UNIVERSITÀ DEGLI STUDI DI PALERMO

DOTTORATO DI RICERCA IN INGEGNERIA INFORMATICA

DIPARTIMENTO DI INGEGNERIA INFORMATICA

A Knowledge-Based Expert System in Bioinformatics: an Application to Reverse Engineering Gene Regulatory Network

Tutor
Dott. Ing. Daniele Peri

Candidato
Ing. Massimo La Rosa

Coordinatore
Ch.mo Prof. Salvatore Gaglio

Tesi di Dottorato di Ricerca in Ingegneria Informatica – XXII Ciclo
SSD: ING-INF/05

A Knowledge-Based Expert System in Bioinformatics: an Application to Reverse Engineering Gene Regulatory Network

Massimo La Rosa

Department of Computer Science (DINFO)

University of Palermo, Italy

A thesis submitted for the degree of

PhilosophiæDoctor (PhD)

2011 03

To my Parents, Achille and Giusy, for their help and unique support. Thank you

"*L'illuminismo è l'uscita dell'uomo dallo stato di minorità che egli deve imputare a sé stesso. Minorità è l'incapacità di valersi del proprio intelletto senza la guida di un altro. Imputabile a sé stesso è questa minorità se la causa di essa non dipende da difetto di intelligenza, ma dalla mancanza di decisione e del coraggio di far uso del proprio intelletto senza essere guidati da un altro. Sapere aude! Abbi il coraggio di servirti della tua propria intelligenza! É questo il motto dell'illuminismo.Sennonché a questo illuminismo non occorre altro che la libertà, e la più inoffensiva di tutte le libertà, quella cioè di fare pubblico uso della propria ragione in tutti i campi. Ma io odo da tutte le parti gridare: Non ragionate! L'ufficiale dice: Non ragionate, ma fate esercitazioni militari. L'impiegato di finanza: Non ragionate, ma pagate! L'uomo di chiesa: Non ragionate, ma credete!*"

da Risposta alla domanda: che cos'è l'Illuminismo?, Beantwortung der Frage: Was ist Aufklrung?.
Immanuel Kant

"*Could be worse...could be raining*"

Igor

Abstract

The huge amount of biological data has spread the development of plenty of bionformatics tools, databases and web services. In order to face a computational biology problem, there not exist only a way, but different methodologies and strategies, with their own pros and cons, can be applied.

In this PhD thesis I present a knowledge-based expert system that aims at helping a bionformatics researcher in the choice of the proper strategy and heuristic in order to resolve a bioinformatics issue. The Knowledge Base of the system is structured by means of an ontology and codes the expertise about the application domain. KB is organized into decision-making modules that introduce a set of meta-reasoning levels.

The proposed expert system is the core reasoning component of BORIS (Bionformatics Organized Resources - an Intelligent System) framework, a research project High Performance Computing and Networking Institute of National Research Council (ICAR-CNR). BORIS, based on a hybrid architecture, can be seen as a crossover between Decision Support System and Workflow Management System because it not only provides decision support, but it help the User in the proper configuration and running of algorithms, tools and services implementing the suggested strategies and, at the same time, builds a workflow that traces both the decision-making activity and the execution of tasks and tools.

The whole system will be applied to an actual case study: the reverse engineering of Gene Regulatory Network.

Acknowledgements

I would like to acknowledge Dr. Afonso Urso and Dr. Riccardo Rizzo, the researcher of National Research Council in charge of BORIS project. They really believed in me and without their continous support my PhD Work could not exist.

A special thank goes to Professor Salvatore Gaglio for his precious advices and suggestions throughout my work

I would like to thank my colleagues and friends Alfonso Farruggia, Orazio Farruggia, Roberto Gallea, Marco Morana and Antonino Fiannaca for sharing this experience with me, with its happy and sad moments.

And last but not least, I thank Angela, my Girlfriend, for making the last year so special.

Contents

List of Figures

Glossary

ARACNE Algorithm for the Reconstruction of Accurate Cellular Networks

BDE Bayesian Dirichlet equivalence

BIC Bayesian Information Criteria

BN Bayesian Network

BORIS Bioinformatics Organized Resources: an Intelligent System

CDSS Clinical Decision Support System

CASP Critical Assessment of protein Structure Prediction

CLR Context Likelihood of Relatedness

DAG Directed Acyclic Graph

DBN Dynamic Bayesian Network

DPI Data Processing Inequality

DSS Decision Support System

EMR Electronic Medical Record

FG Factor Graph

GGM Graphical Gaussian Model

GRN Gene Regulatory Network

GUI Graphical User Interface

HGP Human Genome Project

JPD Joint Probability Distribution

KB Knowledge Base

KNN K-Nearest Neighboor

MI Mutual Information

KDSS Knowledge-driven Decision Support System

NCBI National Center for Biotechnology Information

ODE Ordinary Differential Equations

PCR Polymerase Chain Reaction

PPI Protein-Protein Interaction

SOM Self Organizing Map

TF Transcription Factor

WFMS Workflow Management System

WKNN Weighted K-Nearest Neighboor

1

Introduction

In the late 70s, Computer Science algorithms and statistics have begun to be applied for the analysis and the study of problems related to molecular biology. With the first attempts of DNA sequencing and especially with the beginning, in 1990, of the Human Genome Project (HGP) (1), this type of *in silico* approach, rather than *in vivo* or *in vitro*, grew in importance. A new discipline, called Bionformatics, was born with the aim of highlighting the raising need of merging Computer Science methodologies and techniques with the management and analysis of biological data.

It is not simple to provide a synthetic definition of Bioinformatics. According to the National Center for Biotechnology Information (NCBI) (2) "Bioinformatics is the field of science in which biology, computer science, and information technology merge to form a single discipline. The ultimate goal of the field is to enable the discovery of new biological insights as well as to create a global perspective from which unifying principles in biology can be discerned".

The type of biological data typically considered are DNA and protein sequences, protein structures, gene expressions, protein expressions, protein complexes, protein-protein interactions (PPI).

In this scenario, researchers have begun to develop computational techniques in order to analyse these data, applying well established Artificial Intelligence approaches, such as Pattern Matching, Data Mining and Machine Learning algorithms, and adapting them to the biological evidences.

Bioinformatics has provided its major efforts in various application domains, including among the others: sequence alignment, gene identification, drug discovery and design, protein structure alignment, protein structure prediction, prediction of protein-protein interactions, inference of metabolic and regulatory networks.

1.1 Motivation and Goals

In the past few decades the continuous growing amount of biological data, thanks to the developing of high throughput technologies, has also given a boost to the number of both bionformatics tools and algorithms and both to the availability of web services and biological databases.

Nowadays, in fact, researchers facing biological problems are overwhelmed by the huge set of computational techniques and enormous amount of data available: for any problem, there are many possible models and algorithms, each of them with their own characteristics, giving different results. Given a biological issue, there are potentially plenty of different tools that could be used, none of them providing the best possible results. Just to make a quick example, for the prediction of the tridimensional structure of a protein from its amminoacid sequence, also known as primary sequence, there exist more than 70 software (30), called structure predictors, that offer different performances on the basis of the intrinsic properties of the analysed protein. It means there is not just one predictor that always gives the best result, but each software has its own strengths and weaknesses.

This situation has led to an increasing need for a computational system that can respond to the afore mentioned issues.

In this thesis it will be presented an intelligent system, named BORIS, whose main goals can be summarized as follows:

1. to collect the most common and used bionformatics tools and services and to give them a coherent and flexible structure;

2. to offer support to the bionformatics researcher about the decision-making process in the choice of the best suited algorithm and service. This decision-making activity is built on a set of heuristics and strategies representing the expertise about the application domain;

3. to help the bionformatics researcher in the proper configuration and running of the selected tools;

4. to build a path, or workflow, where both the decision phases and the execution phases can be tracked down;

All of these directives will be formalized in the guidelines of BORIS project, presented in the next Chapter. BORIS is the main research project where my PhD work is born and carried on.

In a very general way, it is possible to say that the basic idea of BORIS system is, then, to provide to the researcher, or experimentalist, not only the tools able to resolve a problem, but also the knowledge used in order to justify the choice of those specific tools and strategies. In the generated workflow representing the execution of an experiment, then it will be shown not only a simple succession of tasks, but also what is the conceptual scheme at the basis of that workflow. From this point of view, BORIS system can be seen as a novel intelligent system that represent an innovative crossover between classical decision support systems (DSS) and the most recent workflow management systems (WFMS).

In this thesis I will present what is my contribution to the BORIS project.

1.2 Background

History of Decision Support Systems and expert systems traces back to 70s (3), when Gorry and Scott-Morton (4) used first the term Decision Support System, defining it as "an interacting computer-based system that helps the decision maker in the use of data and models in the solution of unstructured problems". Definitions and features of a DSS are actually very wide: Moore and Chang focus on their extendibility and adaptability to different domains (5); other reviews emphasises the flexibility in the decision contest change (6), while in (7)

is highlighted the chance to deal with semi-structured problems. More recently, authors of (8) introduced the concept of intelligent decision maker by using tools and techniques of Artificial Intelligence in order to gain direct access to expertise for supporting the decision-making process: this integration provides higher accuracy, reliability and utility.

Knowledge-driven DSS (KDSS) is a category of DSS built using an expert system (9). These systems have their own expertise based on knowledge on many aspects of the problem: the application domain, the definitions of problems within that domain and the necessary skill to solve them (10). The knowledge of the system is often coded as a set of rules by one or more human experts: this kind of systems are often referred to as rule-based expert systems.

KDSSs applied in diagnosis in various clinical domains take the name of Clinical DSS (CDSS). In CDSS a medical knowledge base is integrated with a collection of patient data and an inference engine in order to provide medical recommendations about cases of a specific pathology.

MYCIN (12) was one of the first and most famous CDSS used for the diagnosis and treatment of some blood infections. It was written in Lisp (13), and offered support in the recommendations of the type and dosage of antibiotics.

ONCOCIN (14) is a rule-based expert system developed at Stanford University in order to give support with the treatment of cancer patients receiving chemotherapy. ONCOCIN also introduced a sort of flowchart language in order to keep trace of the sequence of decisions over the time.

KON3 (15) is a CDSS that adopts a proper ontology and rules, starting from unstructured databases of medical records and clinical guidelines, in order to give advices for care process patients.

Other currently used CDSS are: ATHENA (16) implementing guidelines for hypertension using Stanford Medical Informatics EON architecture (17); LISA (18) that is a clinical information system for supporting collaborative care in the management of children with Acute Lymphoblastic Leukaemia (ALL); TherapyEdge (19) that is a web-enabled decision support system for the treatment of HIV.

The proposed system improves the classical concept of DSS in many ways. First of all, for all decision steps it suggests a list of suitable strategies or algo-

rithms presenting for all of them a brief description, a series of pros and cons and a list of bibliographic references. The system also helps the user in the proper configuration and running of the strategies or algorithms selected during the decision making process.

However the main feature is that all decisions made during an experiment are ordered and visualized on a workflow where the user can backtrack in order to change a previous decision. Furthermore it is possible to save the whole workflow and its results in order to share or reuse them.

These last features make our systems closer to modern Workflow Management Systems (WFMS) (20). A WFMS is a computer program that is able to manage and define a series of tasks and processes in order to provide one or multiple outputs. WFMS can be applied in different application fields and they can take into account several types of jobs.

In bioinformatics domain, WFMS provide a simple way to build and run a custom experiment using the most common bioinformatics resources, like online databases, software and algorithms. WFMS, however, do not interact with the user, do not have a knowledge base, nor makes decision like KDSS: for this reason our system represents an ideal merging point between classical DSS and emerging WFMS.

The most used and famous WFMS for bioinformatics is Taverna (21): it is able to automatically integrate tools an databases available both locally and on the web in order to build workflows of complex tasks; to run the workflows and to show results in different formats. The system works by means of a Graphical User Interface (GUI) or a script language.

Other WFMS for bioinformatics are Biowep (22), that allows the user to search and run a predefined set of workflows, already tested, validated and annotated; and BioWMS (23), that is a web-based WFMS built upon an agent-based middleware architecture.

1.3 Dissertation outline

The rest of the thesis will be organized as follows:

In Chapter 2 BORIS project will be described: it is the main research project at the basis of my thesis work.

In Chapter 3 my contribution to BORIS project will be presented: it represents the core of my PhD work.

In Chapter 4 the Graphical User Interface (GUI) and the main features of BORIS will be shown.

In Chapter 5 the application of the proposed system to an actual case study will be described.

In Chapter 6 a brief explanation of all tools and methods used in order to develop the proposed system will be provided.

1.4 Publications

During the 3-years PhD course the following publications has been produced:

- FIANNACA A, LA ROSA M., PERI D, RIZZO R (2011). An Intelligent System for Decision Support in Bioinformatics. ERCIM NEWS, vol. 84; p. 35-36, ISSN: 0926-4981

- FIANNACA A, GAGLIO S, LA ROSA M., PERI D, RIZZO R, URSO A (2010). A Proposed Knowledge Based Approach for Solving Proteomics Issues. Revised selected papers. LECTURE NOTES IN COMPUTER SCIENCE, vol. LNCS 6160; p. 304-318, ISSN: 0302-9743, doi: 10.1007/978-3-642-14571-1

- FIANNACA A, GAGLIO S, LA ROSA M., PERI D, RIZZO R, URSO A (2010). A Knowledge Based Decision Support System for Bioinformatics and System Biology. In: Proceedings of CIBB 2010. Palermo, Italy, 16-18 September 2010, ISBN/ISSN: 978-88-95272-87-0

- LA ROSA M., RIZZO R, URSO A, GAGLIO S (2009). Normalized Compression Distance and Evolutionary Distance of Genomic Sequences: Comparison of Clustering Results. INTERNATIONAL JOURNAL OF KNOWLEDGE ENGINEERING AND SOFT DATA PARADIGMS, vol. 1; p. 363-375, ISSN: 1755-3210, doi: 10.1504/IJKESDP.2009.028988

- FIANNACA A, GAGLIO S, LA ROSA M., PERI D, RIZZO R, URSO A (2009). A Proposed Knowledge Based Approach for Solving Proteomics Issues. In: Proceedings of CIBB 2009. Genova, Italy, 15-17 Ottobre 2009, vol. 3, ISBN/ISSN: 978-88-903537-2-7

- FIANNACA A, GAGLIO S, LA ROSA M., PERI D, RIZZO R, URSO A (2009). A Decision Support System for Reverse Engineering Gene Regulatory Networks. In: Proceedings of SysBioHealth Symposium 2009. Milano, Italy, November, 25-27 2009, p. 81-83

- LA ROSA M., RIZZO R, URSO A, GAGLIO S (2008). Comparison of genomic sequences clustering using Normalized Compression Distance and Evolutionary Distance. In: Knowledge-Based Intelligent Information and Engineering Systems. Zagabria, Croazia, September, 3-5 2008, HEIDELBERG: Springer, vol. LNAI 5179, p. 740-746, ISBN/ISSN: 978-3-540-85566-8

2

Bioinformatics Organized Resources - an Intelligent System

In this Chapter BORIS, an acronym standing for Bioinformatics Organized Resources: an Intelligent System, will be presented. BORIS is a research project developed by the National Research Council and it represents the framework in which my PhD work has given its contribution.

BORIS main features and guidelines will be presented, focusing the attention on its hybrid architecture and development paradigms.

My contribution to BORIS project will be shown in the last Section.

2.1 BORIS Project

PhD work has been carried out inside one of the active project of High Performance Computing and Networking Institute of National Research Council of Palermo, Italy (ICAR-CNR), entitled: "B.O.R.I.S, Bioinformatics Organized Resource - an Intelligent System", under the supervision of project manager Dr. Alfonso Urso, belonging to research group "Analisi Intelligente di Dati per la Bioinformatica".

BORIS project was born from a threefold need:

1. to give a solid and coherent structure both to bioinformatics issue and the plenty of tools and services that operate on bioinformatics domain;

2. to offer support to a bioinformatics researcher during the decision-making process of an experiment;

3. to help the user in the building and execution of pipeline of software and services.

2.2 BORIS Guidelines

Following the three global requisites written in the previous Section, a set of guidelines and functional requirements have been outlined.

Main guidelines of Boris project are the design and implementation of a Decision Support System (DSS) that can help bioinformatics researchers to deal with the plenty of tools and services currently available. The system should collect and organize the most common and used bioinformatics resources, such as bioinformatics tools, web services and biological databases in order to make them accessible to a bioinformatics researcher.

The system should deal with the unstructured knowledge typical of a human expert of the domain that turns into the formalization of heuristics and strategies. BORIS should give to the User support in terms of decision-making activity and execution phase. The former requirement means BORIS should suggest to the User what is the proper methodology to follow in order to resolve a problem and, once the strategy has been set, it should suggest the best tool in order to fulfil it. The latter requirement makes BORIS closer to actual Workflow Management Systems (WFMS), since it is also responsible for the configuration and running of all external tools specified during the planning phase.

Moreover BORIS should provide a flexible and modular framework in order to maintain its constituents parts in an independent way and it should offer a developing platform that can be easily updated and enhanced with new functionalities and new kind of application domains. BORIS should define a standard protocol so that developers community can add his own knowledge and expertise to the system.

2.3 BORIS Hybrid Architecture

Basically in Artificial Intelligence two main programming paradigms are employed in order to implement a Decision Support System (DSS): procedural approach and declarative approach. Intelligent architectures based on a declarative approach encodes the knowledge, using a symbolic representation, about a specific problem so that it can be manipulated and managed by an inference engine: that means the knowledge is referred to the control of reasoning process and it represents the concept of "what to do" given a particular environmental context. The purpose of a declarative program is to generate a plan of action. Other features of declarative programming are:

- the possibility to easily maintain and update the system representation;
- the improving of system transparency, since it is known what is the general flow of control of the program;
- a certain slowness of the execution because the program code has to be interpreted
- the system is data-oriented, because according to the input data, the declarative program can generate a different plan

On the other hand, procedural knowledge specifies all the needed instructions in order to achieve a goal. This type of knowledge is explicitly represented and it is aimed at describing "how to do something". This type of approach is particularly suited for deterministic processes in which there is no need of any kind of reasoning activities and there exists only one way to reach the fixed goal. Procedural programming is the most common paradigm in the development of algorithms. A procedural program has also the following qualities:

- it can be seen as a black box with an input and an output;
- it can be hard to debug;
- it is fast to run;

- it is process-oriented;
- it is easy to write for the first time but it can be difficult to maintain, according to the adopted "programming style".

BORIS architecture has been designed following the main idea of integrating these two computational approaches in order to create a hybrid intelligent system that could inherit the main advantages of both paradigms.

Apart from procedural and declarative approach, another methodology has been considered in the developing of BORIS. According to BORIS guidelines, in fact, the system should offer support not only in the form of a simple textual advice, but it has to trace all the decisions confirmed by the user and, moreover, it has to help the user in the proper configuration and running of the suggested instruments. For this reason, it has been considered the so called "process approach", inherited from business process management and currently carried out by modern Workflow Management System (21, 22, 23). With the expression "process approach" it is meant a collection of structured activities, called tasks, that provides a particular service or product and that can be visualized using a flowchart or a workflow.

