Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Design and Development of a Knowledge-Based Expert System for Bioinformatics

Rapporto Tecnico N.:
RT-ICAR-PA-11-01 **maggio 2011**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sede di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Design and Development of a Knowledge-Based Expert System for Bioinformatics

A. Fiannaca[1], S. Gaglio[12], M. La Rosa[1], R. Rizzo[1], A. Urso[1]

Rapporto Tecnico N.: **RT-ICAR-PA-11-01**

Data: **maggio 2011**

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo, Viale delle Scienze edificio 11, 90128 Palermo.
[2] Università degli Studi di Palermo, Dipartimento di Ingegneria Chimica Gestionale Informatica e Meccanica, Viale delle Scienze, 90128 Palermo.

Design and Development of a Knowladge-Based Expert System for Bioinformatics

Antonino Fiannaca[1], Salvatore Gaglio[1,2], Massimo La Rosa[1], Riccardo Rizzo[1] and Alfonso Urso[1]

[1] ICAR-CNR, National Research Council, Viale delle Scienze, Ed. 11, 90128 Palermo, Italy
[2] DICGIM, University of Palermo, Viale delle Scienze, Ed. 6, 90128 Palermo, Italy
fiannnaca@pa.icar.cnr.it, gaglio@unipa.it, larosa@pa.icar.cnr.it, ricrizzo@pa.icar.cnr.it, urso@pa.icar.cnr.it

Abstract

Decision Support Systems and Workflow Management Systems have become essential tools for some business and scientific field. This thesis propose a new hybrid architecture for problem solving expertise and decision-making process, that aims to support high-quality research in the field of bioinformatics and system biology.

The first part of the dissertation introduces the project to which belong this thesis work, i.e. the "Bioinformatics Organized Resources - an Intelligent System" (*BORIS*) project of the ICAR-CNR; the main goal of BORIS is to provide an helpful and effective support to researchers or experimentalist, that have no familiarity with tools and techniques to solve computational problems in bioinformatics and system biology.

In the second part of this paper, the proposed hybrid architecture is described in detail; it introduces a three-dimensional space for the BORIS system, where the viewpoints of declarative, procedural and process approaches are considered. Using the proposed architecture, the system is able to help the experimentalist choosing, for a given problem, the right tool at the right moment, to generate a navigable Workflow at different abstraction layers, extending current workflow management systems and to free the user from implementation details, assisting him in the correct configuration of algorithms/services.

Two case studies are presented respectively about reverse engineering gene regulatory network and extraction of protein complexes from protein-protein interaction networks, in order to show how the system faces a problem and how it interacts with the user.

Contents

List of Figures

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 (36), 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 work 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 this 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).

1.2 Background

The system proposed in this work, aims to improves classical concept of DSS in many ways. First of all, during the execution of an experiment, it traces its evolution by using a workflow of the decisions, enabling this way the possibility for the user to do backtracking in order to change previous decisions. Furthermore it is possible to save the whole workflow and results for sharing and reusing them. When the system suggests a list of suitable strategies or algorithms, it presents, for each of them, a brief description, a series of pros and cons and bibliographic references. Moreover our system not only offers support giving advices and recommendations, but it helps the user in the proper configuration and running of the strategies or algorithms selected during the decision making process. This last features moves our systems towards modern Workflow Management Systems (WFMS) (26) which 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.

1.2.1 Decision Support Systems

Decision Support Systems (DSS) have been created and investigated more than 35 years ago; the developments of DSS begun with building model-oriented DSS in the late 1960s, where the computing systems to help in decision-making process were known as management decision systems (MDS), continuing with theory developments in the 1970s and the implementation of financial planning systems in the early and mid 80s. The implementation of the web-based DSS started in the mid-1990s, with the specification of HTML 2.0, the expansion of the World Wide Web in companies, and the introduction of hand held computing. Today, the Web 2.0 technologies, mobile-integrated communication and computing devices, and improved software development tools have revolutionized DSS user interfaces.

Due to its different application areas, there are several definitions of DDS,

one of the earlier was introduced by Gorry and Scott-Morton (4), that claim a DDS, "an interacting computer-based system that helps the decision maker in the use of data and models in the solution of unstructured problems". Of course, the DSS will collect and analyse the data and then present it in a way that can be interpreted ny humans. Some DSS come very close to acting as artificial intelligent agents. DSS applications are not single information resources, but the combination of integrated resources working together (9).

Some of the main features of a DSS are:

- to incorporates both data and models;
- to learn through the composition of models;
- to improve the effectiveness of decisions, not the efficiency with which decisions are being;
- to assist decision-makers in decision processes in unstructured or semi-structured environments;
- to support and do not replace user judgment;
- to provides a fast response to unexpected situations, caused by changed conditions, by means of the ability to try several different strategies under different configurations;

Although the user interface (UI) is not in the previous list, it holds a crucial aspect of DSSs. Systems with user interfaces that are cumbersome or unclear or that require unusual skills to be understood, are rarely useful and accepted in practice and could lead the user to a wrong interpretation of results. On the contrary, UI should play a tutoring role, teaching to users how the DSS reasons about domain model, improving their own thinking. A good user interface to DSSs have to support model construction and model analysis, reasoning about the problem structure in addition to numerical calculations and both choice and optimization of decision variables.

Generally there are two main approaches (10) to supporting decision making in DSS, according to the quality of human intuitive reasoning strategies, implementing the expertise of DSSs. The first aims at building support procedures or systems that imitate human experts. This category contains expert systems, that are computer programs based on rules elicited from human domain experts. These systems can supporting decision making in the same way human experts can do. They are based on intuitive human reasoning and lack soundness and formal guarantees with respect to the theoretical reliability of their results. The cons of the expert system approach is that along with imitating human thinking

and its efficient heuristic principles, they also imitate its undesirable aws (11). The second approach is oriented to the application of formal methods; in fact, it is based on the assumption that the most reliable method of dealing with complex decisions is through a small set of normatively sound principles of how decisions should be made. This point of view makes these systems philosophically distinct from those based on ad hoc heuristic artificial intelligence methods, such as rule-based systems. According to the second approach, the goal of a DSS is to support unaided human intuition, just as the goal of using a calculator is to aid human's limited capacity for mental arithmetic.

In the following a category of DSSs based on expert system is reported.

Knowledge-driven DSS (KDSS) are person-computer systems with specialized problem-solving expertise (15). KDSS are composed by three components (16):

- the knowledge (stored as rules, frames, or probabilities) of relations among problems and indicators related to a particular topic or domain,
- the "Skill" or methods for solving some of the problems
- the capability of give the reasoning behind a conclusion it has reached.

In general, a knowledge-driven DSS suggests or recommends actions to targeted users. This type of DSS has specialized problem-solving expertise relevant to a specific narrow task.

KDSS have been most applied in diagnosis in various clinical domains. The so called Clinical DSS (CDSS) (17), typically integrates a medical knowledge base, patient data and an inference engine in order to provide medical recommendations about specific cases. CDSSs form a significant part of the field of clinical knowledge management technologies, since they can support the clinical process and use of knowledge from diagnosis and investigation keeping patients on research and chemotherapy protocols, tracking orders, referrals follow-up, and preventive care. Moreover they are responsible of medical treatment plan processes, promoting use of best practices, condition-specific guidelines, and population-based management (12).

MYCIN (18) was a rule-based expert system designed to diagnose and recommend treatment for certain blood infections (antimicrobial selection for patients with bacteremia or meningitis). It was later extended to handle other infectious diseases. Clinical knowledge in MYCIN is represented as a set of IF-THEN rules with certainty factors attached to diagnoses, that use a basic backward chaining reasoning strategy. MYCIN was developed in the mid-1970s by Ted Shortliffe and colleagues at Stanford University. It is probably the most famous early expert system, described as "the first convincing demonstration of the power of the rule-based approach in the development of robust clinical decision-support systems" (13). An extended version of this DSS, EMYCIN (Essential MYCIN), was

developed at Stanford in 1980 and was used to build diagnostic rule-based expert systems such as PUFF, a system designed to interpret pulmonary function tests for patients with lung disease.

A rule-based medical expert system for oncology protocol management, called ONCOCIN (20), was developed at Stanford University. It was designed in order to assist physicians with the treatment of cancer patients receiving chemotherapy. ONCOCIN was one of the first DSS which attempted to model decisions and sequencing actions over time, exploiting a customized flowchart language, in fact it used an application area where the history of past events and the duration of actions are important.

Another CSS was developed in Italy, as a joint effort among companies, university and regional government agencies. This project, known as Kon3 (21), is oriented to the development of technologies for a sharable knowledge based on Clinical Practice Guidelines at a reasonable cost and effort, and in a form that can be integrated gracefully and supportively into the clinicians workflow via functions of the local clinical information system. the knowledge base of KON3 is composed by guideline and semantic information representation, whose ontology is based on Knowledge representation about patients data, oncology taxonomy (Breast Cancer) and guidelines model.

Other currently used CDSS are: ATHENA (22), implementing guidelines for hypertension using Stanford Medical Informatics EON architecture (23); LISA (24) that is a clinical information system for supporting collaborative care in the management of children with Acute Lymphoblastic Leukaemia (ALL); TherapyEdge (25) that is a web-enabled decision support system for the treatment of HIV.

1.2.2 Workflow Management Systems

Workflow Management Systems (WFMS) are computer systems that allow organizations to define and control the various activities associated with a business process. Most WFMSs allow the opportunity to measure and analyze the execution of the process so that continuous improvements can be made, either in short-term (e.g., the reallocation of tasks to balance the workload at any point in time) or long-term (e.g., redefining portions of the workflow process to avoid bottlenecks in the future).

In this way, they can define a proper workflow for for each type of jobs or processes, according to user needs. WFMSs also integrate with other systems in order to provide a process structure which employs a number of independent systems, organizing resources and documents from diverse sources like document management systems, production applications, etc. That all can be integrated because Workflow Management Systems manage the dependencies required for

the completion of each task.

The most of Workflow Management Systems, including the one presented in this work, have some typical features (14):

- A tool for the process definition: it is a graphical or textual tool for defining the business process, according to user needs and computer application.
- The Simulation/Prototyping/Piloting process: it is possible to simulate or create prototype and/or pilot versions of a particular workflow, in order to try and test a process.
- Initiation and Control of tasks: the business process is initiated and each resource is scheduled and/or engaged to complete each activity as the process progresses.
- Invocation of applications able to view and manipulate data: all the documents, including temporary outputs can be invoked to allow workers to create, update, and view processed data in real time.
- Print a Worklists: WFMSs can allow each user to identify their current tasks, anticipating or estimating the workload, that can be visualized as well.
- Automation of task: Computerized tasks can be automatically invoked. This might include such things as letter writing, email notices, or execution of production applications. Task automation often requires customization of the basic workflow product.
- Tracking and Logging of Activities: all the Information about each task can be logged, in order to let user able to later analyze the process and check the results of certain tasks.

For these reasons, WFMS benefits including the opportunity to improve both the underlying business process and the existing organizational structure, since all the activity steps, roles, and rules are built into the system and less intervention needed to manage the business process. In addiction, they allow for the separation of information technology from workflow management, integrating the business process directly under the control of the system users.

The most used and famous WFMS for bioinformatics is Taverna (27), an application tool that has been created by the myGrid team and funded through the OMII-UK, an open-source organization that empowers the UK research community. Taverna 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. It allows for the automation of experimental methods through the use of a number of different (local or remote) services from a very diverse set of domains (from biology, chemistry and medicine to music, meteorology and social sciences), managing more than 3500 services such as remote resources and analysis tools, Web and grid services. The system works by means of a GUI that integrate a graphical workflow designer with drag and drop workflow components, that is available as a desktop Workbench, Server, through a portal or on a cloud.

A WFMS created for bioinformatics, known as Bioinformatics Workflow Enactment Portal (BioWEP) (28), was developed by Italian National Institute for Cancer Research Genoa (IST). This portal is a web-based client application that allows the user to search and run a predefined set of workflows, already tested, validated and annotated. It is oriented to the simplify access for all researchers, supporting the selection and execution of predefined workflows, obtained by an exhaustive set of biomedical databases.

Another web-based system for bionformatics built upon an agent oriented middleware architecture is BioWMS (29); application domain features are embedded inside the agents knowledge and proactiveness and mobility inside the agent behaviour. Since agents are workflow executors, the resulting workflow engine is a multi-agent system typically open, flexible, and adaptive.

2

Bioinformatics Organized Resources - an Intelligent System

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

2.1 BORIS Project

This 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.

3

Architecture of Decision Support System

The new hybrid architecture has been designed in order to include some features of three different approaches: the procedural, the declarative and the process one. This way the proposed system takes advantages of both decision support systems and workflow management systems.

3.1 Hybrid Architecture

There are two main approaches for making the architecture of a decision support system, in fact they can be represented procedurally or declaratively.

Architectures with declarative representations have knowledge in a format that may be manipulated decomposed and analyzed by its reasoners, i.e the knowledge about a domain is intricate with the control of reasoning process, and thus is implicitly represented. Architectures with procedural representations encode how to achieve a particular result, i.e. the knowledge is explicitly represented and separated from the reasoning procedures.

In artificial intelligence, the procedural knowledge is often represented as finite-state machine or computer program, whereas an AI system based on declarative knowledge is typically based on a domain-independent planning algorithm that indicate how to use the system skills to reach a goals.
Examples of procedural processes in AI are:

- The Subsumption Architecture by (39) that is a reactive robot architecture arranged in order to decompose complicated intelligent behaviour into many simple behaviour modules that implement a particular goal of the agent.

- The Procedural Reasoning System (PRS) (40), that is a framework for constructing real-time reasoning systems that can perform complex tasks in dynamic environments using the Belief-Desire-Intention software model.
- Some programming languages as C, Java, Perl and JavaScript, that declare the control flow.
- Some procedural programs as the Linux Kernel or the Apache Server.

Examples of declarative processes in AI are:

- Dynamic Control Architecture by (41), where the agent acts in a complex dynamic environment, having only an unstructurated and broken knowledge about this environment.
- Homer by (42) implements a robot submarine that is designed to act, reason and reflect on its experience: it can plan how to achieve its instructions, modifying its plans as required during execution.
- Some programming languages as SQL, YACC and markup languages such as HTML, that contain the logic of a computation without describing its control flow.
- Some functional languages as Prolog and Lisp.

In the table 3.1 some characteristics of declarative vs procedural approach are reported. This table clearly shows some advantages/disadvantages of these knowledge representation techniques.

The proposed system aims at integrate both points of view, in order to merge their advantages, offering to the user an exploration of the space of the problem, as exhaustive as possible. Sometimes, whether represented knowledge is viewed as declarative or procedural is not an intrinsic property of the knowledge base, but is a function of what is allowed to read from it (37). For example, if production systems may view themselves, then they are declarative, otherwise they are procedural.

According to the coexistence between these two knowledge representation related to the user point of view, the proposed architecture use both declarative and procedural approaches at different times, taking advantage of their different advantages.

In the past, a similar approach was adopted by (38), on the design of the ATLANTIS architecture for mobile robots. Based on the observation that an

Table 3.1: Comparison between Declarative and Procedural approaches in artificial intelligence and programming.

DECLARATIVE APPROACH	**PROCEDURAL APPROACH**
The representation of knowledge about objects, events and their relationships and states is static.	All the control information necessary to use the knowledge is embedded in the knowledge itself.
It defines the rules about "**what to do**" with knowledge and not how to do it.	It encodes "**how to achieve**" a specific result, requiring an interpreter to follow instructions specified in knowledge.
It is slow, because the system requires code interpretation.	It is fast to use, because all the processes have a direct execution.
The system transparency is improved, easing system governance.	It works as a black box and could be hard to debug.
The system is data-oriented.	The system is process-oriented.
Turn out to be easy to update the system representation, facilitating system maintenance;	It is easy to write, because the knowledge is defined step by step in an explicit way.

environment can be investigate at different levels, that require some proper mechanisms for dealing with them. For example the planning could be important in a level, whereas a quick reaction might be critical for the life of the robot in the other levels. For this reason author defined two different layer for its robot: the control layer, that uses a procedural knowledge, and the deliberative layer, that uses a declarative knowledge.

The hybrid architecture for the decision support system developed in this work, according to the software architecture of the BORIS project (see section 4.1), not only aims to exploit both declarative and procedural approaches, but integrate also another approach from workflow management systems, i.e. the process approach.

The term "process approach" is inherited by business process management,

that is a collection of structured activities (or tasks) that produce a specific service or product (or a goal) for a particular typology of customers. Usually, it can be visualized with a flowchart as a sequence of activities or a workflow of tasks.

Therefore, the process approach is a management strategy where managers supervise the interaction between these processes, and the inputs and outputs that glue these processes together. Each process is an integrated set of activities that uses resources to transform inputs into outputs or, in other words, a system exists whenever several processes are interconnected using such *input-output relationships*.

This point of view is used by several WFMS platforms (47), where the process model describes the behavioral aspect of a workflow specification, such as the process evolution from its initial state to one of its final states. The elementary unit of the workflow created with the proposed system is the task, that is interrelated via connectors, such as join and split elements. Then there are subprocesses that allow the modularization of each generated workflow in terms of self-contained activity fragments, according to strategies/heuristics taken into account.

3.1.1 DSS space

As stated in the previous section, the hybrid system introduced in this work collects at the same time three different knowledge representation: declarative, procedural and process approaches.

The coexistence of these different approaches to the same architecture is guaranteed by assuming a working space that is arranged in a three dimensions space, where each axis represents one of the previously cited approaches. When the system runs, a point inside the DSS space will identify the state of the system, whereas the projection of this point over each axis, will indicate the contribution of each approach.

As depicted in the figure 3.1, the axes of hybrid architecture are respectively: *Abstraction Layer*, *Decision Making Level* and *Workflow Timeline*.

In the following some characteristics of each axis:
Abstraction Layer Axis (based on Procedural Approach):

- It shows "how to achieve" a specific result for an input problem at different abstraction layer .
- It builds a workflow of operations, dealing with the direct execution of each task and sub-task.
- It runs all the algorithms and services, taking care of the management and organization of issues related to inputs-outputs interface.

Decision Making Level Axis (based on Declarative Approach):

- It decides about "what to do" with the DSS knowledge, according to rule-based engine.
- It works with unstructured data.
- It uses strategies and heuristics in the knowledge base to generate some consistent models, for the problem solving process.
- It manages all decision making steps.

Workflow Timeline Axis (based on Process Approach):

- It allows reconfiguration of each selected tool or service, with back-tracking feature.
- It allows to select alternative paths or restart the workflow from a process selected by the user.
- It collects all the intermediate results, saving the process representation of the problem.
- It traces, step by step, the workflow evolution of the system.

All these axes represent discrete values; in facts, a problem can be represented at the highest abstraction layer, at the lowest abstraction layer or at some intermediate abstraction layers. In the same way, a workflow is done by means of some discrete steps, according to tools executions. As will be explained later, also the Decision Making Axis represents discrete value, because it depicts the successions of each transitions of decision making steps of the system; in other words, for each decision step the system reach a new state.

3.2 Decision Making Activity

The development of reasoning systems is an important area of research in Artificial Intelligence. This work uses a procedural reasoning system that have to operate with BORIS software architecture described in section 4.1. The decision-making capabilities of the system indicated how the system integrates both a directed reasoning according to user request, and the ability to takes account of available resources and knowledge.

As defined by (43), "*Decision making is the study of identifying and choosing alternatives based on the values and preferences of the decision maker. Making a*

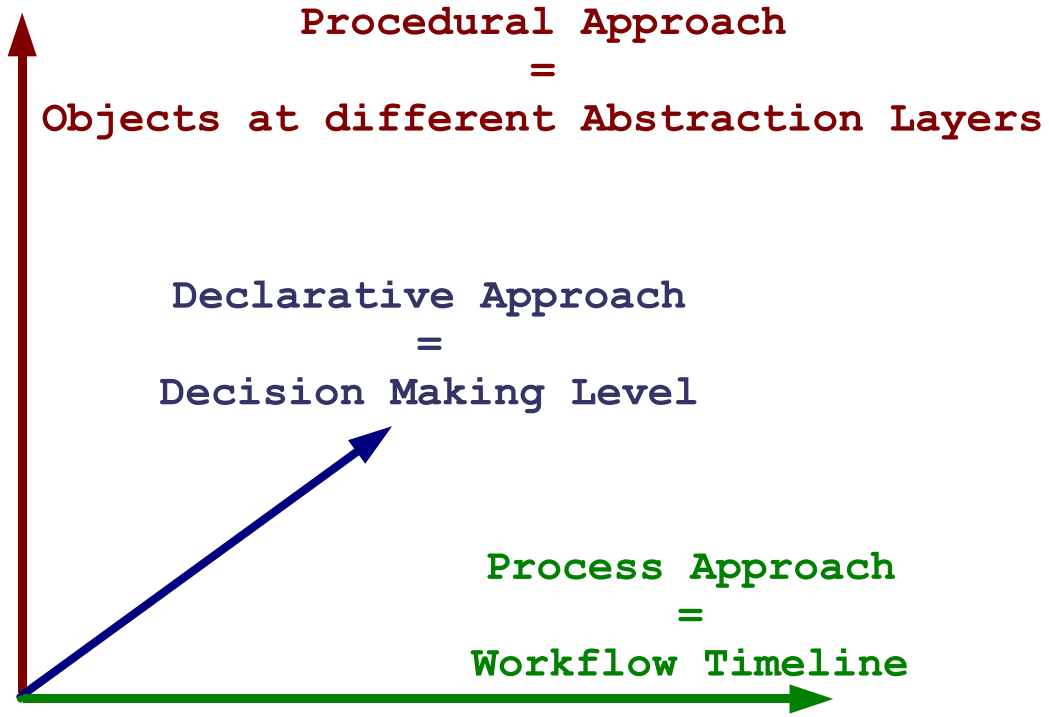


Figure 3.1: Space of Decision Support System. The hybrid architecture introduces tree point of view for the problem, i.e. abstraction layers ((based on Procedural Approach), decision making levels (based on Declarative Approach) and workflow timeline (based on Process Approach).

decision implies that there are alternative choices to be considered, and in such a case we want not only to identify as many of these alternatives as possible but to choose the one that best fits with our goals, objectives, desires, values, and so on.".

