achieved results are presented hereafter.

Grid Execution Performances

The test analyses comprised pedigrees composed of 38 subjects, including individuals genotyped with different genotyping chips of markers from 10k up to 1 million SNPs each. For some individuals a subset of the available marker dataset was used to simulate a smaller chip and this gave an indication about the scalability of our approach. The software adopted to carry out the test was the Merlin application. Considering the trade-off between CPU load and memory requirements, a subset of 50 SNPs was evaluated as the optimal workload for a Grid node and was assigned to each program run; the total number of runs needed to process all SNP data produced by each genotyping chip was split into jobs with an estimated duration of around 6 hours each on a mid-range multi-core CPU. Once a sample challenge for each size of the available chips was generated, the collections of jobs were submitted twelve times at varying dates and times to consolidate the significance of the performance results.

The database for the monitoring system was also used to get accurate timings of job execution. In order to obtain the exact duration of each job execution during all phases of its life (submission, queuing, execution on the remote worker node and delivery of results) we logged into the monitoring database the *datetime* of all the connections made by every job to the monitoring database itself.

Test results are summarized as follows: Table 5.1 shows the duration of the challenges in hours and Figure 5.6 shows the relative plot: the values

[1]Home page link: www.edges-grid.eu

relative to the Grid infrastructure, depicted as black dots in the graph, represent the median value for each pool of challenges submitted; the ones relative to the single CPU, estimated on the basis of a single job duration, is linear to the number of jobs as expected from sequential running procedure, while the cluster series is obtained from previous computations and shows a saturation trend. The two leftmost columns show the challenges characteristics; columns 3, 4, 5 show the challenges durations expressed in hours as resulted for the different environments; finally, columns 6 and 7 show the speedup relative to the sequential execution. The Grid results are median values in a pool of 12 replicas; the single CPU values are estimated assuming linear trend of computational time. Comparing the results of the different computation infrastructures we can see that distributed analysis pipelines with big datasets, i.e. with a high number of linkage variables, achieved a speedup up to more than 72x compared to a mid range dual-core 2 GHz CPU execution; the speedup of the cluster is obviously higher, even if, in spite of the theoretical number of cores, its efficiency is only three times higher than the Grid. This is explained by the fact that this is not a dedicated cluster and not all the nodes were accessible during the test.

Table 5.1: Data derived from different genotyping chips were analyzed with the Merlin software using 3 computational infrastructures: our Grid-based system, a 70-node/280-core Cluster and a single 2GHz CPU Workstation (estimated time).

Genotyping Chip	**Jobs (6h)**	**Computational Cost (h)**			**Speedup**	
		Single CPU	**Cluster**	**Grid**	**Cluster**	**Grid**
10 k SNPs	6	33	8	19	4.1	1.7
66 k SNPs	35	220	9.5	28	23.2	7.9
100 k SNPs	60	333	10	30	33.3	11.1
317 k SNPs	172	1056	13	38	81.2	27.8
370 k SNPs	206	1233	15	39	82.2	31.6
500 k SNPs	278	1665	16	42	104.1	39.6
670 k SNPs	373	2233	18	42	124.1	53.2
1 M SNPs	556	3332	20	46	166.6	72.4

The tests also demonstrate that the average performance of the proposed system exhibits a trend comparable to the cluster, due to the similar workload distribution technique adopted (not MPI). Considering marker chips greater than 100 k, the advantage of both of the distributed architectures gets proportionally bigger compared to the single CPU, due to the difference between linear increase of computational time for the sequential run and the saturation trend of the parallelized data flow obtained distributing the workload on the computing elements.

Figure 5.6 shows the overall lasting times for 96 challenges (12 for each

size from 10 to 556 jobs of similar duration) were considered to obtain this graph: boxes represent 25th and 75th percentile, whiskers represent minimum and maximum, dots indicate the median values. Red line shows the estimated performances of a single 2 GHz CPU, blue line is for Cluster. It can also be noted that with the increase of the challenge size (number of jobs) the variability in the total execution time gets higher, reaching a plateau for a jobs number above 250. This is probably due to the higher probability of getting some of them stuck or lost, with the need of resubmission by the VNAS component of the system, and to the reaching of a balance between the resubmission rate and the loss of jobs.

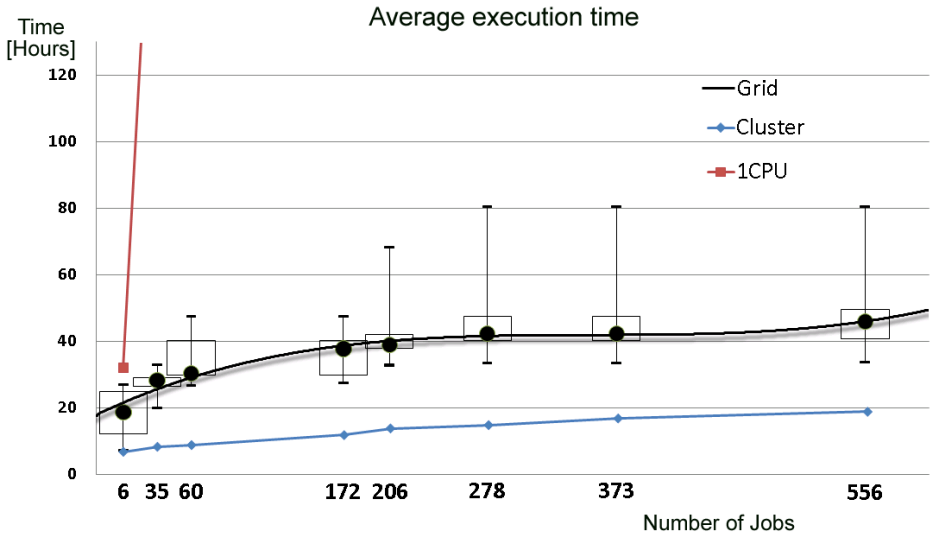


Figure 5.6: Linkage Analysis Challenges average duration.

Analyzing the distribution of the total execution times for a pool of 10,500 jobs with a running time of about 3 hours (Figure 5.7), it can be noticed that more than half of the jobs are completed in less than 10% of the total completion time of the challenge they belong to: tails of few long lasting jobs heavily affect the performances of the proposed system, and more generally those of the Grid environment. This confirms that the total job execution time and the overall challenge duration as a consequence, is dominated by the computing element's register and queuing times and the worker node's execution times. This observation may lead to the development of some kind of "unfair" counter-measures to get rid of the problem, such as the duplication and parallel multiple resubmission of the last queued jobs [104], but this would lead to some overhead in the Grid resources usage and was intentionally not implemented in our present work.

The large variability of the queuing system of the Grid environment may be also seen by examining the central box in Figure 5.8. This Figure shows

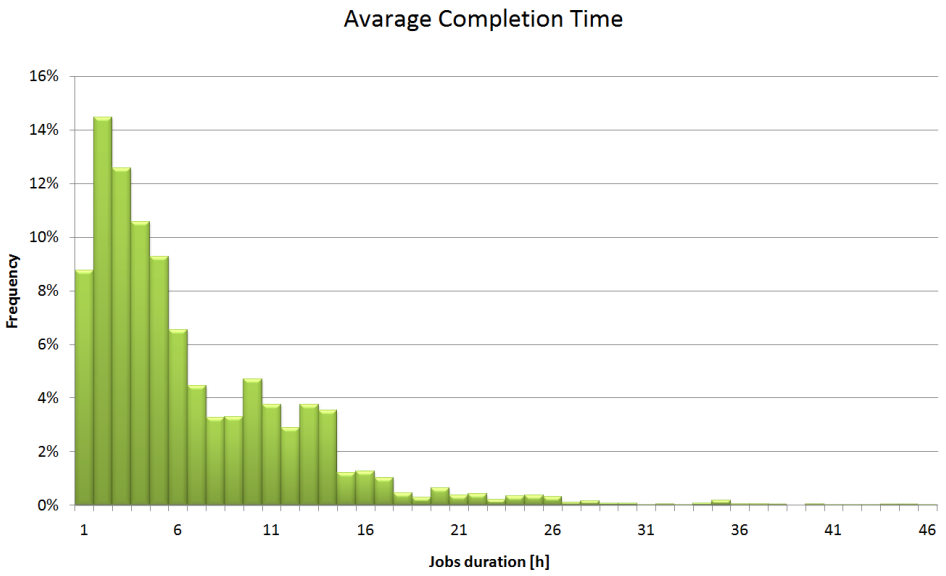


Figure 5.7: The distribution of total execution times for a pool of 10,500 jobs with a running time of about 3 hours: the duration of each job is normalized with the total completion time of the relative challenge.

the components in time of the average life span of a Grid job characterized by a running time of about 3 hours on a 2 GHz multi-core CPU and being part of a medium size challenge (nearly 200 jobs). Boxes represent 25th and 75th percentile, whiskers represent minimum and maximum, dots indicate the median values. The first two elements at the left hand side represent data formatting and jobs submission time spent by our application. These increase linearly with the challenge size even if some efforts were spent to optimize the parallelization of the whole process, using a multi-threaded algorithm to submit the jobs to a list of known-good Grid WMS in parallel, so to increase the submission rate.

Observations may be made on the great variability of the performances of computing resources in Grid, or at least those accessible within the Biomed V.O. In Table 2 we report the results of a quick computational routine used to benchmark the available working nodes. The top performers and the worst elements may differ by a factor 50 or even more and jobs that are assigned to very slow machines can either fail (best case: triggering a subsequent resubmission), or otherwise run so slowly that they effectively negatively affect the overall completion time of the entire challenge.

The performed tests show that the computation efficiency of the distributed approach of our Grid-based system grows with the size of the challenge, being still lower than a dedicated Cluster but much higher than that of a sequential run on a single processing unit. It must also be highlighted

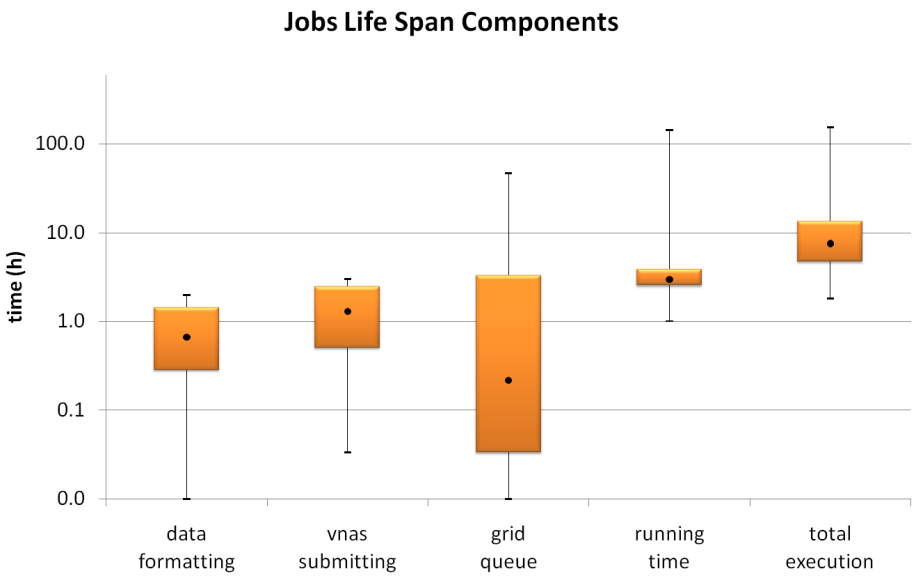


Figure 5.8: The duration of each component of the life cycle of the same pool of jobs.

that computations with a higher number of individuals in the pedigree tree, that may result infeasible on the desktop workstation due to long and non affordable completion time or memory overflow, can be performed on the distributed infrastructure with the proposed splitting approach and to the large and globally high performing (considering memory and computing power) set of resources available in the EGEE Grid.

5.1.6 Validation

To confirm the effectiveness of the proposed approach, an analysis on Atrial Flutter data was performed in the context of a joint research activity within our institute. Atrial flutter (AFL) relates to cardiac conduction disorders, a multifactorial pathology which can be caused by a combination of factors such as developmental and congenital defects, acquired injury or ischemia and/or portions of the conduction system, or less commonly due to inherited diseases altering the functioning of the cardiac conduction system [120].

We collected the genotypes of an isolated family to understand how the monogenic disease can be associated to a specific SNP marker. The patients were genotyped using the Infinium II Assay-HumanHap BeadChip 370k SNPs. The pedigree structure of the family is depicted in Figure 5.9, which is produced by our system on the basis of the pedigree data: circles refer to female sex and boxes to male, while the affected status is represented by the coloring scheme: black-filled for affected, empty-filled for healthy, and gray-filled with a question mark for unknown phenotype. Using the web in-

Table 5.2: Good and the Bad Nodes report: top ten of the fastest (left) and slowest (right) worker nodes exploited in a computation benchmark quick test (17 minutes duration on a dual core 2.5 GHz CPU; S_p is speedup value). Full domain names of the nodes were deliberately anonymized.

Good Nodes			***Bad Nodes***		
Node Name	Running **t [min]**	S_p	S_p	Running **t [min]**	**Node Name**
wn11.***	4	4.3	0.16	106	wn-104-15-32-a.***
grid-wn0474.***	5	3.4	0.16	110	farm009.***
grid-wn0545.***	7	2.6	0.15	111	wn-104-(...)04-a.***
node242.***	7	2.4	0.14	120	wn228.***
sbgwn77.***	7	2.4	0.14	122	node290.***
wn-204-(..)13a.***	7	2.4	0.13	127	farm022.***
grid022.***	7	2.4	0.09	182	wn-104-(...)14-a.***
lcg0857.***	8	2.1	0.08	215	bohr2219.***
n31.***	8	2.1	0.04	392	grid020.***
n33.***	8	2.1	0.04	417	bohr4918.***

terface, we configured the pipeline and set each linkage parameter. All Grid related parameters, such as the number of jobs, were automatically set by the system, hiding grid complexities to the end user. We set parameters for data quality control adopting strategies such as error detection and removal. Merlin was used to search for Mendelian inconsistencies within each of the SNP sets, discarding genotypes that gave contradictory information about gene inheritance in the pedigree.

Moreover we set the application to perform markers filtering and linkage disequilibrium modeling [114, 115] setting up the pre-processing linkage analysis pipeline through the web interface within the linkage box. An inter-marker linkage disequilibrium analysis (LD) was performed in our dataset in order to test the inflation of the nonparametric multipoint LOD score. Then, a strategy for SNP selection based on allele frequencies (MAF) estimations combined with pair-wise correlation coefficient r^2 calculations have been applied; we then modeled the Linkage Disequilibrium (LD) and we obtained 12173 markers.