2.3.1 BORIS Reference Space

The three-folded approach, i.e. declarative, procedural, process, at the basis of BORIS architecture has been formalized through the generation of a three dimensional reference space where each kind of approach has been mapped into a different axis. Then each point in this reference space, that we can simply call system space, represents the state of the system, whereas the projection of this point into the single axis gives the contribution of each approach.
The three axes are called: Abstraction Layer, Decision Making Level and Workflow Timeline and their reference system is shown in Fig. 2.1. Here the basic features and ideas behind each axis will be provided.

1. Abstraction Layer Axis, mapping the Procedural approach:
    - it is responsible for the "how to do" part of the system

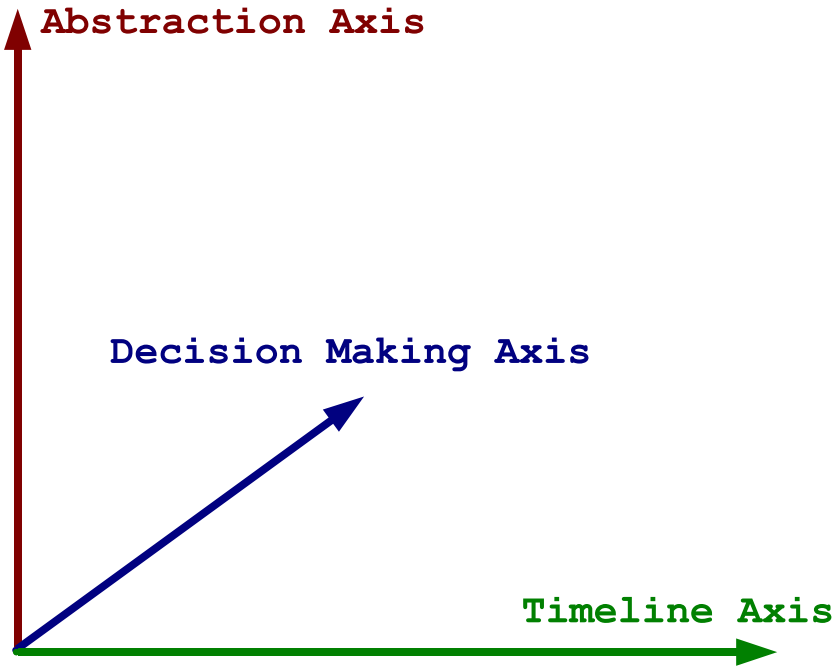

Figure 2.1: The 3-dimensional reference space of BORIS system. In each axis it is mapped one of the different adopted computational approach: in the abstraction axis it is mapped the procedural approach; in the decision-making axis it is mapped the declarative approach; in the workflow timeline axis it is mapped the process approach.

- it decomposes input problem into a set of one or more sub-problems of lesser complexity;
- it runs algorithms and services, guaranteeing fully interoperability among them.

2. Decision Making Level Axis, mapping the Declarative approach:

   - it is responsible for the "what to do" part of the system;
   - it implements the reasoning activity
   - it deals with unstructured data;
   - it uses strategies and heuristics in order to generate a plan of action;

3. Workflow Timeline Axis, mapping the Process approach:

    - it builds a workflow of operations, according to the executed tools by the Abstraction Axis;
    - it allows the configuration of each algorithm and service in the workflow;
    - it provides backtracking features, in the sense that it is possible to select any part of the workflow in order to obtain a different flow of execution or to follow alternative paths during the decision making activity;
    - it can save the generated workflow for sharing it with other users or to re-use it in other experiments.

The three axes contain only discrete values. On the abstraction axis we have high values representing high abstraction layers referred to the whole problem, while low values represent operations at low abstraction layers like for instance the execution of a specific algorithm. In the decision making axis it is traced the succession of every reasoning step of the system. In the timeline axis the workflow is updated only when a tool or service is run, so that it is possible to follow the progression of the experiment and eventually to modify it.

2.4 Contribution to BORIS

My PhD work has been developed according to BORIS functional requirements and it has been designed with the aim of integration with BORIS hybrid architecture.

My PhD work consisted in the design and implementation of a knowledge-based expert system representing the core component of the reasoning process of the Decision Support System. The structure of the expert system responds to the three-folded approach of BORIS architecture.

My work focused first of all on the organization and definition of a Knowledge Base (KB) used to provide expertise in bioinformatics domain. The knowledge-base is built upon an ontology, representing an essential framework in order to

obtain a coherent and consistent KB. The KB has been enriched with a set of rules, extracted from more than 50 scientific papers, that code the skill of a human expert of the domain. Rules are the key elements to let the inference engine of the expert-system make reasoning and to deduce new knowledge and conclusion in the form of suggested operations. The final aim of the expert-system is, in fact, to make, by consulting the KB, those necessary inferences in order to produce a suggestion that will be managed by the Decision Support System and the workflow manager. The KB is organized so that it can be directly mapped to the three dimensional space defined in the previous section, thanks to the definition of a set of *decision making modules* and their organization as a topological tree.

The expert system belongs to a more complex software architecture, characterized by a multi-layer structure that allows modularity and the possibility to change and improve every single component without modifying the rest of the system. Apart from the expert system my work also consisted in the definition, design and implementation of the Graphical User Interface (GUI) of BORIS system. Furthermore the system has been given the possibility to communicate and interface with a large set of bioinformatics tools and services in order to accomplish and run the proposed methodologies obtained as result of the decision making activity.

Finally the proposed system, fully integrated into BORIS architecture, has been tested on a bioinformatics scenario. Considering the state-of-the-art, it has been chosen the case study regarding the reverse engineering of gene regulatory networks (GRN). Inferring a GRN is crucial in order to realize the biological mechanism that regulates the gene expression phenomenon inside cells. There exists a lot of different approaches, strategies and techniques to face this problem, according to the available type of data and resources, so our system can offer support, both in the decision making process and in the execution phase. More than 20 research papers found in literature have been used in order to populate the KB with the necessary skill, and strategies and heuristics has been coded into a set of specific rules. A complete experiment will be shown, highlighting the behaviour of BORIS with regards to the hybrid architecture and the rule-based expert system.

3

BORIS Software Architecture

In this Chapter, the software architecture of the Rule-Based expert system developed for the BORIS global framework is presented. The design ad implementation of this expert system is one of the main contribution of my thesis work.

First of all the whole architecture will be described and then its most important parts will be described in detail, stressing on the organization of the Knowledge Base of the system through an ontology and introducing the concept of Decision-Making module, that are responsible for the reasoning activity of the system.

Finally we will see how the software architecture integrates with the rest of BORIS hybrid structure, following its main requisites and guidelines.

3.1 Three-layer Architecture

Boris software architecture has been developed as a three layers structure. The layered architecture of the proposed system, shown in Fig. 3.1, is inspired by its main goal: to separate the researcher from the tools in order to let him focus on the problem.

The user interacts with the system through a Graphical User Interface (GUI) and the wrapper component that are in the interface layer. The wrapper is the module that manages the communication between the executor in the Controller layer and and the GUI. The GUI sends user's commands to to the wrapper; it formats this messages in the form of queries to the Reasoner. Wrapper module,

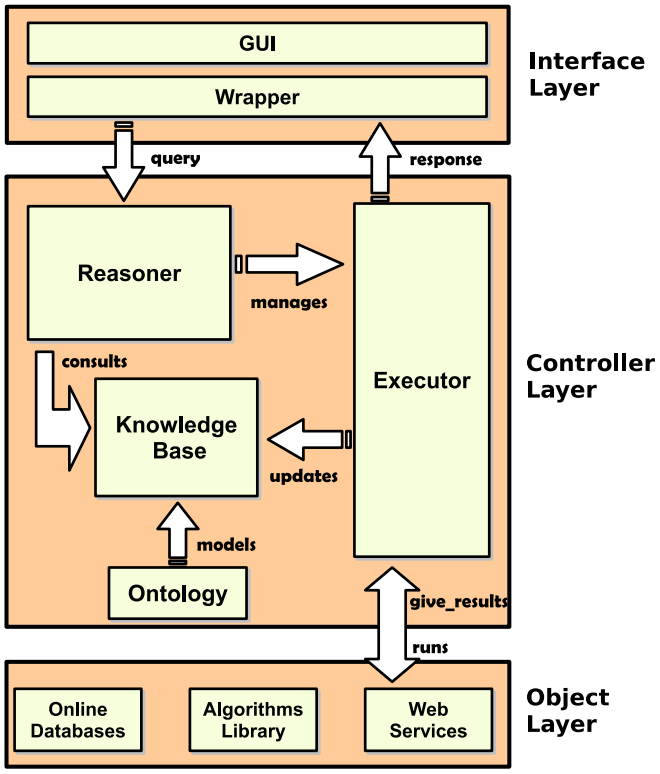

Figure 3.1: Software Architecture. Three main layers interact each other to make the system work. The Interface layer is responsible for the interaction between the User and the system. It implements a GUI that manages the input/output operation. The Controller layer holds a Knowledge-Based expert system: it is able to make inferences on the application domain by consulting the skill coded into the Knowledge-Base. KB is organized and maintained through an ontology. The decisions taken by the Reasoner (inference engine) are passed to the Executor that will schedule and put them in action. The Object layer represents all the tool and services the system can gain access, both locally and on the Internet. Every time a new web service or software is available, the upper layer just needs a simple interface in order to use them.

moreover, allows to to easily change the GUI without interferences to the other parts of the system. The main components and their meaningful features of the GUI will be described in Chapter 4.

The Controller Layer includes a knowledge-based expert system (9). Knowledge-

Base (KB) contains and codes the expertise of the system about the application domain. KB can be populated with information provided by human experts of the domain or extracted by research papers found in literature: so far we used for our KB almost 50 scientific papers. Knowledge Base is composed of facts and rules: facts represent single pieces of information; rules, having the typical form **IF** <*precondition on fact is TRUE*> **THEN** <*do action*>, are used in order to make the system able to do inferences about the domain. The rules, acting on facts, have to be considered single steps of reasoning used for the coding for heuristics, guidelines and strategies adopted by an expert of the domain. The KB is built upon an ontology, in order to provide a fixed and expandable structure to the KB itself. Facts, then, represent instances of the concepts defined in the ontology. The relationships between ontology, facts and rules are shown in Fig. 3.2

The design and the main features of our ontology will be described in the next subsection.

The Reasoner is the inference engine. It, by consulting the knowledge base and according to user's query and input data received from the upper layer, has to decide and suggest what are the suited strategies and tools useful in order to solve user's request. All the decision taken by the Reason are sent to the Executor. It has an internal agenda, in order to schedule the action to perform, and moreover has access to the Object Layer which contains all the tools and software available to the system. The Executor can update the KB with intermediate results obtained during the execution of an experiment.

The Object layer represents all the low level parts that will be run by the executor, according to the decision taken by the reasoner. The Object layer can be considered as a big container, made up of different compartments, corresponding to different class of software and tools. In this layer we considered algorithms and tools and the access to the most common web services and online databases for bioinformatics. All the components of Object layer are developed by third parties and are not subject of our study.

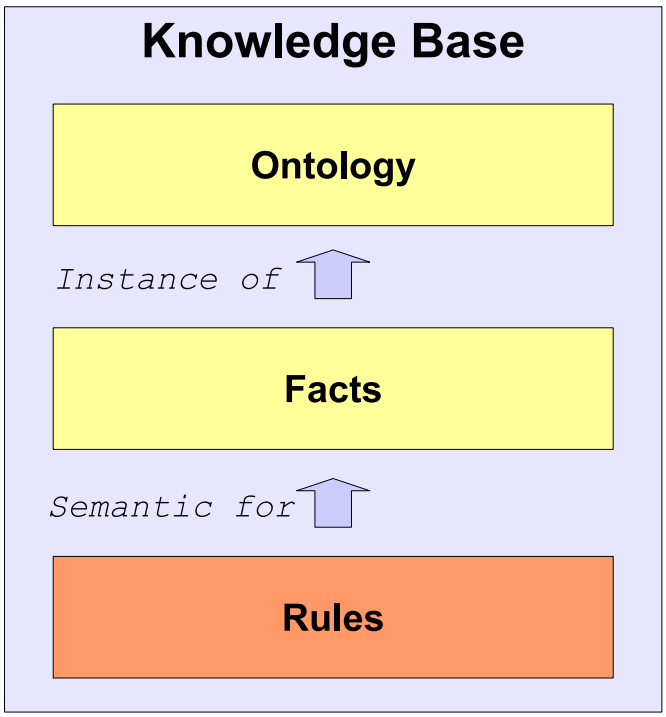

Figure 3.2: Knowledge-Base three main components: Facts are the instances of the concepts defined into the Ontology; the Rules work on facts in order to give semantic and the possibility to make inferences over the facts.

3.2 Ontology Design

In order to build a complete and exhaustive Knowledge Base, three basic components are needed: facts, rules and an ontology of the domain.

In Computer Science, an ontology is a formal representation of the knowledge about a specific domain. It provides a conceptual schema for all facts to be represented. The main reasons for developing an ontology are:

- to share the structure of information among other people or software agents;
- to allow the reuse of domain knowledge;
- to give a well structured, robust and consistent conceptual schema for all facts to be represented

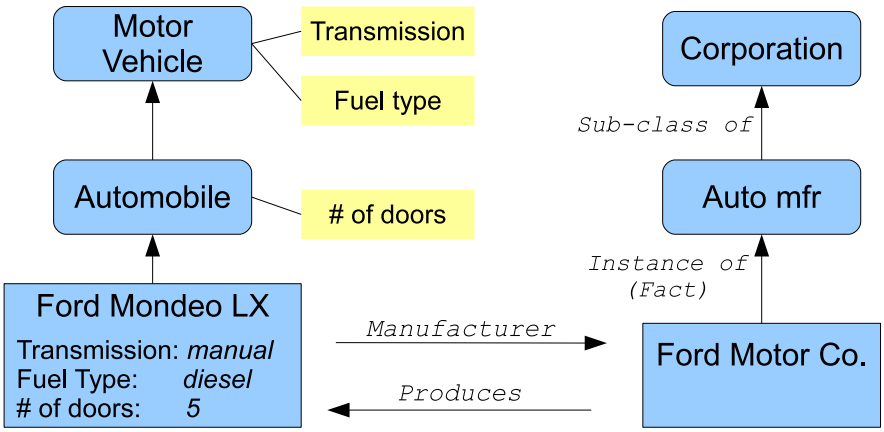

Figure 3.3: An example of ontology in the vehicle domain. The rounded rectangles are the classes characterized by properties or attributes (the yellow boxes); the other rectangles are instances of the classes: in the instances each attribute has a value. Finally there are the relationships between the instances, indicated through the black arrows. Each relationship is given a label representing the type of bind.

- to enable the chance of easily update and extend the KB with new concepts.

Ontology is composed of classes (concepts) organized in a hierarchical structure. Classes are characterized by properties, also called attributes, that describe various features of the concept itself; and relationships with other classes of the domain. Given this definition, facts of the KB represent instances of concepts defined into the ontology.

In Fig. 3.3 it is shown a very simple scheme of an ontology, describing the motor vehicle domain. There we have the concept Automobile and its super concept Motor vehicle, with the set of attributes indicated in yellow. A child concept inherits all the properties from its parent concept. "Ford Mondeo LX" is an instance of Automobile class: in general an instance has all its attributes with a specific value. Moreover, this instance has a mutual relationships of the type manufacturer/producer with another instance of the domain, "Ford Motor Co.". The latter element is an instance of "Auto mfr" class that is a subclass of "Corporation" concept.

In practical terms, developing an ontology includes:

1. identifying and defining classes in the ontology,
2. organizing the classes in a taxonomic (subclasssuperclass) hierarchy,
3. defining slots (attributes) and describing allowed values for these slots,
4. filling in the values for slots for instances, this way obtaining facts for the KB.

In the development of the ontology at the basis of our KB, we decided to focus on and model three main sub-domains:

1. the set of tasks we can do on bioinformatics domain;
2. the tools, software and algorithms currently used in bioinformatics;
3. the type of biological data we have as input and that we have to analyse (we call it generically "Domain").

These three sub-domains are shown in Fig. 3.4, where we can also see the kind of relationships among them: Tasks *operate on* a specific biological data, following the idea "what we can do according to the available type of data"; on the other hand Tasks *use* Tools, in the sense: "in order to do something, what are the suited tools?".

All three main branches of our ontology are modelled according to an hierarchy of classes and subclasses. Each concept is then characterized with a set of attributes and relationships with other concepts: sub-classes, representing more specialized concepts, have all the attributes of their own super-classes plus other specific properties.

The Tasks part of the ontology describes what are the most common bioinformatics operations we can do on biological data. Here, at the moment, we identified three main areas of our interest: Protein Analysis; Protein-Protein Interaction; Gene Regulatory Network. The hierarchical structure of Tasks ontology is shown in Fig. 3.5

Protein Analysis is one of the biggest challenge in bioinformatics: it is a very hard issue to understand how proteins work in biological processes. In

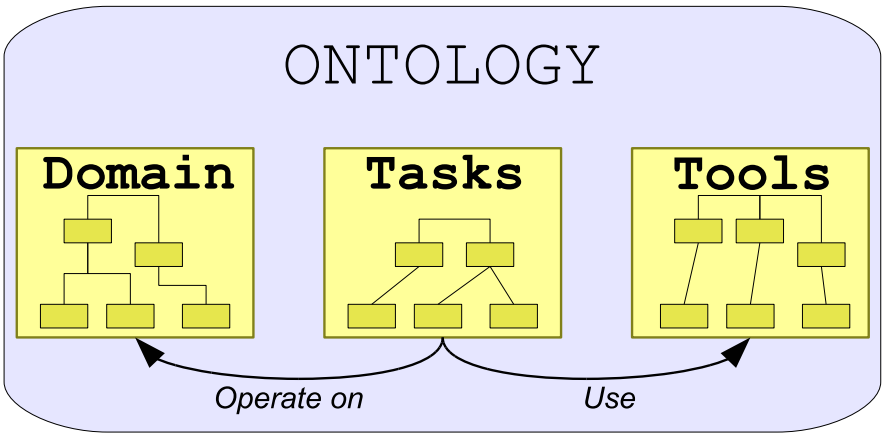

Figure 3.4: Three main ontology sub-domains: Tasks model the set of operations it is possible to do on the bioinformatics domain; Tools model the set of algorithms and services that implements the Task instances; Domain models the biological data to analyse. The generic relationships among the three subdomains are shown: Tasks "operate on" Domain's instances and "uses" Tools' instances.

facts, the proteome of a specific organism differs even from cell to cell, this is because a single gene can code for over 1,000 proteins and each protein can express several functionality, according to other interacting proteins. According to bioinformatics topics classification in (24), protein analysis is divided into four classes of problems: protein structure prediction, protein annotation, protein function prediction, and protein localization prediction.