According to the guideline suggested by (44), the decision making process used in this work is composed by the following steps, reported in the figure 3.2:

1. **Problem identification**:

   When the system receives the user request, it has to first identify the root causes and then produce a problem statement (also in case of complex decision problems) that describes both the initial conditions and the desired conditions.

2. **Requirement setting**:

   The system has to analyse all the constraints describing the set of the admissible solutions to the problem detected in step 1, i.e. for any possible

solution it has to decide unambiguously whether a strategy is acceptable or not.

3. **Alternatives identification**:

   Alternatives strategies or heuristics offering different approaches for finding a solution have to be evaluated by the system, in order to better match with the user desired goal and the boundary conditions.

4. **Attributes definition**:

   It is necessary to define discriminating criteria to measure how well each alternative achieves the goal or almost a sub-goal. According to (44), criteria should be able to discriminate among the alternatives and to support the comparison of the performance of the alternatives, complete, operational and meaningful.

5. **Decision-making tool selection**:

   Although it could exist several tools for solving a decision problem, the selection of the appropriate tool depends on the concrete decision problem, as well as some characteristics of a tool (requirement of additional resources, computational complexity) or computing power. The selected tool is proposed to user with a list of pros and cons.

6. **Alternative tools evaluation**:

   Since more than a tool can satisfy discriminating criteria, the system must show to the user a set of the most promising alternative tools/services, once again with a list of pros and cons for each tool/service. In complex problems, the proposed alternatives may also call the attention of the user, that could add further goals or requirements to the decision model.

The decision-making activity of the system is organized in functional modules; a representation of these module is depicted in figure 3.3. Each module has its own knowledge and skills, takes care of a specific part of the reasoning process and is responsible for making decisions about a well defined task. Typically, this knowledge is unstructured or semi-structured, because information retrieved by different sources is often ambiguous or incomplete. Also included in each module are strategies and/or heuristics, as well as all the rules that are required by the rule-based engine for developing reasoning on the specific task.

In addiction, there are some modules containing also a subset of rules that are able to launch tools and services responsible for the implementation of a specific methodology. Directives contained in these rules are able to suggest to the user the most suitable tool, among a collection of similar tools.

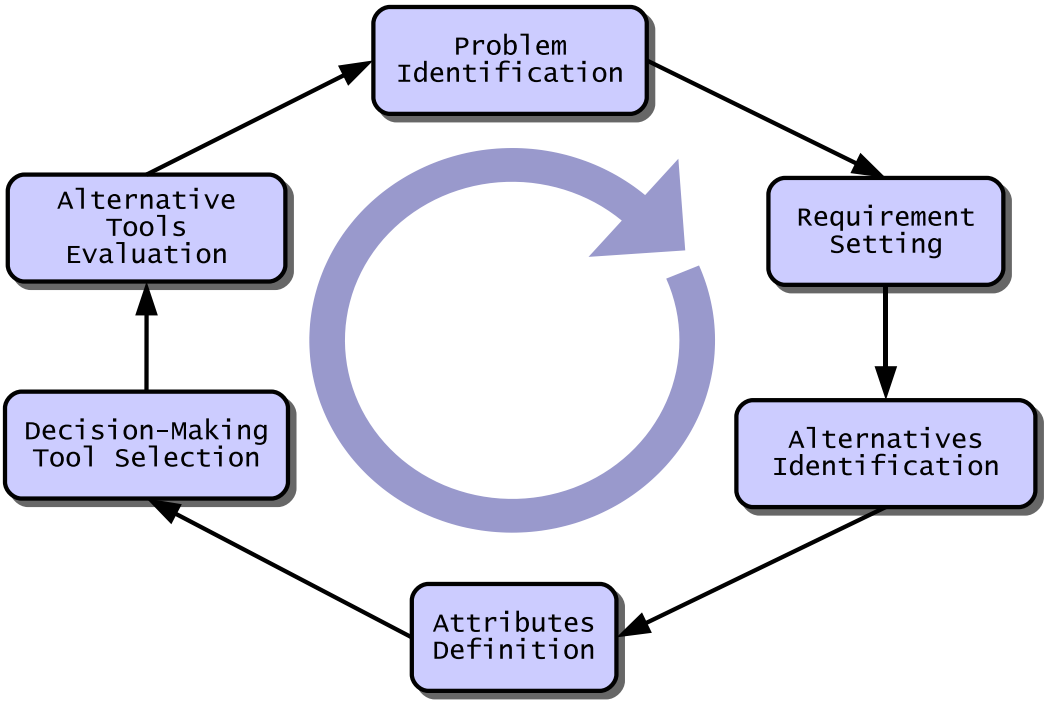


Figure 3.2: Decision Making Activity. The closed loop runs during all the decision making activity, in order to solve tasks and sub-tasks.

3.2.1 Meta Reasoning Tree

Since several problems are very wide, the management of these problems could be hard and, consequently, some large decision modules are needed. For this reason, it is convenient to split problems into sub-problems, building a hierarchy of modules and sub-modules, containing tasks that are able to model only simple issues. The data structure used to link tasks (modules) is the hierarchal tree. The tree allows to represent relationship among problem and sub-problem in a suitable way, with respect to the logical organization adopted in decision making process. In particular, the depth of a node with respect to the root node in the tree is arranged according to the meta levels adopted by the system during the reasoning activity.

By means of the decision making axis of the DSS tree-dimensional space, the user can navigate through the hierarchy of the entire reasoning tree for exploring sub-modules in different meta-levels; this way, user can see in a glass-box the rules behind the reasoning of the system. User can also interact with the system in order to learn about strategies and heuristics leading the decision making activity in figure 3.2. As a meta-meta-level can control a meta-level in a process of reasoning about reasoning itself, in the same way a module can operate a reasoning

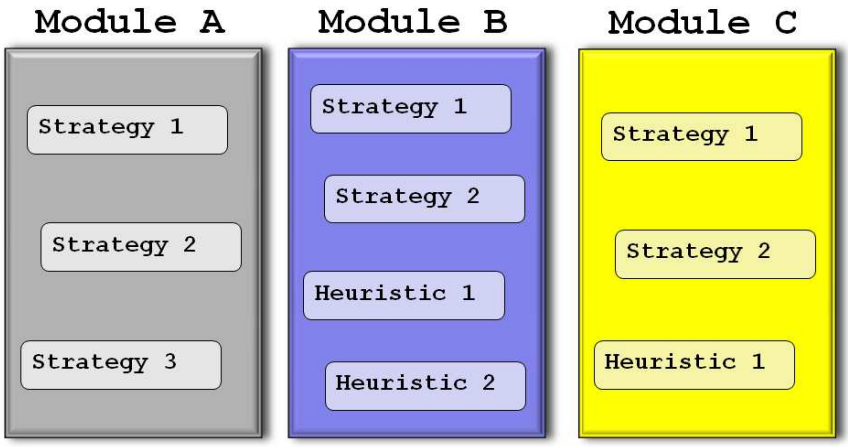


Figure 3.3: Decision-Making Modules. Each module contains all the strategies and/or heuristics for a well defined task. In addiction some modules are also responsible for the management of directives related to tools and services.

over a child module (that lie in a deeper position on the tree), demanding some operation to him.

A representation of the decision making tree is reported in figure 3.4; it contains three meta-reasoning levels, arranged according to the previous cited idea.

Of course, each child is able to solve a specific task that its parent can only propose to solve, without having the knowledge about it. Communication between decision modules is managed from parent to child, in facts the parent module A can give focus to child module A.1 in order to request the solution about a specific sub-problem and, in turn, the module A.1 can give focus to its child A.1.1 to solve a sub-sub-problem.

All the modules lying at lowest meta-reasoning level (i.e. modules at *Meta-Level X.Y.Z* in the figure 3.4) contains rules that are responsible for management of tools and services, because they are "nearest" to the execution layer of workflow process; this way the system can suggest what are the most suitable algorithms, assisting the user in their proper configuration.

Of course, also the other modules could contain some directives for tool/service execution; for example, it can happen when the system request an input data analysis, that is necessary to make a decision at highest MRL.

3.2.2 From Meta Reasoning Tree to DSS Space

This subsection aims to project the representation of meta-reasoning levels from the hierarchical tree to a new dynamic treemap, reported in the following. The in-

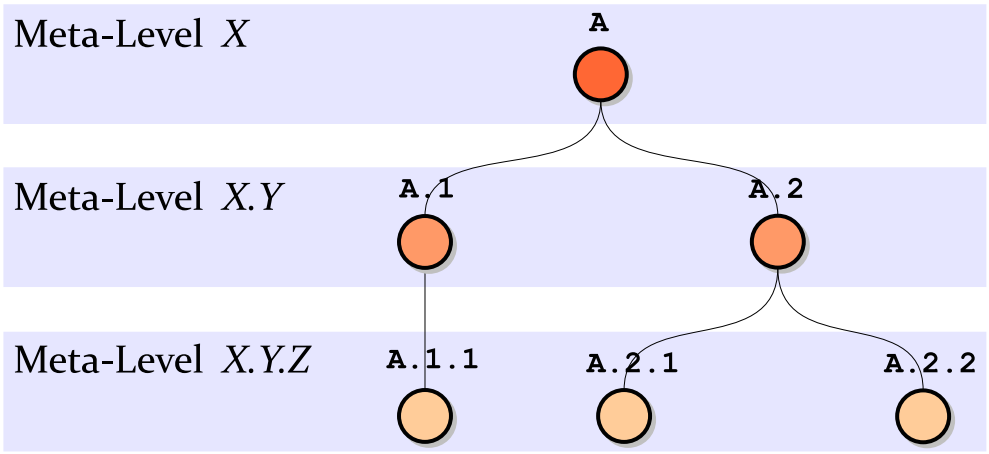


Figure 3.4: Meta reasoning tree. Decision making modules are distributed into some different meta-reasoning levels, according to problem/sub-problem hierarchy.

troduced representation allows to integrate all the MRLs into the tree-dimensional space of hybrid architecture proposed in this work.

3.2.2.1 Dynamic Treemap Representation

Treemap was first designed by Shneiderman (46) during the 1990s, in order to producing a compact visualization of directory tree structures in hard disks. It is a two-dimensional space-filling approach to the visualization of hierarchical information structures, obtained by means of a set of boxes representing nodes of tree: individual nodes within their bounding boxes determines the content information statically presented in a treemap. It is very effective in showing attributes of leaf nodes using size and color coding, providing an overall view of the entire hierarchy and making the navigation of large hierarchies much easier. In general, treemap enables users to compare nodes and sub-trees even at varying depth in the tree, and help them to detect mutually related properties among nodes.

Treemap is able to depict both the organization of information associated with the hierarchy, and the content information associated with each box.

Use of treemap representation fixes some disadvantages related to the previous used representation; the main disadvantages of using the hierarchical tree representation is the lack of content information. In facts, each node has only a simple text label. Additional information, such as the duration of a decision making module with respect to the time line of a workflow, can not be depicted into the decision making tree. In the same manner, no information about which abstraction layers are used when a module is running can be shown using the

hierarchical tree representation.

The treemap visualization technique adopted in this work makes use of the system 3D-space, in order to map the full hierarchy onto a rectangular region in a space-filling manner. For the proposed hybrid architecture, a 3D treemap for MRL browsing that can show overlapping between modules has been introduced. In addiction some functionalities related to time line execution have been integrated. In facts, a sort of `Dynamic Treemap` has been introduced, where each box representing a module has a width related to its duration inside the execution of working process and an height related to the number of different abstraction layers it take in account during the task processing. The representation of decision making modules inside the Dynamic Treemap, follows the workflow generation step-by-step and it is time-dependent (from which the term "dynamic").

An example of the dynamic treemap representation is shown in the bottom of the figure 3.5. The meaning of this figure is described in the following.

3.2.2.2 Communication among Decision Making Modules

Decision modules are represented into the introduced 3D space by means of the previously cited dynamic treemap. This solution integrates all the information about the interaction among modules as well as the relationship among meta-reasoning level. In addiction, this representation assures an appropriate user interaction, providing all the features available for the exploration of the hybrid architecture space.

Figure 3.5 shows an example of the communication among modules during the decision-making process, through different meta-levels. The top of the figure reports a tree where each node represents a module, where parent-child relations are oriented from the highest MRL to the lowest MRL. The root of the tree is the reasoner having the main directives for the resolution of a selected problem.

Each module can have n children: therefore each module in meta-reasoning level A can make a decision according to its own proper knowledge about the problem and, moreover, it can assign a task to another child module at lower meta-level reasoner, that has further and more specific information about the task solving the sub-problem: the "give focus" line between modules is highlighted in the figure with an oriented arrow. Accoring to the reasoning process, all the arrows pointing to modules lying at lower MRL (parent to child) correspond to assignment of a sub-task, whereas all the arrows pointing to modules lying at higher MRL (child to parent) represent a return of focus that confirm the child module has solved the sub-task. The number near to the arrow represents the order of focus transactions: the entire example in the figure is composed by four sequential steps. During this process, each parent module stays awake until all of its children are running, because it has to supervise and process the stack of

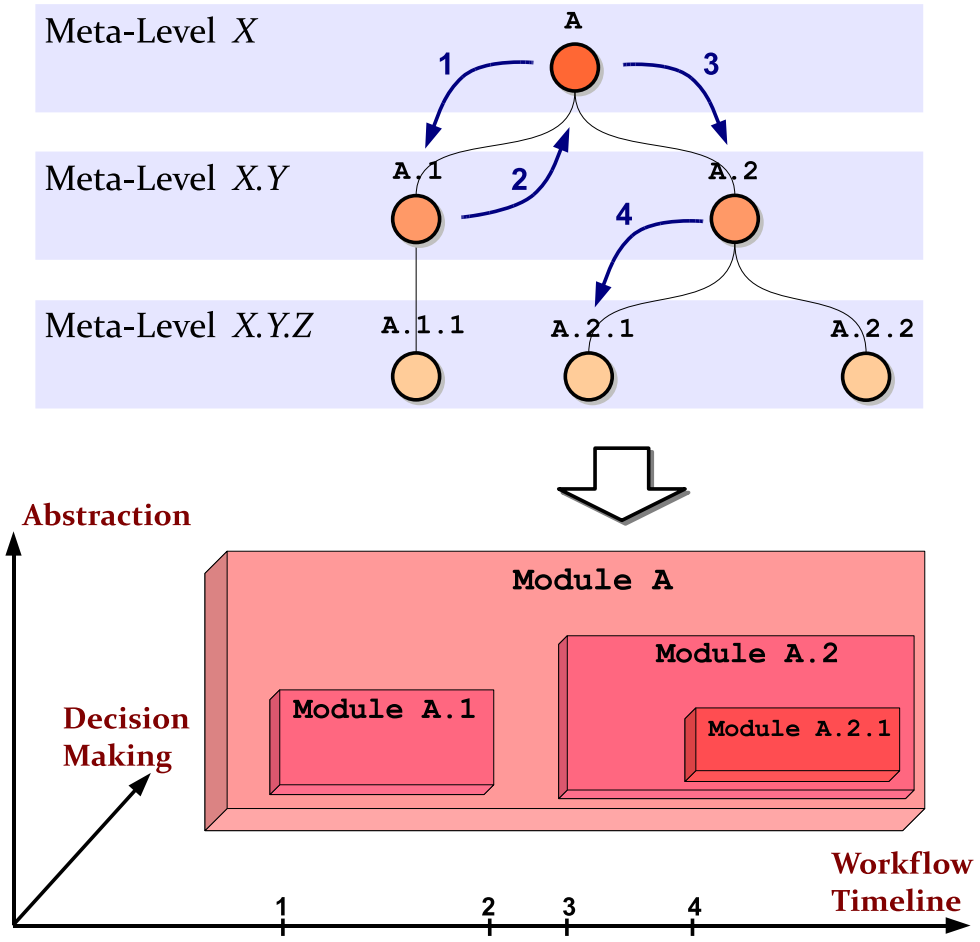


Figure 3.5: From meta reasoning tree to hybrid architecture space. Decision making modules are represented by means of a dynamic treemap.

results.

The bottom of the figure 3.5 reports the dynamic treemap representation of the reasoning process, created inside the 3D space of DSS. This figure join the treemap representation with the workflow timeline: in this manner the user can take into account, at every moment of the workflow evolution, the active decision making modules. The dynamic tree is built step-by-step from the right to the left (according to workflow timeline axis orientation); boxes representing modules used during the experiment appear when these are active and they are bounded when the module give focus back to the parent. A parent will grow up under all its children boxes, because it will manage their results; for example, the module `A`in the figure, will wait for the conclusion of the task solved by the module `A.2.1`. Bounding boxes representing modules at different meta-reasoning levels

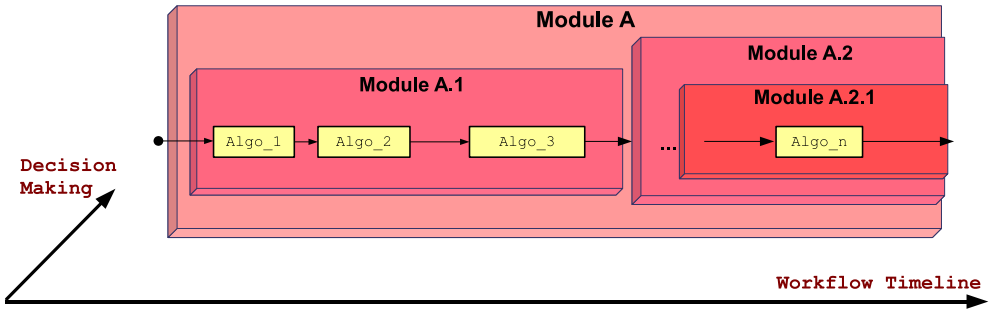


Figure 3.6: Representation of a simple sequence of algorithms. The workflow is projected into the 3D space; no information about abstraction layer is reported, because only the object layer of the system is considered.

are overlapped according to the decision making axis, that takes into account the depth of meta levels, from highest MRL to the lowest MRL. The projection of the dynamic tree over the abstraction layer axis will be discussed in the next subsection. In order to solve a specific request, the module `A` at meta-reasoning level `X` is enabled: At step 1 it call module `A.1` to solve a sub-task. At step 2 the module `A.1` has completed its reasoning and give focus back to the parent module. At step 3 the module `A` call the module `A.2` to solve another sub-task. At step 4 the module `A.2` have not enough knowledge about the sub-task and send a sub-request to the module `A.2.1` at MRL `X.Y.Z` to resolve a sub-sub-task.

3.3 Workflow Generation

Workflow generation starts from the results of the decision making process produced by the rule-based engine, where main goal, sub-tasks, business processes and internal/external tools are specified. They are responsible both to define all the aspects of a process that are relevant to controlling and coordinating the execution of the tasks have been executed and to provide all the information needed for design and implement the final process.

In general, the obtained workflow is a collection of tasks organized to accomplish some business process. A task is performed by one or more softwares (e.g. preprocessing tools), or by means the human interaction (e.g., providing input commands), or a combination of these. In addition the workflow defines the order of task invocation, task synchronization, and information flow (dataflow).