Based on AFL use case disease model and pedigree structure, we parameterized the specific linkage analysis options that would highly affect the overall work load and on which the splitting procedure would divide input set in smaller ones. We set up two analyses: non-parametric and parametric. Multipoint non-parametric linkage analyses (NPL all statistic) for SNP markers was configured to run Merlin, computing SNP clustering by similarity. For the parametric linkage analysis (PLA) a dominant model with a susceptibility allele with population frequency of 0.001 has been assumed,

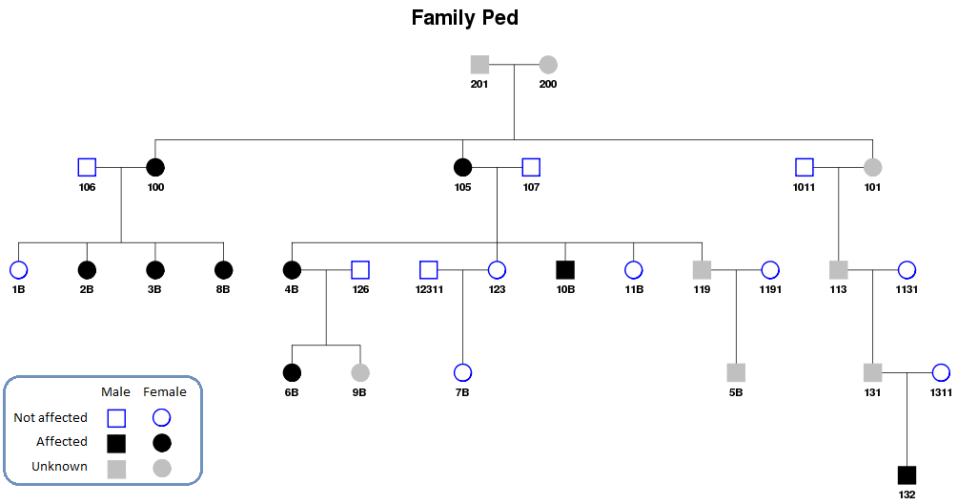


Figure 5.9: The pedigree of the family analyzed for the Atrial Flutter disease.

modeling recombination with Identical-By-Descent schema. These parameters increased the computational cost by a linear factor of 3 and, given these configuration parameters, the splitting procedure produced 305 SNP sets containing 40 SNPs each: each job had a running time on a 2.8 GHz CPU of about 1.20 hours. On the Grid environment we run each job in parallel exploiting our linkage analysis approach.

Examining logs generated by our application we found that 73 jobs were resubmitted for worker node problems, such as maximum queue time exceeded (a parameter in VNAS) or error exit status.
The challenge execution time was divided as follows: 16% during pre-processing operations and splitting procedure running, 3% of job submission/resubmission, 41 of queuing time, 34% of computational time, 5% of retrieving output operations, 1% of output post-processing time.

We got the results in about 36 hours, demonstrating a real gain using a distributed environment, such as the Grid. Once the jobs finished and all results were merged together, we found two candidate regions within the chromosomes 1 and 19, as showed in Figure 5.10. The next biological step will be the sequencing for the two regions.

Using the system described in this work it was possible to perform a whole genome data analysis for genetic linkage without an expensive dedicated computational infrastructure by exploiting the remote resources of the EGEE Grid. The next phase in this research will be to confirm the association results depicted above.

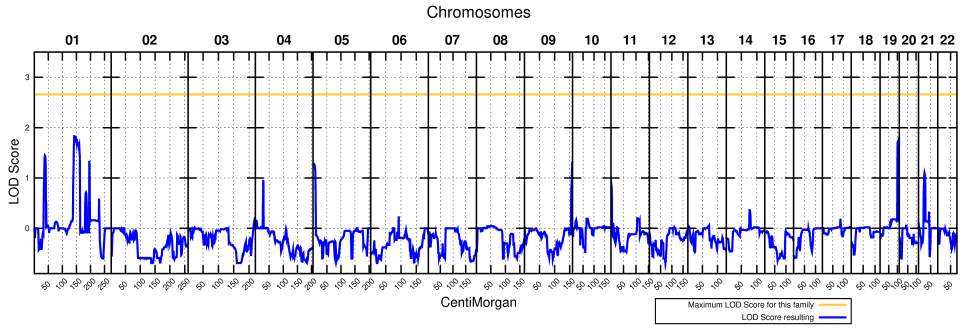


Figure 5.10: The LOD Score of the whole genome analysis for Atrial Flutter adopting the heuristic approach on the distributed grid environment.

5.2 SNP Ranker

5.2.1 Introduction

In the recent past years, genotyping technologies knew a great development and studies about genotype markers are increasing in importance [121, 122]. Among them, the evaluation of SNPs is revealing very promising. SNPs are nowadays widely exploited for Genome Wide Association Studies (GWAS) [123] [124] [125], identification of Copy Number Variations (CNV) [126], observations of Population Stratification [127] and so forth. Since SNPs represent established genomic differences, their knowledge can be exploited to characterize each subject from others on study by correlating specific phenotype with a corresponding genomic pattern.
The total number of SNPs in the whole human genome exceeds 12 millions and each SNP is related to different genomic properties depending on its position on the DNA strand: i.e. a SNP can be located within inter-gene regions or within intra-gene ones. Nowadays chip technologies allow to analyze up to one million SNPs for each patient due to chemical and physical limits that affect probe density. To overcome this limit, together with technological improvements, researchers are trying to define reasonable strategies to filter the initial amount of 12 millions SNPs.
The first approach to optimize the SNPs probeset relies on the concept of Linkage Disequilibrium [128] (LD). LD mapping exploits a statistical similarity measure between adjacent SNPs and computes how much two SNPs are related each other, thus defining what is the genetic information improvement using both or just one of them. LD mapping is thus used to optimize the information contained into 1 million SNPs arrays. The second method able to reduce the number of SNP probes within a chip regards the possibility to create disease-oriented chip. This approach not only allows adapting the analysis to specific genetic studies but even to produce smaller arrays. In fact, this aspect relies on a crucial topic: which strategy

can be followed to select the subset of SNPs suitable to create a specific disease-oriented chip.

This section is related to SNPs' probeset identification for producing genotyping arrays dedicated to pathologies, starting from genes or biological processes involved in such diseases. The implemented tool scores different biomolecular features for SNPs associated to a set of genes, that is given as input and that can be expanded through an ontology-based engine. Once the SNPs final scores are computed, the system provides a ranked list of the most significant SNPs associated to the input set of genes.
Moreover *SNPRanker* can be used for gene enrichment through the identification of the ontological similarity between the input genes and whole set of human genes. This allows to extend the initial gene list, by including also genes presenting a similar biological function (according to the considered ontology) thus potentially involved in the same disease.

Related Works

The analysis of genomic variations is usually performed following two main approaches: statistical methods or machine learning techniques. The main difference is that while the exploitation of statistical methods requires a data model involving a set of a-priori hypotheses and parameters values, the use of machine learning approaches do not need a-priori evaluations, since models and rules are derived directly from a training set of data and the system is trained to fit a general model which will be then adopted for all other data. An alternative solution is offered by data mining approaches where users can visualize, plot and reorder data without fixed models and, if the system supports customizations, they can also validate their own new models on data and infer new knowledge.
Statistical methods are widely used in epidemiology and got many positive results in application studies [129, 130]. Nevertheless, statistical approaches are often computationally intensive, especially when dealing with large amount of data produced by high-throughput techniques, and this approach is often impracticable for most of the research laboratories. Therefore, many scientists found machine learning approaches very attractive; so far many studies exist exploiting machine learning approaches and are providing encouraging results [131, 132].
Machine learning methods are among the most promising approaches due to the flexibility and adaptation to data. When exploiting *supervised* methods, the training set must be carefully selected, since the model is created on it: the training set must therefore embed all peculiarities of the considered data type, thus allowing the model a-priori knowledge. In our case data are single nucleotide genomic variations (SNPs) and the a-priori knowledge is represented by features that characterize each SNP or the related gene and protein.

Data mining methods are mostly employed in business fields, for example for intelligent customer support and business analyses. In the genetic context a few significant works have been produced, for example in [123] the authors mine SNPs from families; we need a tool for scoring SNPs based on a priori information and where users can infer knowledge by setting parameters, like data mining facilities usually support.
Only a few applications exploit machine learning in genotyping context. An example, concerning the genes ranking, is the so called *gene prioritization* [73], a method that, given a set of training genes, considers a number of features from them, which represent the a-priori knowledge available from multiple data sources. Given a set of test genes, the cited system computes features values and ranks test genes with respect to their similarities, achieving the prioritization and highlighting the most important genes, with respect to the selected features.

No methods are available in literature to achieve, in SNPs context, results similar to gene prioritization through data mining and machine learning approach. For this reason, we designed and implemented a new method for evaluating SNPs by features scoring.

5.2.2 Methods

To better understand the motivations of SNP Ranker, we reported in the panel 5.11 the idea of the system. What users usually have while dealing with annotations is a large matrix with a set of properties. An example is represented by the part A of Figure 5.11 where a user retrieved three genes and the included SNPs, three for Gene A, five for Gene B and two for Gene C. For each SNP, users selected a set of annotations, here five ones. Now, if a researcher would rank SNPs, how should do it? How can he choose the most relevant SNPs? We first answered to this question through a data integration approach and then finding a valid conversion for all annotations into a numerical format.

The first step is the numeric format conversion: for each annotation of a SNP we use a function to map the textual annotation into a numeric value. The part B of the panel shows the original annotation table with numerical format. Dealing with numbers gives us the way to use custom data representation; as an example is also possible to extract radar plots from the table in part B and get a Gene representation, as shown in part C of the panel.

Now that we have all annotations as numbers, we can design a new method to mine new knowledge from data and rank SNPs based on user parameters. How to support this activity? Our idea is to assign a weight vector to each feature and finally combine all features for each SNP. The idea of the weight vector id represented in the part D of the panel, the violet column. This weight is applied to all features for each SNP. Finally, part

E of the panel, we combine all data with a general scoring function and aggregate values. The simplest idea could be to sum values. Once we get a single score for each SNP, we can easily rank SNPs, both within each single gene or across all genes.

The core of the designed system can be decoupled into three levels:

- *features set*: choice of the features characterizing each SNP;
- *evaluation function*: definition of the function that provides a final score for each SNP;
- *gene list enrichment*: exploitation of gene ontological annotation to enrich the initial list of genes provided as input.

Following subsections present in details the levels mentioned above and the related characteristics of the implemented prototype.

Features set

The term "features" indicates a set of characteristics, related to each SNP, the machine learning approach relies on, in a direct way or through the gene and the gene product knowledge. The set of features chosen to characterize each SNP represents an a-priori knowledge of the scientific problem.
Since the described approach considers as input a list of genes, the system must consider biomolecular elements as related to genes in a *vertical* perspective. Therefore information is considered in both a top-down view, from proteins interactions to sequences of nucleotides, and a bottom-up perspective, starting from genes characterization and climbing up to processes and complex biological systems.
The features set includes genomic (Minor Allele Frequencies, localization on the DNA sequence), proteomic (InterPro [82] domains), interactomic (hub proteins), phenotypic information (essential genes).
In the following, a complete list of the considered features will be described.

Hub proteins The evaluation of the number of protein-protein interactions (PPIs) established by a protein, i.e. the degree of the protein in the PPIs network, is an important aspect in assessing the biological relevance of a SNP occurring within or close to the gene that encodes for that protein. Indeed, it is known that *hub* proteins play a crucial role for the cell functioning and, in fact, are often encoded by essential genes. The database used in this work integrates PPIs data from HPRD [133] and BioGRID. By means of this information it is possible to score a SNP (x) associated to a gene with the following characteristic function, which considers the number

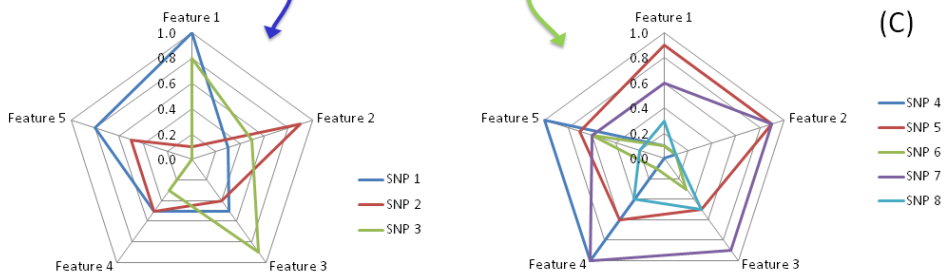
(A)

	Gene A			Gene B					Gene C	
Annotations	SNP 1	SNP 2	SNP 3	SNP 4	SNP 5	SNP 6	SNP 7	SNP 8	SNP 9	SNP 10
Gene Feature	OneVal	OneVal	OneVal	OtherV	OtherV	OtherV	OtherV	OtherV	LastV	LastV
Protein Feature	A+B	A+B	A+B	C*D	C*D	C*D	C*D	C*D	F	F
SNP Feature 1	0.5	0.4	0.9	0.0	0.5	0.3	0.9	0.5	0.2	0.8
SNP Feature 2	2.0	6.0	7.0	1.0	8.0	5.0	76.0	3.0	22.0	5.0
SNP Localization	coding	intron	intron	flanking	condig	intron	flanking	flanking	intron	coding

(B)

	Gene A			Gene B					Gene C	
Annotations	SNP 1	SNP 2	SNP 3	SNP 4	SNP 5	SNP 6	SNP 7	SNP 8	SNP 9	SNP 10
Feature 1	1.0	0.1	0.8	0.1	0.9	0.1	0.6	0.3	0.4	0.8
Feature 2	0.3	0.9	0.5	0.1	0.9	0.1	0.9	0.1	0.5	0.6
Feature 3	0.5	0.4	0.9	0.0	0.5	0.3	0.9	0.5	0.2	0.8
Feature 4	0.5	0.5	0.3	1.0	0.6	0.1	1.0	0.4	0.9	0.3
Feature 5	0.8	0.5	0.0	1.0	0.7	0.6	0.6	0.2	1.0	0.6

(C)



(D)

	Gene A			Gene B					Gene C		
Annotations	SNP 1	SNP 2	SNP 3	SNP 4	SNP 5	SNP 6	SNP 7	SNP 8	SNP 9	SNP 10	Weights Vector
Feature 1	0.2	0.0	0.2	0.0	0.2	0.0	0.1	0.1	0.1	0.2	0.2
Feature 2	0.1	0.4	0.2	0.0	0.4	0.0	0.4	0.0	0.2	0.2	0.4
Feature 3	0.5	0.4	0.9	0.0	0.5	0.3	0.9	0.5	0.2	0.8	1
Feature 4	0.5	0.5	0.3	1.0	0.6	0.1	1.0	0.4	0.9	0.3	1
Feature 5	0.5	0.3	0.0	0.6	0.4	0.4	0.4	0.1	0.6	0.4	0.6

(E)

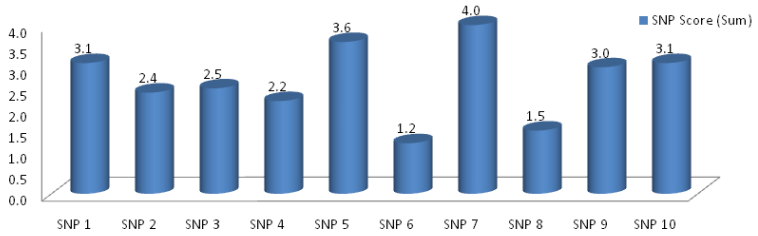


Figure 5.11: The SNP Ranker idea. A panel with generic description and motivation.

of PPIs k_x established by the protein encoded in the gene:

$$f_1(x) = \begin{cases} 1, & k_x \geq \alpha \\ 0, & k_x < \alpha \end{cases}$$

where α is the number of PPIs requested to be considered as hub. By default $\alpha = 20$, according to [134]: through the web interface the user is able to modify this value.