- Structure Prediction: the structure of a protein represents a key feature in its functionality (25). Unfortunately, the prediction of 2D and 3D structures is an NP hard problem in general, because most of the proteins are composed by thousands of atoms and bounds and the number of potential structures is very large. For this reason, in order to approximate the real structure of a protein, several optimization techniques based on machine learning approaches have been implemented and a competition (CASP (30)), aiming at improving prediction techniques in the years, has been instituted;

- Function Prediction: another challenge is to determine protein function at

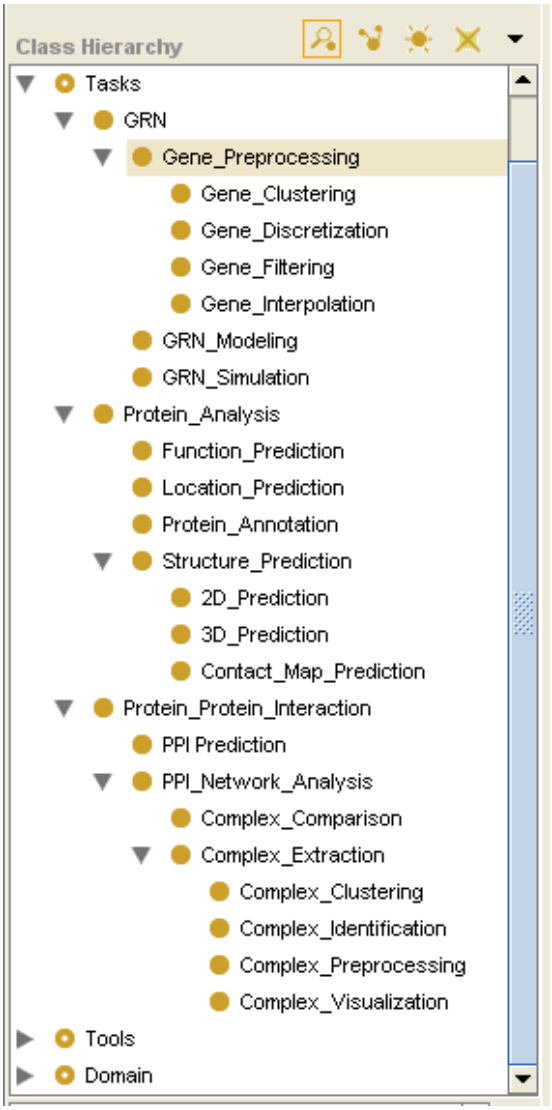

Figure 3.5: Hierarchy of classes and subclasses for the Task part of the proposed ontology.

the proteomics scale. In fact, although in a model organism many individual proteins have a known sequence and structure, their functions are currently unknown. In particular, a single protein can express different function according to some environmental parameters, therefore it is not enough to identify which proteins are responsible for diseases or are advised for medical treatments, if the specific functions are unknowns. Approaches to the

function prediction are based on different techniques (28): some of these are related to protein sequence and structure, the other ones use protein-protein interaction patterns and correlations between occurrences of related proteins in different organisms.

- Location Prediction: the prediction of protein localization aims at determining localization sites of unknown proteins in a cell. By means of this study, it is possible to cope with problems like genome annotation, protein function prediction, and drug discovery. The location of protein into the cell can be calculated through experimental approaches (29), but they are time and cost consuming, thus a computational technique able to screen possible candidates for further analyses, appears a desirable solution.

- Protein Annotation: available databases and technical information on proteins form the raw material of the proteomics. A correct organization of these input data prevents a misleading interpretation of elements. A critical phase in this process is a correct annotation of properties and main features of proteins. This step is based on the classification of scientific texts and the information extraction in the biological domain (27), and it copes with the identification problems. In the biological field the nomenclature is highly variable and ambiguous, especially for protein name identification, where both the use of phenotypical descriptions and the gene homonym/alias management have influenced the nomenclature.

A central role in biological mechanism of a cellular process is covered by the analysis of protein-protein interaction (PPI). Nowadays a large amounts of PPI data have been identified with many technologies, but only a few of them are account as real interaction with an emerging function. Moreover, at biological pathway level, the functionality is not linked to a simple pair of proteins, but arises with protein complex, that is a collection of PPIs. Analysis of protein-protein interaction, as well as identification and extraction of protein complexes, represents an hard task for machine learning algorithms (26), because uncertain information about interconnection and functionality of each protein could lead to erroneous interpretation. Inside Protein-Protein Interaction we distinguished subtasks like

PPI prediction and PPI network analysis. PPI network analysis is composed of techniques for the extraction of protein complexes and the comparison of protein complexes.

Finally Complex Extraction class is made of the following subclasses:

- Complex Clustering is the identification of set of protein complexes characterized by common features according to a similarity metric;
- Complex Identification is the classification of unknown protein complexes given a training set of known complexes;
- Complex Preprocessing is the set of operations used in order to prepare the input dataset the complex clustering or complex identification tasks;
- Complex Visualization is a set of techniques that allow the visualization, using different styles, of the protein complexes.

GRN ontology is made of subtasks concerning preprocessing of data, network inference and visualization. Inside Preprocessing jobs, we also considered:

- Gene clustering is the individuation of set of genes, called clusters, that exhibit similar expression values, according to a specific metric (31, 32). These gene are also defined "coexpressed". Each gene cluster is given an expression value equal to the mean value of all its elements or equal to the value of its most representative gene (cluster centre);
- Gene interpolation consists in the increasing of the number of data points (expression values) in order to obtain more accurate results (33);
- Gene discretization is a numerical procedure to transform continuous expression values into discrete values, because some tools need this type of input values, such as Bayesian Networks (34);
- Gene filtering is a set of procedures that allow to select a subset of input genes according to some user defined constraints (35, 55).

Instances of Tasks ontology has the following attributes:

- *description:* a brief explanation;
- *entry:* the type of biological data needed;
- *exit:* the type of output data produced;
- *precondition:* required task to be previously run
- *pros:* a list of task's advantages;
- *cons:* a list of task's weak points;
- *reference:* one or more bibliographic references;
- *input type* (only for GRN instances): the type of input data supported (see Microarray instance description)

The Tools component is also structured in an hierarchy: here, at the top level, we distinguish between algorithms, that are run locally, and services, that run remotely. At the moment we included filters, algorithms on graphs, graphical models, machine learning algorithms etc... The complete hierarchy is shown in Fig. 3.6.

A generic instance of Tools ontology is characterized by the following attributes:

- *description:* a brief explanation on its main features;
- *input:* type of input data (file format);
- *output:* type of output data;
- *parameters:* number and type of input parameters, if needed;
- *pros:* a list of algorithm's strong points;
- *cons:* a list of algorithm's weak points;
- *complexity:* computational complexity;
- *reference:* one or more bibliographic references;

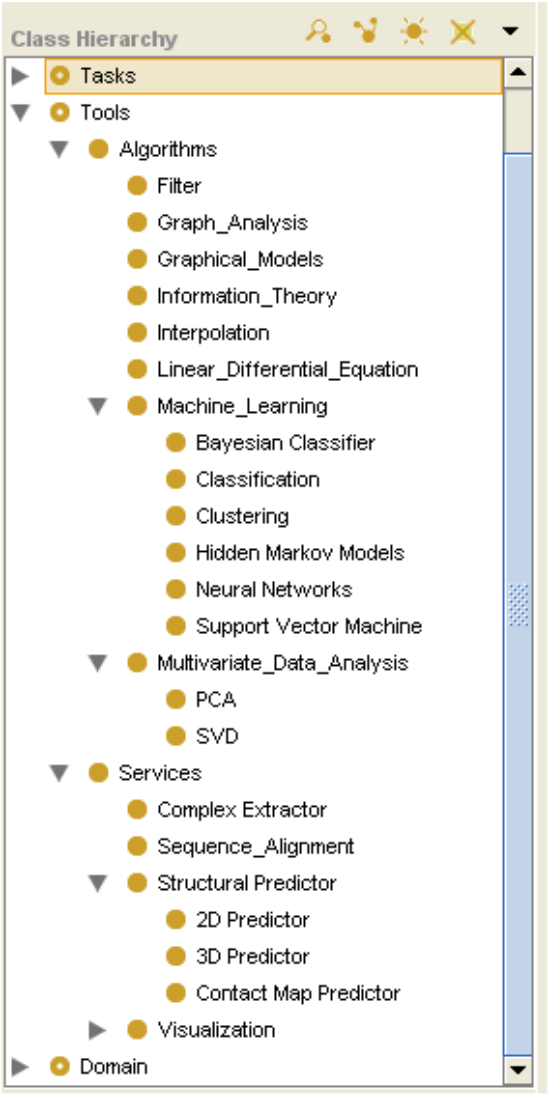

Figure 3.6: Hierarchy of classes and subclasses for the Tools part of the proposed ontology.

In the last main branch of our ontology we modelled the type of biological data we want to analyse: we considered genomic data, proteomic data and transcriptomic data (see Fig. 3.7). Here we focused especially on the modelling of microarray data, since this is at the basis of the developed scenario presented in Chapter 5

Microarray class has the following attributes:

- *db:* the biological database the dataset belongs to;
- *genes:* the number of input genes;
- *samples:* the number of input samples;
- *experiment:* the kind of experiment: time-series or steady state (see Section 5.2.1)
- *species:* the type of biological species (if known);
- *missing values:* the presence or less of missing values.

Apart from an hierarchy of classes and subclasses, an ontology is characterized by the relationships among those classes. We are interested above all in the relationships between Tasks and Domain on one hand, because we want to identify what is the type of biological data needed to perform a specific operation; on the other hand we are interested on the relationships between Tasks and Tools, because we want to know what are the available instruments that actually implements strategies and heuristics coded in the Tasks ontology. For this reason, At the top level we have defined a mutual relation between Tasks and Tools: an instance of Tools "resolves" an instance of Tasks, that conversely is "resolved by" an instance of Tools. We want this way point out that a particular software or algorithm is suited to be applied to a particular bioinformatics task.

3.3 Decision-Making modules

In order to make the system more efficient and structured, facts and rules of the KB are organized into a set of *decision-making modules*.

A decision-making module, from now on simply *module*, is a collection of specific facts and rules with common features. We can assign to each module a well defined scope and purpose, a specific slice of the decision-making process.

For example, we can have modules suited for taking decisions about preprocessing operations, visualization, clustering and so on that can be used in different application domain.

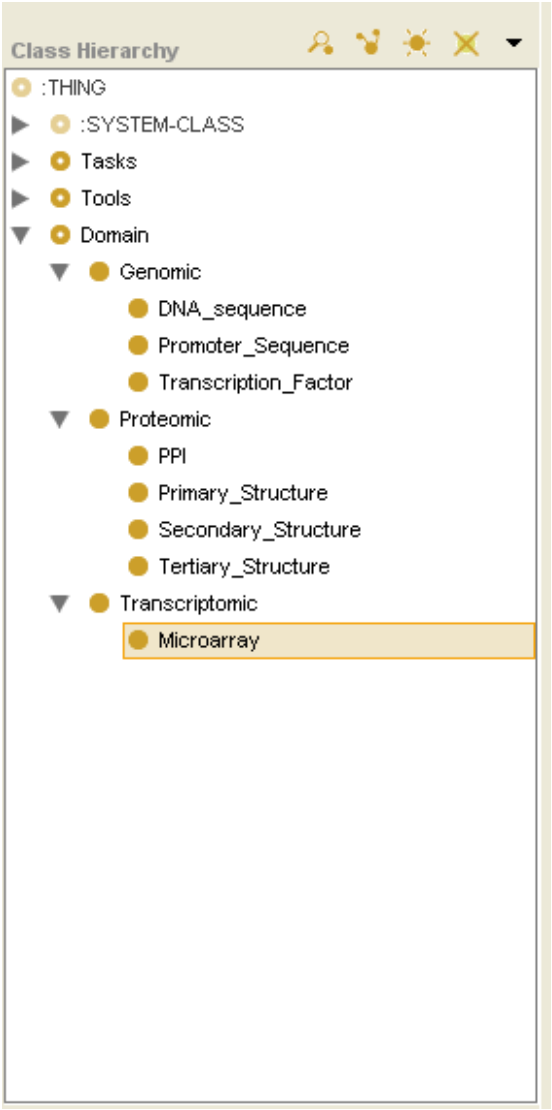

Figure 3.7: Hierarchy of classes and subclasses for the Domain part of the proposed ontology.

All the modules are organized in a tree, representing the relationships of specialization or generalization that exist between modules. Modules can have one or more children and the parent module is responsible for the activation of its sub-modules. Each level in the tree represents a different meta-reasoning level. We define it a meta-reasoning because a parent module makes decision on

the activation of one of its child module, that in turn has the expertise to take decisions on more specialized activities: so we can say that there is a high level reasoning whose results is another kind of reasoning at a lower level.

The mechanism of modules activation, also called *focusing*, is managed by special rules: when the preconditions of these rules, the **IF** part, are satisfied, their action, the **THEN** part, is to give the focus to a child module. A parent module activates a child module when it needs specialized knowledge, i.e. more specific facts and rules, in order to complete its decision-making activity.

When the module ends its job, the focus is automatically returned to the parent module. The tree representation of modules can be converted in a clearer one using a treemap (36), as in Fig. 3.8. The treemap allows to immediately visualize the topology of the tree using a set of nested boxes: the parent nodes "englobe" their own children nodes.

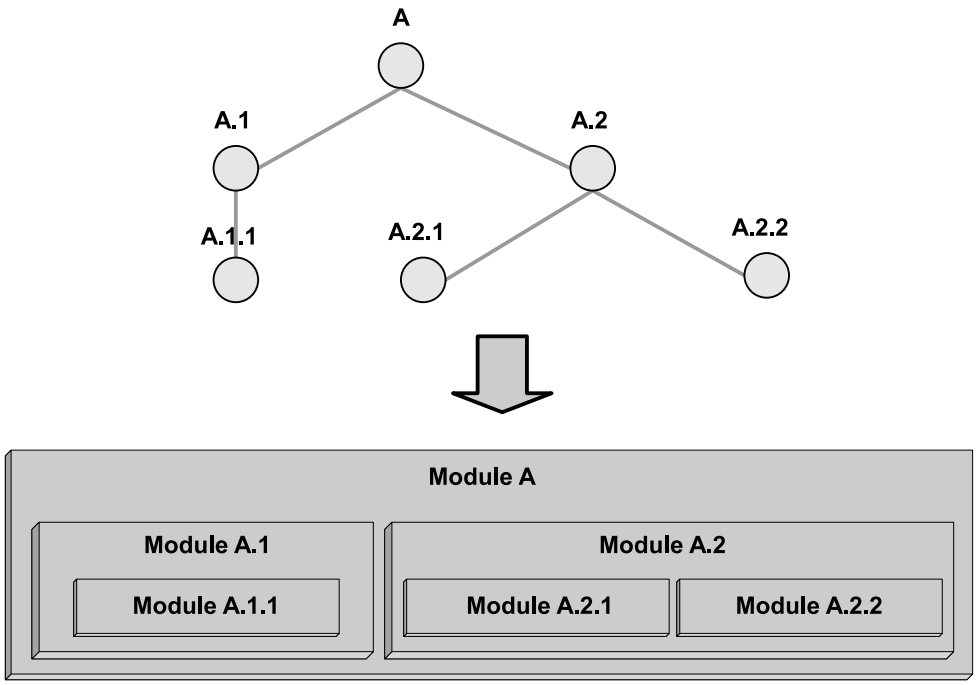

Figure 3.8: The tree structure among modules is projected into a treemap representation. Relationships between nodes mean a parent module is responsible for the activation of children nodes. In the treemap, this relation is depicted through a set of nested boxes.

Modules organization and its features has been designed in order to fully integrate the expert system architecture into the Boris hybrid architecture described in Section 2.3. With regard to the 3-axes reference space of Fig. 2.1, in fact, decision-making modules stand into the decision axis, because they represent the reasoning activity of the system. Then if the abstraction axis is considered, it is possible to map it with the hierarchy of tasks and sub-tasks defined into the ontology: at the lowest level of abstraction there are the instances of the Tools component of the proposed ontology. Finally if the timeline axis, that is responsible for tracking the executed strategies and tools into a workflow, is also considered, it is possible to obtain the scheme of Fig. 3.9.

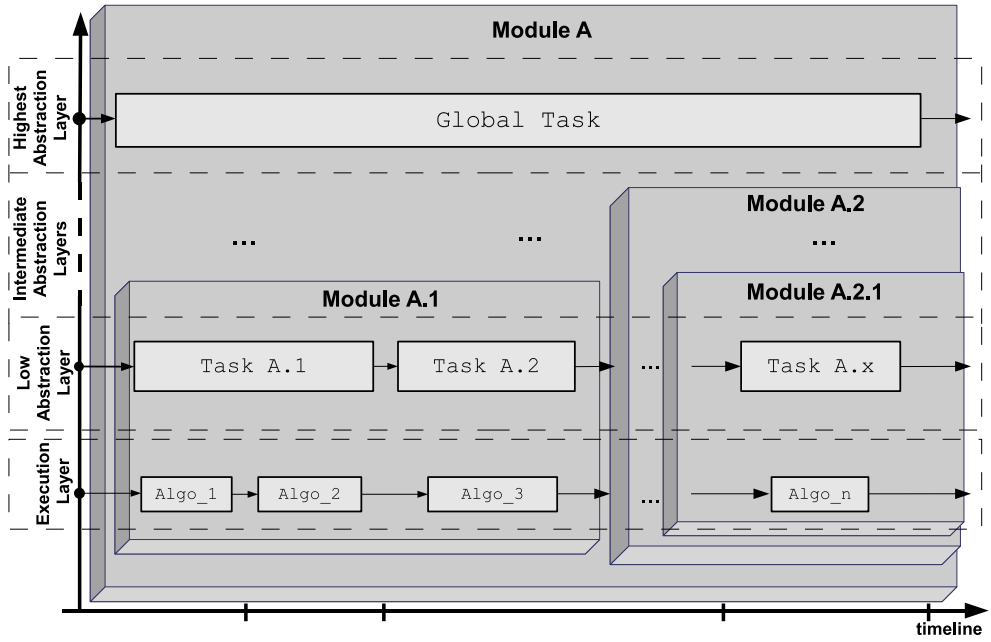

Figure 3.9: Example of complete workflow produced by our system. Gray boxes on the background, arranged as a treemap, are the modules responsible for the decision-making process about current experiment. On the vertical axis there are abstraction layers in which the experiment is decomposed: it is evident that decisions are taken at various abstraction levels. Along the horizontal axis, representing the timeline axis, we have the developing of the workflow, with the rectangles representing strategies and tools already run.