In figure 3.6 a simple workflow inside the system space is shown. For the sake of simplicity, the abstraction layer axis is not depicted in this figure: only the object layer is reported. The proposed hybrid architecture supports the evolution, replacement, and addition of workflow applications, as well as the re-engineering

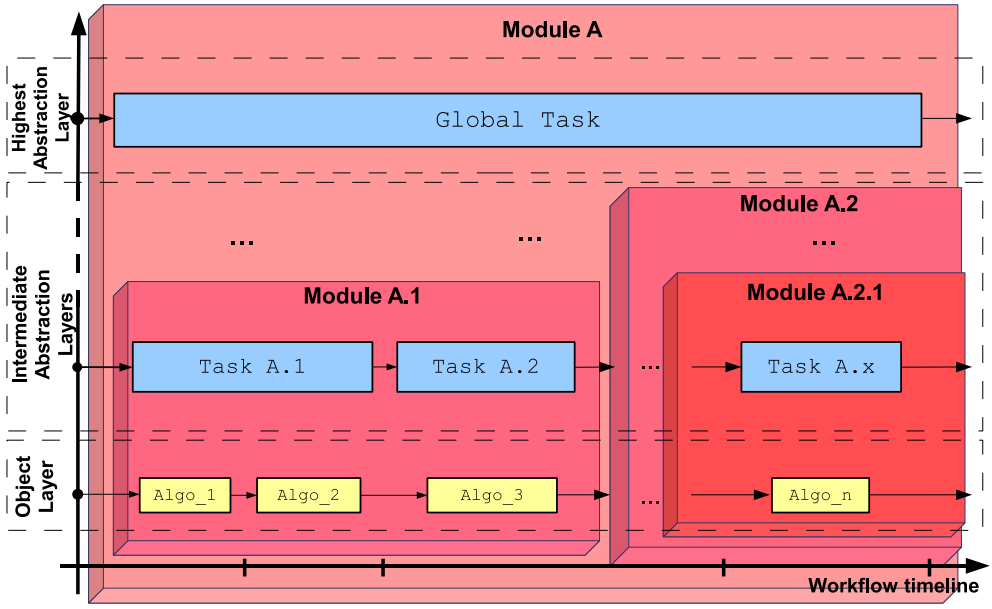


Figure 3.7: An overview of the DSS 3D space. There is a workflow for each abstraction layer, according to the user's point of view.

of system components and processes; in facts, users can interact with the system modifying the sequence of tools, changing algorithms and/or parameters and exploring decision making modules responsible for suggestion of strategies.

The figure also shows that, in order to resolve a required task, more than an algorithm could be managed by the same decision module; for example, tree algorithms (here called `Algo_1`, `Algo_2` and `Algo_3`) have been executed under the supervision of the module `A.1`, before this one can resolve its sub-task and, then, give focus back to parent module `A`.

3.4 Abstraction Layer

The proposed DSS faces each user query at different abstraction layers, according to its complexity. In facts, it shows several views of a problem: from the top abstraction layer (i.e. the problem itself) to the bottom abstraction layer, the object layer (i.e. the workflow of tool/service instances).

Figure 3.7 shows how meta-reasoning levels, abstraction layers and workflow timeline interacts each other during the building of a generic workflow. Reasoning starts with the reasoning of module `A` at highest abstraction layer that manages the different tasks needed to fulfill the users request, identified as the "Global Task". The set of tasks is arranged according to the hierarchy of problems and sub-problems of minor complexity, and at the lowest abstraction layer

there are the specific algorithms and/or services that will be run in order to solve a general complex problem. At each intermediate abstraction layer, it is possible to see the same problem faced at the higher abstraction layer split in operational tasks, that have been detected by the reasoning process as candidate for solving a sub-problem; in other words, decision making modules suggest some strategies/heuristics for problem solving, proposing a sequence of tasks that are visualized at one or more abstraction layers. At the lowest abstraction layer, the system shows all the suggested algorithms and services to run, assisting the user in their proper configuration.

The module `A` at the highest MRL is the main module, responsible for the supervision of the entire process. Following the time axis, it gives the focus to meta level `A.1`, which proposes, through its facts and rules, to launch Task `A.1` and Task `A.2` done by means of `Algo_1` and `Algo_2` (for Task `A.1`), and `Algo_3` (for Task `A.2`). After that, the focus goes back to module `A` that pass it to module `A.2` and so on. This type of multi-layer workflow representation is the actual output of our system.

4

BORIS Software Architecture

In this Chapter, the software architecture of the Rule-Based expert system developed for the BORIS global framework is presented.

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.

4.1 Three-layer Architecture

Boris software architecture has been developed as a three layers structure. The layered architecture of the proposed system, shown in Fig. 4.1, 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, 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 5.

The Controller Layer includes a knowledge-based expert system (15). 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

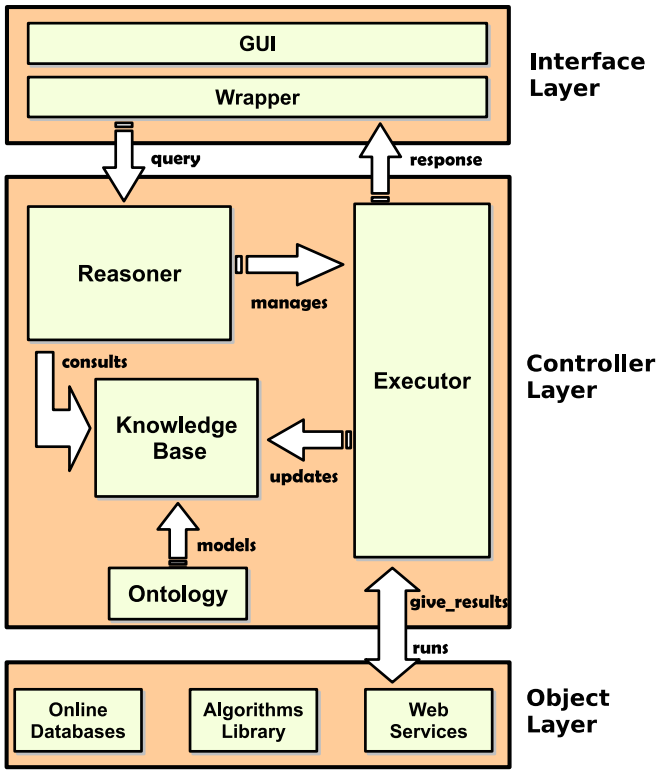


Figure 4.1: Software Architecture. Three main layers interact each other to make the system work. The Interface layer is responsible for the interaction between the User and the system. It implements a GUI that manages the input/output operation. The Controller layer holds a Knowledge-Based expert system: 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.

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

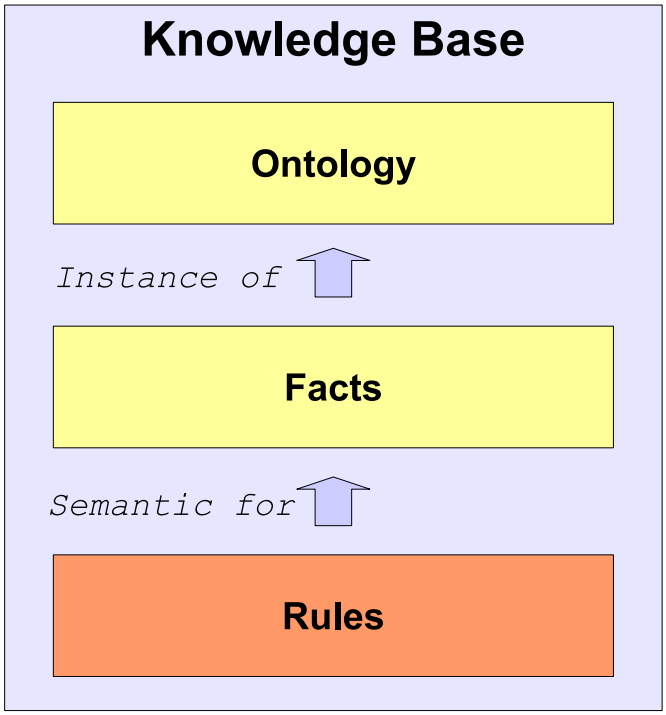


Figure 4.2: Knowledge-Base three main components: Facts are the instances of the concepts defined into the Ontology; the Rules work on facts in order to give semantic and the possibility to make inferences over the facts.

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. 4.2

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.

4.2 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
- 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. 4.3 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,

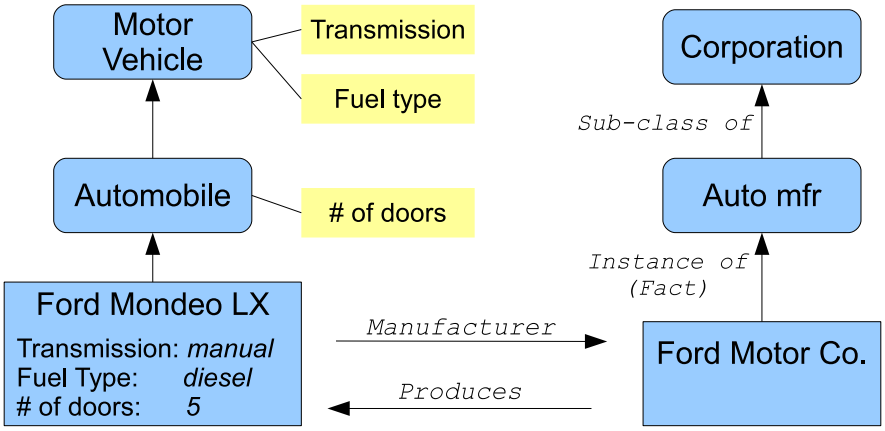


Figure 4.3: An example of ontology in the vehicle domain. The rounded rectangles are the classes characterized by properties or attributes (the yellow boxes); the other rectangles are instances of the classes: in the instances each attribute has a value. Finally there are the relationships between the instances, indicated through the black arrows. Each relationship is given a label representing the type of bind.

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. 4.4, 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

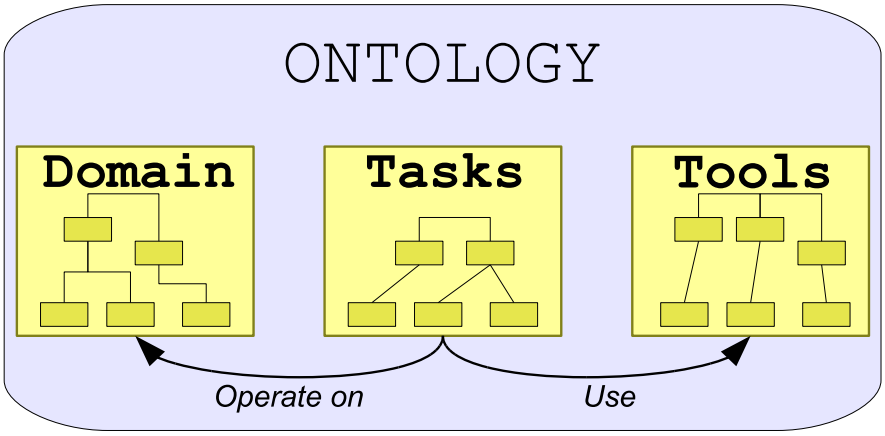


Figure 4.4: Three main ontology sub-domains: Tasks model the set of operations it is possible to do on the bioinformatics domain; Tools model the set of algorithms and services that implements the Task instances; Domain models the biological data to analyse. The generic relationships among the three subdomains are shown: Tasks "operate on" Domain's instances and "uses" Tools' instances.

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. 4.5

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 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 (30), 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 (31). 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

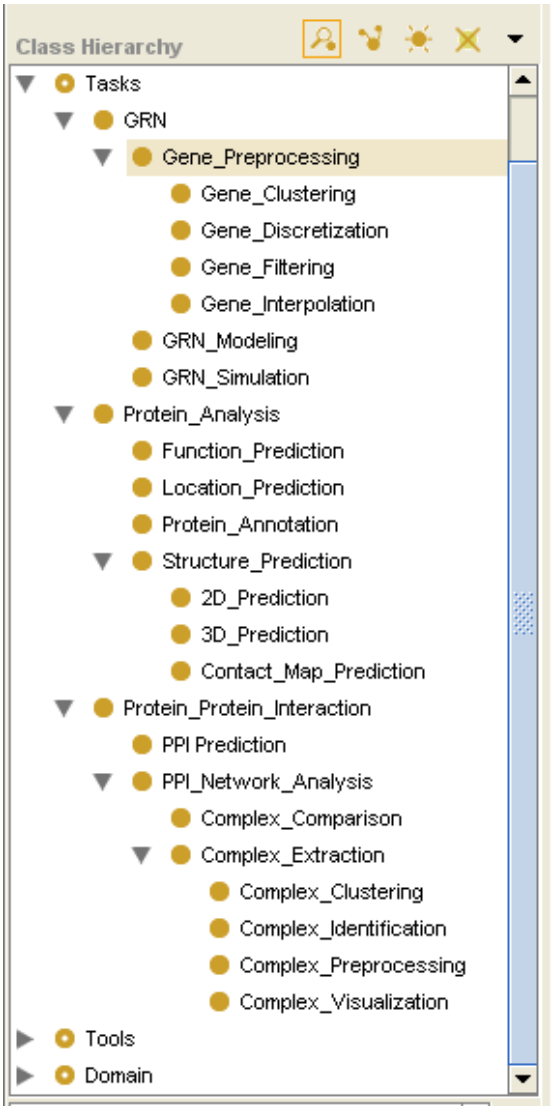


Figure 4.5: Hierarchy of classes and subclasses for the Task part of the proposed ontology.

(36)), aiming at improving prediction techniques in the years, has been instituted;

- Function Prediction: another challenge is to determine protein function at 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 (34): 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 (35), 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 (33), 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 (32), 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 (48, 49). 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 (50);
- 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 (51);
- Gene filtering is a set of procedures that allow to select a subset of input genes according to some user defined constraints (52, 72).

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)

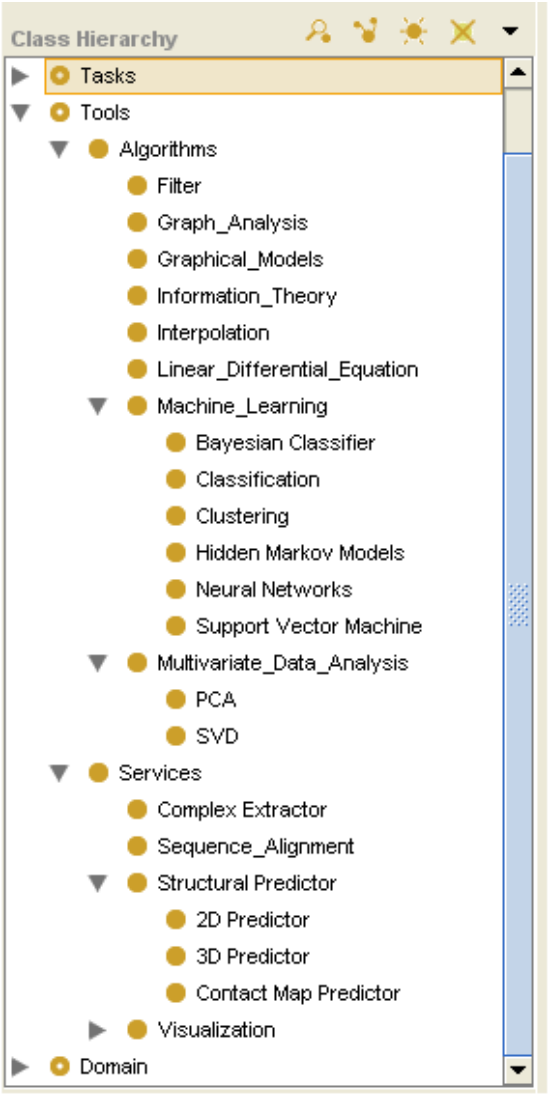


Figure 4.6: Hierarchy of classes and subclasses for the Tools part of the proposed ontology.

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. 4.6.

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;

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. 4.7). Here we focused especially on the modelling of microarray data, since this is at the basis of the developed scenario presented in Chapter 6

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 6.2.1)
- *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.

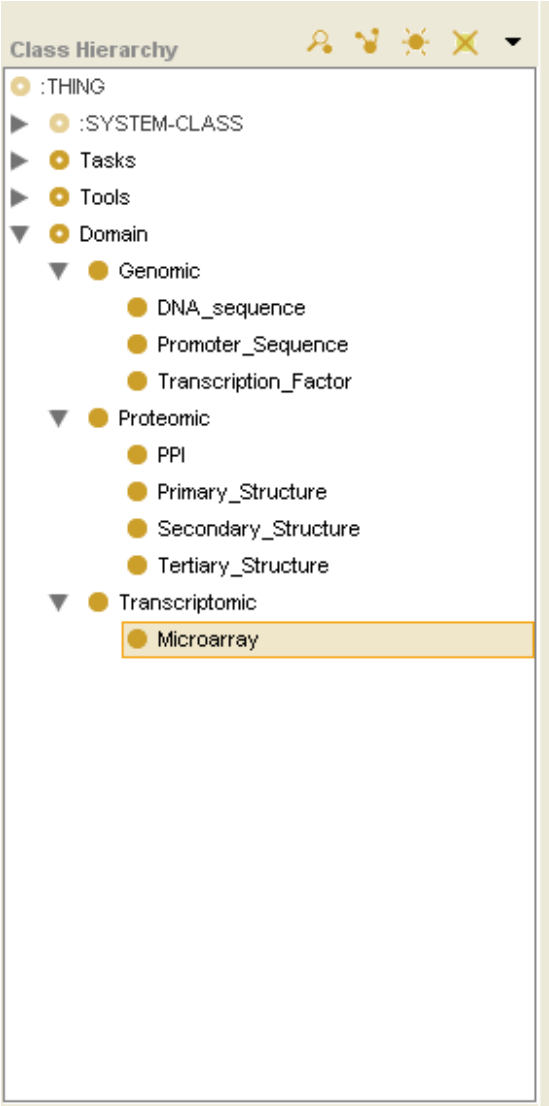


Figure 4.7: Hierarchy of classes and subclasses for the Domain part of the proposed ontology.

4.2.1 Mapping between the ontology and 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*, as seen in Section 3.2.

From this point of view, 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.

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.

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 3.1. With regard to the 3-axes reference space of Fig. 3.1, 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. 3.7.
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.

5

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.

5.1 Boris' Graphical User Interface

In Figure 5.1 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

5.1.1 Profile Panel

The Profile Panel, standing in the top part of Fig. 5.1, 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;

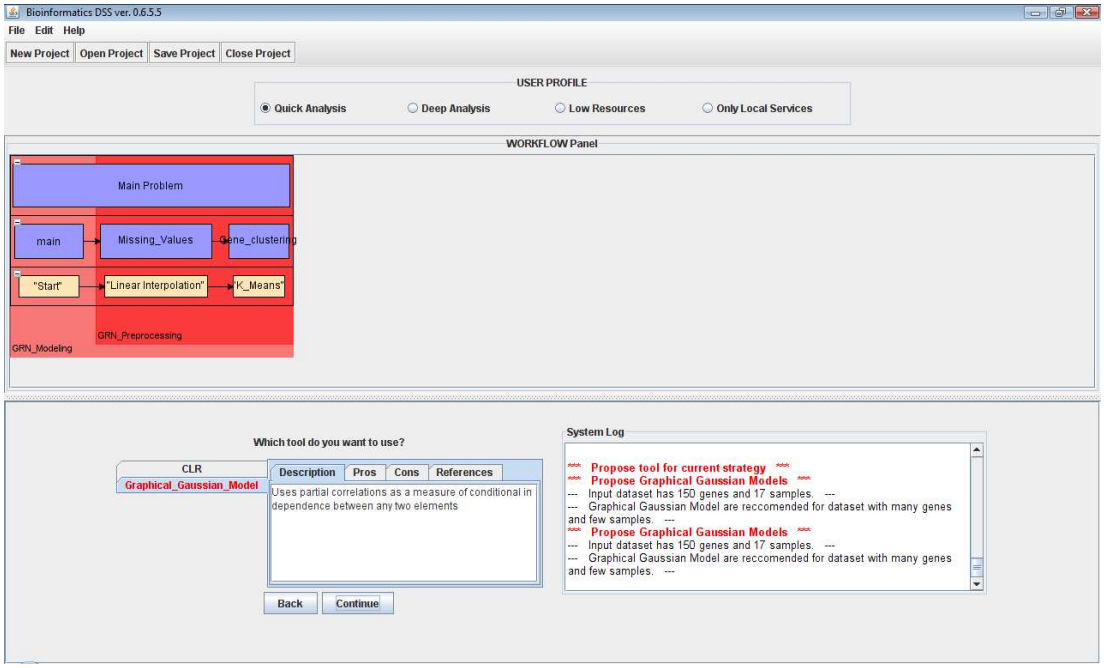


Figure 5.1: A caption of the Graphical User Interface (GUI) of BORIS system during a typical experimental session.

- 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.

5.1.2 Workflow Panel

Workflow Panel, that we can see in Fig. 5.2, shows the building of the workflow. 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 rep-

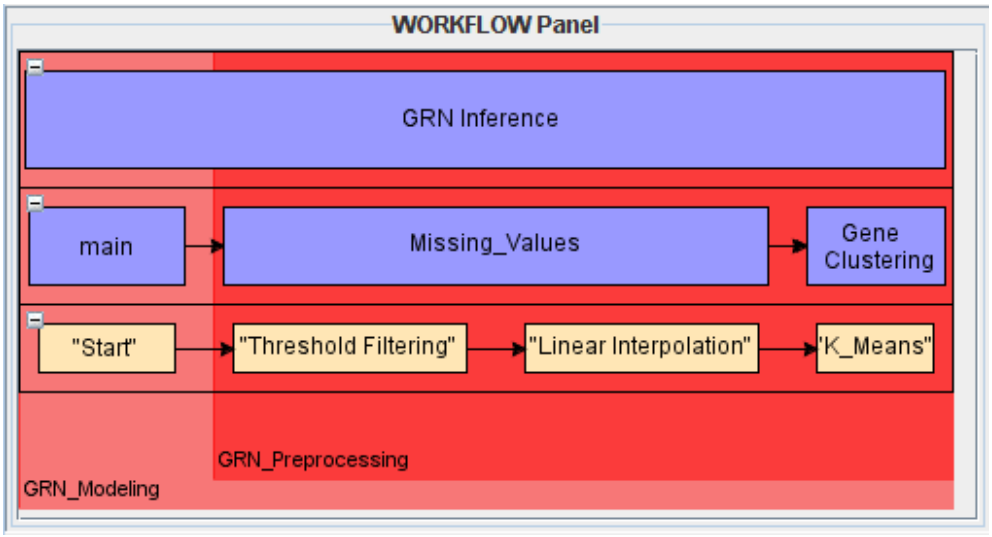


Figure 5.2: Workflow Panel component of BORIS system. It shows the active decision-making modules (pink boxes on the background), the adopted heuristics and strategies (blue rectangles), and the run algorithms and services (yellow rectangles).

resents 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.