Protein domain A SNP can create a missense or even a frame shift in the coding sequence, thus causing changes in the protein amino acid sequence, which can have a deep impact on the protein function according to the region where the modification occurs. In fact, if a change is localized within a domain, thus being functionally important, its effect on the biological function is potentially more relevant than if it is placed within a inter-domain region.
Information related to the domain localization of the SNPs can be obtaining by linking InterPro Domain Architecture (IDA) data (which report the localization of the protein domain according to the amino acid position within the protein chain) with the knowledge concerning the amino acid position (which indicates the position of the amino acid modified by the SNP). This information can be easily accessed from the integrative database.
The score of this feature is provided by a characteristic function, which applies greater values to SNPs occurring into protein domains (D) encoding regions.

$$f_2(x) = \begin{cases} 1, & x \in D \\ 0, & x \notin D \end{cases}$$

Minor Allele Frequencies The Minor Allele Frequency (MAF) represents the frequency of the less frequent allele of a SNP in a specific population: it defines how much an allele (and thus its SNP) is relevant for a population. MAF score can even be employed to measure the "penetrance" of a disease in a population, in case the minor allele is more diffused in the affected phenotype than in the unaffected one in the control population.
Since we are designing a general system for different studies and applications, all populations considered in the HapMap project [135] have been included, with their MAF values. The user is required to choose the MAF of interest.
For this feature, the scoring function is expressed as the original MAF value $m_x \in \Re | (0 \leq m_x < 0.5)$ obtained from the HapMap project:

$$f_3(x) = m_x.$$

When the "1000 Genomes Project" [136] will provide more accurate and updated values about MAF scores, we will update these values.

Localization on the DNA sequence According to the annotation provided by dbSNP [79], functional relationship between SNPs (and possibly alleles) and genes are defined. Relying on this annotation the following categories have been considered $C = \{$ "unknown", "coding-synonymous", "intron", "near-gene 3"', "near-gene 5"', "nonsense", "missense", "frameshift", "untranslated 3"', "untranslated 5"'$\}$.
According to the user's specific analysis the whole set C or subsets of it can be considered. Considering the vector $\mathbf{c} = (c_1, ..., c_10)$, whose elements $c_i \in 0, 1$ indicate the selection or the exclusion of the elements of C, the scoring function can be formalized as follows:

$$f_4(x) = \mathbf{c}^T \times \mathbf{v}$$

where $\mathbf{v}$ is a weight vector.

Essential genes Another important feature for SNP scoring is the kind of gene where the polymorphism occurs. It is known that some genes are *essential* to support cellular life, i.e. if their products are not correctly produced the cell hardly survives. The knowledge base used in this work includes data providing such information for *homo sapiens* [137]. Considering the set of human genes G and the subset of essential genes $E \subset G$, the scoring function is the following characteristic function:

$$f_5(x) = \begin{cases} 1, & x \in E \\ 0, & x \in G - E \end{cases}$$

Core scoring function

In order to obtain a significant score for each SNP, features' values must be processed through a *core scoring function*. This engine allows to compute the final SNP value as a real number, considering genes and genes' products information embedded in the defined set of features.
Given the a-priori knowledge embedded in the described set of features, the user can interact with this information in order to better adapt it to his scientific studies: this is possible by associating each feature to a *weight* that represents the importance of that feature for the calculation of the final SNP score. The default values for all the elements in the vector $\vec{w}$ of the features appears in the tool web page: this model assigns the same importance to each feature, without assuming any specific perspective while performing the analysis. The scoring function g maps the values returned

by the features scoring functions $f_1, \ldots, f_5$ and the weights vector $\mathbf{w} \in \mathbb{R}^5$ to a single final value, which is used to calculate the final SNPs ranked list:

$$g : \mathbb{B} \times \mathbb{B} \times \mathbb{R} \times \mathbb{R} \times \mathbb{B} \times \mathbb{R}^5 \to \mathbb{R}$$

where $\mathbb{B} = \{0, 1\}$. The scoring function g has been defined in two forms; as the sum 5.1 or the product 5.2 (which determines a more restrictive SNPs selection) of the values returned by f_1, f_2, f_3, f_4, f_5 according to the weights $\mathbf{w}$

$$g : (f_1, f_2, f_3, f_4, f_5, \mathbf{w}) \mapsto w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4 + w_5 f_5 \tag{5.1}$$

$$g : (f_1, f_2, f_3, f_4, f_5, \mathbf{w}) \mapsto w_1 f_1 \times w_2 f_2 \times w_3 f_3 \times w_4 f_4 \times w_5 f_5 \tag{5.2}$$

Gene list enrichment

Ontologies are controlled vocabularies hierarchically organized. Their exploitation allows not only the use of standardized and recognized descriptive terms, but even to infer relations among objects that are annotated through ontologies. In the implemented system the ontology layer is exploited, other than for annotation aims, to enrich the list of genes provided to the system as input. From the input genes list g_1 the system generates the list $g_2 \supseteq g_1$, which includes also the genes that present a high semantic similarity with the genes in g_1. Four main methods exist in literature to carry on this task: three methods [138, 139, 140] determine the semantic similarities of two terms based on their distances to the closest common ancestor term and/or the annotation statistics of their common ancestor terms.
A crucial drawback of these methods is that the distances to the closest common ancestor term cannot accurately represent the semantic difference of two terms: if two terms sharing the same parent are near the root of the ontology, thus being more general and less informative, they should have larger semantic difference than two terms having the same parent and being far away from the root of the ontology. Moreover, measuring the semantic similarity of two ontological terms based only on the number of common ancestor terms cannot discern the semantic contributions of the ancestor terms to these two specific terms. The fourth method [141] evaluates these limits and provides an alternative solution for measuring ontological terms similarity: the measure is based on the graph of the considered ontology.
Within the proposed system, the [141] and the [140] strategies have been implemented, thus providing the user the possibility to choose the preferred method.

All the requirements led to design the system as Figure 5.12 shows. In the upper part of the Figure there are data sources that we integrated. The

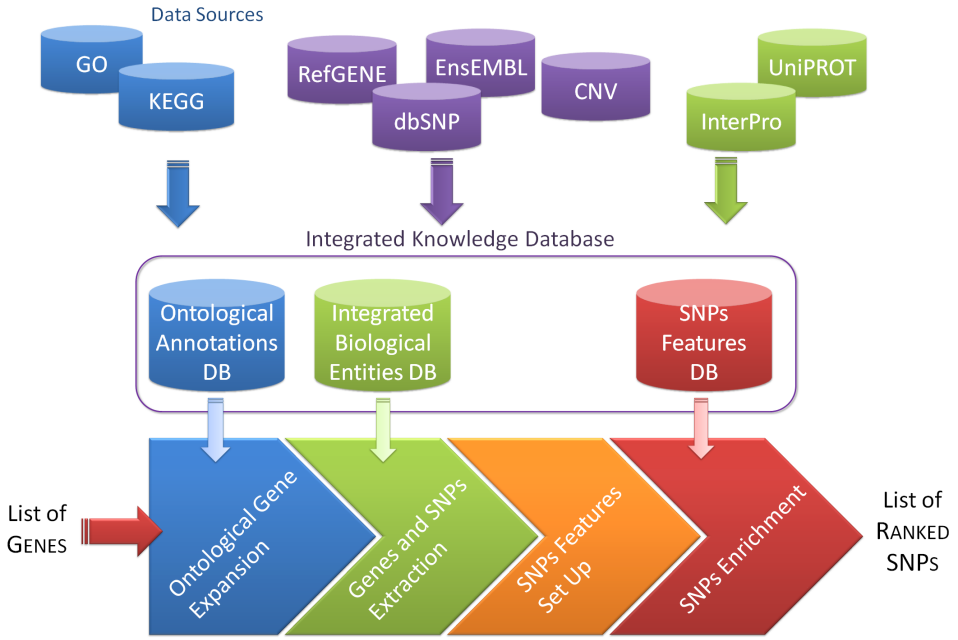


Figure 5.12: SNP Ranker: system design and process visualization.

method and the processing pipeline is represented with arrow boxes, from the input step to the output of a list of ranked SNPs. Deeper details of internal activities are discussed in the next sections.

5.2.3 Interaction with Framework

Figure 5.13 contextualizes how the SNP Ranker application interacts with the framework.

SNP Ranker born to support users on SNP mining as a web application. For this reason, in the SNP Ranker use case users directly interact with web interface too, and they do not directly access genotyping data or knowledge data. Thus, in the biological space as Figure 5.13 shows, users have a directed arrow to the biological problem and through the SNP Ranker interface they can access annotation and knowledge data, the orange arrow to the red block. In the next release of the system, users will also exploit experimental data, this is the meaning of the dashed arrow to the blue box "Experimental data".

The box "Knowledge data" is the entry and the results' visualization point to the system and thus to the *framework space*. Once users customize parameters for the mining activity in the "Knowledge data" box, the system passes to the "Computational layer" where it processes data and finally gets results back to "Knowledge data" interface; this is the meaning of the solid double violet arrow between these two blocks. The other dashed violet

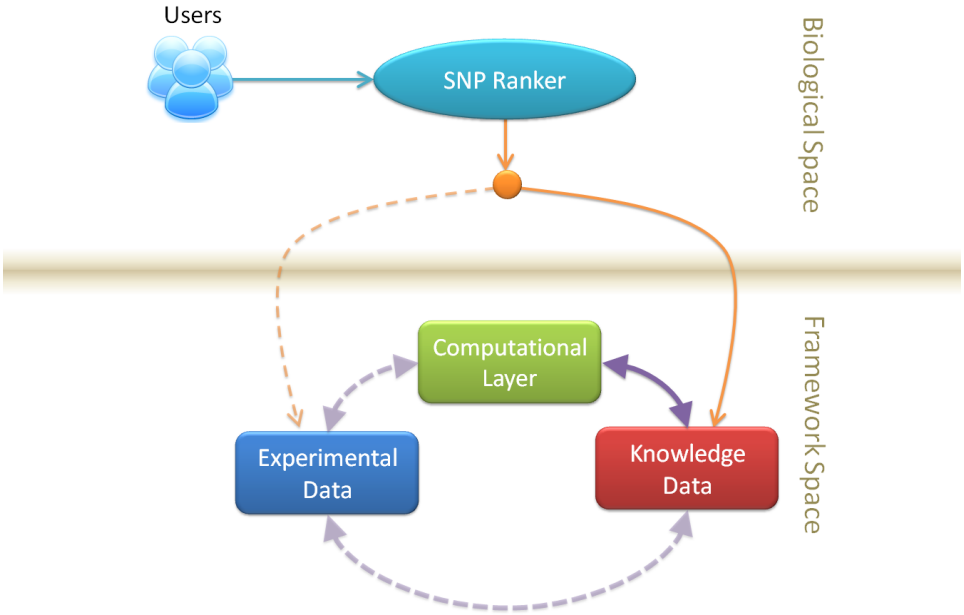


Figure 5.13: SNP Ranker interaction with the framework.

transparent arrows among the three blocks represent next features of the new release to be implemented: since we will include also experimental data, such information will be joined to "knowledge data" (lower violet arrow) and will be processed in the "computational layer" (left violet arrow).

Since the framework has been designed by three conceptual steps, we summarize here how the use case exploits each layer:

- *Vertical Integration*, related to the *Knowledge data* block, represents the core of the SNP Ranker system since all data are annotation about SNPs; annotations come from our internal database and cover genes, proteins, SNPs and systems;
- *Horizontal Integration*, related to *Experimental data* block, has not yet been included here;
- *Computational Layer*, related to *Computational layer*, is related to data processing. For SNP Ranker we did not needed high computational infrastructures, so far a single server with high performances is enough, thus we only adopted a single server solution; under the same assumption, is not needed to write a performance section on the computational layer. Since next version of the system will deal with experimental data we planned to use also Grid EGEE technology.

5.2.4 Application Implementation

The schema of the designed system is presented in Figure 5.14. The system

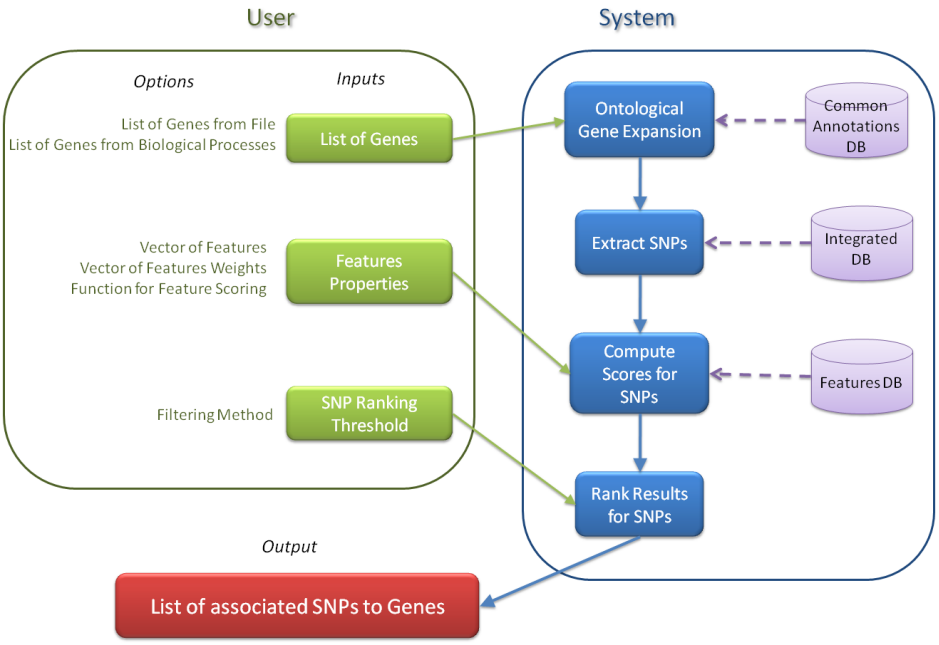


Figure 5.14: General schema of the system

guarantees the flexibility and suitability to users scientific applications by allowing several parameters customizations that enable features set up values to better test hypotheses.
User can access the system, provide the input list, modify system options and retrieve the final results through a PHP and JavaScript based web interface. Available options will be widely explained in next paragraphs and some screen-shots of the developed web site will be shown.