In this type of workflow representation, the decision making modules, in their

treemap organization, are in the central part. The horizontal and vertical axis are respectively the abstraction axis, with the above mentioned features, and the timeline axis. The rectangles that intersects the decision-making modules at the various abstraction layers are the executed tools and services, if they are at the bottom layer, or the strategies and heuristics that use them, if they are at higher abstraction layers. The highest abstraction layer is the main goal of the running experiment. Since this scheme includes the temporal dimension, the treemap used for the modules is a bit different from the classic one. The entire tree structure of the modules is not converted into the treemap and projected into the workflow, but only the modules activated during the execution of the experiment are shown.

4

System Overview

In this Chapter a brief explanation of the main features and components of the Graphical User Interface (GUI) of the proposed system is given. The GUI has been designed according to one of the main aim of Boris project, i.e. integrating the functionalities of a Decision Support System with the ease and usability requirements of a Workflow Management System.

4.1 Boris' Graphical User Interface

In Figure 4.1 we show a typical caption of the GUI of our system during the execution of an experiment. Here we can see four main components, that will be presented in detail in the following subsections:

- Profile Panel
- Workflow Panel
- Strategy Panel
- System Log

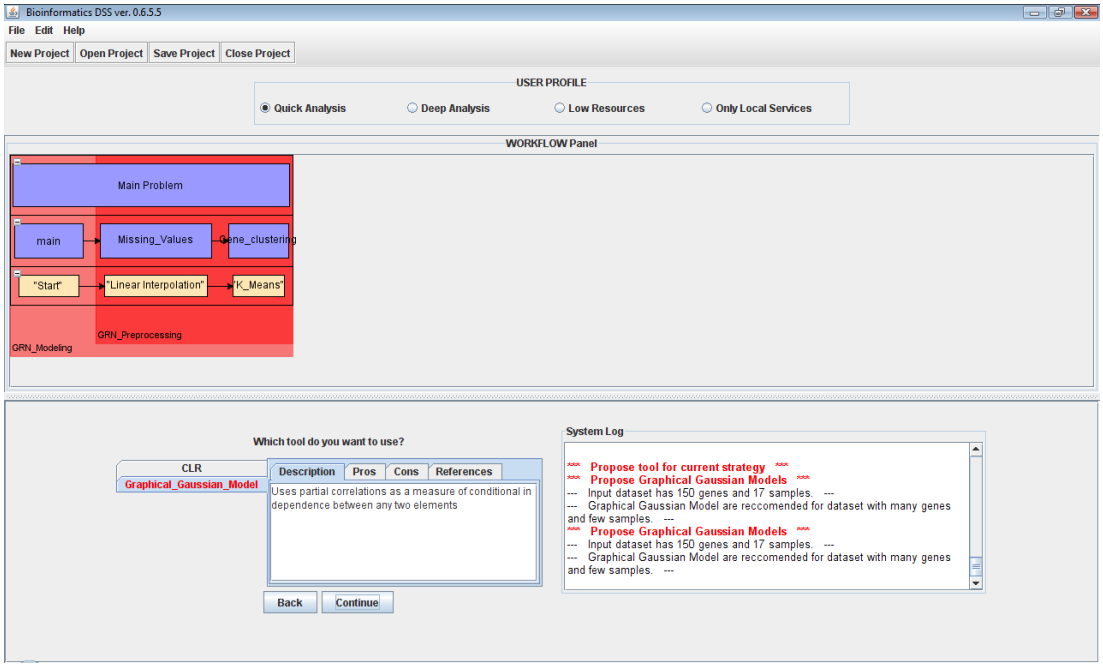

Figure 4.1: A caption of the Graphical User Interface (GUI) of BORIS system during a typical experimental session.

4.1.1 Profile Panel

The Profile Panel, standing in the top part of Fig. 4.1, allows the User to select a profile that will be considered in the choice of strategies and tools for the selected problem. The available profiles are:

- Quick Analysis: the User prefers tools with low computational time;
- Deep Analysis: the User prefers the most accurate tools, without time or resources constraints;
- Low resources: the User prefers tool that needs low computational resources:
- Only local services: the User prefers the execution of local tools and software.

The User can change the selected profile anytime during the experiment, so that he can combine different models according to his preferences.

4.1.2 Workflow Panel

Workflow Panel, that we can see in Fig. 4.2, shows the building of the workflow.

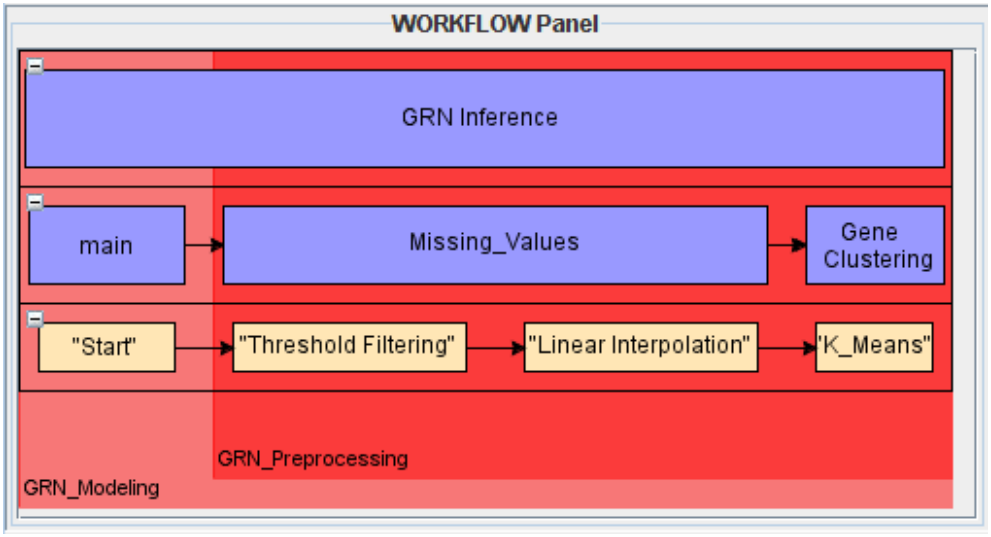

Figure 4.2: Workflow Panel component of BORIS system. It shows the active decision-making modules (pink boxes on the background), the adopted heuristics and strategies (blue rectangles), and the run algorithms and services (yellow rectangles).

It visualizes the hierarchy of tasks and subtasks used to solve the problem organized in different abstraction layers according to their complexity level: at the top level we have the main problem to resolve and at the bottom level we have the actual algorithms and tools run by the system. The intermediate levels represents strategies and heuristics used to decompose and to resolve the main goal. Strategies and corresponding algorithms are shown in rectangular boxes.

Active decision-making modules, representing the reasoning activity of the expert system, are depicted using bounding boxes on the background. The workflow is interactive: right clicking on the different part, a context-sensitive pop-up

menu allows the User to do different actions: for example if he selects an algorithm block he can change input parameters and re-run it; if he selects a strategy block it is possible to select an alternative tool; while if a box representing a module is selected it is possible to start over the whole part of the workflow for which it is responsible, in terms of decision-making activity, that module, in order to explore alternative paths, if any.

4.1.3 Strategy Panel

Strategy Panel, shown in detail in Fig. 4.3, describes available strategies and algorithms for a particular task. For each of them it is provided a general description, a list of pros, cons and bibliographic references. The suggested strategy or tool is highlighted with red text. All of these information is provided by the Knowledge Base.

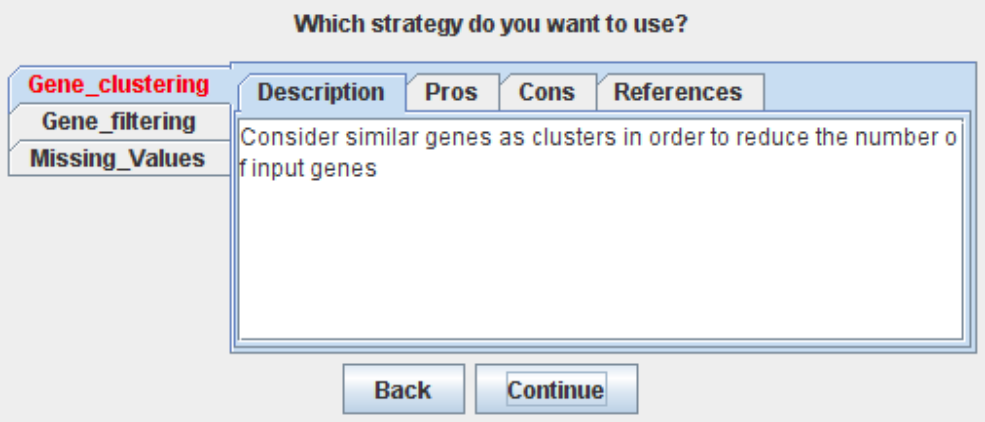

Figure 4.3: Strategy Panel showing the available strategies, heuristics and tools for the given task. The red element is the suggested one.

4.1.4 System Log

System log allows to the User to know every single operation done by the system during the execution of an experiment. It shows the reasoning behind each proposed strategy/tool, writing the motivation of each activated rule; the result

of each executed process; the pathway of the workflow, if there are some possible forks, and the final results of the experiment. The User can scroll the log in order to read the history of the running experiment. The different kinds of communication have different text coloration.

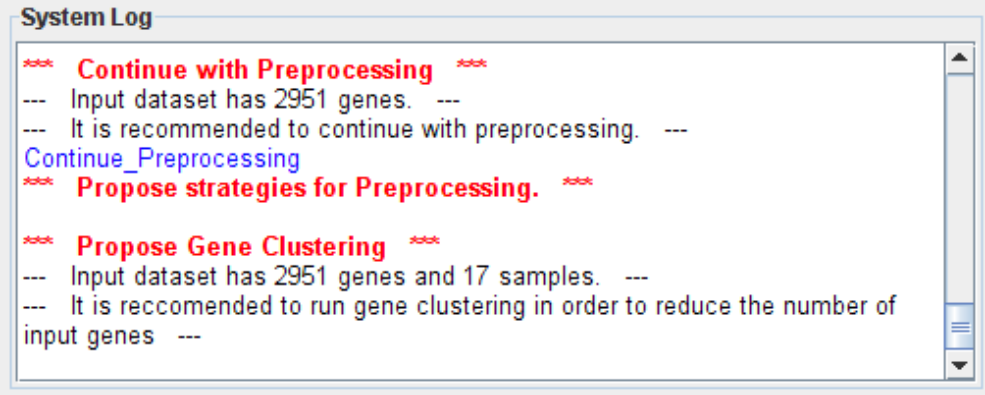

Figure 4.4: System Log of Boris system. Activated rules and their motivation, intermediate and final results, available forks in the workflow and progression of the experiment are shown.

5

Case Study: Reverse Engineering Gene Regulatory Network

In this Chapter an application of the proposed system to an actual case study in Bioinformatics will be presented. The scenario is the inference of a Gene Regulatory Network from an input dataset of gene expression values. First of all I will present what is the biological issue; then I will give a brief explanation of the most common bioinformatics approaches and tools and finally it will be seen how our system can offer support in the choice, configuration and run of those tools. During the system demonstration, we will also show the "back-end" of the system, that is how the system works with regards to BORIS hybrid architecture (see Chapter 2) and its software implementation (see Chapter 3)

5.1 Biological Problem

Gene regulation is the cellular control governing the rates at which genes are transcribed into mRNA: this biological phenomenon is called gene expression. Gene expression depends on physical signals from the environment or within an organism cell. When one of these signals reaches cell nucleus, a protein, called Transcription Factor (TF) is activated. TF, then, binds to the promoter region, that is a specific upstream region, of a target gene and triggers the RNA polymerase enzyme to transcribe DNA to RNA. TFs can be seen as controller of the on-off switch mechanism of gene expression: repression (down-regulation) or

induction (up-regulation) of output. The molecular readout of a gene are then mRNA, which is transcribed from DNA, and protein, which is translated from RNA. In Fig. 5.1 it is shown a schematic representation of gene regulation.

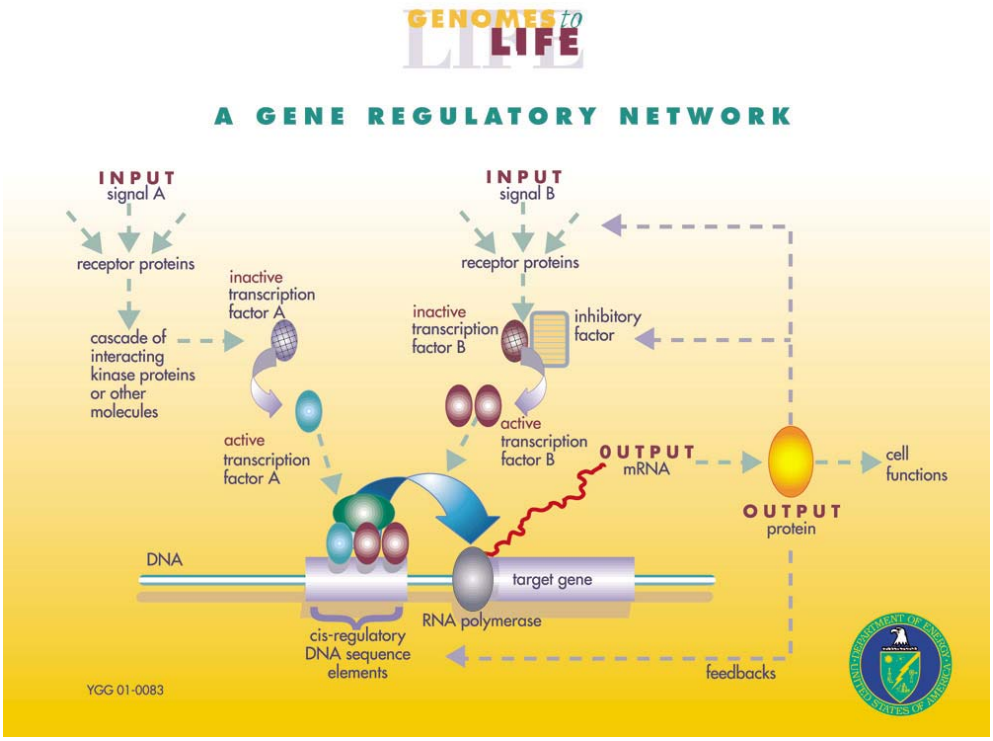

Figure 5.1: A synthetic representation of gene regulation biological phenomenon. This picture is taken from U.S. Department of Energy Genome Programs, http://genomics.energy.gov

5.2 Bioinformatics Approach

Gene regulation is a very complex biological phenomenon and it is not full understood yet (37). In Bioinformatics and System Biology this mechanism is studied and modelled by means of Gene Regulatory Networks (GRN): a GRN is a directed graph where nodes represent genes and other regulatory elements, such as transcription factors (TF), protein complexes and so on, and edges are regulatory

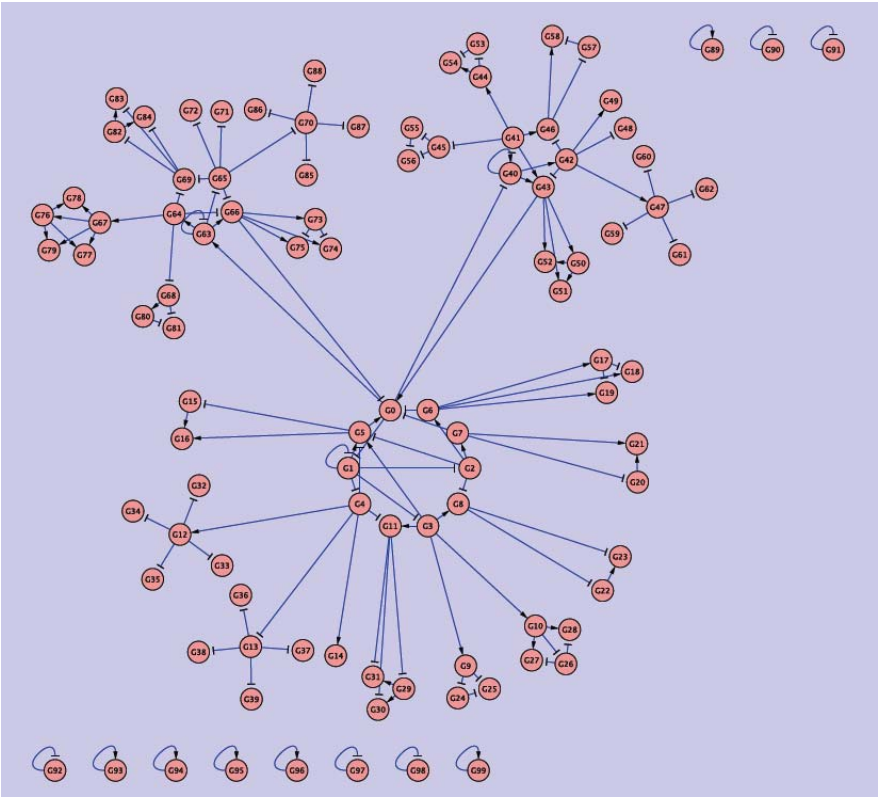

Figure 5.2: Sample gene regulatory network (directed graph)

relationships among them. Basic input for inferring a GRN is a dataset of gene expression values obtained through microarray technology. An example of GRN is shown in Fig. 5.2.

From a computational point of view, modelling a GRN is a reverse engineering problem, since from the output of gene regulation, that is gene expression, we want to infer the network, with its topology and parameters, that provided those outputs.

5.2.1 Microarray Technology

Gene Expression is measured by means of microarray technology (38). Microarray chips are devices that enable the scientist to simultaneously measure the transcription level of every gene within a cell. Microarrays are commercially available

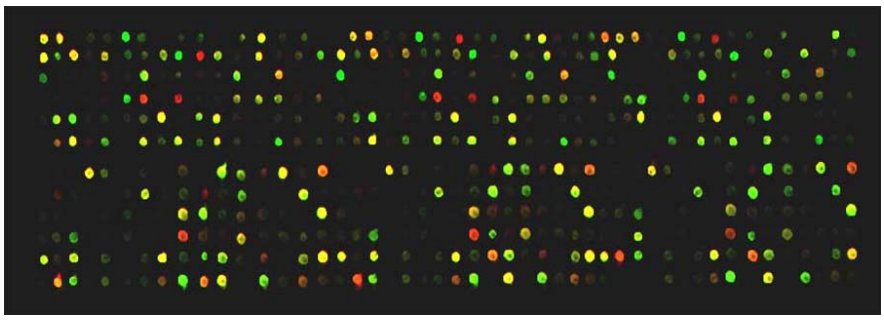

Figure 5.3: Example of microarray image

from a number of companies, like for instance Affymetrix, Invitrogen and Sigma-Genosys. The chip is usually constructed by amplifying all the genes within the selected genome, yeast, for example, using polymerase chain reaction (PCR) (39) methodology. The PCR products would then be "spotted" onto the chips by a robot, as single-stranded DNA that is linked by covalent bonds to the glass slide. The spots would be positioned in an array on a grid pattern, where each spot contains many identical copies of an individual gene. A discussion of the chemistry involved in creating a microarray can be found on the technology page of the Affymetrix website. The position of the genes are recorded by spot location, so that the appropriate gene can be identified any time a probe hybridizes with, or binds to, its complementary DNA strand on the chip.