5.1.3 Strategy Panel

Strategy Panel, shown in detail in Fig. 5.3, 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.

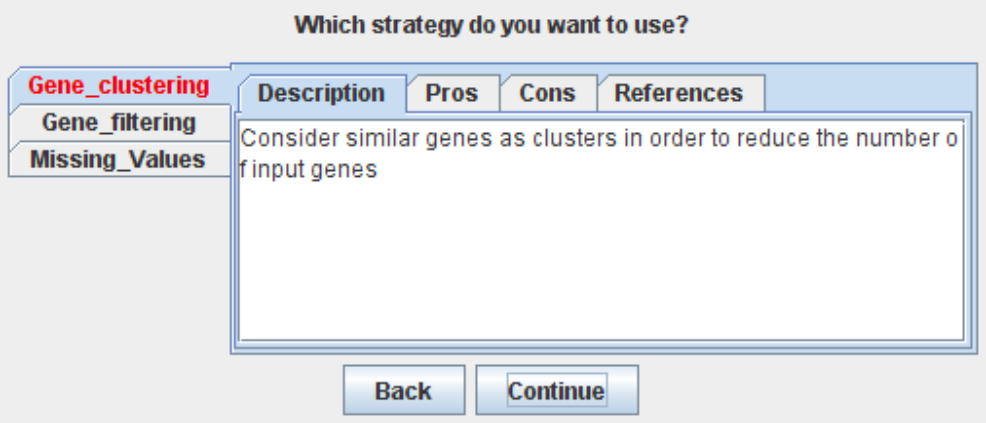


Figure 5.3: Strategy Panel showing the available strategies, heuristics and tools for the given task. The red element is the suggested one.

5.1.4 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.

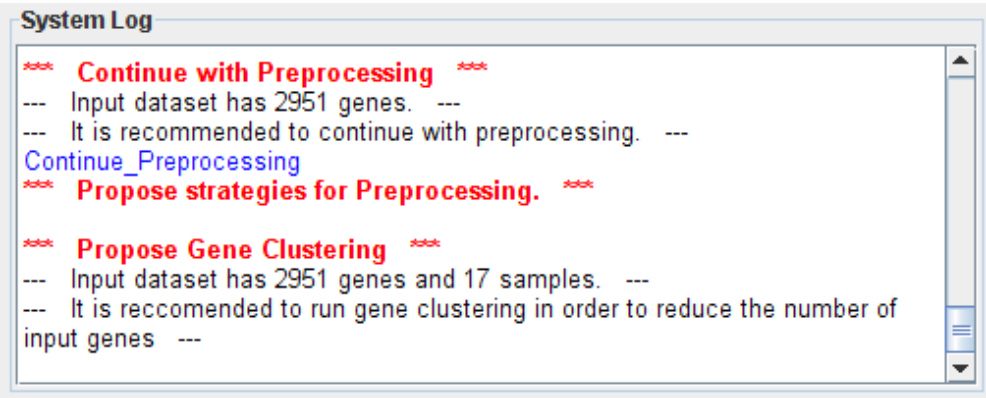


Figure 5.4: System Log of Boris system. Activated rules and their motivation, intermediate and final results, available forks in the workflow and progression of the experiment are shown.

6

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 4)

6.1 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. 6.1 it is shown a schematic representation of gene regulation.

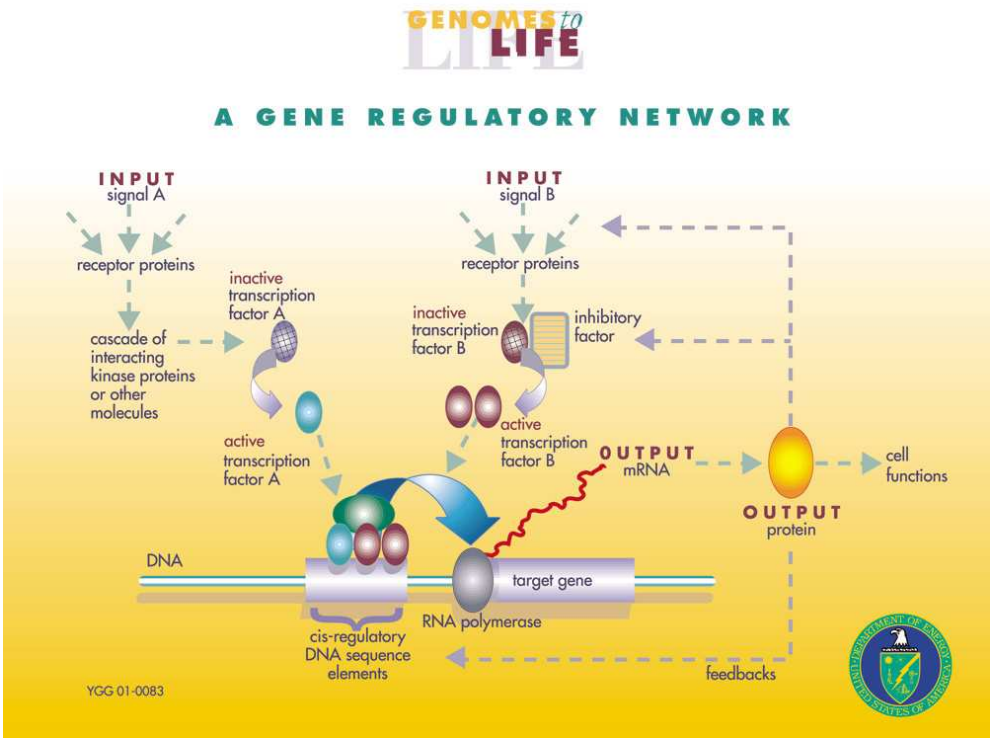


Figure 6.1: A synthetic representation of gene regulation biological phenomenon. This picture is taken from U.S. Department of Energy Genome Programs, http://genomics.energy.gov

6.2 Bioinformatics Approach

Gene regulation is a very complex biological phenomenon and it is not full understood yet (54). 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 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. 6.2.

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.

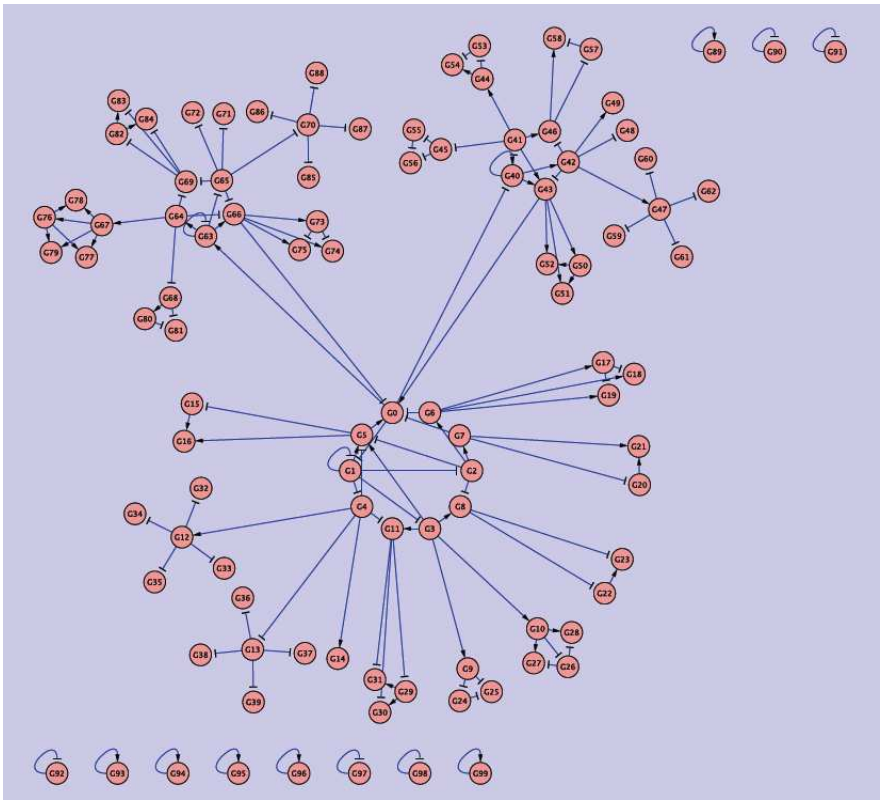


Figure 6.2: Sample gene regulatory network (directed graph)

6.2.1 Microarray Technology

Gene Expression is measured by means of microarray technology (55). Microarray chips are devices that enable the scientist to simultaneously measure the transcription level of every gene within a cell. Microarrays are commercially available 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) (56) 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

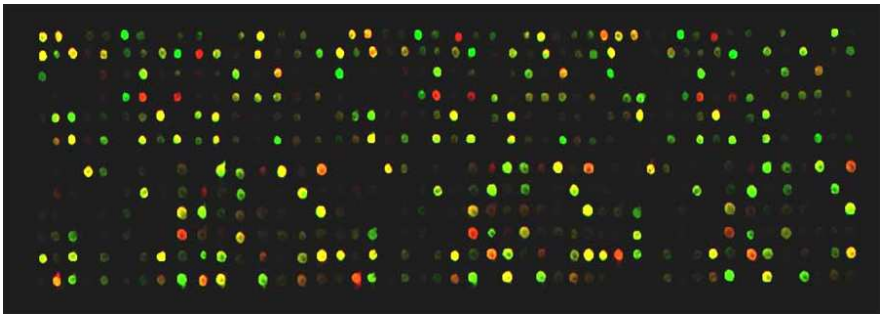


Figure 6.3: Example of microarray image

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 (57) 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. 6.3, 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.

6.3 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 (58, 59, 60). 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 (61, 62), Factor Graph (63), Boolean Networks (64), correlation methods (65), Ordinary Differential Equations (ODE) (66, 67).

To be more precise, at the moment Bayesian Network (68), Graphical Gaussian Models (69), and correlation methods using ARACNE (65) and Context Likelihood of Relatedness (CLR) (70) algorithms, are supported. Moreover, Boris can also offer support for preprocessing of input data, using algorithms for Gene Clustering (48, 49), Gene Filtering (52) and Gene Interpolation (50); and for visualization of networks, using Graphviz software (78) and Cytoscape (79).

6.3.1 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:

$$\mathrm{MI}_{ij} = H_i + H_j - H_{ij} \tag{6.1}$$

where H represents the entropy and it is defined as:

$$H_i = -\sum_{k=1}^{n} p(x_k) \log(p(x_k)) \tag{6.2}$$

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.

6.3.1.1 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 (80) for the estimation of the MI (6.1) 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 (81), 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.

6.3.1.2 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 (82), 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.

6.3.1.3 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:

$$\rho_{ij} = \frac{C^{-1}_{ij}}{\sqrt{C^{-1}_{ii} C^{-1}_{jj}}} \tag{6.3}$$

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. (6.3). 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 (71) 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.

6.3.2 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:

$$P(X_i, \ldots, X_n) = \prod_{i=1}^{n} P(X_i = x_i \,\|X_j = x_j, \ldots, X_{j+p} = x_{j+p}) \tag{6.4}$$

The $p + 1$ genes, on which the probability is conditioned, are the parents of gene i in the graph and represent its regulators. Equation (6.4) is obtained using the Bayes Theorem:

$$P(A, B) = P(B \,\|A) * P(A) = P(A \,\|B) * P(B) \tag{6.5}$$

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

$$P(B \,\|A) = \frac{P(A \,\|B) * P(B)}{P(A)}. \tag{6.6}$$

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 (6.6) 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 6.3.1), in order to restrict the space of possible models and to speed up the entire computation. Moreover, as stated in (60), 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 (68) have been introduced.

6.3.2.1 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.

6.4 Experimental Dataset

The dataset used in this system demonstration is extracted from the genome of Saccharomyces cerevisiae (yeast) (83) 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.

6.5 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 3.1.1.

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. 6.4). This list of supported bionformatics problems is obtained through the Task ontology presented in Section 4.2. 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 6.4 in csv format. Selected User Profile is *Quick Analysis*.
After selecting the problem and inserting input dataset, decision-making activity starts. In Fig. 6.5 we can see what are the decision-making modules responsible for the current experiment:

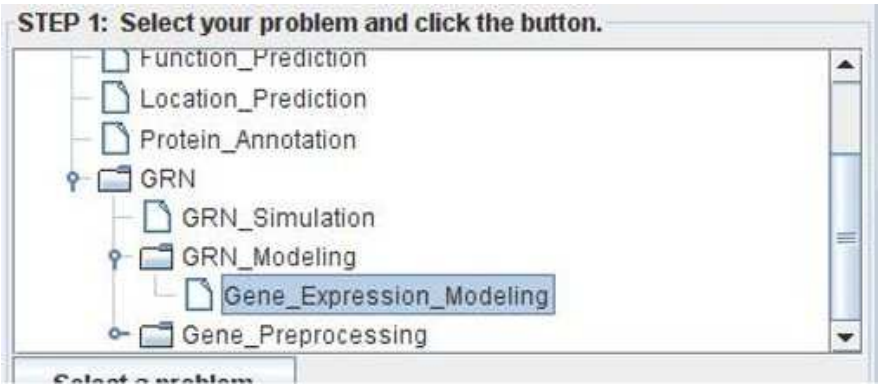


Figure 6.4: Available bioinformatics problems supported by BORIS system.

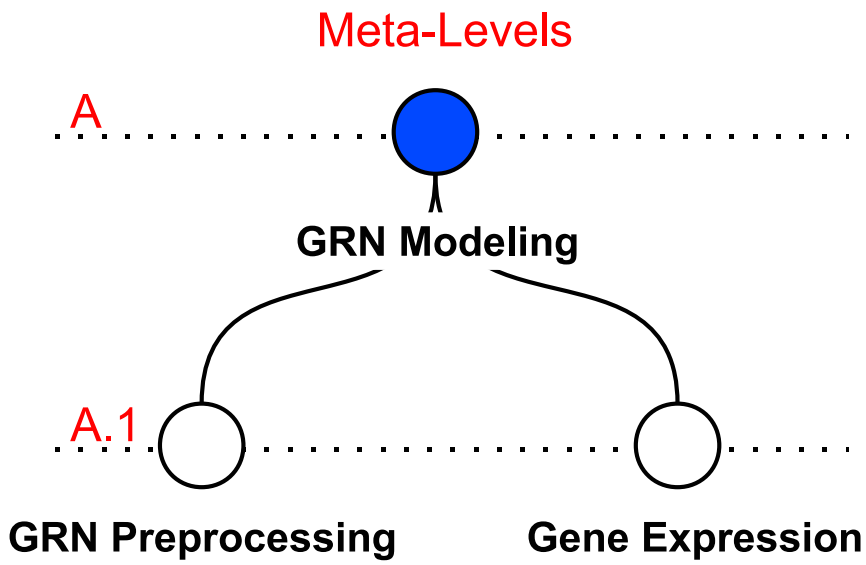


Figure 6.5: Decision-Making modules responsible for the reasoning activity related to the reverse engineering GRN scenario.

- *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. 6.6 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.

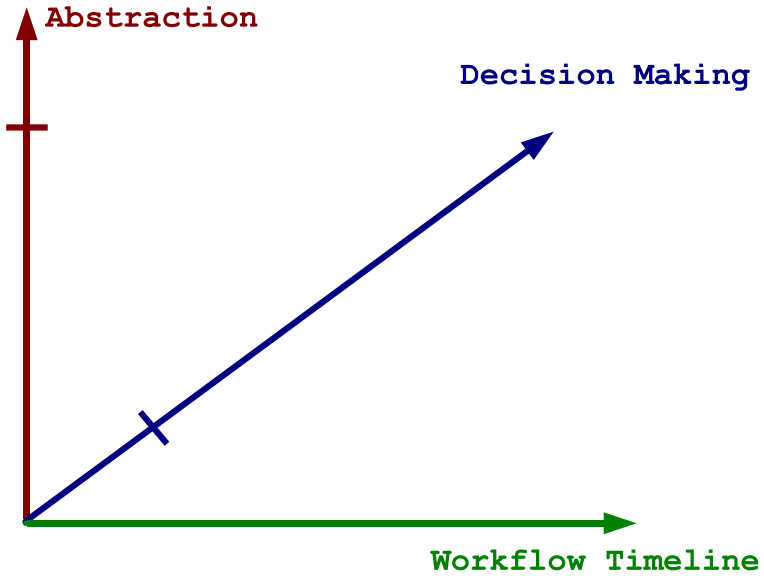


Figure 6.6: The initial state of the system with regard to the 3-dimensional system reference space defined in Section 3.1.1

According to the attributes of Microarray template described in Section 4.2, 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 6.4, 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. 6.7).

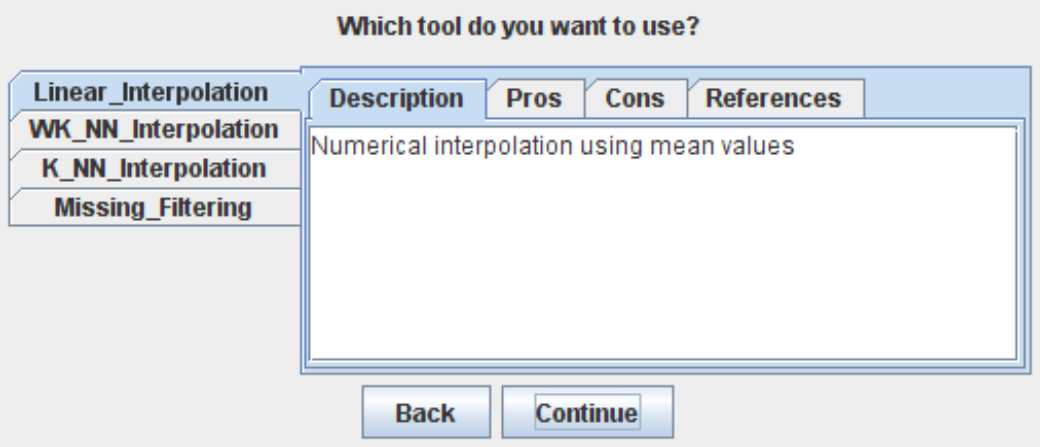


Figure 6.7: Available techniques for dealing with missing values.

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 (85);
- K_NN interpolation: missing values are interpolated using K-Nearest Neighbour algorithm (84);
- Linear interpolation: missing values are interpolated by means of linear interpolation (50).

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. 6.8), 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. 6.9).

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

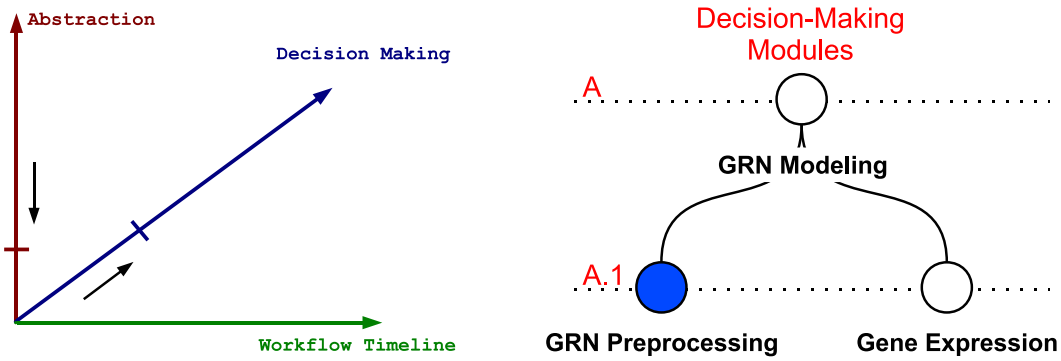


Figure 6.8: State of the system in the 3-axes reference space (left) and corresponding activation status of decision-making modules (right) during preprocessing operations.