System input

The system takes as input an arbitrary list of genes. The dataset can arise from experimental sources (genes of interest originated from a laboratory experiment or from bibliographic research related to a specific scientific aspect): it can be provided as a list of comma separated standard gene symbols. Alternatively, the set of genes can be retrieved by considering the Gene Ontology for a specific biological process of interest, thus obtaining all genes annotated with the defined GO Biological Process term. Process selection is supported by an auto-completion JavaScript function.

Ontology-based expansion

In order to promote the identification of new SNPs relations and to define a custom "model" for scoring SNPs that better allows data mining and new hypotheses formulation about SNPs influence, especially on genetic pathologies, a crucial function is available within the system, which enriches the input list g_1 by adding new genes that are biologically related with them. This step is performed through the exploitation of ontology similarity measures. Depending on the interests of the user, for each gene in g_1 the system retrieves a number of genes with the highest semantic similarity according to a specific ontology, such as the Gene Ontology, exploiting Wang similarity measure [141] and Schlicker one [140]. Similarity score is retrieved by a pre-calculated matrix, which provides an affinity measure for all couples of genes. Since the matrix creation is computationally intensive, a pre-filtering phase has been applied in order to select couples of genes which present at least one common ontology term.

SNPs extraction

The whole system relies on considering and evaluating data and metadata concerning the biomolecular building blocks. In order to obtain the SNPs associated to the enriched list of input genes, the system performs multiple SQL queries on the database. The output of this step is the list of SNPs associated to the input genes, scored but unranked. Since genomic coordinates and thus the definitions of genes can change during assembly upgrades, we keep track of historical data and users can specify the dbSNP release on which query the system.

Score computation

The function exploited to obtain the final score associated to each SNP relying on a-priori knowledge has been introduced in section 5.2.2. From the application point of view, the user can widely interact with the system to perform this step. First of all the user can specify which features to adopt for SNPs scoring: actually he could be interested to consider some characteristics while neglecting others. The default condition is considering all the listed gene and gene products features. Moreover, user can freely assign different weights to each feature, according to the scientific challenge that has to be faced: in specific context some properties might be more valuable than others. By default all features have a weight assigned by authors on the basis of most common genotyping studies. User can even interact with some thresholds considered in the system. He can freely choose: whether applying the ontological expansion or not and what similarity score threshold to consider; the minimum number of interactions valid to consider a protein

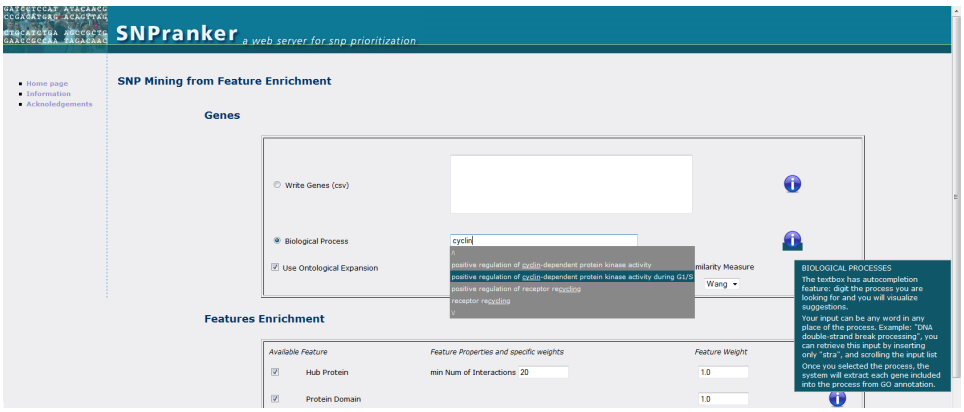


Figure 5.15: Screenshot from the web site. Input gene list definition.

as a hub protein; whether to exploit a sum-based or a multiplication-based scoring function, to provide respectively similar importance to all the chosen features in the computation of the SNPs score or to select just those SNPs that are significant in a specific biological context.

SNPs ranking and Filtering

Once all scores have been computed, the last step consists in ranking all SNPs relying on their score value. Due to the great amount of SNPs potentially reported as output, the user can decide to cut the list. In fact, before running the processing, user can select the percentile where he wants the result list to be cut. Available options are "Percentage" (followed by the corresponding threshold) or "No filters". Final results have to be written in a file and then user can download file through the web link.

5.2.5 Validation and Discussion

SNPRanker is available at the web link http://www.itb.cnr.it/snpranker. The developed system is aimed to support SNPs analysis, particularly interesting for helping SNPs/disease association studies. The input of the system can be either a predefined list of genes, typically whose evidences have been found related to the same pathology, or a set of genes associated to a particular biological process, as shown in Figure 5.15. The ontological expansion is an important tool for studying SNPs related to pathologies, since it allows to extend the analysis to SNPs that could potentially be involved in a pathology onset but that have not being highlighted by more traditional approaches. In fact, this tool permits to increase the number of SNPs in analysis even considering those belonging to genes that present similar semantic annotations with the genes initially considered. For instance,

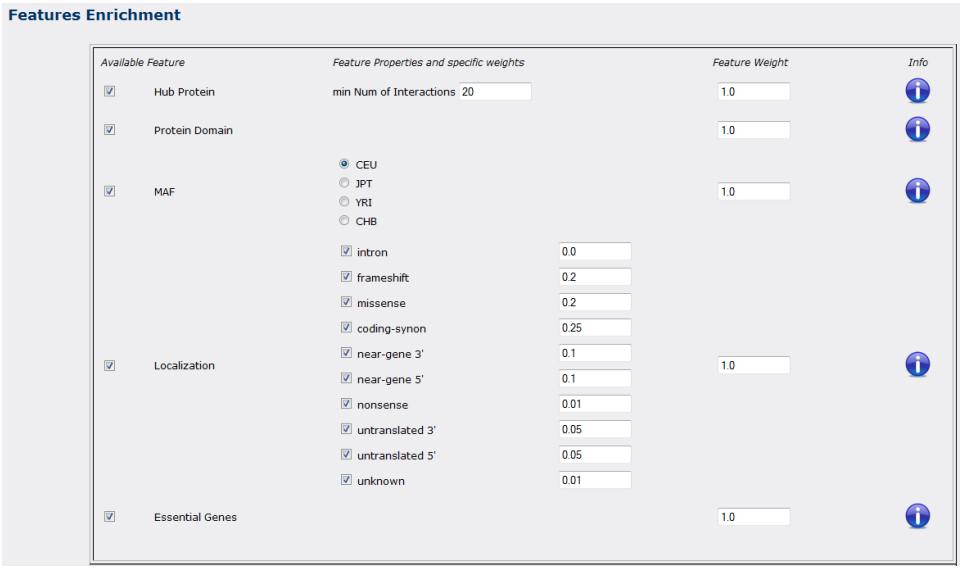


Figure 5.16: Screenshot from the web site. Features selection and weights values setting.

in Table 5.3, we show the top ranked genes showing the highest semantic similarity with the gene *CCND1* encoding for the *cyclin D1*, which controls the cell cycle process. The semantic similarity was calculated by means of the Wang's method [141] considering the Gene Ontology Biological Process. The method successfully identifies the genes annotated similarly to CCND1.

EG ID	EG Symbol	Semantic similarity score
894	CCND2	0.81
896	CCND3	0.81
8941	CDK5R2	0.72
56647	BCCIP	0.72
28984	C13orf15	0.72

Table 5.3: Wang's method [141] semantic similarity scores using the GO Biological Processes: the top 5 ranked genes are listed.

The setting of SNPs feature weights values has been thought as a support for population genetics studies. Actually, this property allows to assign different levels of importance to diverse biomolecular aspects. For instance, depending on the aim of the specific study, it is possible to assign a high relevance to SNPs associated to hub proteins or SNPs occurring in regulatory regions such as the 5' near gene region. An overview of these possibilities is provided in Figure 5.16. Finally, the retrieval of the scored SNPs ranked

list is specifically aimed to support evaluation of genetic diseases.

Starting from the queried gene CCND1 and its semantically more similar genes (CCND2, CCND3,BCCIP, CDK5R, C13orf15) a list of ranked SNPs has been obtained exploiting the developed core scoring function. The whole list includes more than 1500 SNPs characterized by diverse features and thus assuming different degree of importance. Considering just genes with a final score ≥ 0.01 the list can be reduced to 499 genes. Part of the results obtained from CCND1 gene is reported in Figure 5.17. At the top of the ranked list there are 4 SNPs belonging to CCND1 (which codes for an hub protein), and placed on functional protein domains. Information about essential genes in not exploited in the described example, since no genes from the input list (nor in the original version neither in the enriched one) are labeled as essential for life.

An interesting parameter is represented by DNA localization. Within this feature crucial information related to the position of the SNP on the DNA chain is included: different forms of localization can be considered and a weight can be provided to each of them, according to the specific use case. In particular, referring to the considered example, while performing evaluations about DNA localization of listed SNPs it results that most of them are placed on intronic regions (around 60%) on DNA: this is obvious considering the high percentage of intronic regions on DNA strand compared to the esonic areas. According to the considered weight values SNPs present on introns are localized at lower positions within the ranked list, while at the top of it many "frameshift" and "missense" are concentrated. The latter localization types are only around the 3% of the whole set but SNPs in these positions obtained higher scores. Descending the ranked list, SNPs occur that are localised in UTR regions and near the gene. The "unknown" value is important for covering other DNA regions.

Input Genes (including optional ontological expansion):
CCND1, CCND2, CCND3, CCNG1, CDK5R2, BCCIP, C13orf15

Listed below all scored SNPs in the input genes, sorted by genetic position
To change sort order click on the column header
When all SNPs will be computed the output file will be linked at the end of this page.

Gene Symbol	SNP Name	Hub Protein	Domain Protein	SNP MAF	SNP Localization	Essential Gene	Feature Score ▲
CCND3	rs3218089	1	1	0	0.2	0	2.2
CCND3	rs11552778	1	1	0	0.2	0	2.2
CCND3	rs33966734	1	1	0	0.2	0	2.2
CCND1	rs11263523	1	1	0	0.2	0	2.2
CCND1	rs1131439	1	1	0	0.2	0	2.2
CCND1	rs1050971	1	1	0	0.2	0	2.2
CCND1	rs2220247	1	1	0	0.2	0	2.2
BCCIP	rs3208565	1	0	0.46078431372549	0.3	0	1.7607843137255
CCND2	rs3217805	1	0	0.43965517241379	0.25	0	1.6896551724138
CCND3	rs1051130	1	0	0.46551724137931	0.2	0	1.6655172413793
BCCIP	rs4385801	1	0	0.425	0.15	0	1.575
BCCIP	rs12049644	1	0	0.43220338983051	0.1	0	1.5322033898305
CCND3	rs13194688	1	0	0.5	0.01	0	1.51
CCND3	rs4607417	1	0	0.49166666666667	0.01	0	1.5016666666667
CCND3	rs6913232	1	0	0.48333333333333	0.01	0	1.4933333333333
CCND3	rs4333413	1	0	0.475	0.01	0	1.485
CCND3	rs4623235	1	0	0.475	0.01	0	1.485
CCND3	rs7766960	1	0	0.47457627118644	0.01	0	1.4845762711864
CCND3	rs4415146	1	0	0.47413793103448	0.01	0	1.4841379310345
CCND3	rs6920885	1	0	0.47413793103448	0.01	0	1.4841379310345
CCND3	rs4711703	1	0	0.46551724137931	0.01	0	1.4755172413793
CCNG1	rs2069347	1	0	0.475	0	0	1.475
CCND2	rs3217827	1	0	0.47413793103448	0	0	1.4741379310345
CCND1	rs7177	1	0	0.41379310344828	0.05	0	1.4637931034483
CCND3	rs4554318	1	0	0.44827586206897	0.01	0	1.458275862069
CCND2	rs3217936	1	0	0.35593220338983	0.1	0	1.4559322033898
BCCIP	rs3740206	1	0	0.45	0	0	1.45
BCCIP	rs11244667	1	0	0.45	0	0	1.45
BCCIP	rs10159992	1	0	0.44915254237288	0	0	1.4491525423729
CCND2	rs1049606	1	0	0.39166666666667	0.05	0	1.4416666666667
CCND2	rs12299509	1	0	0.43220338983051	0	0	1.4322033898305

Figure 5.17: Screenshot from the web site. Results page showing SNPs ranked list obtained from CCND1 example, discussed in the text. SNPs are sorted by features score. Ontological expansion is performed with similarity threshold set to 0.7; chosen features weights are shown in Figure 5.16.

Chapter 6

Horizons

Chapter Summary

In this chapter, we are discussing about future outcomes and extensions of the Ph.D. project since the work has a long term goal oriented to visual data mining in biomedicine, as mentioned in the introduction. The first idea is to improve the data layer by extending it with the dimension of the Phenotype and the Results of Analyses. The second approach is to extend the system architecture, enriching the core system with new blocks:

1. a Dynamic Query Builder that eases of accessing integrated data;
2. a Workflow engine that enables integration and execution of custom workflows;
3. a Graphical module that plots data and results in their genomic context.

The long term goal of our project is to create a visual data mining system for genetic studies, as pointed in the initial chapter 1. We designed a general framework aimed at dealing with experimental data, annotation data and clinical phenotypes and we implemented our framework with a set of features, related on data management and engineering, including a computational layer. The overall schema that we just described had empirical validations that confirmed us that our approach is valid and potentially useful.

Approaching the long term goal means including a set of new features that improve the system's functionalities and re-engineering some of the existing features. All new features are related both on data and architecture extensions. The initial framework schema, showed in Figure 3.3, is now extended with new design blocks in the "framework space" and new functionalities: Figure 6.1 shows the schema evolutions.
The new design model within the "framework space" includes two new blocks and organizes all blocks into two categories.