Microarray chips measure transcriptomes, which are the entire collection of RNA transcripts within a cell under the given conditions. To use the chip to measure an experimental transcriptome against a reference transcriptome requires cells grown under two different conditions, the experimental conditions and the reference conditions. The mRNA from the two different conditions are harvested separately, and reverse transcriptase (40) is used to transcribe the mRNA into cDNA. The nucleotides used to synthesize the cDNA will be labelled with either a green or red dye, one colour for the reference conditions and the other for the experimental conditions. The microarray chip is then incubated overnight with both populations of cDNAs, and a given cDNA will hybridize with the complementary strand from its gene that is covalently bound to a grid spot on the chip. The chips are washed to remove any unbound cDNAs and then two

computerized images are produced by scanning first to detect the grid spots containing cDNAs labelled with green dye, and second to detect the spots contain red-labelled cDNAs. The computer also produces a merged image, like the one shown in Fig. 5.3, that will show a yellow spot for grid spots that contain both red- and green-labelled cDNAs, indicating transcripts that are expressed under both sets of conditions.

In addition to producing a qualitative image that is easy visualize, a microarray experiment yields quantitative data for each spot, consisting of the measured fluorescence intensity of the red signal, the fluorescence intensity of the green signal, and the ratio of red signal to green signal. It is in storing and analysing the quantitative data that bioinformatics really comes into play in microarray technology. These datasets are incredibly large. For instance, a typical mammalian cell is estimated to have between 10,000 to 20,000 different species of mRNA expressed at a given time.

There exists two different types of microarray datasets: static, or steady-state, and dynamic, or time-series. In static data, an experiment with well-defined conditions is carried out and the observation of gene expression values is done at the presumed steady state of the biological system. Static data, then, captures the effects that the perturbations on initial conditions have ion the final state of the system. This type of data, however, can miss some dynamic events that may be critical in the description of the biological phenomenon described by the GRN.

Dynamic data, in turn, are obtained from time-series experiment, when the gene expression values, also called samples, are taken at precise intervals, or time-points, after a perturbation. Dynamic datasets have the advantage that can capture some fundamental dynamics of the biological system but, on the other hand, may contain redundant information that could penalize the network inference process. Furthermore in this type of experiments it is difficult to find a compromise between the duration of the observation and the interval between two consecutive measurements, since the number of time-points influences the performance of the GRN inference methodologies.

5.3 Bioinformatics tools

Inferring a GRN is an ideal application scenario for our system: looking at the state-of-the-art, in fact, a wide set of algorithms and methods are used for this purpose (41, 42, 43). All of these techniques present pros and cons, and differ each other according to the type of input data (microarray, gene sequences, protein-protein interactions), the applied algorithm, the desired output, the need of specific data format, the accuracy level of the inferred model, the computational time and resources. Moreover the process of modelling a GRN often needs preprocessing steps, like filtering and clustering, and/or postprocessing steps, like simulation and visualization.

Among the most used methodologies there are static and Dynamic Bayesian Networks (44, 45), Factor Graph (46), Boolean Networks (47), correlation methods (48), Ordinary Differential Equations (ODE) (49, 50).

To be more precise, at the moment Bayesian Network (51), Graphical Gaussian Models (52), and correlation methods using ARACNE (48) and Context Likelihood of Relatedness (CLR) (53) algorithms, are supported. Moreover, Boris can also offer support for preprocessing of input data, using algorithms for Gene Clustering (31, 32), Gene Filtering (35) and Gene Interpolation (33); and for visualization of networks, using Graphviz software (61) and Cytoscape (62).

5.3.1 Correlation Methods

Generally speaking, correlation methods are based on Information Theory Models. This kind of approach compares expression profiles from a microarray dataset computing, for each pair of genes, a pairwise correlation coefficient called Mutual Information (MI). Given the gene i and the gene j, their mutual information MI_{ij} is computed as:

$$\mathrm{MI}_{ij} = H_i + H_j - H_{ij} \tag{5.1}$$

where H represents the entropy and it is defined as:

$$H_i = -\sum_{k=1}^{n} p(x_k) \log(p(x_k)) \tag{5.2}$$

Then gene i and gene j are considered connected by an edge if their MI is higher than a specific threshold. The higher the threshold, the sparser is the inferred network. Using an algorithm based on correlation measures, an undirected graph is inferred because it is found only a possible correlation between two genes, without any information about the direction of that relationship.

The computation of the MI requires that each experiment in the microarray dataset be statistically independent each other. That means information-theoretic approaches works both on steady-state gene expression dataset and with time-series experiments only if the sampling time is long enough to consider statistically independent one time point from the other ones.

The inferred relationships among genes, representing the edges of the gene network, computed by means of this type of approach indicate a statistical dependence among gene expression profiles. Information-theoretic models, in fact, does not represent direct casual interaction between two genes.

Correlation-based methods are best suited to infer large-scale networks because of their low computational cost and low data requirement. A major drawback is that they can not model the dynamics of gene regulation and do not consider that multiple genes can influence the regulation.

5.3.1.1 ARACNE

Algorithm for the Reconstruction of Accurate Cellular Networks (ARACNE) is an information-theoretic algorithm for the reverse engineering of gene regulatory networks. It uses a Gaussian kernel estimator (63) for the estimation of the MI (5.1) and moreover it implements a pruning phase of the inferred network with the aim of reducing the number of false-positive interactions, i.e. inferred relationships that do not correspond to actual biological interactions. The pruning is done according to the Data Processing Inequality (DPI) principle (64), which states that if gene i and gene k interacts only through gene j, then $\mathrm{MI}_{ik} \leq \min(M_{ij}, M_{jk})$. DPI principle represents a necessary but not sufficient condition, that means some direct interaction could be eliminated during the pruning phase.

The main purpose of ARACNE is, in fact, to infer a subset of all the regulatory interactions with a high confidence level. It has a low computational cost, it does

not need any prior assumption about the network to compute and it does not require the discretization of input gene expression values.

5.3.1.2 CLR

Context Likelihood of Relatedness (CLR) is an unsupervised network inference method that, given a dataset of gene expression profiles, finds transcriptional regulatory relationships among genes. CLR is an improvement over the relevance network algorithm (65), which considers Mutual Information between each pair of genes in order to estimate the similarity between them according to a certain threshold. CLR algorithm gives an estimate of the relevance of the MI value between each pair of input genes by comparing it with a background distribution of MI values. Given the gene i and the gene j, their background distribution is computed considering the set of MI values of gene i with all other genes, MI_i, and the set of MI values of gene j with all other genes, MI_j. Then the background MI is approximated as a joint normal distribution assuming MI_i and MI_j as independent variables. The key idea at the basis of CLR algorithm is that the mutual information score of the most probable interacting genes should be significantly higher than the background distribution of the MI scores.

CLR algorithm is characterized by a low computational cost, since it is based on an information-theoretic model, and it is suited for the analysis of large-scale gene expression datasets. Moreover, if a list of known transcription factors is available, it can provide a directed acyclic graph, limiting the possible interactions from transcription factors to non-transcription factor genes.

5.3.1.3 Graphical Gaussian Models

Graphical Gaussian Models (GGM) are undirected probabilistic graphical models that are able to find the conditional independence relations among the nodes of a network, considering the prior hypothesis of a multivariate Gaussian distribution of the data. GGM uses partial correlation in order to calculate the conditional independence between each pair of genes in the network. Given the generic gene i and gene j, their partial correlation coefficient p_{ij} is computed by measuring the correlation between them after the effects of all the other genes have been

discarded. The estimation of the covariance matrix of the Gaussian distribution of the data allows the computation of GGM because partial correlation ρ_{ij} is related to covariance matrix $\mathbf{C}$, and its inverse $\mathbf{C}^{-1}$, by the following formula:

$$\rho_{ij} = \frac{C_{ij}^{-1}}{\sqrt{C_{ii}^{-1} C_{jj}^{-1}}} \tag{5.3}$$

Partial correlation is able to distinguish direct interactions among genes, that are the ones of interest for the construction of a regulatory network, from indirect interactions. In order to infer a GRN using a Graphical Gaussian Model partial correlation among the elements belonging to the input dataset is computed by means of Eq. (5.3). Then the distribution of $|\rho_{ij}|$ is analysed and the edges (i, j) with a small value of $|\rho_{ij}|$ are discarded from the graph. So the key element of this method is the estimation of the covariance matrix and its inverse.

GGM produces undirected graphs, therefore it is able to model network with feedback loops. In (54) an improvement over GGM has been done in order to obtain a partially causal network, i.e. a directed graph, in which some edges are given a direction.

One of the major drawbacks of GGMs is the dealing with high dimensional data.

5.3.2 Bayesian Networks

Bayesian Networks (BN) are directed graphical models that allow to identify probabilistic relationships among a set of interacting elements, or random variables. These relationships are represented through a directed acyclic graph (DAG) whose nodes are the random variables and the edges are the conditional relationships among them. In this case study, random variables are input gene expression levels and their regulatory relationships are described by a joint probability distribution $P(X_1, \ldots, X_n)$ where X_i is the i-th gene. The joint probability distribution (JPD) can be decomposed into the product of conditional probabilities if each variable (gene) X_i is independent from its non-descendants, given its parents in the graph:

$$P(X_i, \ldots, X_n) = \prod_{i=1}^{n} P(X_i = x_i \,\|X_j = x_j, \ldots, X_{j+p} = x_{j+p}) \tag{5.4}$$

The $p+1$ genes, on which the probability is conditioned, are the parents of gene i in the graph and represent its regulators. Equation (5.4) is obtained using the Bayes Theorem:

$$P(A, B) = P(B \,\|A) * P(A) = P(A \,\|B) * P(B) \tag{5.5}$$

from which we can derive the so-called Bayes rule:

$$P(B \,\|A) = \frac{P(A \,\|B) * P(B)}{P(A)}. \tag{5.6}$$

Bayesian Networks reflect the stochastic nature of gene regulation. They are used to infer a gene network by finding the best DAG, according to a metric, describing the gene expression dataset. The most common metric, computed using the Bayes Rule (5.6) are the Bayesian Information Criteria (BIC) and Bayesian Dirichlet equivalence (BDe). Learning a BN is an iterative procedure consisting of three main steps: model selection, parameters fitting and network scoring.

During model selection, a candidate DAG is found. Then, given this graph, the best conditional probabilities of each node is computed thanks to the experimental data provided. Finally each candidate graph is scored, by means of one of the above cited metrics, and the model with the highest score is the winner network, since that means it best fit to the data.

The most expensive computational phase is model selection, because the brute-force approach, i.e. enumerating all the possible graph configuration, is a NP-Hard problem. Therefore for this learning phase it is often used an heuristic search method considering techniques such as greedy-hill-climbing, simulating annealing, etc...

In reverse engineering GRN Bayesian Networks represent a very flexible framework because it is possible to combine many type of input data, like for instance TF-DNA interaction data, and also, when available, prior knowledge about the

structure of the searched network. Moreover they can use a network template, obtained for example by other inference techniques like information-theoretic methods (see Section 5.3.1), in order to restrict the space of possible models and to speed up the entire computation. Moreover, as stated in (43), BNs avoid overfitting issues and can deal with incomplete and noisy data.

Classic Bayesian Networks have a very strong limitation in their application to the inference of gene networks because they can not model networks containing a feedback loop, that is a direct cycle. In a gene network, a feedback loop represents a feature that can cause homeostasis. The result of this limitation is that BNs can not work with input dataset containing time-series experiments. In order to overcome this drawback, Dynamic Bayesian Networks (51) have been introduced.

5.3.2.1 Dynamic Bayesian Networks

Dynamic Bayesian Networks (DBN) are an extension of traditional (static) Bayesian Networks that can deal with time-series input data. Here gene-expression values are modelled by means of random variables $X_i[t]$ representing the gene expression level of gene i at time t. DBNs are used under the assumption that the modelled process is stationary, i.e. the relationships between two nodes in the graph do not change over time. DBNs can be specified by a directed acyclic graph (DAG) where its vertices belongs to two separate sets of random variables: $X_1[t], X_2[t], \ldots, X_n[t]$ and $X_1[t+1], X_2[t+1], \ldots, X_n[t+1]$. Moreover there are only directed edges from the nodes of the first set to the nodes belonging to the second one. One last consideration is that if we represents the genes as nodes independent of time, we obtain a direct cyclic graph that is not allowed using static BNs.

5.4 Experimental Dataset

The dataset used in this system demonstration is extracted from the genome of Saccharomyces cerevisiae (yeast) (66) and consists of 3000 genes. This dataset is obtained from a dynamic (time-series) experiment, it has 17 samples (time-points) and it contains some missing values. Saccharomyces cerevisiae is one of

the most studied organism in the field of system biology and gene networks issues. This dataset has been chosen because it has some interesting features, such as the presence of missing values, the high number of input genes and the relatively low number of time-points, that can exploit several important characteristics and suggestions of the proposed system.

5.5 System Running

In this Section, a typical experimental session with BORIS will be shown. The aim of the experiment is to infer a gene regulatory network from the input dataset of gene expression values described in the previous Section. Boris system will give decision support in the choice of the proper strategies and tools and will help the User both in the configuration and running of selected instruments. Moreover, during the description of the experiment, it will be shown the status of the system according to the the 3-axes architecture presented in Section 2.3.1.

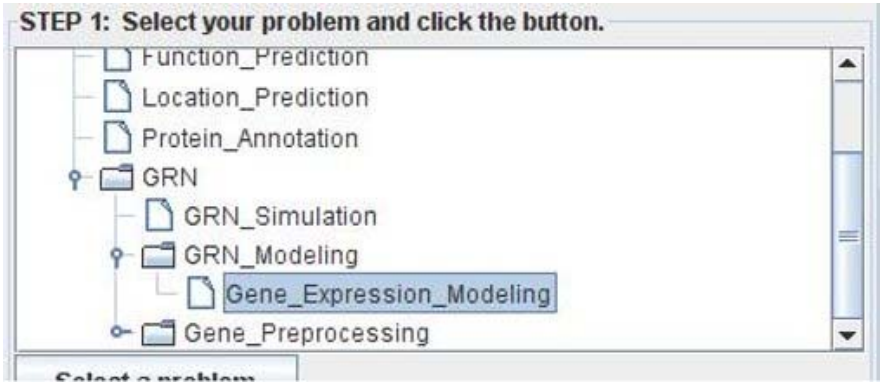

Figure 5.4: Available bioinformatics problems supported by BORIS system.

When the User starts a new session, he can choose the type of the experiment from a list, organized as a tree, of the supported scenarios (Fig. 5.4). This list of supported bionformatics problems is obtained through the Task ontology presented in Section 3.2. Once selected a problem, the User he will be asked to insert an input file, depending on the type of the experiment, so that the system can begin its work. Here he choose "Gene Expression Modeling" as for

the bioinformatics problem and insert the input dataset presented in Section 5.4 in csv format. Selected User Profile is *Quick Analysis*.
After selecting the problem and inserting input dataset, decision-making activity starts. In Fig. 5.5 we can see what are the decision-making modules responsible for the current experiment:

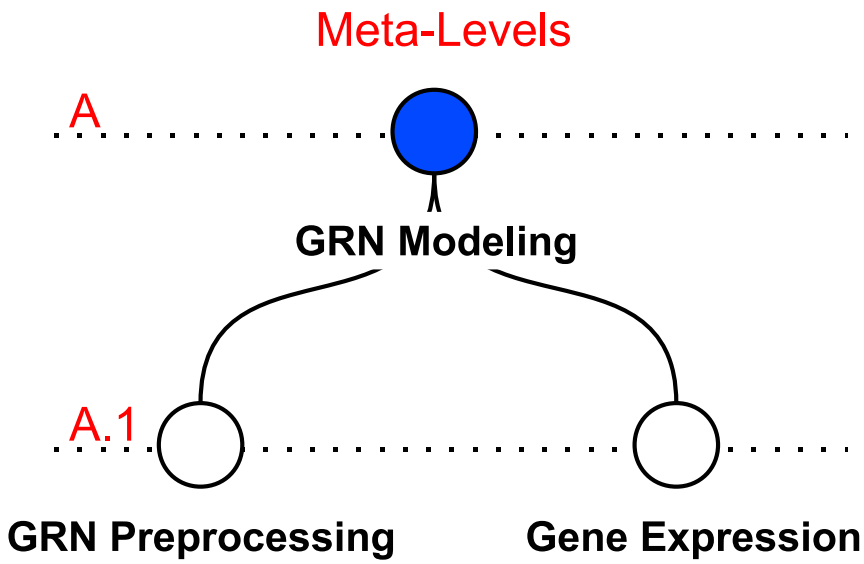

Figure 5.5: Decision-Making modules responsible for the reasoning activity related to the reverse engineering GRN scenario.

- *GRN Modeling*: the supervisor module that manages all the session and that can activate children modules in order to deal with more specific tasks;
- *GRN Preprocessing*: the module responsible for the reasoning part with regard to the preprocessing phase of input data
- *Gene Expression*: the module in charge of the decision-making activity regarding the inference of the gene network.

At the beginning of the experiment, *GRN Modeling* module is active (the blue filled circle): the job of this module is to analyse input dataset in order to extract all the meaningful information that can be used to trigger the rules.

In Fig. 5.6 it is shown what is the current status in the 3-axes reference system: there we can see what are the values of abstraction, decision-making level and workflow timeline. Abstraction axis, characterized by discrete values, has an high value because at the beginning of the experiment the user's request represents the final goal and then it is seen as a complex problem at the top abstraction level. An increment of the value in the decision-making axis means a new decision-making module has been activated. Finally, in the workflow timeline axis, we will see a progression according to the generation of the workflow: the workflow is built every time a tool or service is actually run.

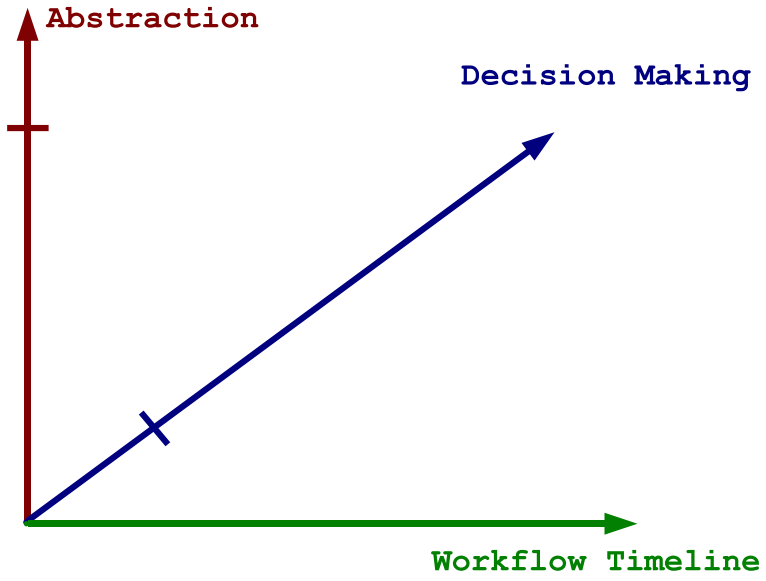

Figure 5.6: The initial state of the system with regard to the 3-dimensional system reference space defined in Section 2.3.1

According to the attributes of Microarray template described in Section 3.2, the number of genes, the number of samples (or time-points), the name of the species, if available, and the type of experiment (steady-state or time-series) are extracted. The latter property is expressly asked to the user, because the system can not infer it by itself.