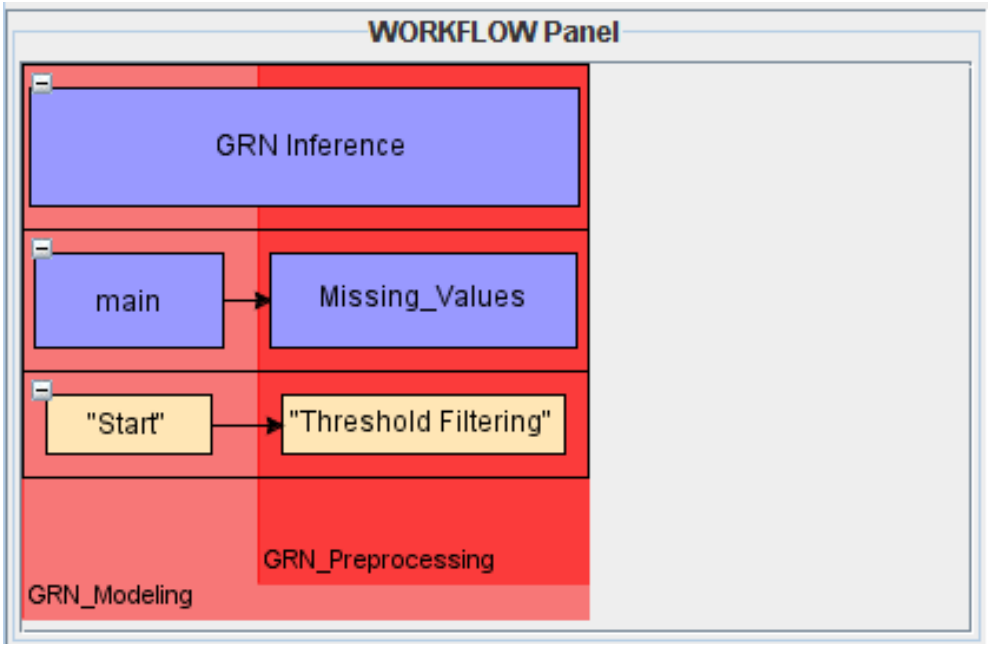


Figure 6.9: Workflow of the current experiment after the first algorithm (Threshold filtering) has been run. It is possible to notice the decision-making modules on the background, the strategy name at middle abstraction layer and the main goal at the top abstraction layer.

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. 6.10).

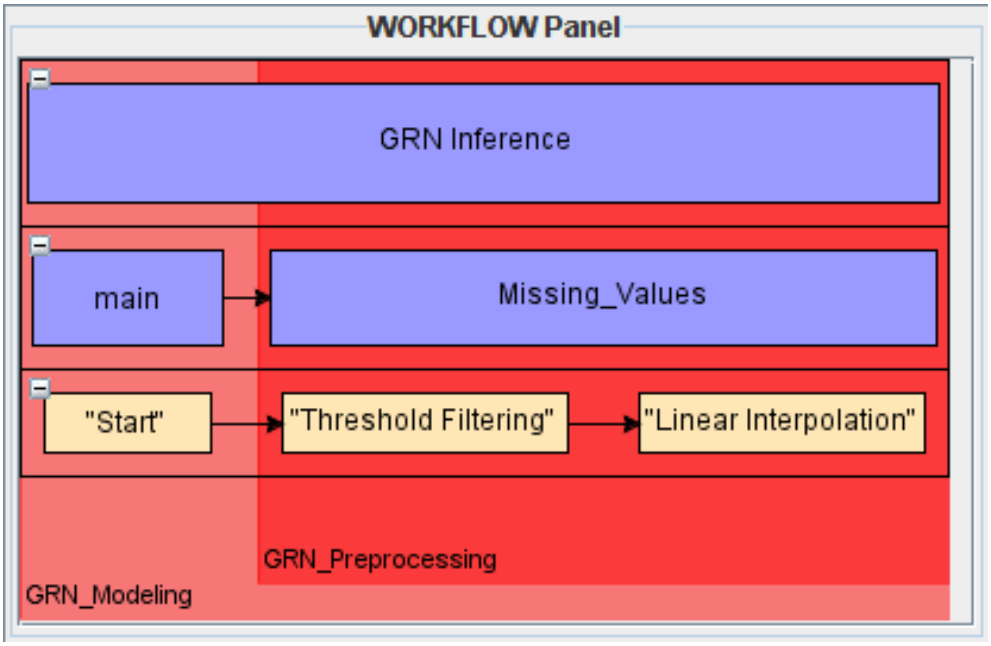


Figure 6.10: Workflow of the current experiment after Linear interpolation.

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 (72);
- Robust Genecycle: an enhanced version of simple Genecycle algorithms, characterized by more accurate results but higher computational time (73, 74).

The supported clustering algorithms are (Fig. 6.11):

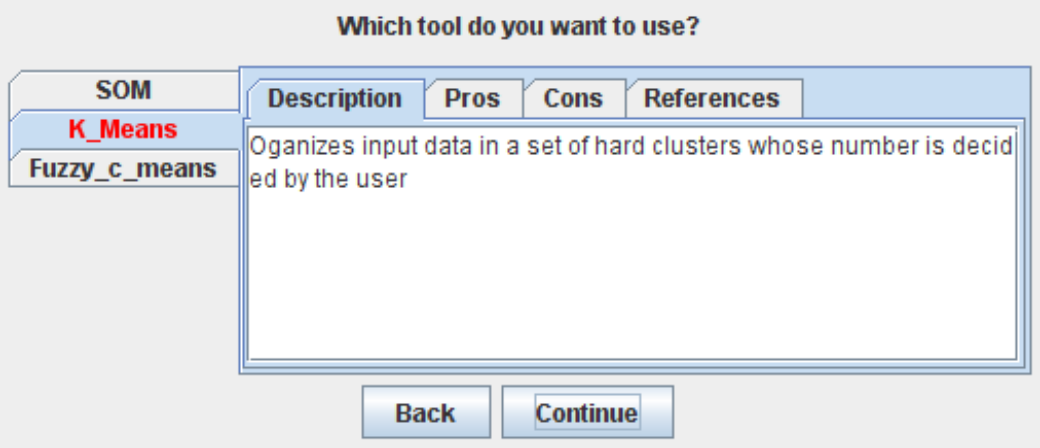


Figure 6.11: Supported clustering tools. K-means, in red, is the suggested one.

- K-Means: one of the simplest clustering algorithm, it puts together input elements into clusters, maximizing *intra* cluster similarity and *inter* clusters diversity (75);
- 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 (76);
- 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 (77).

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. 6.12).

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

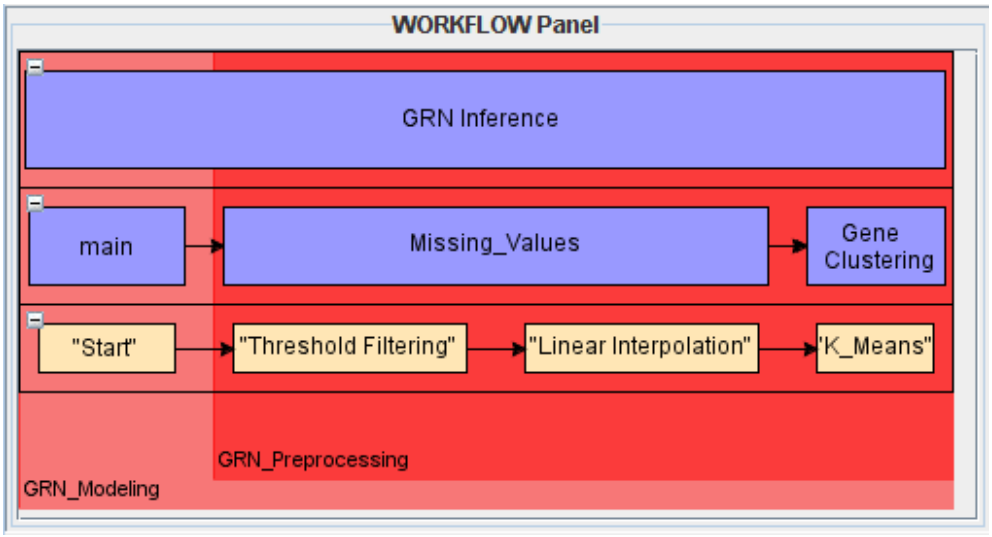


Figure 6.12: Workflow of the current experiment after K-Means has been run. The active decision-making module is *GRN Preprocessing* as well.

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. 6.13: 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. 6.14). 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 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

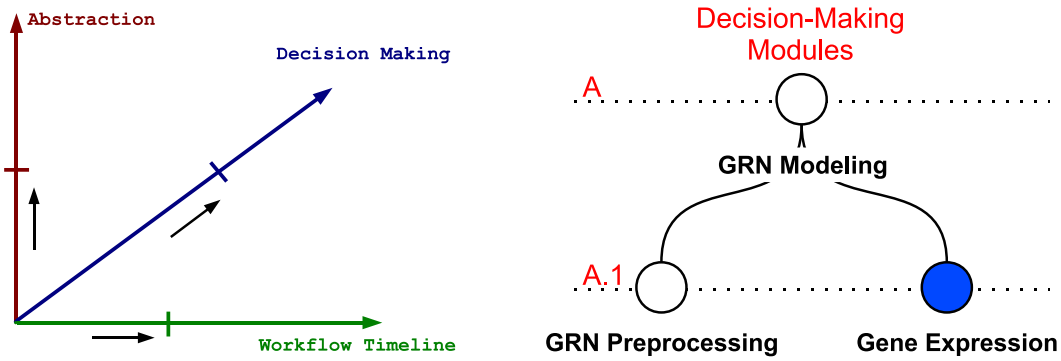


Figure 6.13: State of the system in the 3-axes reference space (left) and corresponding activation status of decision-making modules (right) at the beginning of actual GRN inference phase.

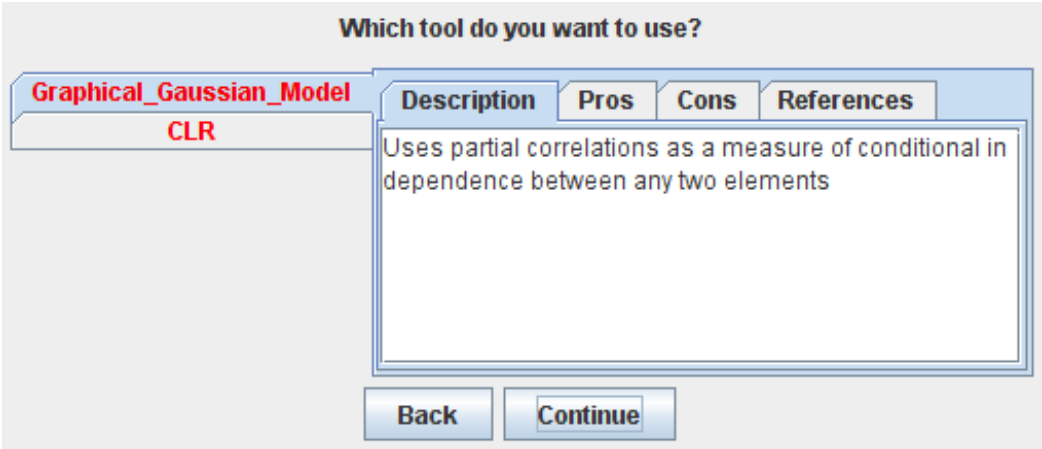


Figure 6.14: Supported tools implementing Correlation methods. Both algorithms are suggested, for different motivations.

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. 6.15) 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. 6.16). The final network is then obtained: in Fig.6.17 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.

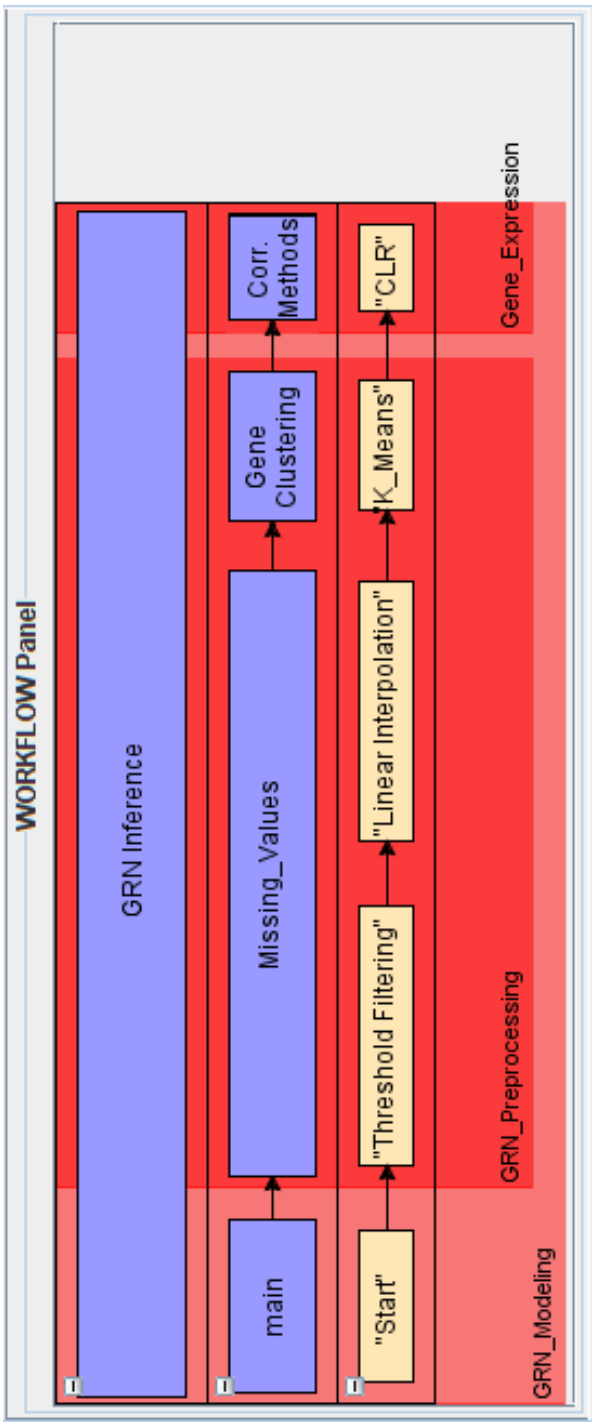


Figure 6.15: Workflow of the current experiment after CLR algorithm has been run. The active decision-making module is *Gene Expression*.

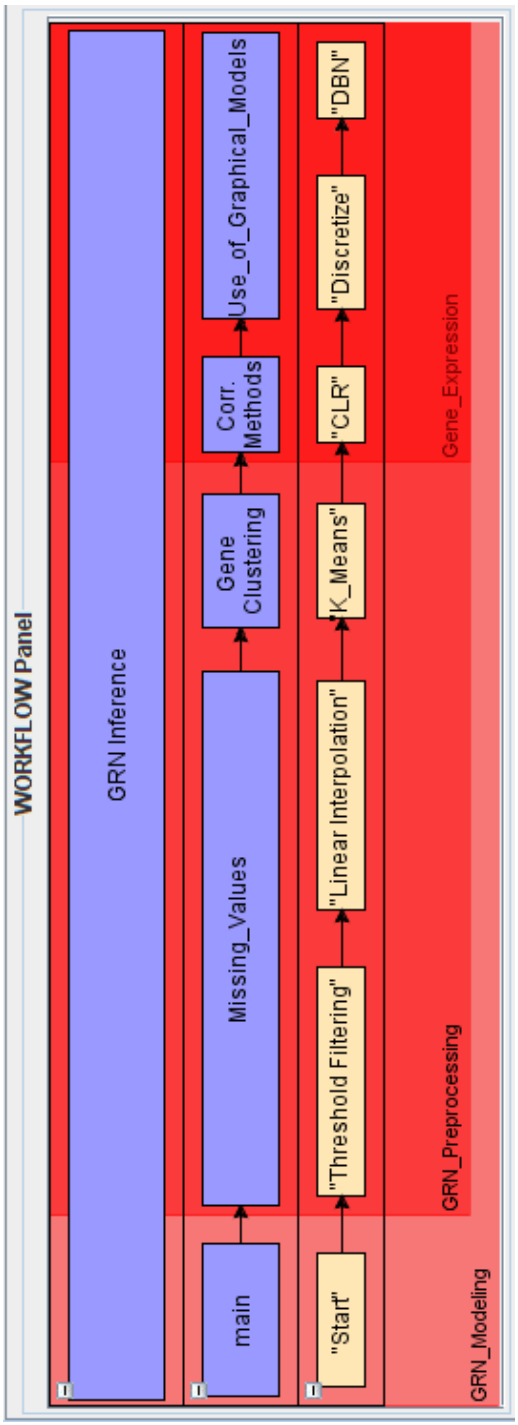


Figure 6.16: Final workflow of the current experiment. It can be eventually saved for sharing or reusing it. 5

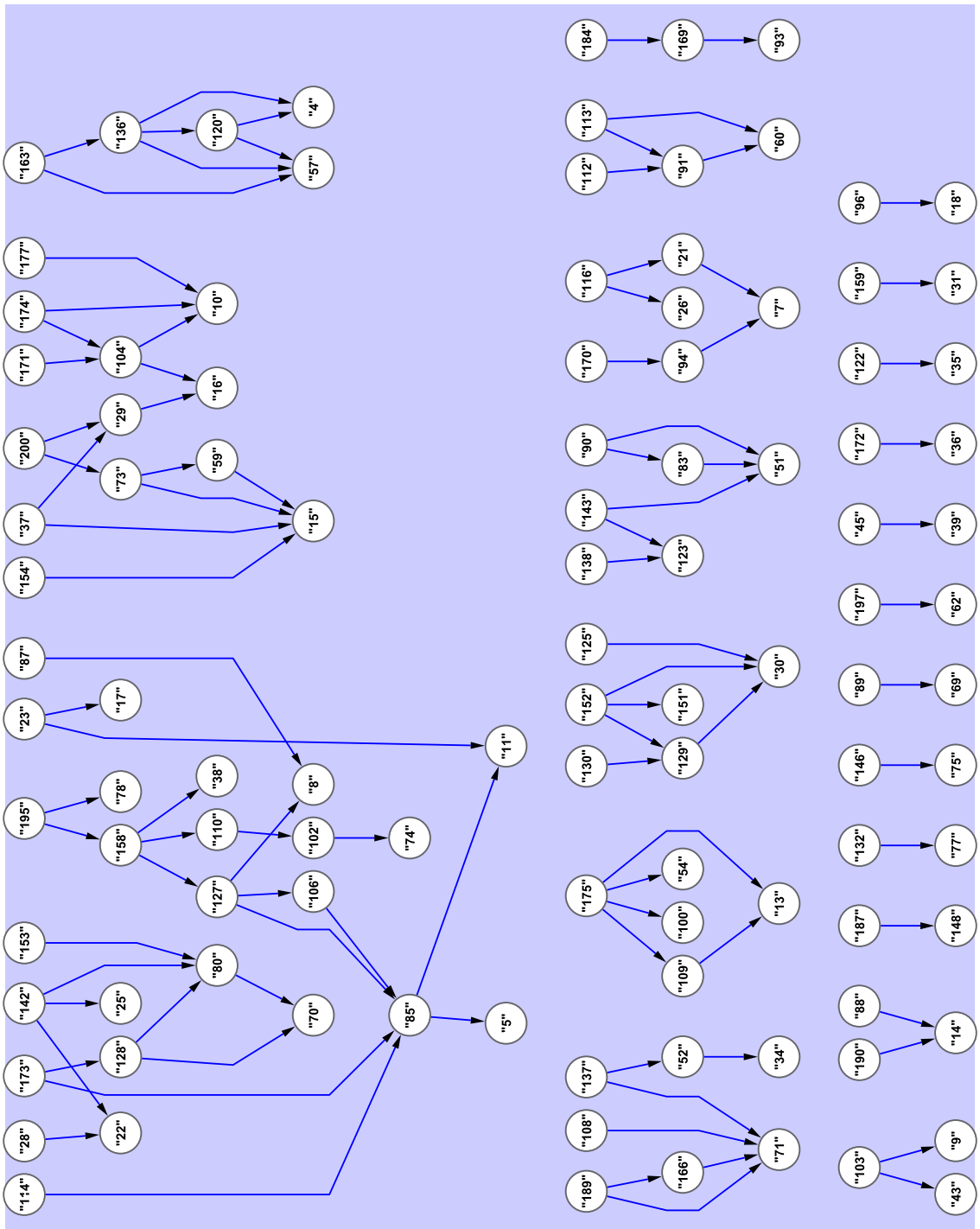


Figure 6.17: GRN inferred from the input dataset. This visualization is obtained by means of Cytoscape software, supported by BORIS system.

7

Case Study: Protein Complex Extraction from Protein-Protein Interaction Networks.

This Chapter contains a case study about extraction of protein complexes from protein-protein interaction networks. A complete analysis of the biological issue is done by means of the BORIS system, in order to show both how the hybrid architecture faces a problem and how the software implementation interacts with the user.

7.1 Biological Problem

Proteins represent the working molecules of a cell, but to fully understand cell machinery, studying the functions of proteins is not enough. The biological activity of a cell is not defined by the proteins functions *per se* (89), what it is really important is the interactions among proteins.

A group of proteins that interact in order to regulate and support each other for specific biological activities is called a protein complex. Protein complexes are one of the functional modules of the cell, an example of this protein function modules are RNA-polymerase and DNA-polymerase.

The concerted action of different functional modules is responsible of major biological mechanisms of a cellular process such as DNA transcription, translation, cell cycle control, and so on. Since a protein could have several binding sites, each protein can belong to more than one complex and exhibit more than one functionality. The basic element of these modules is the protein-protein interaction (PPI). The figure 7.1 shows the relationship between the protein-protein interaction network for the bacterium Mycoplasma pneumoniae and a whole-cell

tomogram. In the network are highlighted five large complexes and the lines that show where some of these structures can be found in the tomogram. For example ATP synthase still need to be located. The tomogram was kindly provided by A. Frangakis (European Molecular Biology Laboratory (EMBL), Heidelberg, Germany).