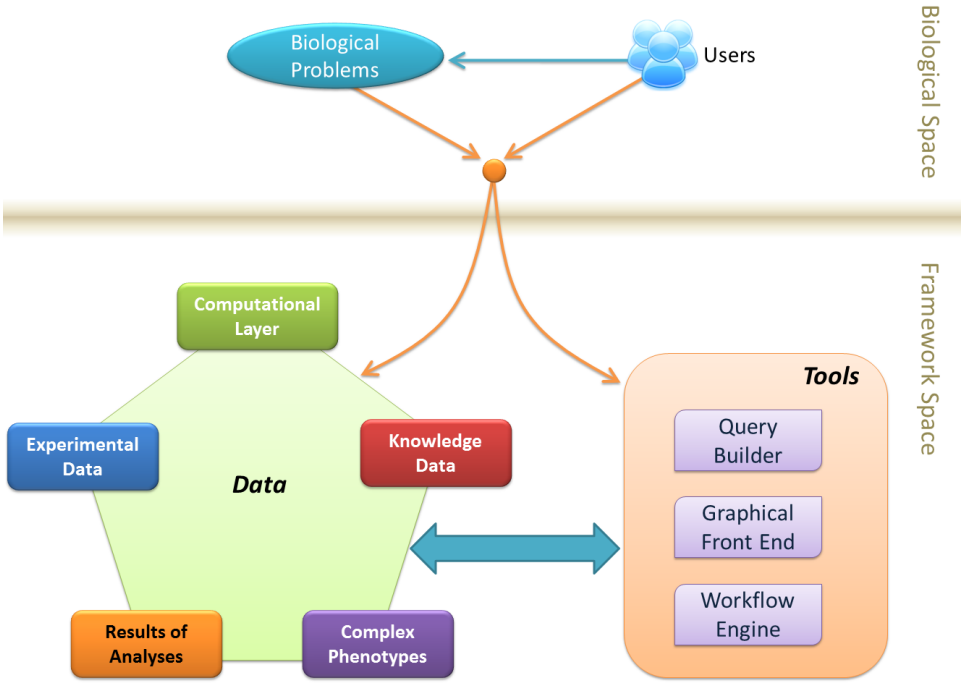


Figure 6.1: The extended versione of the framework.

We added two new data blocks, corresponding to remaining data, as mentioned in the chapter 3: *Results of Analyses* and *Complex Phenotypes*. The block *Results of Analyses* refers to data coming from previous genetic analyses and are mostly related to statistical analyses, for example GWAS results: in this case we collected for each SNP under analysis, the score of significance associated to the p-value in relationship with a specific phenotype. The second block, *Complex Phenotypes*, is added to include the definition of a complex phenotype and associates clinical records with a wide set of variables. An example of complex phenotype is Hypertension: modeling Hypertension means to collect many patient variables, from blood pressure (both systolic and diastolic) to life style monitoring and environment influence. Phenotype also includes patient history and clinical investigations, for example magnetic resonances or X-ray exams: all these data has to converge into the framework, addressing privacy and security issues. Other new blocks are related to system's functionalities and extensions. To achieve a visual data mining system, we have to implement the following tools:

1. a *Dynamic Query Builder*: this component enables an easy way of accessing all integrated data. A query builder usually simplifies data access and query. What is needed here is a *dynamic* query builder: since we need tools that can combine different data from all the different blocks to realize the mining activity and that can drive users

to their custom research, the dynamic query builder has to suggest on the fly what fields can be related to other fields and which range are allowed in the filtering process (query customization). The meaning of dynamic is related to this flexibility during the fields' combination, both on data types and values. We are designing this components using a machine learning approach on a semantic definition of tables' terms and fields, creating a set of self-updating rules.

2. a *Workflow engine*: this component enables integration and execution of custom users workflows. Workflows are a common procedures in structured genetic studies: if users have to follow a sequence of steps (pipelines or workflows) using well known objects, they can exploit available web services to obtain their results. We are designing the integration of service oriented modules and systems, such as Taverna [142], that enables available web services within our framework. In this way we can integrate external components that collaborate with our system to users' goal; this approach follows the direction of community interaction and reuse, also going into the direction of Mesh Up.

3. a *Graphical module*: this component plots data and results in their genomic context. Without a graphical system we could not have the visual data mining. We designed a graphical component that was easy for end users and that could help on localize biological objects in their natural position: following this idea, we are developing a custom genome browser, empowered through new filtering methods (related to data) and linked to the other tools, the Dynamic Query Builder and the Workflow Engine.

Figure 6.1 also presents blocks grouped into two categories:

1. *Data*: the three original design blocks (*Computational Layer*, *Knowledge Data* and *Experimental Data*) converge here and are combined with two new blocks, *Results of Analyses* and *Complex Phenotypes*. The four data blocks along with the computational block constitute the extended[1] category of data, represented in the figure 6.1 by a green pentagon.

2. *Tools*: this category includes all new functionalities of the systems, the three tools just mentioned, in the red rounded rectangle: the *Dynamic Query Builder*, the *Workflow engine* and the *Graphical module*.

The two categories are directly connected, such that each block of a group is reachable by a block of the other group and vice versa (the bidirectional light blue arrow between the categories).

[1]The adjective "extended" is placed to include the block of the computational layer together with the data blocks.

The new components added in the schema 6.1 imply some new critical aspects and issues to be solved, mainly related to two areas: data and architecture. From the *data* perspective, importing new data types such as phenotype reflects into new problems on data management. As an example, phenotypes are usually related to clinical records and patient visits; visits are naturally temporal-based, that means that for each visit physicians monitor and collect patient variables. For this reason, a data management system has to address temporal aspects while dealing with clinical records. Moreover, complex phenotypes need to be semantically related on the clinical and disease variables, creating an ontological representation of these variables. This aspect is important while integrating different data sets of existing cohorts of patients in which data have been collected using different protocols. Some important procedures of normalization, harmonization and data cleaning are required and semantic approaches can support these processes. Semantics is under evaluation also for the vertical data integration re-engineering, trying to adopt a RDF schema integration approach. On the other hand, from the *architecture* point of view, we need to adapt the framework to a Service-Oriented architecture. In fact, workflows require web services and thus the use of standard protocols. We are developing web services to access the integrated database such that the different tools can have a common access level; this approach also means re-design some middleware components, to be compatible with service-oriented architectures.

Next version of the framework will be accessible through web user interfaces that enable direct access to data and to available tools. The general vision of the extended framework from the user point of view is represented in Figure 6.2.

Users will access the framework through the *application layer*, where the available tools reside. In this layer we placed the dynamic query builder, the workflow engine, a set of relevant applications (an example can be SNPRanker) and the graphical front-end. Users can also access the *database layer* by loading data into the system using the available importing interfaces, and upload experimental data and results of analyses. Except for these two types of data, only tools can access the database layer: users will use annotation data and phenotypes by using the query builder or a specific analytical application. In the figure is not represented the computational layer since users will not directly access it, but only by executing jobs from available applications.

To achieve all these new requirements, we are developing a pilot comprehensive project called BDIM ("Biodata Insight for Mining") that integrates the design blocks and attempts to create a unique entry point to genetic visual data mining.

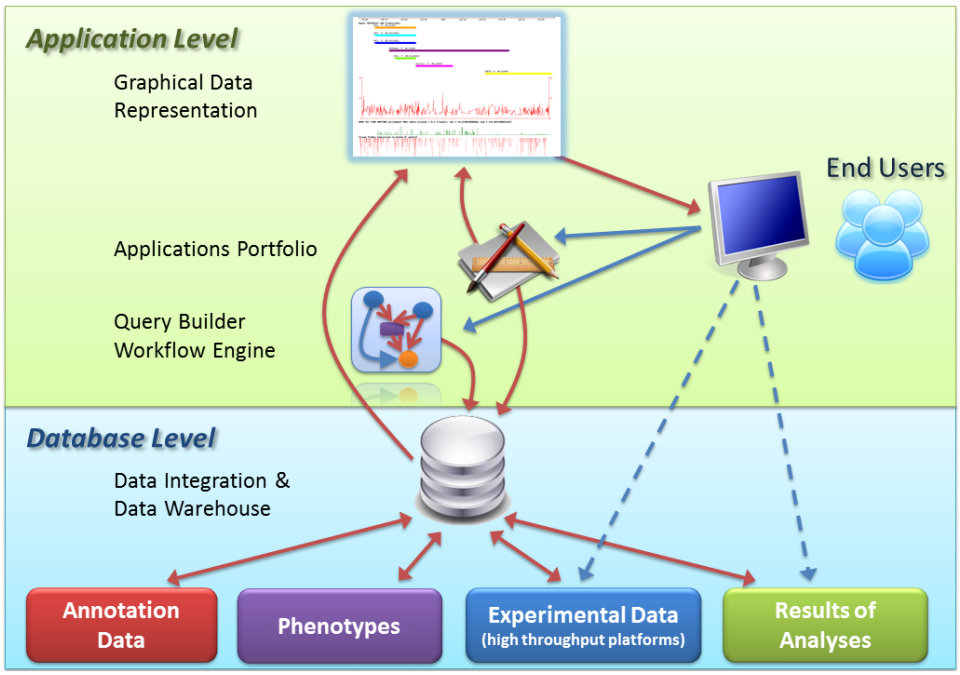


Figure 6.2: The Framework Schema of next versions from a user point of view.

Chapter 7

Conclusions

Chapter Summary

With this final chapter, we summarize what has been done during the Ph.D. project and the rationale that lies under each described activity. The discussion follows the 5 *W*s principle and traces the path of the work under this perspective.

One of the first sentences in the Introduction (Chapter 1) was related to the primary scope of the Ph.D. study: applications of Computer Science and Engineering methods to Genetics and Molecular Biology, to support both data and process engineering and problem solving. During the period of the Ph.D., we deepen the field of Bioinformatics applied to Genetics and in general to Biomedicine since we directly worked with Biologist, Geneticists and Medical Doctors. Most of the problems related to the application domain came from real case studies, arising from our partners Multimedica and Universita' degli Studi di Milano, that encountered issues on their empirical investigations. That is Applied Research: from a contextualized problem, find a solution to overcome existing limitations, to achieve required output with a valid and novel method to approach results that increase the scientific knowledge in your field. In information-gathering, such a process is equivalent to extracting from the 5 Ws model[1] two primary dimensions:

1. *what*: the result to obtain, the original problem;
2. *how*: the method that grants you to obtain a solution to the problem and consistent results.

Given our engineering background, we adopted this dichotomic approach to the Ph.D. project and for this reason the rationale of the work focused on

[1] An explanation of the 5 Ws principle is reported in Wikipedia English: http://en.wikipedia.org/wiki/Five_Ws.

two main goals: the first related to Biomedicine, that is the application of *what* to solve, the problem that we had to deal with; on the other hand, the second one is related to Computer Science discipline, the research field of our scientific faculty, that is the translation of *how* to address and solve the problem. The title of the Ph.D. manuscript expresses both concepts in a very short description: *Data integration for Clinical Genomics*. Data Integration is *how* we solved the problem, *what*, of Clinical Genomics. Clinical Genomics is also a short description of a critical biological objective: to translate genetic studies into clinical benefits, achieving the so called "personalized medicine". The idea of Clinical Genomics is a vision of what researchers are trying to realize; in practice there are many short term issues to be solved. This Ph.D. work is intended to solve some of these problems, starting from data management and providing a framework that supported users on their needs. Starting from this motivation, we gathered requirements from the laboratories with which we have collaborated, understanding their needs and how to support them, as described in Chapter 3.

A common problem of genetic laboratories is processing and analyzing experimental data coming from latest biotechnology platforms, such as *genotyping*, *sequencing* and so on: a research center needs computational resources and flexible information systems for data analysis and management along with the high-throughput biological platforms. Another key point is that researchers usually need to compare experimental data and annotation data, following custom workflows, and analyze them all on a high performance computational environment.

Given these specifications, we designed an extensible framework (Chapter 4) based on three constituent layers inter-connected (design blocks):

1. Experimental data layer that provides data integration of high-throughput platforms (also called *horizontal data integration*);

2. Knowledge data layer, that provides data integration of knowledge data from literature and public databases (also called *vertical integration*);

3. Computational layer, that provides access to distributed environments for data analysis, in our cases GRID (EGEE) and Cluster (Michelangelo - LITBIO) technologies.

Above the three blocks, single biological problems can be supported and custom user interfaces are implemented, as case studies shows in Chapter 5. From our partner laboratories, two main relevant biological problems have been studied:

- Linkage Analysis: we had a large pedigree in which subjects were genotyped with chips of 1 million of SNPs. Since the linkage analysis problem has computational limits, we designed a heuristic method

to overcome computational restrictions and implemented the method in a system to run analyses in distributed environments using our framework. The system has been successfully validated by end-users: promising genetic results have been retrieved and are now under new analyses (sequencing). We also tested performances of the system in Grid and Cluster, reporting compared results.

- SNP selection and ranking: given the problem of ranking SNPs based on a-priori information, we developed a new method for biological data mining on gene knowledge. The method has been implemented in a web tool, called *SNP Ranker*, that now is in use in the laboratories of our partners.

The framework here designed and implemented demonstrated that our approach is relevant, consistent and can have potential impacts on the scientific community. For this reason, this project is a first step toward a long term goal, as explained in Chapter 6: building an integrated platform for visual data mining applied to genetics and personalized medicine. Visual data mining is our vision. We believe that not only statistical methods are useful but also machine learning approaches and new techniques that leverages on researchers' insight and knowledge in the mining activity. This future platform cannot prescind from a solid, consistent and with high quality integrated data base, enabled to distributed environments; in other words what we tried to realize within this Ph.D project.

How to deal with the remaining *W*s? Are *who*, *where* and *when* relevant? The answer is "yes", they are very important and a good researcher has to consider all of them. In fact, *who* represents your collaborators, your research network that is one of the most important resource of each scientist. In our experience, we are in a multi-disciplinary laboratory, the ITB CNR, that gave us the opportunity to learn what is Bioinformatics and how to apply it to case studies. Moreover, without partner laboratories, none of our activities would have been validated with significant biological problems and real data. A good research network is the key for the success of a scientific project because is an opportunity of improving personal professional culture and experience. A similar concept is behind the term *where*: affiliation is not a marginal consideration, since an important research center can support you with consistent resources and contacts. Eventually, *When* identifies the appropriate timing of a research; this term refers to two important notes: (1) the importance of the planning and project management, and (2) in which time it can be relevant for the scientific community.