As stated in Section 5.4, input file has some missing values: that property triggers a rule, whose action is shift the focus to the *GRN Preprocessing* module:

it will be responsible to suggest to the user a possible strategy to deal with this issue (Fig. 5.7).

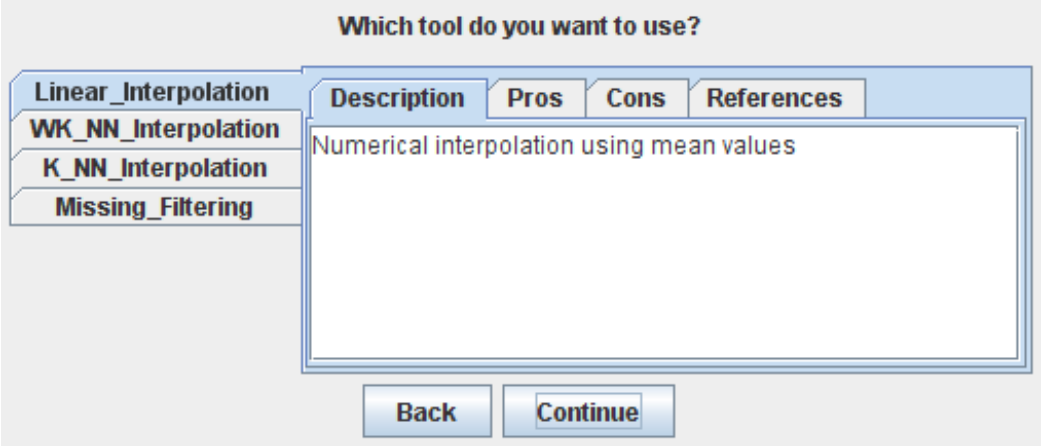

Figure 5.7: Available techniques for dealing with missing values.

The supported strategies are:

- Missing Filtering: the genes with a certain percentage of missing values are pruned;
- WK_NN interpolation: missing values are interpolated using Weighted K-Nearest Neighbour algorithm (68);
- K_NN interpolation: missing values are interpolated using K-Nearest Neighbour algorithm (67);
- Linear interpolation: missing values are interpolated by means of linear interpolation (33).

At this point, if we look at the 3-axes reference system, the abstraction axis has a low value because the proposed strategies are immediately executable representing the lowest level of abstraction. (Fig. 5.8), while the decision-making axis has incremented because a new module has been activated.

The User selects Missing Filtering with a threshold of 25%: the resulting dataset has now 2951 genes, and the first part of the workflow is built (Fig. 5.9).

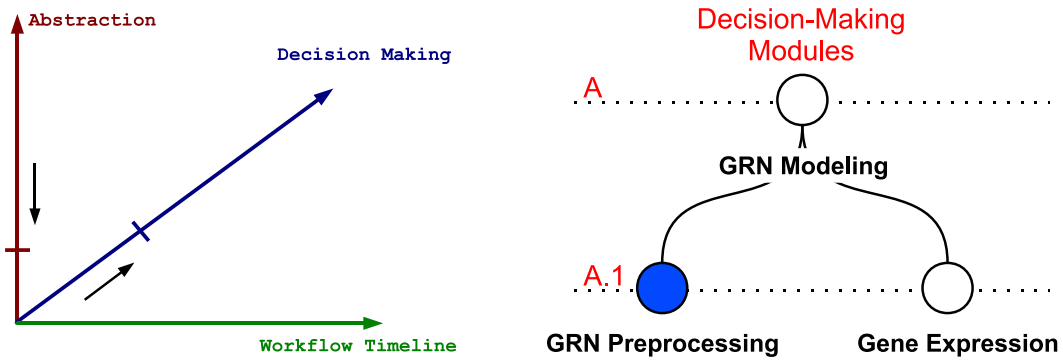

Figure 5.8: State of the system in the 3-axes reference space (left) and corresponding activation status of decision-making modules (right) during preprocessing operations.

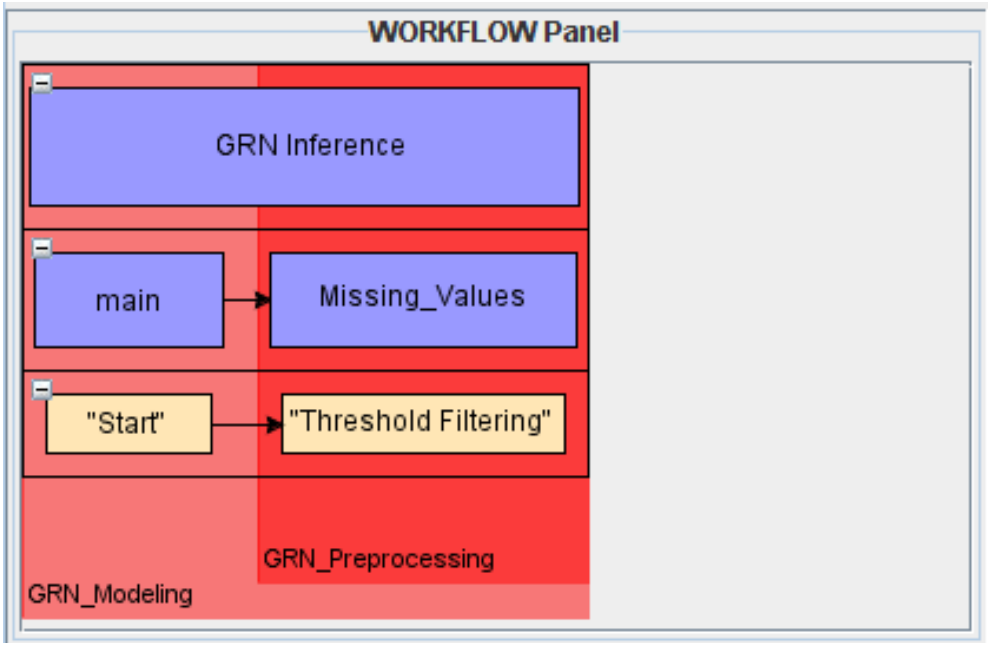

Figure 5.9: Workflow of the current experiment after the first algorithm (Threshold filtering) has been run. It is possible to notice the decision-making modules on the background, the strategy name at middle abstraction layer and the main goal at the top abstraction layer.

Once again input dataset has still missing values (the threshold was not enough), this way the system suggests to the user to continue with preprocessing and, after the affirmative selection of the User, it presents the possible preprocessing strategies, suggesting the *Missing Values* strategy. This time, among the available tools, *Linear Interpolation* is the suggested one because it satisfies the requirement on User's Profile (Quick Analysis) since it is the less expensive algorithm. After the linear interpolation has been done, then the workflow is updated (Fig. 5.10).

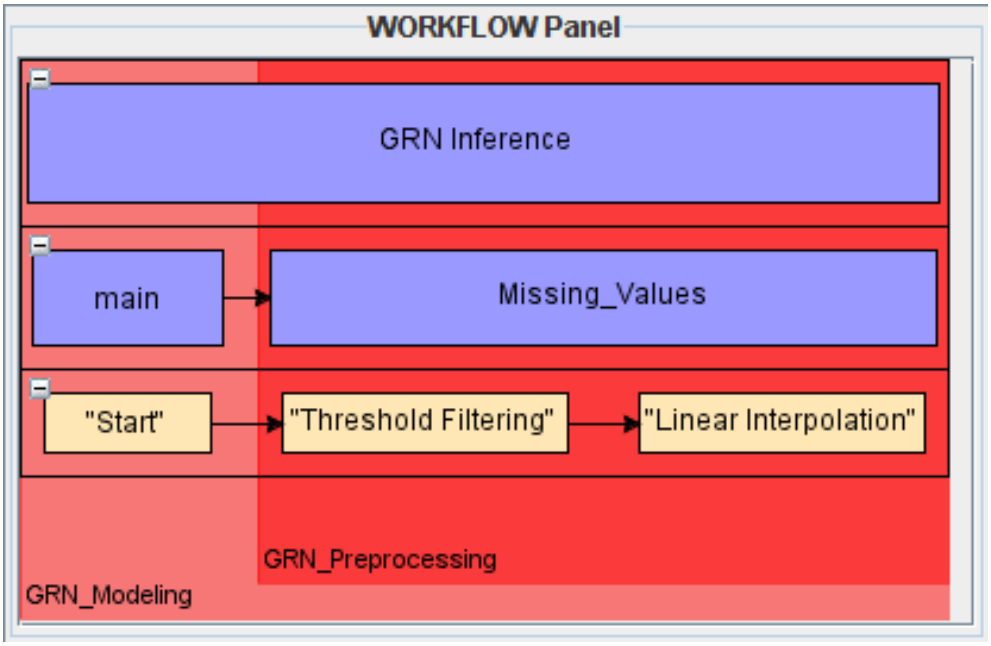

Figure 5.10: Workflow of the current experiment after Linear interpolation.

Input dataset has no more missing values, but the system keeps on suggesting the preprocessing phase because of the activation of the rule that proposes to do preprocessing if input dataset has many genes (more than 1000). Once again the system presents the supported preprocessing operations, and the suggested strategy is *Gene clustering*, because with dataset with many genes (almost 3000) and few sample (17) is the most recommended technique. The remaining strategy, *Gene Filtering*, offers support in the selection of only a subset of input genes. The available gene filtering algorithms in the system are:

- Threshold filtering: input gene with an expression value lower than a user defined threshold are not considered;
- Genecycle: an algorithm that is able to identify periodically expressed genes, supposed to hold meaningful information content, in a time-series gene-expression dataset (55);
- Robust Genecycle: an enhanced version of simple Genecycle algorithms, characterized by more accurate results but higher computational time (56, 57).

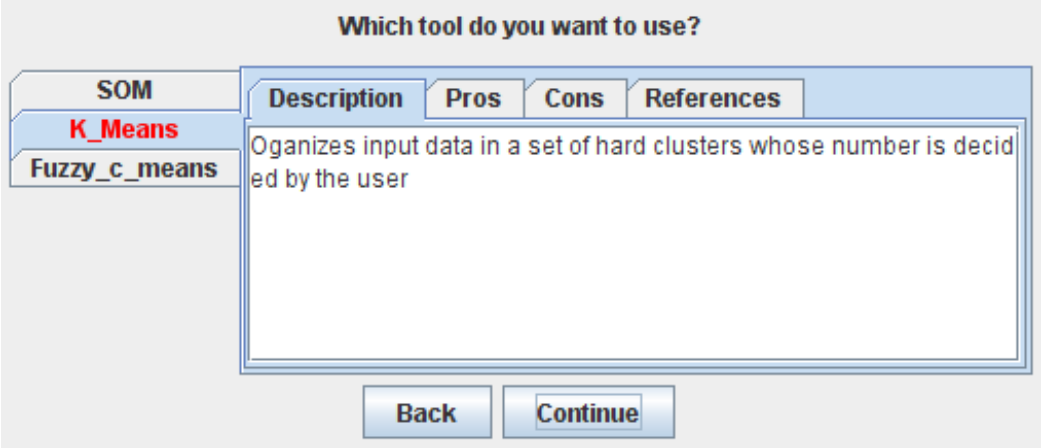

Figure 5.11: Supported clustering tools. K-means, in red, is the suggested one.

The supported clustering algorithms are (Fig. 5.11):

- K-Means: one of the simplest clustering algorithm, it puts together input elements into clusters, maximizing *intra* cluster similarity and *inter* clusters diversity (58);
- Self Organizing Map (SOM): an unsupervised clustering algorithm allowing multidimensional elements to be projected into a (typically) 2D space, providing this way both visualization and clustering information (59);

- fuzzy_c_Mean: similar to K-Means algorithm, but it produces *soft* clusters, i.e. each element is given a score measuring its membership level to each cluster (60).

K-Means, that is the fastest algorithm among the other ones, is the suggested algorithm according to the User's Profile (Quick Analysis). The system then asks the user what is the final number of clusters: K-Means in fact requires the number of output clusters as an input parameter.

In this case, the system will assist the User in the proper configuration of the algorithm emphasizing the effect of the desired number of clusters: the more the number of clusters, the finer the classification of patterns, but if too many clusters are chosen, the resulting clustering can miss important correlation among elements. In this scenario, 200 clusters are selected, K-Means is run and the workflow is updated (Fig. 5.12).

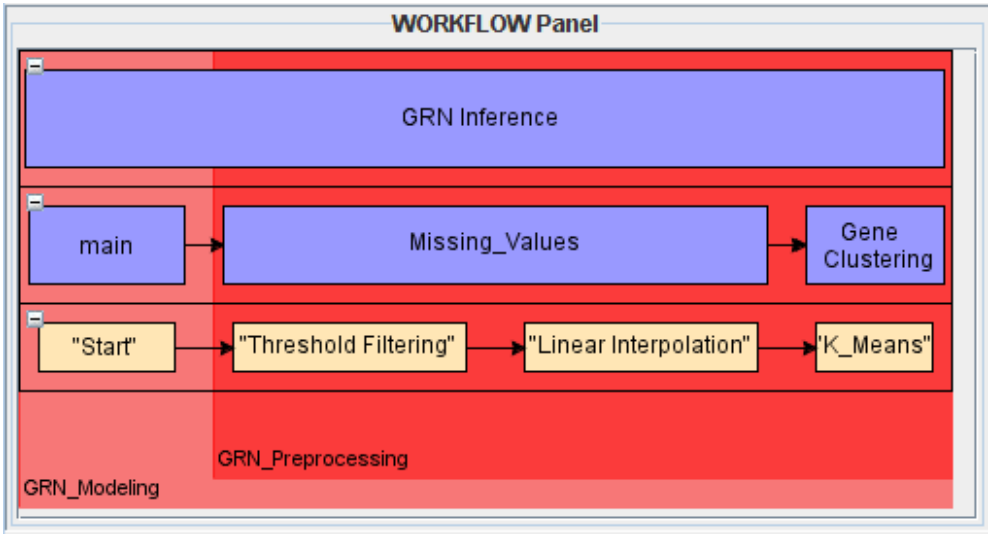

Figure 5.12: Workflow of the current experiment after K-Means has been run. The active decision-making module is *GRN Preprocessing* as well.

After the Gene clustering procedure, preprocessing is no longer needed because there are not missing values and the number of input genes is not very high, so the system suggests to continue with the rest of the experiment. Since *GRN Preprocessing* module has finished its job, it gives back the focus to its

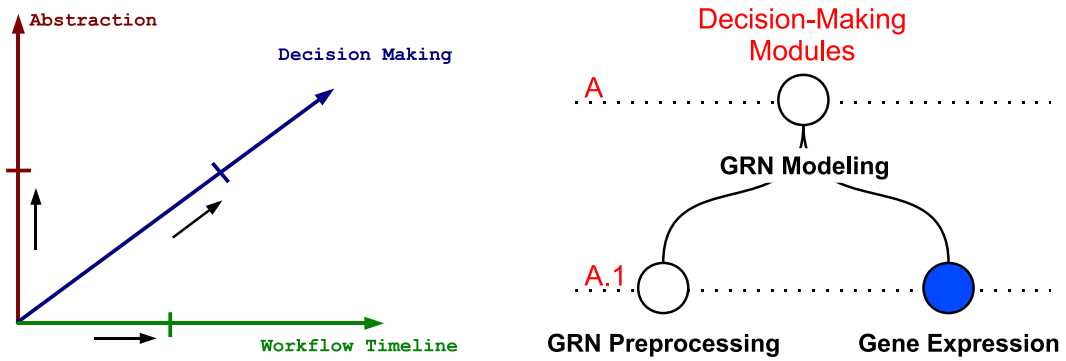

Figure 5.13: State of the system in the 3-axes reference space (left) and corresponding activation status of decision-making modules (right) at the beginning of actual GRN inference phase.

parent module, the *GRN Modeling* module. It is aware, by consulting its KB, the *GRN Preprocessing* has ended, so it can activate the *Gene Expression* module, containing the needed skill in order to infer a GRN.

At the beginning of this phase, the status in the 3-axes reference system can be seen in Fig. 5.13: the abstraction axis has a medium value because the system is reasoning about a sub-problem of the main problem; the timeline axis tracked the building of the workflow so far; in the decision-making axis there is an other incremental step corresponding to the activation of *Gene expression* module.

The system shows what are the possible strategies to infer a gene network: the suggested ones are the correlation based methods, consisting of the use of Graphical Gaussian Models (GGM) and CLR algorithm (Fig. 5.14). Here it is important to point out that both techniques are recommended at the same time for different motivations: that means the two rules that trigger the suggestion of these algorithms are both fired by the Reasoner. If two or more rules, whose effect is to suggest a strategy or a tool, are activated at the same time, they, in fact, do not represent mutually exclusive options.

GGMs are suited for the analysis of datasets with with a number of genes greater than the number of samples, more than ten times in the specific case study; CLR is recommended for the quick analysis of large-scale input dataset, where with large-scale dataset can be considered dataset with more than 150

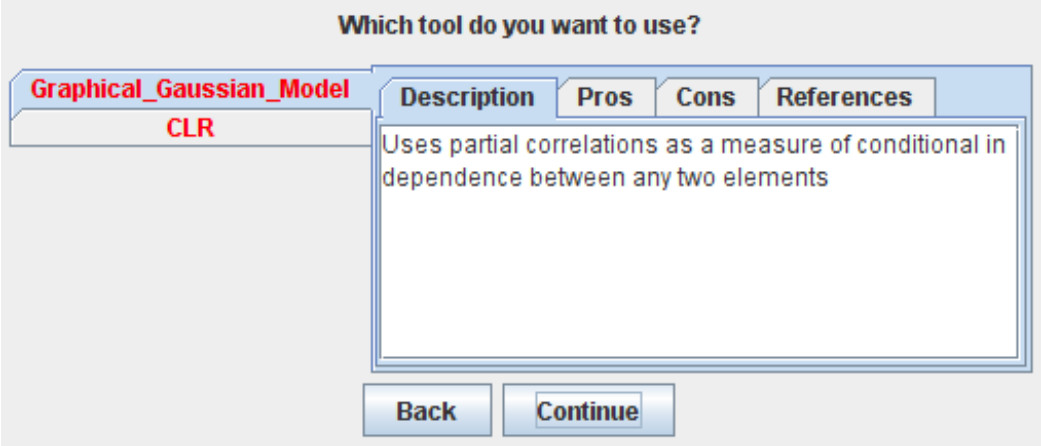

Figure 5.14: Supported tools implementing Correlation methods. Both algorithms are suggested, for different motivations.

elements. In this situation the User decides to run one of the two algorithms, remembering that the backtracking features of the system allows him to reconsider his choice in order to select another possible alternative and, in case, to compare the two different results. Here the User decides to run CLR algorithm, then the workflow is updated (Fig. 5.15) and a first gene network is generated. This network can be saved and/or visualized.