7.2 Bioinformatics Approach

A large amounts of PPI data have been identified for different biological species by using high throughput proteomic technologies. Of course experimentalists can take advantage of using different online databases containing a list of PPIs for each species (DIP (86), MIPS (87), etc..), but unfortunately available datasets are still incomplete and contain non-specific (false positive) interactions (88), in fact only a few of interactions have been verified with small scale experiments (*in vitro*) as real interaction with an emerging function.

Usually, in bioinformatics a collection of these interactions is modelled as a directed graph, the protein-protein interaction network (*PPIN*), where nodes represent proteins and edges represent pairwise interactions: it allows us to exploit graph theory methods and solutions.

The task of exploiting biologically relevant modules in PPINs represent an active research area in bioinformatics, not only for cell understanding, but also for new drugs developing; for example, several authors, as (90), are studying the mechanisms that regulate the evolutionary crossroads of p53 complex, responsible for different aspects of animal life, in developing human cancer cells. Then, identifying protein complexes with emerging function turns into extracting sub-networks with some emerging properties. Because of the importance of isolating functionally coordinated interactions, a lot of models, algorithms and strategies have been introduced to extract interesting PPI subnetwork (soft-clustering, greedy heuristics, probabilistic approaches, etc..), but each of them has proper pros and cons.

Since PPI dataset preprocessing plays a prominent role in PPI Network analysis, several authors aim to increase the reliability of these data. Some preprocessing strategies are aimed to eliminate false positive interactions (*FP*) from dataset obtained by online DBs. For example (91) notices that the interactions not part of dense subnetworks, are more likely to be interactions that are do not exist. To identify these false positives, he combined two topological metrics named Cluster Coefficient(92) and Centrality(93). Also (94) uses the same algorithms, but he adopted a different methodology, integrating individual topological measures into a combined measure by computing their geometrical mean. A different approach to improve the quality of PPI datasets is adopted by (95), that attempts to detect

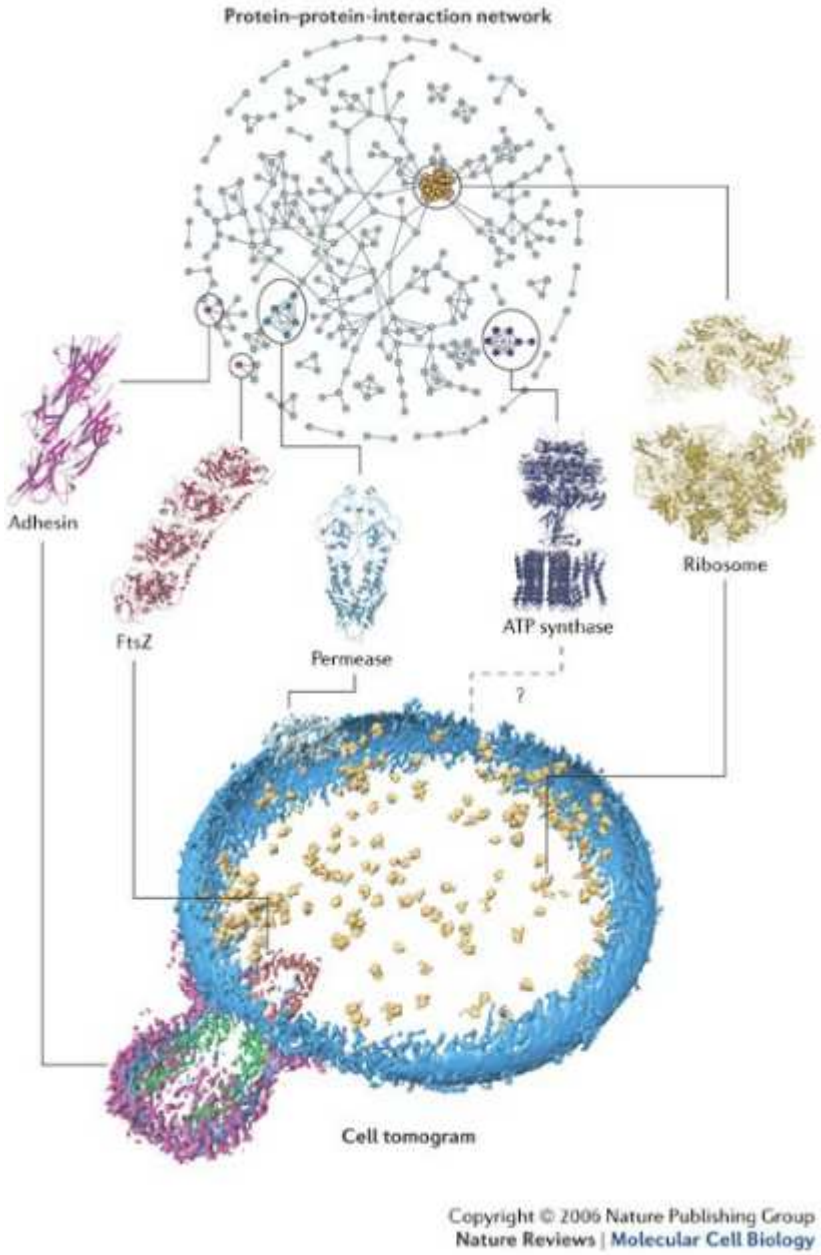


Figure 7.1: Cell tomogram. This figure shows five large complexes inside the PPI network and the corresponding location in the cell tomogram. Figure by Aloy et al. Nature Reviews Molecular Cell Biology 7, 188197 (March 2006).

those interactions that are missed by large-scale experiments or, in other words, he points to predict false negative using a topological analysis.

After having analysed some preprocessing techniques, it is possible to focus on the main goal, that is finding meaningful groups of biological units. A number of approaches have been proposed to solve the protein complex prediction problem,and a lot of them are based on clustering. A well know algorithm introduced by (96), the Molecular Complex Detection Algorithm (*MCODE*), makes use of local graph properties and is aimed at finding densely connected regions in protein interaction networks. Another algorithm based on local search is the Restricted Neighbourhood Search Clustering Algorithm (*RNSC*) developed by (97). This algorithm searches for a low-cost clustering by first composing an initial random clustering, then reducing the clustering cost by a near-optimal strategy. A different strategy is adopted by the Markov Clustering Algorithm (*MCL*)(98), that divides the graph by means of flow simulation. In facts, it separates the graph into different segments, with an iteration of simulated random walks within a graph.

7.2.1 Graph-based methods for analysing PPI networks

Usually cellular networks can be modelled by mathematical graphs $G(V, E)$, using nodes $v \in V$ to represent cellular components, and edges $e \in E$ to represent their various types of interactions. In particular, protein-protein interaction networks are conveniently represented as undirected graphs (99), where the nodes are proteins and two nodes are connected by an undirected edge if the corresponding proteins physically bind.

The representation of complex PPI networks as undirected graphs make it possible to systematically investigate the topology and function of these networks using well-understood graph-theoretical methods that can be used to predict the structural and dynamical properties of the underlying network. Such predictions can help at lower complexity level (local properties), to understand new biological hypotheses regarding both the unexplored PPIs of the network (edges of the graph) and the function of some proteins that are testable with subsequent experimentation. Moreover, at higher complexity level (global properties), mathematical modelling also enables an iterative process of sub-network reconstruction and complex detection, where model simulations and predictions are closely coupled with new experiments chosen systematically to maximize their information content for subsequent model adjustments (100). Thus, the most general level of network analysis comes from global network measures, used for characterizing and comparing the configuration of the nodes and their connecting edges. The most known global property of a PPI network is related to its topology, in fact the most of biological networks have several nodes with only a few connections

and few nodes highly connected; this property is called scale-free topology and it is characterized by a power-law degree distribution that decays slower than exponential. Others topological measures in proteomics are employed such as the *Degree Distribution* (the degree of a node is the number of edges it participates in) and the *Clustering Coefficient* (the number of edges connecting the neighbours of the node divided by the maximum number of such edges), the *Betweenness Centrality* (a measure of the centrality of a node and its influence over data flows in the network), the *Closeness Centrality* (a measure of the closeness of a node, on average, to all the other nodes): in fact they can efficiently capture the cellular network organization.

7.2.2 Algorithms and Tools for Complex Extraction

7.2.2.1 MCODE Algorihtm

The Molecular Complex Detection (MCODE) is a graph theoretic clustering algorithm that detects densely connected regions in large PPI networks, in order to detect molecular complexes. This algorithm was created in 2003 and thenceforth it has been setting the benchmark for complex detection in PPI Networks. It is based on vertex weighting by local neighborhood density and outward traversal from a locally dense seed protein to isolate the dense regions according to given parameters. Moreover it allows fine-tuning of clusters of interest without considering the rest of the network and allows examination of cluster interconnectivity, which is relevant for protein networks.

The MCODE algorithm operates in three stages: (1) vertex weighting, (2) complex prediction and (3) optionally postprocessing by means of certain connectivity criteria. For this algorithm, the PPI Network will be modelled as a undirected graph, where vertices are molecules and edges are molecular interactions; this graph representation allows to apply some graph theoretic methods in order to aid in analysis and solve biological problems. In facts, MCODE exploits a vertex-weighting scheme based on the clustering coefficient to find locally dense regions of a graph and a density measure based on the connectivity level of a graph.

During the first stage, all vertices are weights with their local network density according to properties of the vertex neighborhood. The second stage is the core of the algorithm: it takes as input the previously modified vertex weighted graph, seeds a complex with the highest weighted vertex and recursively includes vertices in the complex whose weight is above a given threshold depending on a given percentage away from the weight of the seed vertex. This process identifies densest regions of the network; obviously the threshold parameter defines the density of the resulting complex. The last stage basically deletes complexes that

do not contain at least a graph of a given minimum degree. Moreover, two optional filters are included, such as 'fluff' option (increasing the size of the complex) and 'haircut' option (removing the vertices that are singly connected to the core complex). If both options are specified, fluff is run first, then haircut.

7.2.2.2 RNSC Algorihtm

The Restricted Neighborhood Search Clustering algorithm (RNSC) partitions the PPI network into clusters based on a cost function that is assigned to each partitioning.

The algorithm is a cost-based local search algorithm, based loosely on the tabu search meta-heuristic (Glover, 1989). In this case, the clustering is equivalent to a partitioning of the network into some sets of proteins. The RNSC efficiently searches the space of partitions and assign a cost of each set of proteins. The algorithm searches using a simple integer-valued cost function as a preprocessor before it searches using a more expressive real-valued cost function. Usually, the algorithm is initialized with random values and it searches for a low-cost clustering by first composing an initial random clustering, then iteratively moving one protein from one cluster to another in a randomized fashion in order to improve the clusterings cost and reach a near-optimal amount. To avoid local minima, the algorithm uses diversification and multiple experiments, that shuffle the clustering by occasionally dispersing the contents of a cluster at random, preventing any possible cycling back to the previously explored partitioning. Notice that, since the RNSC is randomized, different runs on the same input data will result in different clusterings. Three additional criteria are used to achieve high accuracy in predicting protein complexes, i.e. a maximum P-value for functional homogeneity, a minimum density and a minimum size.

7.2.2.3 MCL Algorihtm

The Markov Cluster algorithm (MCL) is a fast and scalable unsupervised cluster algorithm for PPI networks based on simulation of stochastic flow in graphs.

The algorithm simulates a flow process alternating two simple algebraic operations on matrices; the structure of each cluster is bootstrapped via a flow process that is inherently affected by any cluster structure present and the basic algorithm does not include some procedural instructions for assembling, joining, or splitting of protein groups. MCL is composed by two steps: the first step is the expansion, which coincides with normal matrix multiplication: it models the spreading out of flow, becoming more homogeneous; the second step is inflation, which is mathematically speaking a Hadamard power followed by a diagonal scaling, such that the resulting matrix is stochastic again, i.e. the matrix elements on

each column correspond to probability values. The MCL process causes flow to spread out within natural clusters and evaporate in different clusters. The only algorithm parameter is the inflation; it models the contraction of flow, becoming thicker in regions of higher current and thinner in regions of lower current. By varying this parameter, clusterings on different scales of granularity can be found. In the Markov Cluster algorithm, the number of clusters can not be specified in advance.

7.2.2.4 Cytoscape Tool

Cytoscape is an open source bioinformatics software platform for the visualization and analysis of biological network data. Cytoscape core distribution provides a basic set of features for automated graph layout, integrating network data with other data such as expression data and functional annotations, and setting visual attributes according to node or edge attributes, establishing a powerful visual mappings across these data. This tool is widely used in PPI Network analysis, because it can visualize the topological relationship among the protein clusters (or complexes) in the model of global interaction network, revealing which of the clusters is highly connected to other clusters.

7.3 Experimental Dataset

In our experiments, among different available on-line databases of PPIs network, we use the Database of Interacting Proteins (*DIP*). The input dataset used in this scenario is a subset of *Saccharomyces cerevisiae* PPI-Network composed by 34 proteins and 90 interactions, as shown in Table 7.3. This table reports a list of 90 PPIs: for each PPI is shown the uniprotKB ID of the first protein, the uniprotKB ID of the second protein and the PID ID of the interaction between the previous pair of proteins. We chose this very simple dataset because it has been well studied by (104, 105) with small scale experiments (*in vitro*) at biological interaction level. *DIP* also provides a subset of PPIs curated manually by experts, that are called core PPIs.

7.4 System Running

The experiment related to this scenario begins when the user asks the system to extract protein complexes from a PPI-Network and inserts the chosen dataset: from now on, for each decision step the system reach a new state.

At this moment, the experiment is projected into the BORIS 3D space, as depicted in figure 7.2. The transitions from start position (when the system

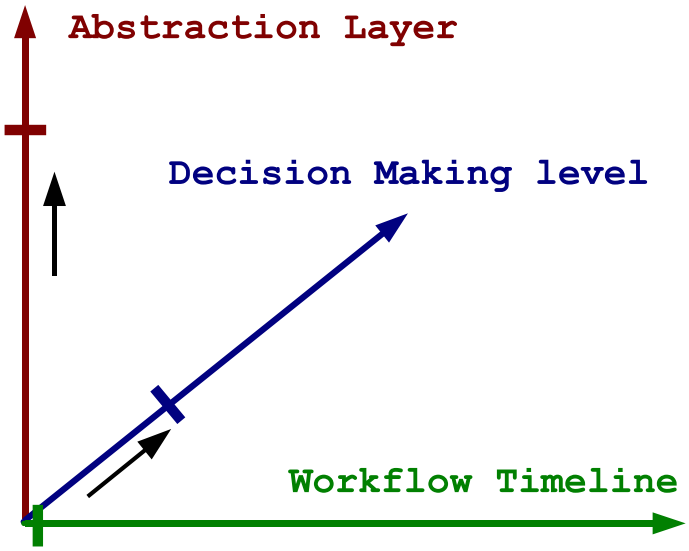


Figure 7.2: Projection of the system state over the hybrid architecture space at the first step of the protein complex extraction scenario.

state is at the point 0,0,0) to the current system state are highlighted in the figure 7.2 with black arrows. The three axes representing the projection of the experiment on the hybrid system, are configured as following: the projection of the current state to the abstraction layer axis reaches the highest level of abstraction, because the system get an overview to the "main goal", i.e. the protein complex extraction; the projection of the current state to the workflow timeline axis is in resting position, because no process was developed and no task was carried out; the projection of the current state to the decision making axis reach the highest meta-reasoning level, according to the decision making tree in figure 7.3.

This figure shows decision making modules responsible for the specific problem; the sub-tree obtained by the entire BORIS knowledge base is arranged in two meta-reasoning level, meta-reasoning level A (MRL A in the figure) and meta-reasoning level A.1 (MRL A.1 in the figure) and it contain the following modules:

- `Complex Extraction`, the parent module that gives directives to two children modules at the bottom, that could be activate in order to deal with more specific tasks;
- `Complex Preprocessing`, the child module that contains the reasoning about strategies and tools able to face the PPI Network preprocessing phase;

- `Complex Clustering`, the child module in charge of the decision-making activity regarding clustering strategies and tools.

In the figure are reported also some of activation rules (in the form of "Object, Attribute, Value") belonging to the Complex Extraction module that are responsible for giving focus to children module, i.e. these rules aim at shifting the reasoning process to a lower meta-reasoning level.

In the bottom of the figure 7.3 is reported the related treemap representation, where it is possible to see how the parent module includes its children modules, as well as the reasoning at higher level contains the reasoning at lower level; the system exploits these rules to suggest user which strategy could be adopted. Finally, some guidelines have been extracted from papers cited in section 7.2, translated into rules and placed into the appropriate module.

At the beginning of the experiment, Complex Extraction module is active: the job of this module is to analyse input data, in order to get the properties and parameters necessary to activate the proper rules; in this simple scenario, we take into account only a few of input features.

First of all, the system compare the PPIs of dataset with a list of core interactions, provided by *DIP* for the *Saccharomyces cerevisiae* species. In this case 67 of 90 interactions are reliable, because they are manually curated. Then the system creates the undirected graph, the *PPIN*, and checks if resulting network is scale-free, that is if its degree distribution follows a power law, at least asymptotically. In this scenario the PPIN is not scale-free. Since several authors(106) demonstrate that most networks within the cell approximate a scale-free topology, then some of our PPIs (edges of the network) could be false positives or new PPIs could be not revealed (false negatives) when DIP dataset was created. For this reason, a rule that propose PPIN preprocessing, in order to change the geometry of the network, is activated.

When the user follows the system advice, according to previous rule, the *PPI Complex Extraction* module gives focus to the child module *Complex Preprocessing* at lower meta-reasoning level, in order to deal with preprocessing task.

According to the analysis phase, the system knows the PPIN contains about 74% of core-interactions. Since has been estimated that approximately half the interactions obtained from high-throughput proteomic techniques may be false positives (107, 108, 109), the rule suggesting to find and delete false positive PPIs is not activated; in fact, cutting edges of PPIN could implicate some core-interactions are deleted and moving core-interactions is lethal for biological networks. For this reason, the rule suggesting to add new PPIs is activated.

When the user agrees to the advice, the system looking for tools implementing this strategy. In this simple scenario, the knowledge-base contains only a tool that can find and add some false negatives in PPIN: the *Detect Defective Cliques*

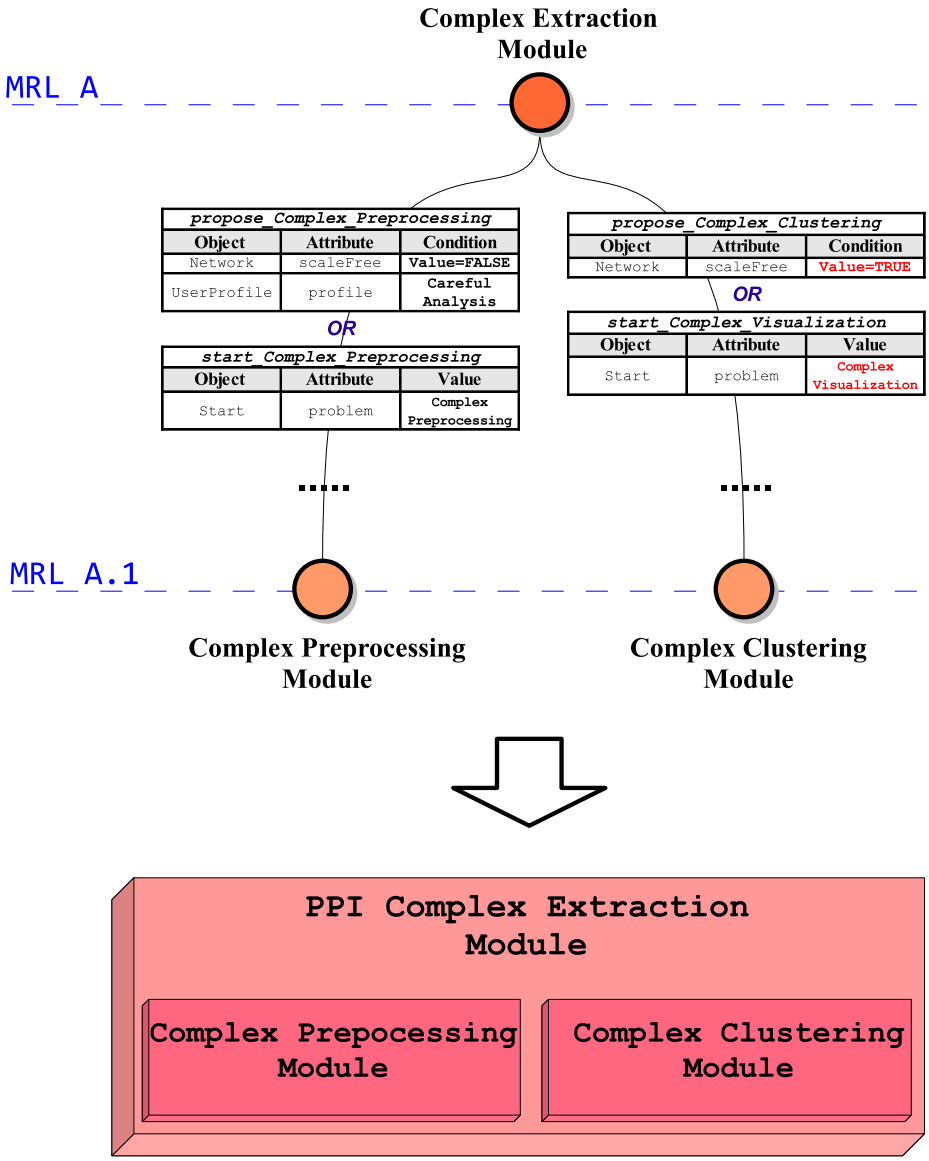


Figure 7.3: Decision making modules responsible for the protein complex problem and related treemap representation. Some transitions for the activation of child modules are reported.

algorithm, created by (95). When the user accepts to run the proposed algorithm, then the system informs that this algorithm requires, as input parameter, the number of common interactions between two defective cliques, and suggests to user a considerable value for the experiment.