Publications

Journals

1. **A. Calabria**, D. Di Pasquale, M. Gnocchi, P.A. Cozzi, A. Orro, G.A. Trombetti, L. Milanesi. *Grid Based Genome Wide Studies on Atrial Flutter.* Journal of Grid Computing, 2010 Sept, DOI: 10.1007/s10723-010-9163-y

2. E. Mosca, R. Alfieri, I. Merelli, F. Viti, **A. Calabria**, L. Milanesi. *A multilevel data integration resource for breast cancer study.* BMC Syst Biol. 2010 Jun 3;4:76.

3. **A. Calabria**, E. Mosca, F. Viti, I. Merelli, L. Milanesi. *SNPRanker: a tool for identification and scoring of SNPs associated to target genes.* J Integr Bioinform. 2010 Mar 25;7(3). doi: 10.2390/biecoll-jib-2010-138.

4. L. Volpi, G. Roversi, E.A. Colombo, N. Leijsten, D. Concolino, **A. Calabria**, M.A. Mencarelli, M. Fimiani, F. Macciardi, R. Pfundt, E.F.P.M. Schoenmakers, L. Larizza. *Targeted Next-Generation Sequencing Appoints C16orf57 as Clericuzio-Type Poikiloderma with Neutropenia Gene.* Am J Hum Genet. 2010 Jan;86(1):72-6. Epub 2009 Dec 10. Erratum in: Am J Hum Genet. 2010 Sep 10;87(3):445.

5. F. Viti, E. Mosca, I. Merelli, **A. Calabria**, R. Alferi, and L. Milanesi. *Ontological enrichment of the Genes-to-Systems Breast Cancer database.* Metadata and Semantic Research, SpringerLink, part 2, pp 171-182, September 2009, DOI 10.1007/978-3-642-04590-5.

6. L. Milanesi, **A. Calabria**, D. Di Pasquale, M. Gnocchi, A. Orro, G.A. Trombetti. *A HPC and Grid enabling framework for genetic linkage analysis of SNPs.* Nuovo Cimento C, Vol 32, Issue 2, pp 249-252, April 2009, DOI 10.1393/ncc/i2009-10414-8

7. **A. Calabria**, D. Di Pasquale, A. Orro, G.A. Trombetti, L. Milanesi. *Genetic Linkage Analysis Challenges On A Distributed Grid Environment.* Herald Vogis 2009, Tom 13, N 1.

Proceedings

1. **A. Calabria**, E.F Osimo, S. Gaudi, F. Macciardi. *BioData Insight: a tool for biological data integration and data mining support.* Human Variome Project, Implementation and Integration Meeting, May 2010, UNESCO, Paris (FR).

2. **A. Calabria**, I. Merelli, L. Milanesi, D. Cusi, F. Macciardi. *SNP Mining from Features Enrichment.* Clinical Genomic Analysis Workshop, March 2010, IBM Research, Haifa (IL)

3. A. Orro, M. Gnocchi, **A. Calabria**, D. Di Pasquale, L. Milanesi. *Genetic Linkage Analysis on the Desktop Grid Infrastructure.* Third AlmereGrid Desktop Grid Experience workshop, March 2010, Almere (NL).

4. F. Viti, E. Mosca, I. Merelli, **A. Calabria**, R. Alferi, and L. Milanesi. *Ontological enrichment of the Genes-to-Systems Breast Cancer database.* International Conference on Metadata and Semantics Research, Oct 2009, Milano (IT).

5. **A. Calabria**, A. Maurino, F. Macciardi and L. Milanesi. *Gene data fusion: a prototype for conflict reporting facility.* Computational Intelligence Methods For Bioinformatics And Biostatistics, Sept 2009, Genova (IT).

6. L. Milanesi, G. Trombetti, **A. Calabria**, D. Di Pasquale, M. Gnocchi, A. Orro. Grid based genetic population analysis challenges for Genetic Linkage Analysis of SNPs EGEE Meeting 2008, Instanbul (TR).

7. F. Torri, S. Lupoli, A. Orro, S. Potkin, E. Salvi, J. Fallon, P. Cozzi, **A. Calabria**, J. Turner, C. Barlassina, V. Tieran, F. Taddeo, C. Cosentino, F. Macciardi. Copy-Number variations in a case-control study of schizophrenia. European Human Genetics Conference, May 2008, Barcelona (ES).

8. **A. Calabria**, D. Di Pasquale, A. Orro, G. Trombetti, L. Milanesi. Genetic Linkage Analysis Challenges On A Distributed Grid Environment. Bioinformatics of Genome Regulation and Structure, June 2008, Novosibirsk (RU).

9. A. Orro, **A. Calabria**, D. Di Pasquale, L. Milanesi. Data management for SNP genotyping technologies. NETTAB Conference, May 2008, Varenna (IT).

10. **A. Calabria**, D. Di Pasquale, G. Trombetti, P. Cozzi, A. Orro, M. Gnocchi, L. Milanesi. *Genome Wide Linkage Analysis and Systems Knowledge: an Integrated Web Framework for Distributed Execution and Results Annotation.* Sysbiohealth Symposium 2008, Bologna (IT).

11. L. Milanesi, E. Mosca, A. Orro, I. Merelli, **A. Calabria**, R. Alfieri, G. Trombetti. ICT and GRID computing for Bioinformatics and Systems Biology. SysBioHealth Conference proceeding 2007, Milano (IT).

12. **A. Calabria**, A. Orro, L. Milanesi. An Information System for Data and Analysis Tracking in Medical Genetic Studies. NETTAB Conference 2007, Pisa (IT).

Presentations

1. **A. Calabria**, E.F Osimo, S. Gaudi, F. Macciardi *BioData Insight: a tool for biological data integration and data mining support.* Human Variome Project, Implementation and Integration Meeting, May 2010, UNESCO, Paris.

2. **A. Calabria**, E. Mosca, F. Viti, I. Merelli, L. Milanesi. *SNPRanker: a tool for identification and scoring of SNPs associated to target genes.* Integrative Bioinformatics Conference, March 2010, Cambridge (UK).

3. **A. Calabria**, A. Maurino, F. Macciardi and L. Milanesi. *Gene data fusion: a prototype for conflict reporting facility.* Computational Intelligence Methods For Bioinformatics And Biostatistics, Sept 2009, Genova (IT).

4. **A. Calabria**, D. Di Pasquale, G. Trombetti, P. Cozzi, A. Orro, M. Gnocchi, L. Milanesi. *Grid based application for snps genome wide studies.* Computational Intelligence Methods For Bioinformatics And Biostatistics, Sept 2009, Genova (IT).

List of Figures

List of Tables

Bibliography

[1] E. Alpaydin, *Introduction to Machine Learning.* The MIT Press, 2004.

[2] L. E. Peterson and X.-W. Chen, "Machine learning in biomedicine and bioinformatics." *Int J Data Min Bioinform*, vol. 3, no. 4, pp. 363–364, 2009.

[3] P. Baldi and S. Brunak, *Bioinformatics: the machine learning approach.* The MIT Press, 2001.

[4] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27 (2), pp. 83–85, 2005.

[5] J. Han and M. Kamber, *Data mining: concepts and techniques*, T. M. K. S. in Data Management Systems, Ed. Morgan Kaufmann, 2006.

[6] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*, T. M. K. S. in Data Management Systems, Ed. Morgan Kaufmann, 2005.

[7] A. D. Weston and L. Hood, "Systems biology, proteomics, and the future of health care: toward predictive, preventative, and personalized medicine." *J Proteome Res*, vol. 3, no. 2, pp. 179–196, 2004.

[8] G. S. Ginsburg and J. J. McCarthy, "Personalized medicine: revolutionizing drug discovery and patient care." *Trends Biotechnol*, vol. 19, no. 12, pp. 491–496, Dec 2001.

[9] M. Bianchessi, S. Burgarella, and M. Cereda, "Point-of-care systems for rapid dna quantification in oncology." *Tumori*, vol. 94, no. 2, pp. 216–225, 2008.

[10] G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "On reconciling data exchange, data integration, and peer data management," in *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* New York, NY, USA: ACM, 2007, pp. 133–142.

[11] W. Sujansky, "Heterogeneous database integration in biomedicine." *J Biomed Inform*, vol. 34, no. 4, pp. 285–298, Aug 2001. [Online]. Available: http://dx.doi.org/10.1006/jbin.2001.1024

[12] M. Mesiti, E. Jiminez-Ruiz, I. Sanz, R. Berlanga-Llavori, P. Perlasca, G. Valentini, and D. Manset, "Xml-based approaches for the integration of heterogeneous bio-molecular data." *BMC Bioinformatics*, vol. 10 Suppl 12, p. S7, 2009. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-10-S12-S7

[13] K. A. Karasavvas, R. Baldock, and A. Burger, "Bioinformatics integration and agent technology." *J Biomed Inform*, vol. 37, no. 3, pp. 205–219, Jun 2004. [Online]. Available: http://dx.doi.org/10.1016/j.jbi.2004.04.003

[14] B. Louie, P. Mork, F. Martin-Sanchez, A. Halevy, and P. Tarczy-Hornoch, "Data integration and genomic medicine." *J Biomed Inform*, vol. 40, no. 1, pp. 5–16, Feb 2007. [Online]. Available: http://dx.doi.org/10.1016/j.jbi.2006.02.007

[15] T. Hernandez and S. Kambhampati, "Integration of biological sources: current systems and challenges ahead." *ACM SIGMOD Record*, vol. 33, p. 3, 2004.

[16] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases." *ACM Computing Surveys (CSUR)*, vol. 22(3), pp. 183 – 236, 1990.

[17] W. Hasselbring, "Information system integration." *Communications of the ACM*, vol. 43, pp. 32–38, 2000.

[18] P. D. Karp, "A strategy for database interoperation," *Journal of Computational Biology*, vol. 2(4), p. 573 • 586, 1995.

[19] A. Kementsietsidis and M. Arenas, "Data sharing through query translation in autonomous sources," in *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*. VLDB Endowment, 2004, pp. 468–479.

[20] L. Wong, "Kleisli: its exchange format, supporting tools, and an application in protein interaction extraction," in *Proc. IEEE Int Bio-Informatics and Biomedical Engineering Symp*, 2000, pp. 21–28.

[21] C. A. Goble, R. Stevens, G. Ng, S. Bechhofer, N. W. Paton, P. G. Baker, M. Peim, and A. Brass, "Transparent access to multiple bioinformatics information sources," *IBM Systems Journal*, vol. 40, no. 2, pp. 532–551, 2001.

[22] C. Goble and R. Stevens, "State of the nation in data integration for bioinformatics." *J Biomed Inform*, vol. 41, no. 5, pp. 687–693, Oct 2008. [Online]. Available: http://dx.doi.org/10.1016/j.jbi.2008.01.008

[23] S. Aymard, D. Fieschi, F. Volot, M. Joubert, and M. Fieschi, "Towards interoperability of information sources within a hospital intranet." *Proc AMIA Symp*, pp. 638–642, 1998.

[24] F. J. van Wingerde, J. Schindler, P. Kilbridge, P. Szolovits, C. Safran, D. Rind, S. Murphy, G. O. Barnett, and I. S. Kohane, "Using hl7 and the world wide web for unifying patient data from remote databases." *Proc AMIA Annu Fall Symp*, pp. 643–647, 1996.

[25] J. Annevelink, C. Y. Young, and P. C. Tang, "Heterogenous database integration in a physician workstation." *Proc Annu Symp Comput Appl Med Care*, pp. 368–372, 1991.

[26] I. M. A. Chen, A. Kosky, V. M. Markowitz, and E. Szeto, "Developing and accessing scientific databases with the opm data management tools," in *Proc. 13th Int Data Engineering Conf*, 1997.

[27] Y. Arens, C. Y. Chee, C.-N. Hsu, and C. A. Knoblock, "Retrieving and integrating data from multiple information sources," *International Journal of Intelligent Cooperating Information Systems*, vol. 2(2), p. 127•158, 1993.

[28] P. P.-S. Chen, "The entity-relationship model•toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1(1), pp. 9 – 36, 1976.

[29] R. Fikes, M. Cutkosky, T. Gruber, and J. V. Baalen, "Knowledge sharing technology," Knowledge Systems Laboratory, Stanford University, Tech. Rep., 1991. [Online]. Available: http://www.ksl.stanford.edu/knowledge-sharing/papers/kst-project.rtf

[30] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing." *International Journal of Human Computer Studies*, 1995. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.6025&rep=rep1&type=pdf

[31] P. Haase and Y. Wang, "A decentralized infrastructure for query answering over distributed ontologies," in *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing.* New York, NY, USA: ACM, 2007, pp. 1351–1356.

[32] S. B. Davidson, C. Overton, and P. Buneman, "Challenges in integrating biological data sources." *J Comput Biol*, vol. 2, no. 4, pp. 557–572, 1995.

[33] L. Zamboulis, N. Martin, and A. Poulovassilis, "Bioinformatics service reconciliation by heterogeneous schema transformation," in *DILS'07: Proceedings of the 4th international conference on Data integration in the life sciences*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 89–104.

[34] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching." *VLDB Journal*, vol. 10 (4), pp. 334–350, 2001.

[35] P. Shvaiko and J. Euzenat, "A survey of schema-based matching approaches," *Journal of Data Semantics IV, Lecture Notes in Computer Science*, vol. 3730/2005, pp. 146–171, 2005.

[36] G. Wiederhold, "Mediators in the architecture of future information systems," *Computer*, vol. 25, no. 3, pp. 38–49, 1992.

[37] C. Schroth and T. Janner, "Web 2.0 and soa: Converging concepts enabling the internet of services," *IT Professional*, vol. 9, no. 3, pp. 36–41, 2007.

[38] A. Bonifati, E. Chang, T. Ho, L. V. Lakshmanan, R. Pottinger, and Y. Chung, "Schema mapping and query translation in heterogeneous p2p xml databases," *The VLDB Journal*, vol. 19, no. 2, pp. 231–256, 2010.

[39] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos, "The hyperion project: from data integration to data coordination," *ACM SIGMOD Record*, vol. 32 (3), pp. 53–58, 2003.

[40] M. D. Hossain and I. Kiringa, "Querying communities of interest in peer database networks," in *DBISP2P'05/06: Proceedings of the 2005/2006 international conference on Databases, information systems, and peer-to-peer computing*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 227–234.

[41] V. Kantere, D. Tsoumakos, and N. Roussopoulos, "Querying structured data in an unstructured p2p system," in *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*. New York, NY, USA: ACM, 2004, pp. 64–71.

[42] A. Kementsietsidis, M. Arenas, and R. J. Miller, "Mapping data in peer-to-peer systems: semantics and algorithmic issues," in *International Conference on Management of Data, Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003.

[43] G. C. Minsk, L. S. Poh, H. Wei, and T. P. Siew, "Web 2.0 concepts and technologies for dynamic b2b integration," in *Proc. ETFA Emerging Technologies and Factory Automation IEEE Conf*, 2007, pp. 315–321.