After that, the systems invites the User to continue with the experiment because the inferred network, obtained with a fast but poor accurate algorithm (CLR), can be considered as a template input network in order to find a better one using a Dynamic Bayesian Network (DBN). If the User agrees with the system, input dataset is first "discretized" since DBN works with discrete values, once again the workflow is updated (Fig. 5.16). The final network is then obtained: in Fig.5.17 a visualization of the inferred GRN obtained through Cytoscape is shown. The nodes without any connections with other nodes are not plotted.

At the end of the experiment, the User can save the workflow, start a new session or exit the program.

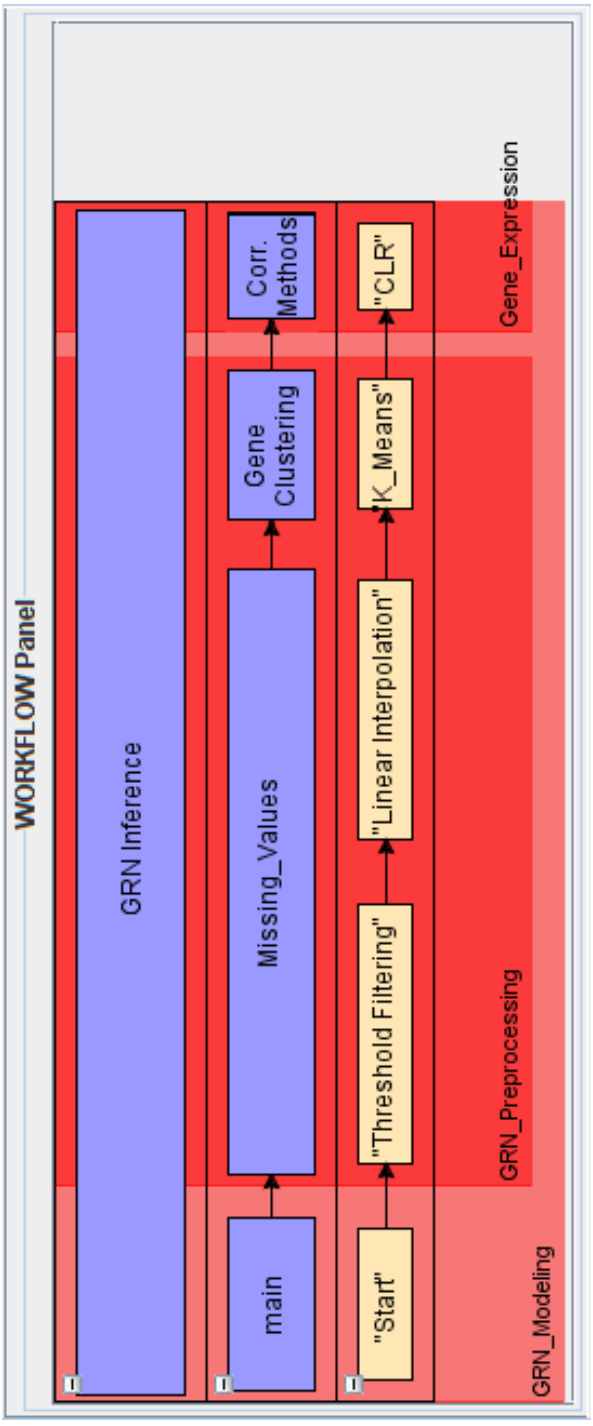

Figure 5.15: Workflow of the current experiment after CLR algorithm has been run. The active decision-making module is *Gene Expression*.

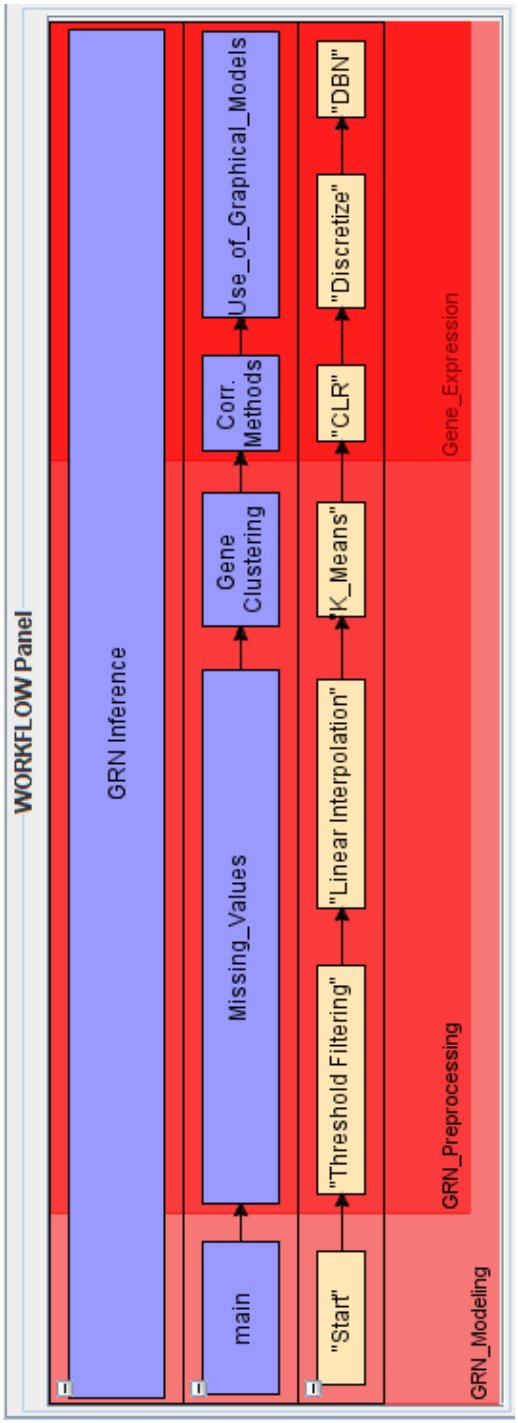

Figure 5.16: Final workflow of the current experiment. It can be eventually saved for sharing or reusing it. 5

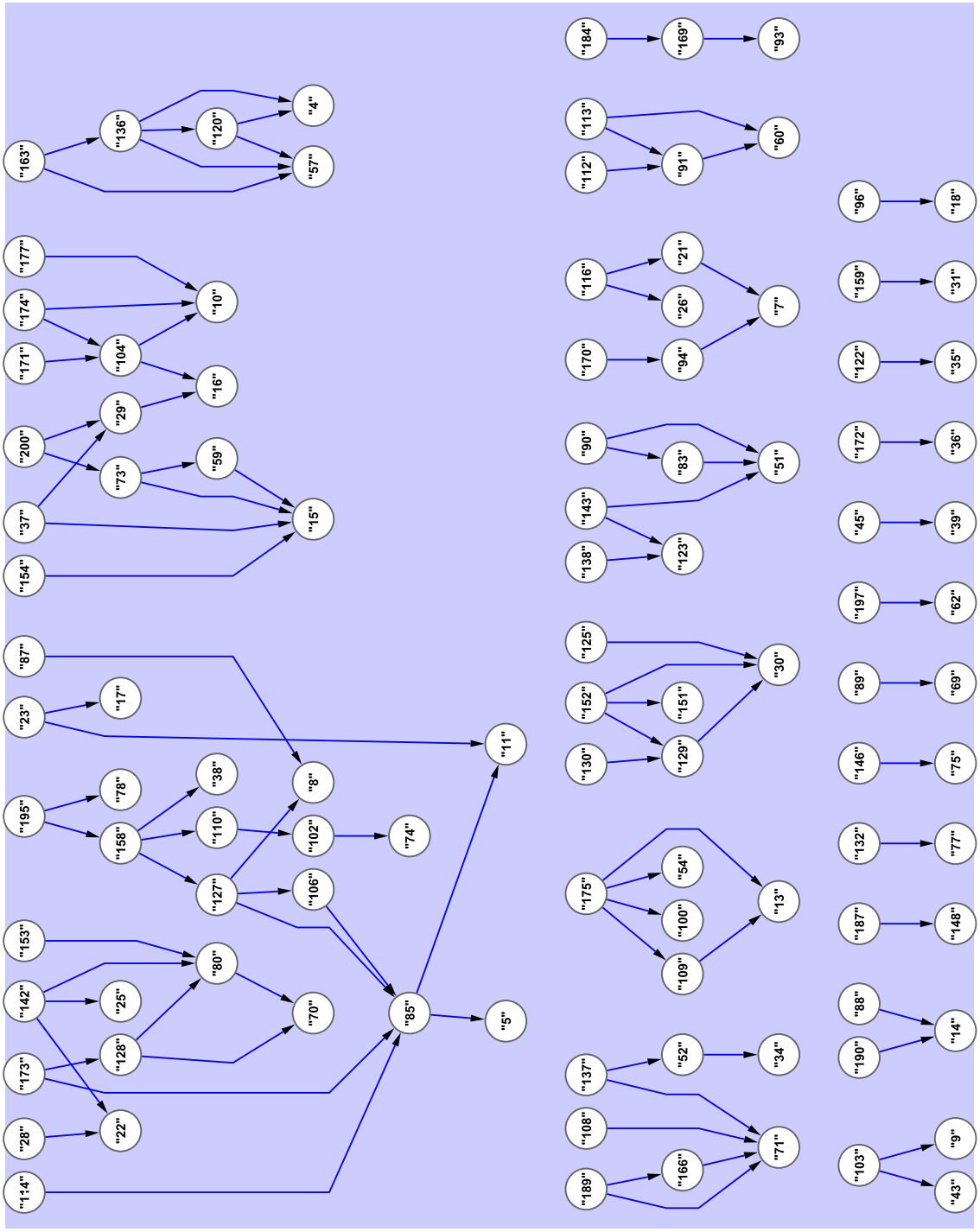

Figure 5.17: GRN inferred from the input dataset. This visualization is obtained by means of Cytoscape software, supported by BORIS system.

6

Materials & methods

In this Chapter the instruments and tools adopted for the development of the proposed system will be described.

Main features and characteristics of a Rule-Based system will be provided and the specific properties of Jess, the rule Engine for the Java platform, will be exploited.

Finally some of the basic concepts of Protege, the tool used for the design of the ontology, will be highlighted.

6.1 Rule-Based System

A Rule-Based system is an intelligent system that is able to make conclusions, or inferences, from a set of initial knowledge, called facts, by means of rules, representing reasoning activity. Rules are usually written in the traditional *if-then* statement of programming languages: the *if* part is called predicate or premises; the *then* part is called action or conclusion.

Rule-Based systems are not general purpose: they are designed and employed for a specific application domain. A domain represents the system's scope, that is all the set of information the rules could possibly work with.

Rule-Based systems are also known as Expert system, since they capture the knowledge of human experts in a particular domain. With this definition, the rules are intended to code the expertise, the skill and the heuristics typical of human experts.

The main difference between rule-based systems and common computer programs is their programming paradigm. Computer programs use a procedural approach, in the sense the programmer decide "what to do", "how to do" and in what order. Rule-based systems, on the other hand, use a declarative approach: a declarative program only tells the computer "what to do", but it does not give instructions about "how to do". That means declarative programs need some kind of runtime system that is able to use those declarative information in order to make conclusions, or inferences.

A declarative approach is well suited above all for solving problems without a clear algorithm solution, like for instance classification, prediction, diagnosis that have some heuristics or guidelines rather than a predefined set of instructions.

6.1.1 Architecture of a Rule-Based System

Main components of a typical Rule-based system are the Knowledge-Base (KB) and the inference engine.

KB contains both the pieces of information, called facts, and the rules. Facts can be seen as tables in a relational database, where each element has a set of attributes and relationships with other elements of the database. Each rule is in the form **IF** *precondition on facts is true* **THEN** *execute action* and it is activated when some constraints on the values of facts' attributes are satisfied. The set of all facts is also known as working memory.

The inference engine is made of three elements:

- a pattern matcher;
- an agenda;
- an execution engine.

The pattern matcher in an algorithm that is able to check the KB and realize what are the rules that can be activated according to the content of working memory. The pattern matching phase is the most expensive in terms of time and resources during the inference mechanism, for this reason a lot of research has

been done in this context in order to optimize this issue. It is important to point out that activated rules are not immediately executed, or "fired".

All activated rules, in fact, are written into the agenda, that is responsible for the scheduling of the rules to be fired. The agenda can resolve execution conflicts, that means it can decide in which order rules activated at the same time should be fired, using a conflict strategy. Common strategies take into account the complexity of each rule, its age, that is how much time it is stored into the agenda, and eventually some special properties like for instance priority values.

Finally the execution engine, after the agenda has decided the order in which rules have to be fired, can actually execute the right part of the rules. Firing a rule can have several effects: it can produce new knowledge, in the sense of new facts to be added to the KB; it can invoke other programming languages that define what happen when that rule fires; it can call external algorithm and tools whose results can, at last, update the KB.

The whole mechanism of the inference engine is not static, but it works as a cycle, or reasoning loop, as we can see in Fig. 6.1. The pattern matcher checks the KB for activated rules and stores them into the agenda; the agenda, through a conflict resolution strategy, decides the firing scheduling of the rules; the execution engine runs the right part of rules according to the order provided by the agenda, obtaining eventually new information that updates KB and that can trigger the activation of other rules; and then this mechanism can restart. New facts can be added to the KB also by the user, if he submit new inputs.

6.2 Jess: the Rule Engine for the Java Platform

The Rule-Based system of Boris has been implemented using Jess (71), the Rule Engine for the Java Platform. Jess is written totally in Java and it can be easily embedded in our framework. Jess inference engine uses RETE algorithm (72) as pattern matcher: this algorithm will be briefly described in the next Subsection. The agenda works with two different conflict resolution strategies: depth and breadth. With depth strategy, the default one, the most recent activated rules are fired first; with breadth strategy, rules are fired according to their activation

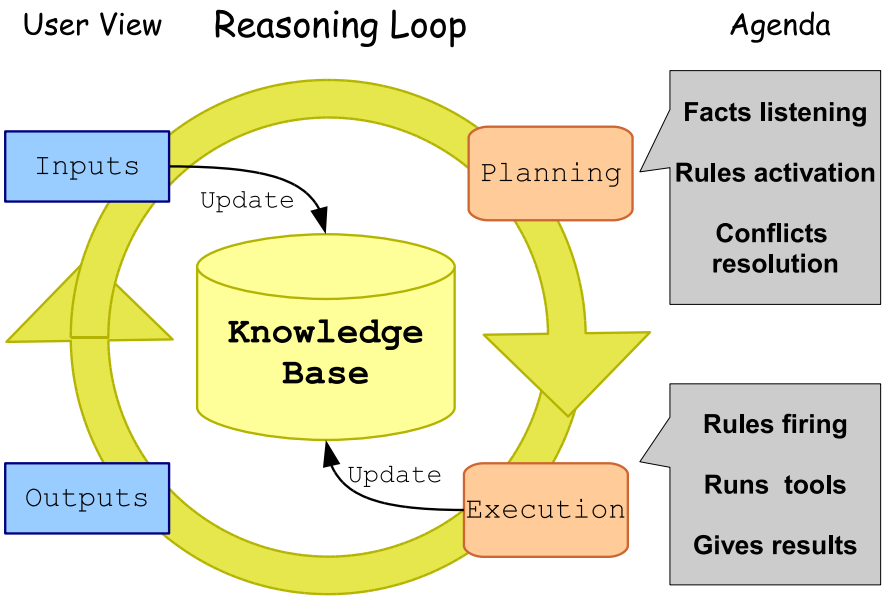

Figure 6.1: Reasoning Loop

order: this way the most activated rules fire last. In both strategies firing order can be modified changing rules priority.

Jess' working memory can be organized into modules: each module has its own set of facts and rules. Only one module a time can be active, or in other words can have the "focus", and only the rules belonging to the active module can be fired. By default the MAIN module has got the focus; the other modules can receive the focus when special rules, whose action is to shift the focus, are fired. The entire mechanism is managed by a stack, with the active module on the top and the other modules below, according to the order of the shift of focus. This way, when a module ends its job, the focus is automatically returned to last active module.

6.2.1 Rete algorithm

As stated in the above Sections, the main task of the pattern matcher component of an inference engine is to check the KB in order to find what rules are satisfied and activated so that they can be fired according to the scheduling of the agenda. A *brute force* approach, consisting in the analysis of every rules' premise against

the KB would be inefficient and difficult to scale for large working memories.

The Rete algorithm represents an efficient way to deal with the pattern matching issue. Over time, it has been enhanced and refined in past rule based system such as OPS5 (73), ART (74) and CLIPS (75): Jess implements the highest performance version. Rete algorithm improves simple pattern matching approach considering only new or deleted facts of working memory to be tested against the rules at each reasoning step. Moreover it stores past test results across iterations of the rule loop. Rete, that is the Latin word for net, organizes the pattern matcher by means of a network of interconnected nodes, so that the few facts interested in the inference mechanism are tested against a subset of rules could eventually match.

The performance of Rete algorithm with regards to the simple pattern matcher algorithm depends on the number of reasoning cycles. During the first reasoning loop, in fact, since Rete has to analyse all the facts of the working memory because there are not previous results to compare, the performance between the two algorithms are basically the same. Rete will, instead, outperform the basic algorithm for all the reasoning cycles after the first one.

6.3 Protege Ontology Editor

The knowledge base and the underlying ontology have been implemented with Protege (69, 70), that is one of the largest adopted tool for building an ontology and populate it with pieces of information that represent the knowledge of the system. Protege, through a clear and simple graphical user interface allows to define classes, to define their properties and relationships, to build hierarchies of concepts, to create instances. Moreover Protege is supported by a set of third parties plugins that extend its functionalities, adding for example visualization capabilities, using Jambalaya (76) or Ontoviz plugins (77), or a simple way to export instances into Jess facts by means of JessTab plugin (78). Protege is based on a Java implementation, so that it provides a set of Java APIs in order to ease its own interoperability with other systems.

6.4 Implementation Details

The computational instruments described in the previous Sections and adopted to implement the Knowledge-Based expert system belonging to BORIS framework, interact each other according to the scheme shown in Fig. 6.2. The main control program of the expert system, also implementing the GUI seen in Chapter 4, is written in Java. In this way it is possible to gain access both to protege editor, in order to get the initial knowledge, and both to Jess inference engine, in order to eventually assert new facts depending on the User's interaction. Protege and Jess, being both written in Java, provide a set of interface classes that simplify the communication with other Java programs. Jess accesses the knowledge base defined into Protege and, through JessTab plugin, assert the facts into its own working memory to allow the beginning of the inference process.