When the user accepts the proposed value, the system executes the algorithm, that finds a new potential FN interaction between the proteins *P60010*

and *P33338*. At this moment, the PPIN is composed by 34 proteins and 91 interactions; the user could either continuing the experiment or executing another preprocessing tool (in cascade or restarting the preprocessing phase).

A virtual caption of the system at this moment is shown in figure 7.4. In the top of this figure it is possible to see the tree-dimensional space of hybrid system and the decision making module tree. The projection of the system state on the decision making axis shows that the notch is slided up, with respect to figure 7.2, to indicates the system will make reasoning at MRL A.1, in particular the complex preprocessing module is the active one. The red notch that identify the abstraction level is gone to the lowest layer, i.e. the object layer. Finally the workflow timeline get a step ahead, because an instance of *Detect Defective Clique* tool has been executed. On the top-right of the figure, the active module, responsible for strategies and tools related to complex preprocessing is highlighted with blue color.

In the bottom of the figure is reported a part of the BORIS GUI (see *Appendix A*) that shows the workflow of the experiment. As explained in section 3, the workflow is projected inside the BORIS 3D-space; the executed process is developed on tree different abstraction layers: at the highest layer that is the main goal (Complex Clustering); at the intermediate abstraction layer that is the complex preprocessing sub-goal (to add False Negatives) and at the lowest abstraction layer, the object layer that is the instance of executed tool (detect defective cliques). In this caption, there are also decision making modules used till now. The Complex Extraction module, the biggest red box, contains the whole experiment, while the Complex Preprocessing box has been activated only for the task related to strategies and execution of the network preprocessing.

If the user wants to try another solution before continuing the experiment and does not want to accept the system advices, he could choose follow the strategy to find and delete false positive PPIs. In this case, the system saves results obtained so far and proposes to run one of those algorithms that satisfy the selected strategy. The user selects the *Betweenness Centrality* algorithm from among three different tools available into the knowledge-base, because the system indicated this is the algorithm with the lowest computational cost. The result of *Betweenness Centrality* algorithm is a PPIN with 34 proteins, 88 interactions and 65 core-interactions; then the system advices the user to change strategy and/or modify parameters because 2 core-interaction has been deleted.

Figure 7.5 shows the workflow the system built so far. In the figure it is possible to see how *PPI Complex Extraction* module contains all the workflow elements; it supervise the main problem at highest abstraction layer, giving the other directives to *Complex Preprocessing* module. The latter is responsible of some strategies for verifying and purifying the network and have knowledge about tools used for data manipulation. At intermediate abstraction layer, the child

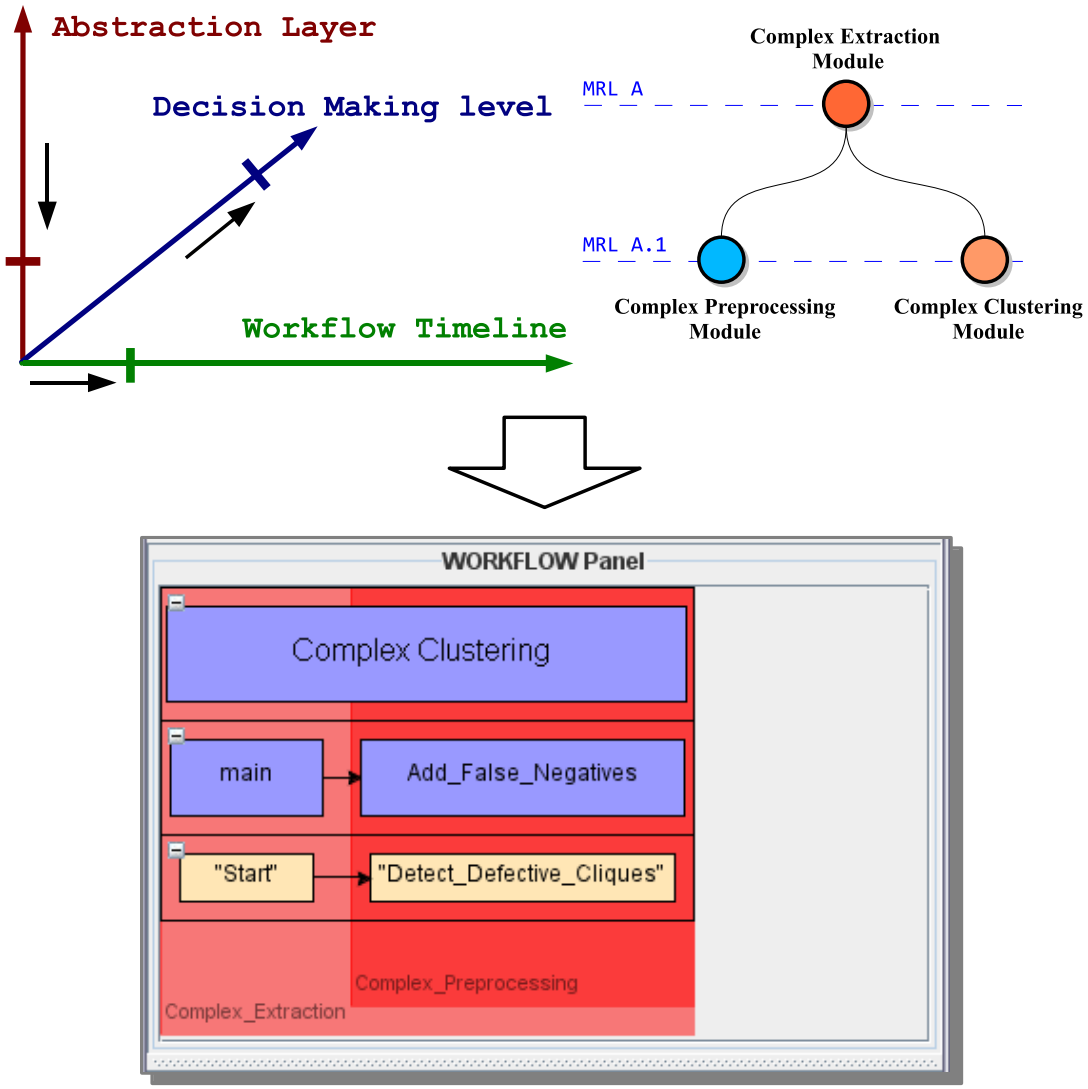


Figure 7.4: Case study at the preprocessing phase. The projection of the state of system over the tree axis is reported, the active module is highlighted and the multi-level workflow representing the system output is shown.

module contains the strategies used in this experiment: in facts the user tried first to add new PPIs and then to delete false positive PPIs; obviously, both these strategies have the same PPIN as input, according to the user choices. The instance of tools used for processing data are shown at lower abstraction layer and their order in the figure follows the implementation timeline.

Notice that some numbers with a yellow background are highlighted in the workflow panel. They represent the available paths the workflow management system integrated in hybrid architecture offers to the user, that agree with the

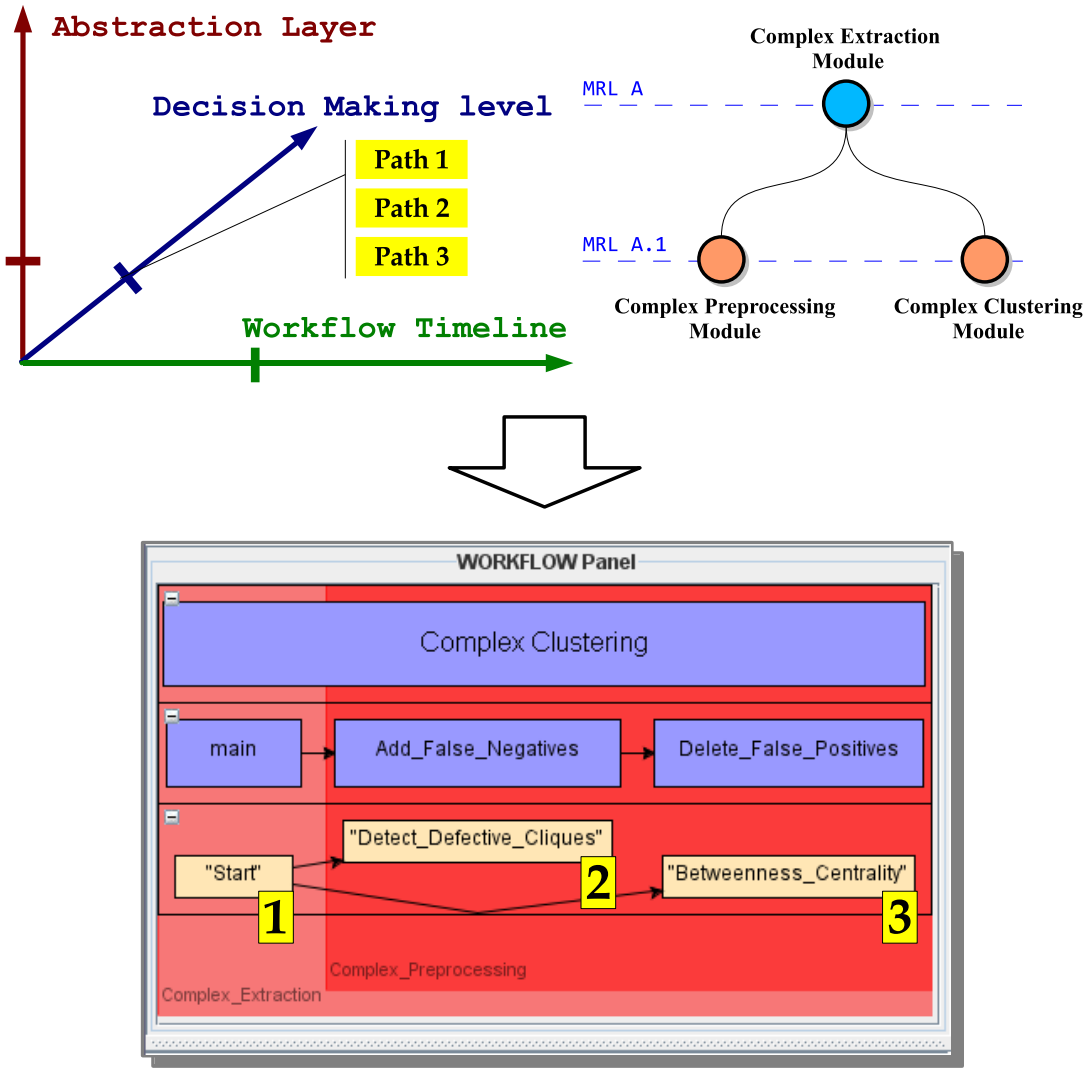


Figure 7.5: Selection of the preprocessing tool. After the execution of tree different tools, the system proposes to the user the available outputs for data analysis.

tree decision states showed in the BORIS 3D-space. In facts, in this scenario, the user can choose among three pathway: he could accept the system advice and continue the workflow elaboration from the output number 2 (defined as "Path 2" in the BORIS 3D-space); he could refuse the system advise and select the output number 3 (defined as "Path 2" in the BORIS 3D-space); he could refuse the main suggestion, i.e. the preprocessing strategy, by-passing the complex preprocessing module and continuing the workflow elaboration from the the output number 1, i.e. the input file (defined as "Path 1" in the BORIS 3D-space);

When the user chooses the appropriate output to continue the experiment,

Table 7.1: Some features of the three protein complex prediction algorithms: RNSC, MCODE and MCL.

Comparative table among RNSC, MCODE and MCL			
	RNSC	**MCODE**	**MCL**
Use Local search approach	Yes	Yes	No
Support multiple assignment of protein	No	Yes	No
Support weighted edge	No	No	Yes
Use a fast and scalable algorithm	No	No	Yes
Is suitable for sparse graph	Yes	Yes	No
High sensibility to FP & FN PPIs	No	Yes	Yes
...	...	...	...

the *PPI Complex Extraction* module knows the data input has been preprocessed and gives focus to the child *Complex Clustering*. Also the latter module knows the preprocessing phase is done, thus it uses this information for suggesting an appropriate clustering strategy. The authors (94, 110) demonstrated *MCODE* is sensitive to noise in the PPIN and the preprocessing phase can increase the algorithm performance. Other authors (101, 102) notice that *MCL* and *RNSC* work almost in the same manner in terms of precision and recall, whether PPIN are noisy or purified. Moreover *MCODE* algorithm has been widely used with protein-protein interaction networks belonging to the species *Saccharomyces cerevisiae*, so that the system can suggests standard parameters for this species. For these reasons, the system proposes to use the aforementioned algorithm, based on the local search analysis, for clustering. When the user accepts the advice and confirm proposed parameters, the system runs the *MCODE* algorithm. Now the user can either ending the experiment or executing another clustering tool, having as input PPIN the output of the preprocessing phase. If the user wants to try another tool, he can consider descriptions, pros and cons that are available for each strategy and algorithm contained into the system. In this case, he notices that *MCL* algorithm is described as the faster than the other algorithms and, moreover, it does not appear so bad with dense graphs; then, the user chooses to run *MCL* algorithm, based on the flow simulation analysis.

All the information about cited algorithms (i.e. MCODE, RNSC and MCL) are included in knowledge base and represented as facts; each suggestion is obtained by means of some rules. A comparative schema among the three algorithms is reported in the table 7.1, in order to highlight some their characteristics.

Features reported in the first column, have been obtained by means of scientific

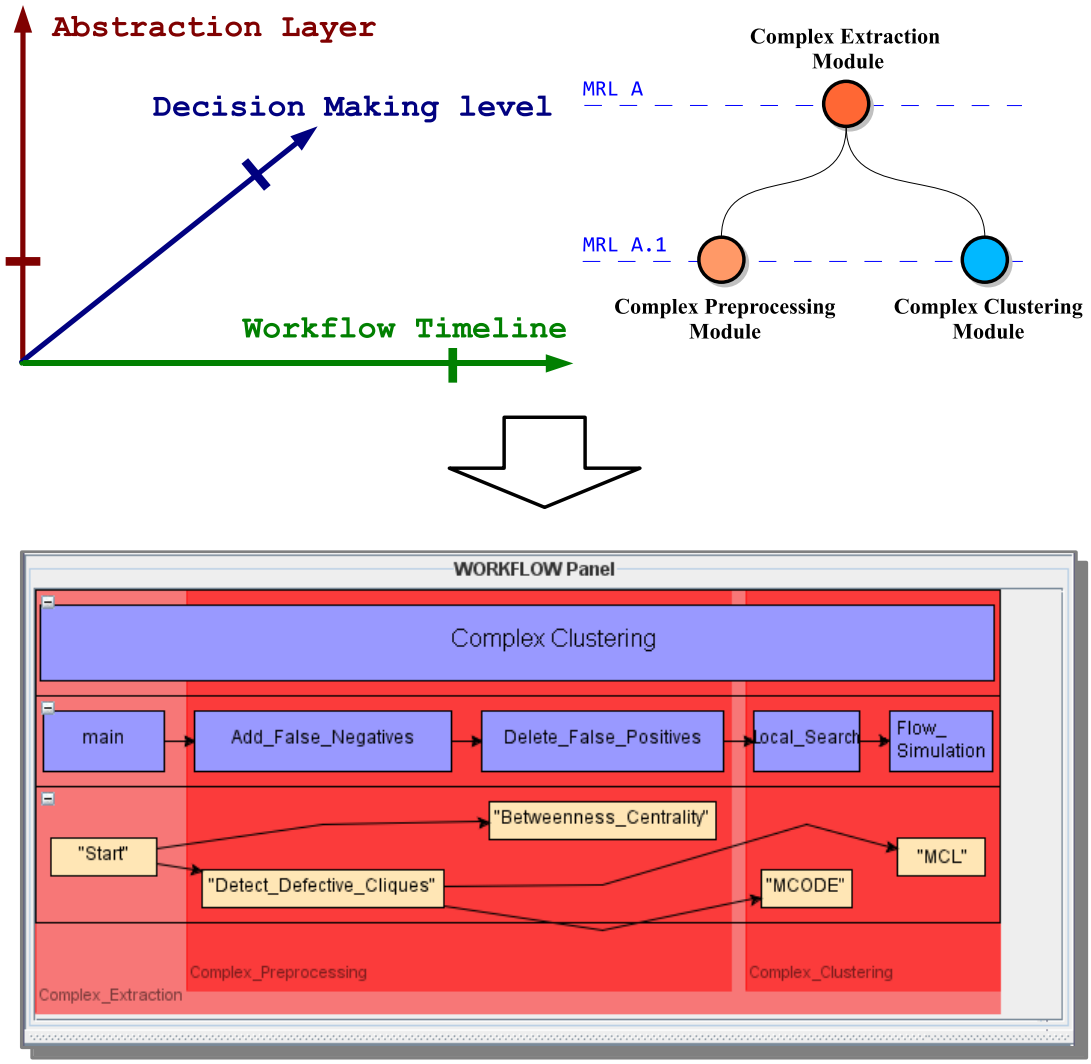


Figure 7.6: Workflow of the whole experiment. The system shows in a tree-like structure all the strategies and algorithms have been used, according to abstraction layers.

papers and humane expertise and represents some discriminant features that has been used in order to generate some rules that will be, eventually, selected by the rule-based engine. Notice that some boundary conditions could imply the activation of more than one rule that satisfy the user request: in these cases, the rule-based engine is responsible to compare all the active rules and, then, the one with higher priority is executed before the other.

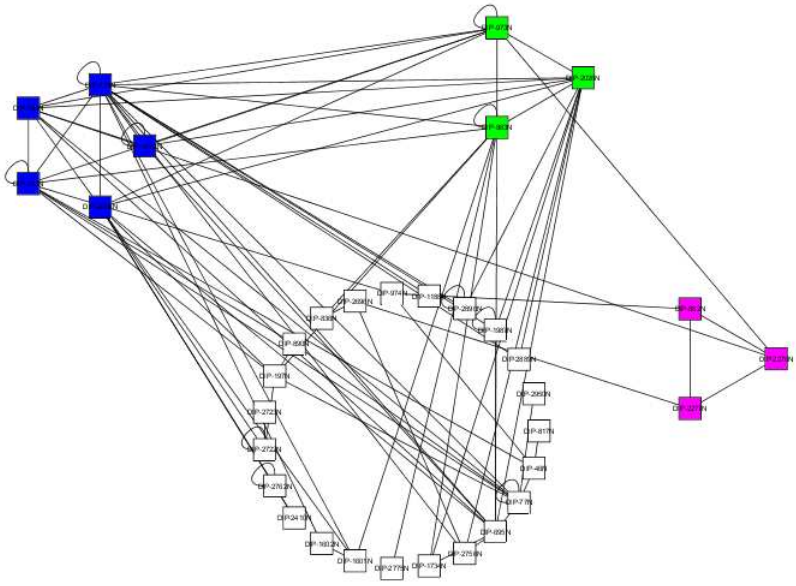
The final workflow is shown in Fig. 7.6. At the intermediate abstraction layer are depicted all the strategies within the boundaries of their respective

Output 1 DDC preprocessing + MCODE Clustering	
Cluster	**Proteins**
1	P33338, Q12446, P32793
2	Q12134, P15891, P53933, P39743, P60010, P32793
3	P48562, Q06648, P19073
Output 2 DDC preprocessing + MCL Clustering	
Cluster	**Proteins**
1	P53933, P32944, P38274
2	P60010, P17555, P40450, P41832, Q03048, P38793, P46680
3	P15891, P48232, Q12270, P32790, Q12134, P33338, P39743, P32793, P25343, Q12168, P38266, P47129, P40325, Q06604, P38837
4	P13517, Q06648
5	Q06440, Q03088

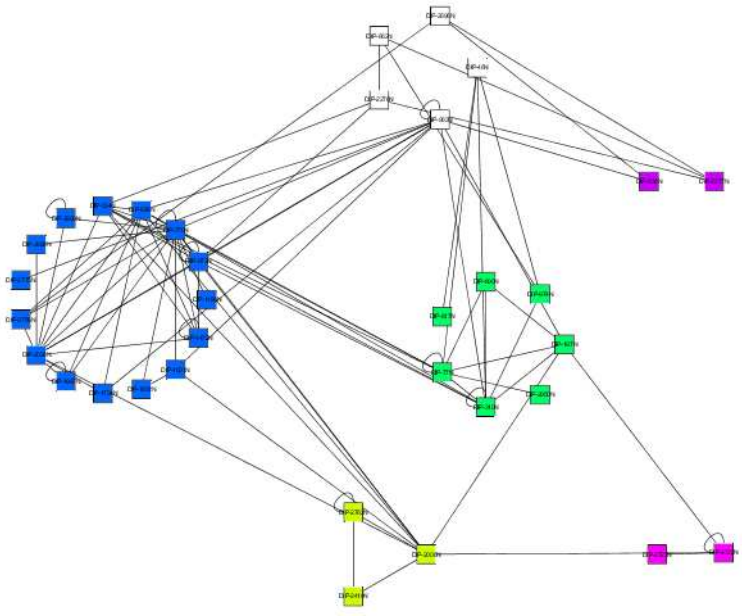
Table 7.2: System outputs. The implemented workflow gives 2 outputs: the former with 3 complexes and the latter with 5 complexes.

decision modules, whereas at the lowest abstraction layer there are all the tools implemented in this scenario. The above picture shows also the BORIS 3D-space; once again it is possible to notice that both the red notch of the abstraction axis is located in the lower position, since the MCL algorithm has been just executed and the active decision module is the "Complex Clustering" module. For the next step, the selection of clustering strategy, the behavior of the system is similar to the preprocessing phase, in fact the user could choose between two clustering algorithms.