[44] V. Hoyer, K. Stanoesvka-Slabeva, T. Janner, and C. Schroth, "Enterprise mashups: Design principles towards the long tail of user needs," in *Proc. IEEE Int. Conf. Services Computing SCC '08*, vol. 2, 2008, pp. 601–602.

[45] L. D. Stein, "Integrating biological databases." *Nat Rev Genet*, vol. 4, no. 5, pp. 337–345, May 2003. [Online]. Available: http://dx.doi.org/10.1038/nrg1065

[46] I. C. Gerling, S. S. Solomon, and M. Bryer-Ash, "Genomes, transcriptomes, and proteomes: molecular medicine and its impact on medical practice." *Arch Intern Med*, vol. 163, no. 2, pp. 190–198, Jan 2003.

[47] V. Maojo and C. A. Kulikowski, "Bioinformatics and medical informatics: collaborations on the road to genomic medicine?" *J Am Med Inform Assoc*, vol. 10, no. 6, pp. 515–522, 2003. [Online]. Available: http://dx.doi.org/10.1197/jamia.M1305

[48] T. E. Klein, J. T. Chang, M. K. Cho, K. L. Easton, R. Fergerson, M. Hewett, Z. Lin, Y. Liu, S. Liu, D. E. Oliver, D. L. Rubin, F. Shafa, J. M. Stuart, and R. B. Altman, "Integrating genotype and phenotype information: an overview of the pharmgkb project. pharmacogenetics research network and knowledge base." *Pharmacogenomics J*, vol. 1, no. 3, pp. 167–170, 2001.

[49] J. Augen, "The evolving role of information technology in the drug discovery process." *Drug Discov Today*, vol. 7, no. 5, pp. 315–323, Mar 2002.

[50] B. L. Claus and D. J. Underwood, "Discovery informatics: its evolving role in drug discovery." *Drug Discov Today*, vol. 7, no. 18, pp. 957–966, Sep 2002.

[51] D. Karolchik, R. M. Kuhn, R. Baertsch, G. P. Barber, H. Clawson, M. Diekhans, B. Giardine, R. A. Harte, A. S. Hinrichs, F. Hsu, K. M. Kober, W. Miller, J. S. Pedersen, A. Pohl, B. J. Raney, B. Rhead, K. R. Rosenbloom, K. E. Smith, M. Stanke, A. Thakkapallayil, H. Trumbower, T. Wang, A. S. Zweig, D. Haussler, and W. J. Kent, "The ucsc genome browser database: 2008 update." *Nucleic Acids Res*, vol. 36, no. Database issue, pp. D773–D779, Jan 2008. [Online]. Available: http://dx.doi.org/10.1093/nar/gkm966

[52] D. Karolchik, A. S. Hinrichs, T. S. Furey, K. M. Roskin, C. W. Sugnet, D. Haussler, and W. J. Kent, "The ucsc table browser data retrieval tool." *Nucleic Acids Res*, vol. 32, no. Database issue, pp. D493–D496, Jan 2004. [Online]. Available: http://dx.doi.org/10.1093/nar/gkh103

[53] E. Birney and E. Team, "Ensembl: a genome infrastructure." *Cold Spring Harb Symp Quant Biol*, vol. 68, pp. 213–215, 2003.

[54] S. Davidson, O. Buneman, J. Crabtree, V. Tannen, G. Overton, and L. Wong, "Biokleisli: Integrating biomedical data and analysis packages." *Bioinformatics: Databases and Systems*, vol. 3, pp. 201–211, 1999.

[55] R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N. W. Paton, C. A. Goble, and A. Brass, "Tambis: transparent access to multiple bioinformatics information sources." *Bioinformatics*, vol. 16, no. 2, pp. 184–185, Feb 2000.

[56] L. Donelson, P. Tarczy-Hornoch, P. Mork, C. Dolan, J. A. Mitchell, M. Barrier, and H. Mei, "The biomediator system as a data integration tool to answer diverse biologic queries." *Stud Health Technol Inform*, vol. 107, no. Pt 2, pp. 768–772, 2004.

[57] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. L. Dong, Y. Kadiyska, G. Miklau, and P. Mork, "The piazza peer data management project." *ACM SIGMOD Record*, vol. 32 (3), pp. 47–52, 2003.

[58] S. Kim and A. Misra, "Snp genotyping: technologies and biomedical applications." *Annu Rev Biomed Eng*, vol. 9, pp. 289–320, 2007. [Online]. Available: http://dx.doi.org/10.1146/annurev.bioeng.9.060906.152037

[59] T. Hubbard, D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down, R. Durbin, E. Eyras, J. Gilbert, M. Hammond, L. Huminiecki, A. Kasprzyk, H. Lehvaslaiho, P. Lijnzaad, C. Melsopp, E. Mongin, R. Pettett, M. Pocock, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik, and M. Clamp, "The ensembl genome database project." *Nucleic Acids Res*, vol. 30, no. 1, pp. 38–41, Jan 2002.

[60] M. D. Mailman, M. Feolo, Y. Jin, M. Kimura, K. Tryka, R. Bagoutdinov, L. Hao, A. Kiang, J. Paschall, L. Phan, N. Popova, S. Pretel, L. Ziyabari, M. Lee, Y. Shao, Z. Y. Wang, K. Sirotkin, M. Ward, M. Kholodov, K. Zbicz, J. Beck, M. Kimelman, S. Shevelev, D. Preuss, E. Yaschenko, A. Graeff, J. Ostell, and S. T. Sherry,

"The ncbi dbgap database of genotypes and phenotypes." *Nat Genet*, vol. 39, no. 10, pp. 1181–1186, Oct 2007. [Online]. Available: http://dx.doi.org/10.1038/ng1007-1181

[61] R. G. G. of the Gene Ontology Consortium, "The gene ontology's reference genome project: a unified framework for functional annotation across species." *PLoS Comput Biol*, vol. 5, no. 7, p. e1000431, Jul 2009. [Online]. Available: http://dx.doi.org/10.1371/journal.pcbi.1000431

[62] M. Asslaber and K. Zatloukal, "Biobanks: transnational, european and global networks." *Brief Funct Genomic Proteomic*, vol. 6, no. 3, pp. 193–201, Sep 2007. [Online]. Available: http://dx.doi.org/10.1093/bfgp/elm023

[63] K. K. Kakazu, L. W. K. Cheung, and W. Lynne, "The cancer biomedical informatics grid (cabig): pioneering an expansive network of information and tools for collaborative cancer research." *Hawaii Med J*, vol. 63, no. 9, pp. 273–275, Sep 2004.

[64] C. Ricss, V. Alvarez, F. Cava, J. V. Garcna-Lario, A. Hernandez, C. V. Jiminez, J. Minchinela, C. Perich, and M. Simsn, "Integration of data derived from biological variation into the quality management system." *Clin Chim Acta*, vol. 346, no. 1, pp. 13–18, Aug 2004. [Online]. Available: http://dx.doi.org/10.1016/j.cccn.2004.03.022

[65] B. R. Zeeberg, W. Feng, G. Wang, M. D. Wang, A. T. Fojo, M. Sunshine, S. Narasimhan, D. W. Kane, W. C. Reinhold, S. Lababidi, K. J. Bussey, J. Riss, J. C. Barrett, and J. N. Weinstein, "Gominer: a resource for biological interpretation of genomic and proteomic data." *Genome Biol*, vol. 4, no. 4, p. R28, 2003.

[66] D. A. Benson, M. S. Boguski, D. J. Lipman, J. Ostell, B. F. Ouellette, B. A. Rapp, and D. L. Wheeler, "Genbank." *Nucleic Acids Res*, vol. 27, no. 1, pp. 12–17, Jan 1999.

[67] S. C. Potter, L. Clarke, V. Curwen, S. Keenan, E. Mongin, S. M. J. Searle, A. Stabenau, R. Storey, and M. Clamp, "The ensembl analysis pipeline." *Genome Res*, vol. 14, no. 5, pp. 934–941, May 2004. [Online]. Available: http://dx.doi.org/10.1101/gr.1859804

[68] V. Curwen, E. Eyras, T. D. Andrews, L. Clarke, E. Mongin, S. M. J. Searle, and M. Clamp, "The ensembl automatic gene annotation system." *Genome Res*, vol. 14, no. 5, pp. 942–950, May 2004. [Online]. Available: http://dx.doi.org/10.1101/gr.1858004

[69] S. M. J. Searle, J. Gilbert, V. Iyer, and M. Clamp, "The otter annotation system." *Genome Res*, vol. 14, no. 5, pp. 963–970, May 2004. [Online]. Available: http://dx.doi.org/10.1101/gr.1864804

[70] E. A. Bruford, M. J. Lush, M. W. Wright, T. P. Sneddon, S. Povey, and E. Birney, "The hgnc database in 2008: a resource for the human genome." *Nucleic Acids Res*, vol. 36, no. Database issue, pp. D445–D448, Jan 2008. [Online]. Available: http://dx.doi.org/10.1093/nar/gkm881

[71] M. Rebhan, V. Chalifa-Caspi, J. Prilusky, and D. Lancet, "Genecards: integrating information about genes, proteins and diseases." *Trends Genet*, vol. 13, no. 4, p. 163, Apr 1997.

[72] M. Safran, V. Chalifa-Caspi, O. Shmueli, T. Olender, M. Lapidot, N. Rosen, M. Shmoish, Y. Peter, G. Glusman, E. Feldmesser, A. Adato, I. Peter, M. Khen, T. Atarot, Y. Groner, and D. Lancet, "Human gene-centric databases at the weizmann institute of science: Genecards, udb, crow 21 and horde." *Nucleic Acids Res*, vol. 31, no. 1, pp. 142–146, Jan 2003.

[73] S. Aerts, D. Lambrechts, S. Maity, P. V. Loo, B. Coessens, F. D. Smet, L.-C. Tranchevent, B. D. Moor, P. Marynen, B. Hassan, P. Carmeliet, and Y. Moreau, "Gene prioritization through genomic data fusion." *Nat Biotechnol*, vol. 24, no. 5, pp. 537–544, May 2006. [Online]. Available: http://dx.doi.org/10.1038/nbt1203

[74] T. D. Bie, L.-C. Tranchevent, L. M. M. van Oeffelen, and Y. Moreau, "Kernel-based data fusion for gene prioritization." *Bioinformatics*, vol. 23, no. 13, pp. i125–i132, Jul 2007. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btm187

[75] G. R. G. Lanckriet, T. D. Bie, N. Cristianini, M. I. Jordan, and W. S. Noble, "A statistical framework for genomic data fusion." *Bioinformatics*, vol. 20, no. 16, pp. 2626–2635, Nov 2004. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/bth294

[76] G. R. G. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble, "Kernel-based data fusion and its application to protein function prediction in yeast." *Pac Symp Biocomput*, pp. 300–311, 2004.

[77] M. Y. Galperin and G. R. Cochrane, "Nucleic acids research annual database issue and the nar online molecular biology database collection in 2009." *Nucleic Acids Res*, vol. 37, no. Database issue, pp. D1–D4, Jan 2009. [Online]. Available: http://dx.doi.org/10.1093/nar/gkn942

[78] A. G. Rust, E. Mongin, and E. Birney, "Genome annotation techniques: new approaches and challenges." *Drug Discov Today*, vol. 7, no. 11 Suppl, pp. S70–S76, Jun 2002.

[79] E. M. Smigielski, K. Sirotkin, M. Ward, and S. T. Sherry, "dbsnp: a database of single nucleotide polymorphisms." *Nucleic Acids Res*, vol. 28, no. 1, pp. 352–355, Jan 2000.

[80] E. W. Sayers, T. Barrett, D. A. Benson, E. Bolton, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. Dicuccio, S. Federhen, M. Feolo, L. Y. Geer, W. Helmberg, Y. Kapustin, D. Landsman, D. J. Lipman, Z. Lu, T. L. Madden, T. Madej, D. R. Maglott, A. Marchler-Bauer, V. Miller, I. Mizrachi, J. Ostell, A. Panchenko, K. D. Pruitt, G. D. Schuler, E. Sequeira, S. T. Sherry, M. Shumway, K. Sirotkin, D. Slotta, A. Souvorov, G. Starchenko, T. A. Tatusova, L. Wagner, Y. Wang, W. J. Wilbur, E. Yaschenko, and J. Ye, "Database resources of the national center for biotechnology information." *Nucleic Acids Res*, vol. 38, no. Database issue, pp. D5–16, Jan 2010. [Online]. Available: http://dx.doi.org/10.1093/nar/gkp967

[81] U. Consortium, "The universal protein resource (uniprot) in 2010." *Nucleic Acids Res*, vol. 38, no. Database issue, pp. D142–D148, Jan 2010. [Online]. Available: http://dx.doi.org/10.1093/nar/gkp846

[82] S. Hunter, R. Apweiler, T. K. Attwood, A. Bairoch, A. Bateman, D. Binns, P. Bork, U. Das, L. Daugherty, L. Duquenne, R. D. Finn, J. Gough, D. Haft, N. Hulo, D. Kahn, E. Kelly, A. Laugraud, I. Letunic, D. Lonsdale, R. Lopez, M. Madera, J. Maslen, C. McAnulla, J. McDowall, J. Mistry, A. Mitchell, N. Mulder, D. Natale, C. Orengo, A. F. Quinn, J. D. Selengut, C. J. A. Sigrist, M. Thimma, P. D. Thomas, F. Valentin, D. Wilson, C. H. Wu, and C. Yeats, "Interpro: the integrative protein signature database." *Nucleic Acids Res*, vol. 37, no. Database issue, pp. D211–D215, Jan 2009. [Online]. Available: http://dx.doi.org/10.1093/nar/gkn785

[83] H. Berman, K. Henrick, H. Nakamura, and J. L. Markley, "The worldwide protein data bank (wwpdb): ensuring a single, uniform archive of pdb data." *Nucleic Acids Res*, vol. 35, no. Database issue, pp. D301–D303, Jan 2007. [Online]. Available: http://dx.doi.org/10.1093/nar/gkl971

[84] K. M. Aoki-Kinoshita KF, "Gene annotation and pathway mapping in kegg." *Methods Mol Biol*, vol. 396, pp. 71–91, 2007.