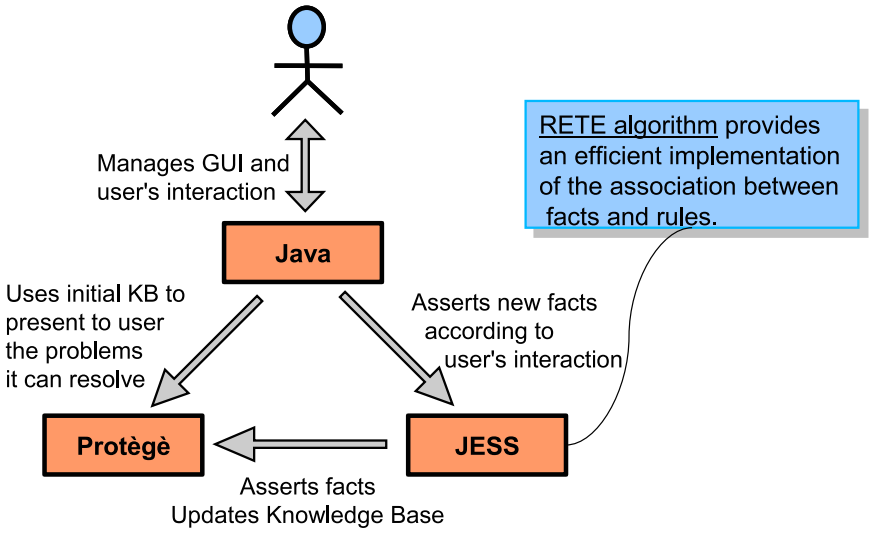

Figure 6.2: The interaction scheme among the computational tools adopted by the expert system belonging to BORIS framework-

7

Conclusions and Future Work

In this thesis, a Knowledge-Based expert system for bioinformatics domain has been presented. The Knowledge Base, populated thanks to the expertise extracted from more than 50 scientific papers, is based on an ontology of concepts. The proposed ontology provides a robust and coherent structure to the knowledge base and moreover it offers a simple way for maintaining and expanding it with new expertise. The designed ontology models three main global classes, interacting each other. The Tasks ontology represents what are the operations it is possible to carry on a specific kind of input biological data; the Tools ontology models the algorithms, software and services implementing the instances defined in the Tasks ontology; the Domain ontology gives the most important features and properties of the biological data to be analysed. Moreover the KB, consisting of facts and rules, is organized in a set of decision-making modules, each of them is responsible for a specific slide of the reasoning activity. The decision-making modules are arranged into a topological tree, where each level in the tree defines a meta-reasoning level, since the inference result of a high level decision-making module is the activation of a lower level module, representing a specialized reasoning task.

The expert system and all the PhD work has been developed inside a research project of National Research Council of Italy. The name of this project is BORIS (Bioinformatics Organized Resources: an Intelligent System). BORIS is born with the main goal of providing to the bioinformatics community a simple and

at the same time powerful instruments that is able to offer decision support during the execution of a bioinformatics experiment. Given the plenty of services, strategies, tools and algorithms available, it is often very difficult to discern what are the best suited methodologies and techniques for a given problem. BORIS proposes an hybrid architecture, integrating a declarative approach, with regards to its decision-making activity; a procedural approach, with regard to its capability to run and configure the selected tools; and a process approach because it generates a workflow that traces all the taken decision and executed tools during a typical session. Focusing on these two main features, i.e. the decision-making process and the workflow building, BORIS system can be seen is an ideal joint between classical decision support system and more recent workflow management system.

BORIS system has been tested with an actual case study: the reverse engineering of gene regulatory network. In this thesis a typical experimental session is shown, highlighting the original features of the system and how the three different approaches of its hybrid architecture work together.

In the near future, the whole BORIS framework will be turned into a web application so that it will be freely accessible by the community.

Looking at the future developing progress, the proposed expert system will be provided with an editor and formal guidelines that will offer the possibility to introduce new knowledge and expertise in a very simple way. New application scenario in bionformatics domain will be added, and at the same time the existing scenarios will be updated and enhanced when new tools and services will be available.

The ontology organization into the three-folded main classes (Tasks, Tools, Domain) provides a very general purpose knowledge arrangement. That means that the expert system can be adapted with few modifications to other application domain, like for instance the clinical field. The system, in fact, can be used in order to combine the characteristics of an electronic clinical workflow with an Electronic Medical Record (EMR). The former represents a decision support system that can assist a medic in the diagnosis and prognosis activities. Its suggestion can be given according to the patient's EMR, so that its previous

medical history will be taken into account. The EMR will be then updated with the current medical cures.

References

[1] Human Genome Sequencing Consortium International, "Finishing the euchromatic sequence of the human genome", *Nature*, vol. 431, pp. 931–945, 2004. 1

[2] National Center for Biotechnology Information NCBI 1

[3] D. J. Power, "Brief History of Decision Support Systems", DSSResources.COM. 3

[4] A. Gorry and M. S. Scott-Morton, "A Framework for Information Systems", *Sloan Management Review*, vol. 13(1) pp. 56–79, 1971. 3

[5] J. H. Moore and M. G. Chang, "Design of Decision Support System", *Database*, vol. 12, pp. 8–14, 1980. 3

[6] B. J. Parker, "Decision support systems: the reality that seems hard to accept", *Omega*, vol. 14(2), pp. 135–143, 1986 3

[7] P. G. W Keen, M. S. Scott Morton, "Decision support systems : an organizational perspective", *Reading, Mass.,* Addison-Wesley Pub. Co 1978. 3

[8] U. Cortes, M. Sanchez-Marre, L. Ceccaroni, I.R.-Roda and M. Poch, "Artificial Intelligence and Environmental Decision Support Systems", *Applied Intelligence*, vol. 13(1), pp. 225-239, 2000. 4

[9] M. K. El-Najdawi, A. C. Stylianou, "Expert support systems: integrating AI technologies", *Commun. ACM*, vol. 36(12), pp. 55–ff, 1993. 4, 17

[10] D. J. Power, "Decision Support Systems: Concepts and Resources for Managers", Westport, CT Greenwood/Quorum, 2002. 4

[11] J. Wyatt , D. Spiegelhalter, "Field trials of medical decision-aids: potential problems and solutions", Clayton P (Eds.), *Proc. 15th Symposium on Computer Applications in Medical Care, Washington* pp. 3–7, 1991.

[12] B. G. Buchanan and E. H. Shortliffe, "Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project", *AAAI*, 1984. 4

[13] G. L. Steele, "Common Lisp the Language", *2nd Edition, Digital Press,* 1990. 4

[14] E. H. Shortliffe, A. C. Scott, M. B. Bischoff, et al., "ONCOCIN: an expert system for oncology protocol management", *International Joint Conference on Artificial Intelligence*, pp. 876-881, 1981. 4

[15] M. Ceccarelli, A. Donatiello, D. Vitale, "KON3: a Clinical Decision Support System, in oncology environment, based on knowledge management", *IEEE International Conference on Tools with Artificial Intelligence*, vol. 2 pp. 206–210, 2008. 4

[16] M. K. Goldstein, B. B. Hoffman, R. W. Coleman et al., "Implementing clinical practice guidelines while taking account of changing evidence: ATHENA DSS, an easily modifiable decision-support system for managing hypertension in primary care", *Proc AMIA Symp.*, pp 300–304, 2000. 4

[17] M. A., Musen, S. W. Tu, A. K. Das and Y. Shahar, "EON: A component-based approach to automation of protocol-directed therapy", *Journal of the American Medical Information Association*, vol. 3(6), pp. 367–388, 1996. 4

[18] J. P. Bury, C. Hurt, C. Bateman et al., "LISA: A Clinical Information and Decision Support System for Collaborative Care in Childhood Acute Lymphoblastic Leukaemia", *Proceedings of the annual AMIA Annual Symposium*, 2002. 4

[19] R. Boulme, D. Gonzalez and JC. Schmit, "Storing genotypic resistance data and linking to other clinical information", *XV International AIDS Conference*, Bangkok, Thailand, 2004. 4

[20] D. Hollinsworth, "The Workflow Reference Model", *Tech Rep TC00-Workflow Management Coalition*, 1994. 5

[21] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, T. Oinn, "Taverna: a tool for building and running workflows of services", *Nucleic Acids Res*, vol. 34, 2006. 5, 12

[22] P. Romano, E. Bartocci, G. Bertolini, F. De Paoli, D. Marra, G. Mauri, E. Merelli and L. Milanesi, "Biowep: a workflow enactment portal for bioinformatics applications", *BMC Bioinformatics*, vol. 8, 2007. 5, 12

[23] E. Bartocci, F. Corradini, E. Merelli, L. Schortichini. "BioWMS: a Web-based Workflow Management System for Bioinformatics", *BMC Bioinformatics*, vol. 8(1), 2007. 5, 12

[24] P. Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armananzas, G. Santafe', A. Perez and V. Robles. "Machine learning in bioinformatics", *Briefing in bioinformatics*, vol. 7(1), pp. 86–112, 2005. 22

[25] V. Robles, P. Larraaga, J.M. Pea, E. Menasalvas, M.S. Prez, V. Herves. "Bayesian networks as consensed voting system in the construction of a multi-classifier for protein secondary structure prediction". *Artificial Intelligence in Medicine*, vol. 31, pp. 117–136, 2004. 22

[26] H. Yamakawa, K. Maruhashi, Y. Nakao, "Predicting Types of Protein-Protein Interactions Using a Multiple-Instance Learning Model". *LNCS*, vol. 4384, pp. 42–53, 2007. 24

[27] D. Hanisch, K. Fundel, H.T. Mevissen, R. Zimmer, J. Fluck, "ProMiner: rule-based protein and gene entity recognition", *BMC Bioinformatics*, vol. 6, Suppl 1:S14, 2005. 24

[28] J.C. Whisstock, A.M. Lesk. "Prediction of protein function from protein sequence and structure". *Quarterly Reviews of Biophysics*, Cambridge University Press, vol. 36(3), pp. 307–340, 2003. 24

[29] E.C. Su, H.S. Chiu, A. Lo, J.K. Hwang, T.Y. Sung, W.L. Hsu. "Protein subcellular localization prediction based on compartment-specific features and structure conservation". *BMC Bioinformatics*, vol. 8, 2007 24

[30] "CASP: Critical Assessment of Techniques for Protein Structure Prediction". http://predictioncenter.org/index.cgi 2, 22

[31] K. Yeung, M. Medvedovic, R. Bumgarner, "Clustering gene-expression data with repeated measurements", *Genome Biology*, vol. 4, 2003. 25, 43

[32] M.B. Eisen, P.T. Spellman, P.O. Brown, D. Botstein "Cluster analysis and display of genome-wide expression patterns", *Proc Natl Acad Sci*, vol. 95 pp. 14863–14868, 1998. 25, 43

[33] P. Dhaeseleer, X. Wen, S. Fuhrman, R. Somogyi, "Linear modeling of mRNA expression levels during CNS development and injury", *Proceeding of the Pacific Symposium on Biocomputing*, pp. 41-52, 1999. 25, 43, 52

[34] D. Pe'er, A. Regev, G. Elidan, N. Friedman, "Inferring Subnetworks from Perturbed Expression Profiles", *Bioinformatics*, vol.17, Suppl. 1 2001. 25

[35] Y. Wang, T. Joshi, X.S. Zhang, D. Xu, , L. Chen, "Inferring gene regulatory networks from multiple microarray datasets", *Bioinformatics*, vol. 22(19), pp. 2413-2420, 2006. 25, 43

[36] B. Johnson, B. Shneiderman, "Tree-maps: a space-filling approach to the visualization of hierarchical information structures", *Proceedings of IEEE Conference on Visualization*, pp. 284–291, 1991. 30

[37] K. Chen, N. Rajewsky, "The evolution of gene regulation by transcription factors and microRNAs", *Nat Rev Genet*, vol. 8(2), pp. 93–103, 2007. 39

[38] S. Cates, "Microarray Experiments", *Connexions Web site*, http://cnx.org/content/m11050/2.17/, Oct 1, 2007. 40

[39] J.M. Bartlett, D. Stirling, "A Short History of the Polymerase Chain Reaction", *Methods in Molecular Biology*, vol. 226, pp. 3–6, 2003. 41

[40] S.L. Berger, D.M. Wallace, R.S. Puskas, W.H. Eschenfeldt, "Reverse transcriptase and its associated ribonuclease H: interplay of two enzyme activities controls the yield of single-stranded complementary deoxyribonucleic acid", *Biochemistry*, vol. 22(10), pp. 2365–72, 1983. 41

[41] Y. Huang, I. Tienda-Luna, Y. Wang, "Reverse engineering gene regulatory networks", *IEEE Signal Processing Magazine*, vol. 26(1) ,pp. 76–97, 2009. 43

[42] K. H. Cho, S. M. Choo, S. H. Jung, J. R. Kim, H. S. Choi, and J. Kim, "Reverse engineering of gene regulatory networks", *Systems Biology*, pp. 149–163 2007. 43

[43] M. Hecker, S. Lambeck, S. Toepfer, E. van Someren, R. Guthke, "Gene regulatory network inference: Data integration in dynamic models–A review", *Biosystems*, vol. 96(1), pp 86–103, 2009. 43, 48

[44] M. Zou, and S. D. Conzen "A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data", *Bioinformatics*, vol. 21(1), pp. 71–79, 2005. 43

[45] S. Y. Kim, S. Imoto and S. Miyano, "Inferring gene networks from time series microarray data using dynamic Bayesian networks", *Briefings in Bioinformatics*, vol. 4(3), pp. 228–235, 2003. 43

[46] I. Gat-Viks, A. Tanay, D. Raijman, R. Shamir, "A probabilistic methodology for integrating knowledge and experiments on biological networks", *J. Comput. Biol.*, vol. 13(2), pp. 165–181, 2006. 43

[47] H. Lahdesmaki, S. Hautaniemi, I. Shmulevich, O. Yli-Harja, "Relationships between probabilistic Boolean networks and dynamic Bayesian networks as models of gene regulatory networks", *Signal Process.*, vol. 86(4), pp. 814–834, 2006. 43

[48] A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Favera, R. A. Califano, "ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context", *BMC Bioinform.* vol. 7, 2006. 43

[49] T. S. Gardner, D. di Bernardo, D. Lorenz, J. J. Collins, "Inferring genetic networks and identifying compound mode of action via expression profiling", *Science*, vol. 301, pp. 102–105, 2003. 43

[50] D. di Bernardo, M. Thompson, T. Gardner, S. Chobot, E. Eastwood, A. Wojtovich, S. Elliott, S. Schaus, J. Collins, "Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks", *Nat. Biotechnol.*, vol. 23(3), pp. 377-383, 2005. 43

[51] J.P. van Berlo, E.P. van Someren, M.J. Reinders, "Studying the Conditions for Learning Dynamic Bayesian Networks to Discover Genetic Regulatory Networks". *Simulation* vol. 79(12), pp. 689–702, 2003. 43, 48

[52] J. Schfer, K. Strimmer, "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics", *Statist Appl Genet Mol Biol*, vol. 4(32), 2005. 43

[53] Faith *et al.* "Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles", *PLoS Biol*, vol. 5, doi:10.1371/journal.pbio.0050008, 2007. 43

[54] R. Opgen-Rhein, K. Strimmer, "From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data", *BMC Syst. Biol.*, vol. 1(37), 2007. 46

[55] S. Wichert, K. Fokianos, K. and Strimmer, "Identifying periodically expressed transcripts in microarray time series data", *Bioinformatics* vol. 20, pp. 5–20, 2004. 25, 55

[56] M. Ahdesmaki, H. Lahdesmaki, R. Pearson, H. Huttunen, O. and Yli-Harja, "Robust detection of periodic time series measured from biological systems", *BMC Bioinformatics* vol. 6(117), 2005. 55

[57] M. Ahdesmaki, H. Lahdesmaki, A. Gracey, I. Shmulevich, , and O. Yli-Harja, "Robust regression for periodicity detection in non-uniformly sampled time-course gene expression data", *BMC Bioinformatics* vol. 8(233), 2007. 55

[58] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations", *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967. 55

[59] T. Kohonen, "Self-Organizing Maps", Third extended edition, Springer, 2001. 55

[60] J. C. Bezdek, J. M. Keller, R. Krishnapuram and N. R. Pal, "Fuzzy Models and Algorithms for Pattern Recognition and Image Processing", Springer, NY, 1999. 56

[61] http://graphviz.org/ 43

[62] P. Shannon P et al., "Cytoscape: a software environment for integrated models of biomolecular interaction networks", *Genome Research*, vol. 13(11), 2003. 43

[63] J. Beirlant, E. Dudewicz, L. Gyorfi, E. van der Meulen, "Nonparametric entropy estimation: An overview", *Int J Math Stat Sci*, vol. 6(1), pp. 17–39, 1997. 44

[64] T.M. Cover, J.A. Thomas, "Elements of Information Theory". New York: John Wiley & Sons; 1991. 44

[65] A.J. Butte, I.S. Kohane, "Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements", *Pac Symp Biocomput*, pp. 418-429, 2000. 45

[66] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, R. W. Davis, "A Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle", *Molecular Cell*, vol. 2(1), pp. 65–73, ISSN 1097-2765, DOI: 10.1016/S1097-2765(00)80114-8. 48

[67] T. Cover, P. Hart, "Nearest neighbor pattern classification", *IEEE Transactions on Information Theory*, vol. 13(1), pp. 21–27, 1967. 52

[68] S.A. Dudani, "The distance-weighted k-nearest-neighbor rule", *IEEE Trans. Syst. Man Cybern.*, SMC-6, pp. 325-327, 1976. 52

[69] The Protege Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu. 66

[70] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubzy, H. Eriksson, N. F. Noy, and S. W. Tu, "The evolution of protege: An environment for knowledge-based systems development", I*nternational Journal of Human-Computer Studies*, vol. 58, pp. 89-123, 2003. 66

[71] Sandia National Laboratories, "Jess: The rule engine for the JavaTM platform", Available at http://herzberg.ca.sandia.gov/jess/, 2003. 64

[72] C. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", *Artificial Intelligence*, vol. 19, pp 17-37, 1982 64

[73] L. Brownston, R. Farrell, E. Kant, N. Martin, "Programming Expert Systems in OPS5", Addison-Wesley, 1985. 66

[74] B.Clayton, "ART Programming Tutorial", Vol 1–3 Department of Artificial Intelligence, University of Edinburgh. 66

[75] J. C. Giarratano, G. D. Riley, "Expert Systems: Principles and Programming", Third Edition, Course Technology, 1998. 66

[76] SHriMP research group, "Jambalaya - Information Browser for Protege", Department of Computer Science University of Victoria. 66

[77] "OntoViz: a visual browser for ontologies", http://ontoviz.sourceforge.net/. 66

[78] H. Eriksson, "Using JessTab to Integrate Protg and Jess", *IEEE Intelligent Systems*, vol. 18(2), pp. 43–50, 2003. 66

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other Italian or foreign examination board.

The thesis work was conducted from January 2008 to February 2011 under the supervision of Dr. Daniele Peri at University of Palermo.

Palermo, 2011-15-02