[85] L. Matthews, G. Gopinath, M. Gillespie, M. Caudy, D. Croft, B. de Bono, P. Garapati, J. Hemish, H. Hermjakob, B. Jassal,

A. Kanapin, S. Lewis, S. Mahajan, B. May, E. Schmidt, I. Vastrik, G. Wu, E. Birney, L. Stein, and P. D'Eustachio, "Reactome knowledgebase of human biological pathways and processes." *Nucleic Acids Res*, vol. 37, no. Database issue, pp. D619–D622, Jan 2009. [Online]. Available: http://dx.doi.org/10.1093/nar/gkn863

[86] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers, "Biogrid: a general repository for interaction datasets." *Nucleic Acids Res*, vol. 34, no. Database issue, pp. D535–D539, Jan 2006. [Online]. Available: http://dx.doi.org/10.1093/nar/gkj109

[87] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene ontology: tool for the unification of biology. the gene ontology consortium." *Nat Genet*, vol. 25, no. 1, pp. 25–29, May 2000. [Online]. Available: http://dx.doi.org/10.1038/75556

[88] A. Calabria, A. Maurino, F. Macciardi, and L. Milanesi, "Gene data fusion: A prototype for conflict reporting facility," in *Computational Intelligence methods for Bioinformatics and Biostatistics*, 2009.

[89] F. Gagliardi, B. Jones, F. Grey, M.-E. Bigin, and M. Heikkurinen, "Building an infrastructure for scientific grid computing: status and goals of the egee project." *Philos Transact A Math Phys Eng Sci*, vol. 363, no. 1833, pp. 1729–1742, Aug 2005. [Online]. Available: http://dx.doi.org/10.1098/rsta.2005.1603

[90] G. A. Trombetti, "Enabling computationally intensive bioinformatics applications on the grid platform," Ph.D. dissertation, DEIS department, Universita degli Studi di Bologna, 2008. [Online]. Available: http://amsdottorato.cib.unibo.it/922/

[91] J. M. Alonso, J. M. Ferrero, V. Hernandez, G. Molto, J. Saiz, and B. Trenor, "A grid computing-based approach for the acceleration of simulations in cardiology," vol. 12, no. 2, pp. 138–144, 2008.

[92] V. Breton, N. Jacq, V. Kasam, and M. Hofmann-Apitius, "Grid-added value to address malaria," vol. 12, no. 2, pp. 173–181, 2008.

[93] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," vol. 15, no. 11, pp. 1427–1436, 1989.

[94] S. Jin, G. Schiavone, and D. Turgut, "A performance study of multiprocessor task scheduling algorithms," *J. Supercomput.*, vol. 43, no. 1, pp. 77–97, 2008.

[95] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems." *Journal of Parallel and Distributed Computing*, vol. 61 6, pp. 810–837, 2001.

[96] M. Cecchi, F. Capannini, A. Dorigo, A. Ghiselli, F. Giacomini, A. Maraschini, M. Marzolla, S. Monforte, F. Pacini, L. Petronzio, and F. Prelz, "The glite workload management system." *Advances in Grid and Pervasive Computing*, vol. 5529/2009, pp. 256–268, 2009.

[97] V. V. Korkhov, J. T. Moscicki, and V. V. Krzhizhanovskaya, "Dynamic workload balancing of parallel applications with user-level scheduling on the grid." *Future Generation Computer Systems*, vol. 25 1, pp. 28–34, 2009.

[98] C. Germain-Renaud, C. Loomis, J. T. Moscicki, and R. Texier, "Scheduling for responsive grids," *Journal of Grid Computing*, vol. 6, Number 1, pp. 15–27, 2008.

[99] T. Maeno, "Panda: Distributed production and distributed analysis system for atlas." *Journal of Physics: Conference Series*, vol. 119, Part 6, 2008.

[100] H.-C. Lee, J. Salzemann, N. Jacq, H.-Y. Chen, L.-Y. Ho, I. Merelli, L. Milanesi, V. Breton, S. C. Lin, and Y.-T. Wu, "Grid-enabled high-throughput emphasis emphasistype="italic"¿in silico screening against influenza a neuraminidase," vol. 5, no. 4, pp. 288–295, 2006.

[101] I. Sfiligoi, "glideinwms: a generic pilot-based workload management system." *Journal of Physics: Conference Series*, vol. 119 6, 2008.

[102] A. Pugliese, D. Talia, and R. Yahyapour, "Modeling and supporting grid scheduling." *Journal of Grid Computing*, vol. 6, 2, pp. 195–213, 2007.

[103] R. McClatchey, A. Anjum, H. Stockinger, A. Ali, I. Willers, and M. Thomas, "Data intensive and network aware (diana) grid scheduling." *Journal of Grid Computing*, vol. 5, 1, pp. 43–64, 2007.

[104] B. Rood and M. J. Lewis, "Grid resource availability prediction-based scheduling and task replication." *Journal of Grid Computing*, vol. 7, 4, pp. 479–500, 2009.

[105] E. Floros and E. Loomis, "Interactive and real-time applications on the egee grid infrastructure," *Remote Instrumentation and Virtual Laboratories Service Architecture and Networking.*, pp. 263–273, 2010.

[106] R. M. Rahman, K. Barker, and R. Alhajj, "Replica placement strategies in data grid." *Journal of Grid Computing*, vol. 6, 1, pp. 103–123, 2007.

[107] I. Sfiligoi, O. Koeroo, G. Venekamp, D. Yocum, D. Groep, and D. Petravick, "Addressing the pilot security problem with glexec." *Journal of Physics: Conference Series*, vol. 119, 5, 2008.

[108] G. A. Trombetti, R. J. P. Bonnal, E. Rizzi, G. D. Bellis, and L. Milanesi, "Data handling strategies for high throughput pyrosequencers." *BMC Bioinformatics*, vol. 8 Suppl 1, p. S22, 2007. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-8-S1-S22

[109] G. R. Abecasis, S. S. Cherny, W. O. Cookson, and L. R. Cardon, "Merlin–rapid analysis of dense genetic maps using sparse gene flow trees." *Nat Genet*, vol. 30, no. 1, pp. 97–101, Jan 2002. [Online]. Available: http://dx.doi.org/10.1038/ng786

[110] E. Lander and L. Kruglyak, "Genetic dissection of complex traits: guidelines for interpreting and reporting linkage results." *Nat Genet*, vol. 11, no. 3, pp. 241–247, Nov 1995. [Online]. Available: http://dx.doi.org/10.1038/ng1195-241

[111] R. C. Elston and J. Stewart, "A general model for the genetic analysis of pedigree data." *Hum Hered*, vol. 21, no. 6, pp. 523–542, 1971.

[112] E. S. Lander and P. Green, "Construction of multilocus genetic linkage maps in humans." *Proc Natl Acad Sci U S A*, vol. 84, no. 8, pp. 2363–2367, Apr 1987.

[113] M. Fishelson, N. Dovgolevsky, and D. Geiger, "Maximum likelihood haplotyping for general pedigrees." *Hum Hered*, vol. 59, no. 1, pp. 41–60, 2005. [Online]. Available: http://dx.doi.org/10.1159/000084736

[114] L. Kruglyak, M. J. Daly, M. P. Reeve-Daly, and E. S. Lander, "Parametric and nonparametric linkage analysis: a unified multipoint approach." *Am J Hum Genet*, vol. 58, no. 6, pp. 1347–1363, Jun 1996.

[115] A. Kurbasic and O. Hfssjer, "A general method for linkage disequilibrium correction for multipoint linkage and association." *Genet Epidemiol*, vol. 32, no. 7, pp. 647–657, Nov 2008. [Online]. Available: http://dx.doi.org/10.1002/gepi.20339

[116] A. Piccolboni and D. Gusfield, "On the complexity of fundamental computational problems in pedigree analysis." *J Comput Biol*, vol. 10, no. 5, pp. 763–773, 2003. [Online]. Available: http://dx.doi.org/10.1089/106652703322539088

[117] J. Andrade, M. Andersen, A. Sillin, C. Graff, and J. Odeberg, "The use of grid computing to drive data-intensive genetic research." *Eur J Hum Genet*, vol. 15, no. 6, pp. 694–702, Jun 2007. [Online]. Available: http://dx.doi.org/10.1038/sj.ejhg.5201815

[118] J. Hernandez-Sanchez, J.-A. Grunchec, and S. Knott, "A web application to perform linkage disequilibrium and linkage analyses on a computational grid." *Bioinformatics*, vol. 25, no. 11, pp. 1377–1383, Jun 2009. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btp171

[119] A. Dix, J. E. Finlay, G. D. Abowd, and R. Beale, *Human-computer interaction.* Pearson, 2003.

[120] A. L. Waldo and G. K. Feld, "Inter-relationships of atrial fibrillation and atrial flutter mechanisms and clinical implications." *J Am Coll Cardiol*, vol. 51, no. 8, pp. 779–786, Feb 2008. [Online]. Available: http://dx.doi.org/10.1016/j.jacc.2007.08.066

[121] P. I. W. de Bakker, R. Yelensky, I. Pe'er, S. B. Gabriel, M. J. Daly, and D. Altshuler, "Efficiency and power in genetic association studies." *Nat Genet*, vol. 37, no. 11, pp. 1217–1223, Nov 2005. [Online]. Available: http://dx.doi.org/10.1038/ng1669

[122] D. B. Goldstein and G. L. Cavalleri, "Genomics: understanding human diversity." *Nature*, vol. 437, no. 7063, pp. 1241–1242, Oct 2005. [Online]. Available: http://dx.doi.org/10.1038/4371241a

[123] H. Zhang, L. Liu, X. Wang, and J. R. Gruen, "Guideline for data analysis of genomewide association studies." *Cancer Genomics Proteomics*, vol. 4, no. 1, pp. 27–34, 2007.

[124] P. C. Sham, S. S. Cherny, and S. Purcell, "Application of genome-wide snp data for uncovering pairwise relationships and quantitative trait loci." *Genetica*, vol. 136, no. 2, pp. 237–243, Jun 2009. [Online]. Available: http://dx.doi.org/10.1007/s10709-008-9349-4

[125] W. P. Hanage and D. M. Aanensen, "Methods for data analysis." *Methods Mol Biol*, vol. 551, pp. 287–304, 2009. [Online]. Available: http://dx.doi.org/10.1007/978-1-60327-999-4_20

[126] G. W. C. Tam, R. Redon, N. P. Carter, and S. G. N. Grant, "The role of dna copy number variation in schizophrenia." *Biol Psychiatry*, vol. 66, no. 11, pp. 1005–1012, Dec 2009. [Online]. Available: http://dx.doi.org/10.1016/j.biopsych.2009.07.027

[127] H. K. Tiwari, J. Barnholtz-Sloan, N. Wineinger, M. A. Padilla, L. K. Vaughan, and D. B. Allison, "Review and evaluation of methods correcting for population stratification with a focus on underlying statistical principles." *Hum Hered*, vol. 66, no. 2, pp. 67–86, 2008. [Online]. Available: http://dx.doi.org/10.1159/000119107

[128] D. Altshuler, M. J. Daly, and E. S. Lander, "Genetic mapping in human disease." *Science*, vol. 322, no. 5903, pp. 881–888, Nov 2008. [Online]. Available: http://dx.doi.org/10.1126/science.1156409

[129] C. Infante-Rivard, L. Mirea, and S. B. Bull, "Combining case-control and case-trio data from the same population in genetic association analyses: overview of approaches and illustration with a candidate gene study." *Am J Epidemiol*, vol. 170, no. 5, pp. 657–664, Sep 2009. [Online]. Available: http://dx.doi.org/10.1093/aje/kwp180

[130] P. J. Taub and E. Westheimer, "Biostatistics." *Plast Reconstr Surg*, vol. 124, no. 2, pp. 200e–208e, Aug 2009. [Online]. Available: http://dx.doi.org/10.1097/PRS.0b013e3181addcd9

[131] J. Cheng and P. Baldi, "A machine learning information retrieval approach to protein fold recognition." *Bioinformatics*, vol. 22, no. 12, pp. 1456–1463, Jun 2006. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btl102

[132] L. Hamel, N. Nahar, M. S. Poptsova, O. Zhaxybayeva, and J. P. Gogarten, "Unsupervised learning in detection of gene transfer." *J Biomed Biotechnol*, vol. 2008, p. 472719, 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/472719

[133] T. S. K. Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, S. Kumar, S. Mathivanan, D. Telikicherla, R. Raju, B. Shafreen, A. Venugopal, L. Balakrishnan, A. Marimuthu, S. Banerjee, D. S. Somanathan, A. Sebastian, S. Rani, S. Ray, C. J. H. Kishore, S. Kanth, M. Ahmed, M. K. Kashyap, R. Mohmood, Y. L. Ramachandra, V. Krishna, B. A. Rahiman, S. Mohan, P. Ranganathan, S. Ramabadran, R. Chaerkady, and A. Pandey, "Human protein reference database–2009 update." *Nucleic Acids Res*, vol. 37, no. Database issue, pp. D767–D772, Jan 2009. [Online]. Available: http://dx.doi.org/10.1093/nar/gkn892

[134] R. Aragues, A. Sali, J. Bonet, M. A. Marti-Renom, and B. Oliva, "Characterization of protein hubs by inferring interacting motifs from protein interactions." *PLoS Comput Biol*, vol. 3, no. 9, pp. 1761–1771, Sep 2007. [Online]. Available: http://dx.doi.org/10.1371/journal.pcbi.0030178

[135] T. I. H. Consortium, "A second generation human haplotype map of over 3.1 million snps." *Nature*, vol. 449, pp. 851–861, 2007. [Online]. Available: http://hapmap.ncbi.nlm.nih.gov/hapmappopulations.html.en

[136] N. Siva, "1000 genomes project." *Nat Biotechnol*, vol. 26, no. 3, p. 256, Mar 2008. [Online]. Available: http://dx.doi.org/10.1038/nbt0308-256b

[137] R. Zhang, H.-Y. Ou, and C.-T. Zhang, "Deg: a database of essential genes." *Nucleic Acids Res*, vol. 32, no. Database issue, pp. D271–D272, Jan 2004. [Online]. Available: http://dx.doi.org/10.1093/nar/gkh024

[138] P. Resnik, "Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language." *Journal of Artificial Intelligence Research*, vol. 11, pp. 95–130, 1999.

[139] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy." *Proceedings of 10th Interna-tional Conference on Research In Computational Linguistics*, 1997.

[140] A. Schlicker, F. S. Domingues, J. Rahnenf'ohrer, and T. Lengauer, "A new measure for functional similarity of gene products based on gene ontology." *BMC Bioinformatics*, vol. 7, p. 302, 2006. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-7-302

[141] J. Z. Wang, Z. Du, R. Payattakool, P. S. Yu, and C.-F. Chen, "A new method to measure the semantic similarity of go terms." *Bioinformatics*, vol. 23, no. 10, pp. 1274–1281, May 2007. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btm087

[142] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li, "Taverna: a tool for the composition and enactment of bioinformatics workflows." *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, Nov 2004. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/bth361