Before concluding the experiment, the system proposes to visualize the outputs of MCODE and MCL algorithms with the well know Cytoscape tool (111). Visualization of clustering results, obtained through Cytoscape, are shown in Fig. 7.7. Finally, the user can choose between two outputs shown in Table 7.2, according to its knowledge about the protein complex domain and/or using external evaluation parameters.



(a) MCODE Clustering. Parameters: K-Core=2, Degree Cut-Off=2, Node Score Cut-Off=0.2, Haircut= NO, Fluff= NO, Include Loops= NO



(b) MCL Clustering. Parameter: Inflation (Cluster Granularity)= 2.0

Figure 7.7: Clustering visualization with Cytoscape tool. In the top, the result of MCODE clustering (3 protein complexes); in the bottom, the result of MCL clustering (5 protein complexes).

#	Protein_A	Protein_B	PPI_ID
1	P60010	P15891	DIP-10439E
3	P48562	P15891	DIP-3499E
4	P32790	P15891	DIP-2452E
5	P17555	P15891	DIP-1139E
6	Q12134	P15891	DIP-3500E
7	P60010	P60010	DIP-1145E
8	P41832	P60010	DIP-1155E
9	Q03048	P60010	DIP-1157E
10	Q12446	P60010	DIP-1158E
11	P07274	P60010	DIP-1143E
12	P33338	P60010	DIP-1175E
13	P60010	P46680	DIP-1140E
14	P17555	P46680	DIP-3502E
15	P38274	P53933	DIP-3683E
16	P39743	P53933	DIP-3907E
17	P33338	P53933	DIP-3966E
18	P32793	P53933	DIP-11282E
19	P19073	P41832	DIP-1154E
20	P13517	P28495	DIP-3546E
21	Q06648	P28495	DIP-3547E
22	P48562	P19073	DIP-2580E
23	Q06648	P19073	DIP-2583E
24	Q06648	P48562	DIP-3639E
25	P46680	Q03048	DIP-1346E
26	P53933	Q03048	DIP-14613E
27	Q12446	Q03048	DIP-1161E
28	P53933	Q06440	DIP-3604E
29	Q03048	Q06440	DIP-11816E
30	Q06440	Q06440	DIP-4127E
31	P38274	P38274	DIP-9812E
32	P32944	P38274	DIP-7787E
33	P13517	Q12446	DIP-1160E
34	Q12446	Q12446	DIP-11092E
35	P39743	Q12446	DIP-3699E
36	P32790	Q12446	DIP-1162E
37	P33338	Q12446	DIP-15438E
38	P32793	Q12446	DIP-11095E
39	P41832	P07274	DIP-1164E
40	P40450	P07274	DIP-1166E
41	P17555	P07274	DIP-3762E
42	P53933	P25343	DIP-4047E
43	Q12446	P25343	DIP-4048E
44	P38266	P25343	DIP-1781E
45	P15891	P39743	DIP-1138E

#	Protein_A	Protein_B	PPI_ID
46	Q12168	P39743	DIP-3900E
48	P25343	P39743	DIP-1780E
49	P39743	P39743	DIP-3901E
50	P33338	P39743	DIP-10013E
51	P38266	P39743	DIP-3902E
52	P40325	P39743	DIP-3903E
53	P47129	P39743	DIP-3904E
54	Q06604	P39743	DIP-10016E
55	P32793	P39743	DIP-10017E
56	P53933	P32790	DIP-10020E
57	P39743	P32790	DIP-10011E
58	P17555	P32790	DIP-10018E
59	P40325	P32790	DIP-10019E
60	Q12134	P32790	DIP-11232E
61	Q06604	P32790	DIP-3964E
62	P15891	P33338	DIP-2453E
63	P48562	P33338	DIP-3965E
64	P33338	P33338	DIP-3144E
65	P60010	P17555	DIP-1144E
66	Q03048	P17555	DIP-11822E
67	P39743	P17555	DIP-3029E
68	P17555	P17555	DIP-1177E
69	P38793	P17555	DIP-4014E
70	Q06440	Q03088	DIP-3603E
71	P53933	P32944	DIP-4050E
72	P40325	P40325	DIP-2272E
73	P32793	P40325	DIP-2243E
74	Q12446	P38837	DIP-3700E
75	P47129	P47129	DIP-4186E
76	P32793	P47129	DIP-11280E
77	P39743	P48232	DIP-3906E
78	P39743	Q12134	DIP-10015E
79	P33338	Q12134	DIP-3967E
80	Q12134	Q12134	DIP-6160E
81	P32793	Q12134	DIP-11283E
82	Q12446	Q12270	DIP-3702E
83	P32790	Q12270	DIP-11231E
84	P28495	Q06604	DIP-9981E
85	P32793	Q06604	DIP-11285E
86	P15891	P32793	DIP-11370E
87	Q12168	P32793	DIP-11277E
88	P32790	P32793	DIP-2242E
89	P33338	P32793	DIP-3968E
90	Q12270	P32793	DIP-11284E

Table 7.3: There are 90 PPIs among 34 Proteins for the species *Saccharomyces cerevisiae*. Each row contains two PPIs. For each PPI is shown the first protein uniprotKB ID, the second protein uniprotKB ID and the interaction PID ID between the previous pair of proteins. The complete set of PPIs for this species is available in *Scere20081014.txt* file, provided by PID online database(86).

8

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.

8.1 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.

8.1.1 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. 8.1. 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.

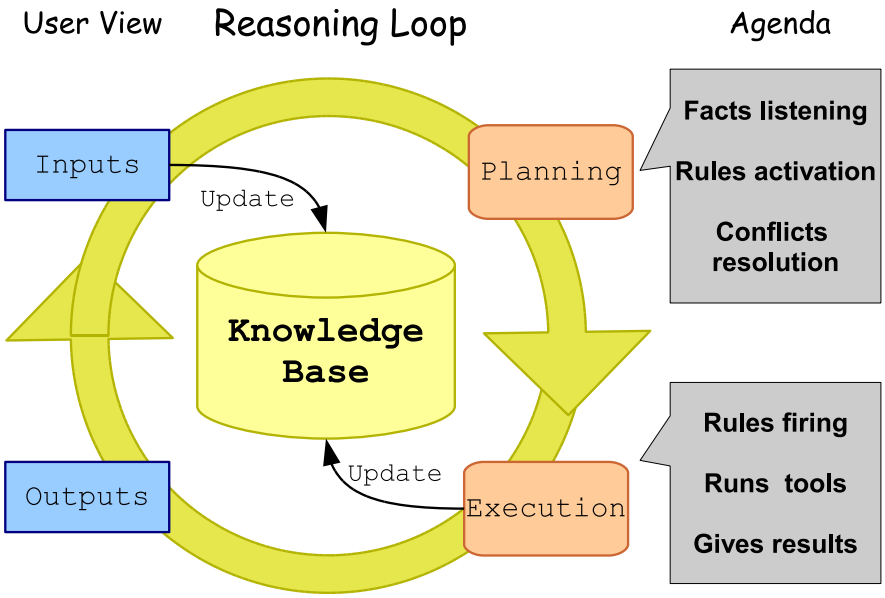


Figure 8.1: Reasoning Loop

8.2 Jess: the Rule Engine for the Java Platform

The Rule-Based system of Boris has been implemented using Jess (114), 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 (115) 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 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.

8.2.1 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 (116), ART (117) and CLIPS (118): 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.

8.3 Protege Ontology Editor

The knowledge base and the underlying ontology have been implemented with Protege (112, 113), 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 (119) or Ontoviz plugins (120), or a simple way to export instances into Jess facts by means of JessTab plugin (121). 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.

8.4 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. 8.2. The main control program of the expert system, also implementing the GUI seen in Chapter 5, 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.

8.5 JGraphX Library

JGraphX is the Java Swing library version of mxGraph (122), a product family of libraries, written in a variety of technologies, that provide features aimed at applications that display interactive diagrams and graphs. Development of JGraphX began as the diploma thesis of Gaudenz Alder at the Swiss Federal Institute of Technology, Zurich and it became a privately owned company in the U.K. in 2000 by David Benson.

The core client functionality of JGraphX is a Java compilable library that describes, displays and interacts with diagrams as part of your larger Java Swing application. JGraphX is primarily designed for use in a desktop environment, although Java does have web enabling features making it possible to deploy JGraphX in web environment.

Among the amount of applications provided by this library, the most important for the implementation of BORIS hybrid architecture are the functionality related to process diagrams, workflow visualization and flowcharts; in facts the

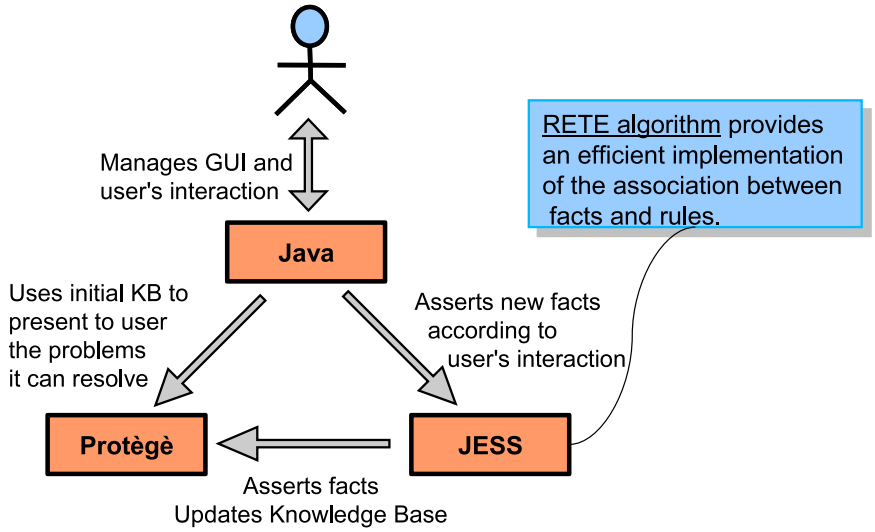


Figure 8.2: The interaction scheme among the computational tools adopted by the expert system belonging to BORIS framework-

main scope of JGraphX library is its visualization functionality and the interaction with the graph model through the web application GUI. JGraphX supports dragging and cloning cells, re-sizing and re-shaping, connecting and disconnecting, drag and dropping from external sources, editing cell labels in-place and so on.

The figure 8.3 shows an example of JGraphX visualization.

8.6 Eclipse Platform

Eclipse is a multi-platform of software development that is mainly composed by an integrated development environment (a small run-time kernel) and an extensible plug-in system (123). The Eclipse Project was originally created by IBM in November 2001 and in January 2004 was created the Eclipse Foundation, an independent not-for-profit corporation that permise the foundation of an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the life-cycle.

The most of the environment is written in Java and, at the beginning, it allowed to develop applications in Java, subsequently by means of various plug-ins, other programming languages have been included. Eclipse integrates the Eclipse Modeling Framework (EMF), that is a modeling framework and code

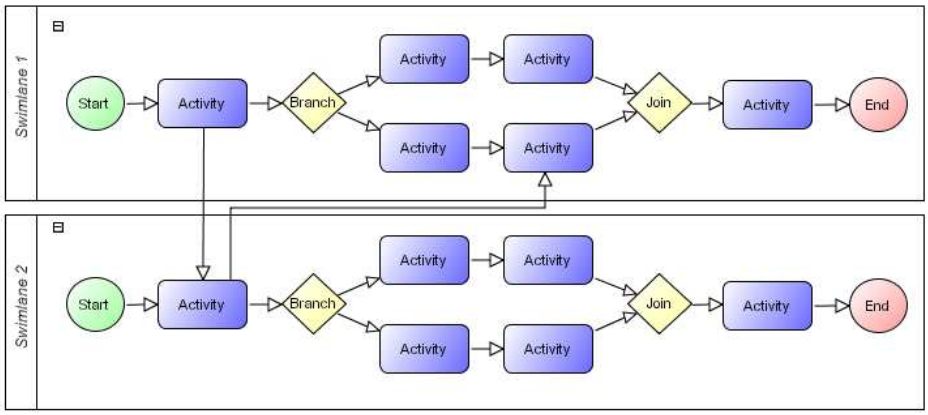


Figure 8.3: JGraphX: an example of the workflow layout. Figure from "*JGraphX User Manual. Copyright (c) David Benson, Gaudenz Alder 2006-2010.*"

generation facility for building tools and other applications based on a structured data model.

The most important thing for this work is there are, among all the available plug-ins, two environment that integrate the afore mentionate tools, i.e. the Jess Developer's Environment (JessDE) and the Protege Frame Editor. By means of these plug-ins, the BORIS hybrid architecture has been provided by the knowledge base and the decision making modules.

9

Conclusions and Future Work

In this work, 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 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 work 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, http://DSSResources.COM/history/dsshistory.html

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

[5] J. H. Moore and M. G. Chang, Design of Decision Support System, Database 12 (1980) 8–14.

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

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

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

[9] S. K. Singh, Database Systems: Concepts, Design and Applications, (2009). 4

[10] M. J. Druzdzel, R. R. Flynn, Decision Support Systems, Encyclopedia of Library and Information Science, Second Edition, Allen Kent (ed.), New York: Marcel Dekker, Inc., (2002). 4

[11] M. Henrion, J. S. Breese, E. J. Horvitz, Decision Analysis and Expert Systems. AI Magazine, 12(4), Winter 1991. 5

[12] Perreault L., Metzger J. A pragmatic framework for understanding clinical decision support. Journal of Healthcare Information Management, 13(2), (1999), 5–21. 5

[13] M.A. Musen, Stanford Medical Informatics: uncommon research, common goals. MD Comput, 16(1), (1999). 5

[14] A. DiCaterino, K. Larsen, M.H. Tang, W.L. Wang. An Introduction to Workflow Management Systems, (1997). 7

[15] M. K. El-Najdawi, Anthony C. Stylianou, Expert support systems: integrating AI technologies, Commun. ACM 36(12) (1993) 55–ff. 5, 26

[16] D. J. Power, Decision Support Systems: Concepts and Resources for Managers, Westport, CT Greenwood/Quorum, 2002. 5

[17] 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 1991. New York: McGraw Hill Inc. 1991, pp. 3–7. 5

[18] B. G. Buchanan and E. H. Shortliffe, Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, AAAI, 1984. 5

[19] Guy L. Steele, Common Lisp the Language, 2nd Edition, Digital Press, 1990.

[20] 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 (1981), 876-881. 6

[21] 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 2 (2008), 206–210. 6

[22] 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. (2000), 300–304. 6

[23] 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 3(6) (1996), 367–388. 6

[24] 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. 6

[25] R. Boulme, D. Gonzalez and JC. Schmit, Storing genotypic resistance data and linking to other clinical information, XV International AIDS Conference, (2004) Bangkok, Thailand. 6

[26] D. Hollinsworth, The Workflow Reference Model, Tech Rep TC00-Workflow Management Coalition, 1994. 3

[27] 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 34 (2006). 7

[28] 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 8 (2007). 8

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

[30] 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, no. 1, pp. 86-112, 2005. 31

[31] 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. 31

[32] 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. 33

[33] 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. 33

[34] J.C. Whisstock, A.M. Lesk. "Prediction of protein function from protein sequence and structure". *Quarterly Reviews of Biophysics*, Cambridge University Press, vol. 36, no. 3, pp. 307-340, 2003. 33

[35] 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 33

[36] "CASP: Critical Assessment of Techniques for Protein Structure Prediction". *http://predictioncenter.org/index.cgi* 2, 32

[37] S. Srinivasan, D. Kumar, V. Jaglan, Agents and their knowledge representations Journal of Ubiquitous Computing and Communication, 5(1), (2010). 12

[38] E. Gat, Integrating planning and reacting in a heterogeneous asynchronous architecture for mobile robots, SIGART Bulletin, 2, (1991), 70–74. 12

[39] R. Brooks, A robust layered control system for a mobile robot. Robotics and Automation, IEEE Journal of, 2(1), (1986), 14-23. 11

[40] M.P. Georgeff, A.L. Lansky, A system for reasoning in dynamic domains: Fault diagnosis on the space shuttle. Technical Note 375, Artificial Intelligence Center, SRI International, (1986). 12

[41] B. Hayes-Roth, Dynamic Control Planning in Adaptive Intelligent Systems, Proceedings of the DARPA Knowledge-Based Planning Workshop, (1987). 12

[42] S. Vere, T. Bickmore, A basic agent. Computational Intelligence, 6, (1990), 41–60. 12

[43] J. Flp, Introduction to Decision Making Methods, Workshop on Biodiversity & Ecosystem Informatics,(2005). 15

[44] D. Baker, D. Bridges, R. Hunter, G. Johnson, J. Krupa, J. Murphy, K. Sorenson, Guidebook to Decision-Making Methods, WSRC-IM-2002-00002, Department of Energy, USA, (2002). 16, 17

[45] B. Shneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, (2nd edition) Reading, MA: Addison-Wesley, (1992).

[46] B. Shneiderman, B. Johnson, Treemaps: a space-filling approach to the visualization of hierarchical information structures, Proc. of the 2nd International IEEE Visualization Conference, San Diego, (1991), 284–291. 20

[47] D. Georgakopoulos, M. Hornick, A. Sheth, An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure Distributed and Parallel Databases, 3, (1995), 119–153. 14

[48] Yeung, Ka, Medvedovic, Mario, Bumgarner, Roger, Clustering gene-expression data with repeated measurements, Genome Biology, vol. 4, 2003. 34, 47

[49] Eisen MB, Spellman PT, Brown PO, Botstein D Cluster analysis and display of genome-wide expression patterns. Proc Natl Acad Sci USA 1998 , 95:14863–14868 34, 47

[50] Dhaeseleer, P., Wen, X., Fuhrman, S., Somogyi, R., 1999. Linear modeling of mRNA expression levels during CNS development and injury. In: Proceeding of the Pacific Symposium on Biocomputing, pp. 4152. 34, 47, 55

[51] Dana Pe'er and Aviv Regev and Gal Elidan and Nir Friedman, Inferring Subnetworks from Perturbed Expression Profiles, Bioinformatics, Vol.17 Suppl. 1 2001. 34

[52] Wang, Y., Joshi, T., Zhang, X.S., Xu, D., Chen, L., 2006. Inferring gene regulatory networks from multiple microarray datasets. Bioinformatics 22 (19), 2413-2420. 34, 47

[53] B. Johnson, B. Shneiderman, Tree-maps: a space-filling approach to the visualization of hierarchical information structures, Proceedings of IEEE Conference on Visualization (1991), 284–291.

[54] K. Chen, N. Rajewsky, "The evolution of gene regulation by transcription factors and microRNAs", *Nat Rev Genet*, Vol. 8, no. 2, pp. 93–103, 2007. 44

[55] ates, S.Microarray Experiments, Connexions Web site. http://cnx.org/content/m11050/2.17/, Oct 1, 2007. 45

[56] Bartlett John M., Stirling David, A Short History of the Polymerase Chain Reaction, Methods in Molecular Biology, vol. 226, pp. 3–6, 2003. 45

[57] Berger SL, Wallace DM, Puskas RS, Eschenfeldt WH. (1983). Reverse transcriptase and its associated ribonuclease H: interplay of two enzyme activities controls the yield of single-stranded complementary deoxyribonucleic acid. Biochemistry, 22(10):2365–72. 46

[58] Y. Huang, I. Tienda-Luna, Y. Wang, "Reverse engineering gene regulatory networks", *IEEE Signal Processing Magazine*, Vol. 26, no. 1,pp. 76–97, 2009. 47

[59] 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. 47

[60] 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, no. 1, pp 86–103, 2009. 47, 51

[61] 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, no. 1, pp. 71–79, 2005. 47

[62] 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, no. 3, pp. 228–235, 2003. 47

[63] 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, no. 2, pp. 165–181, 2006. 47

[64] 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, no. 4, pp. 814–834, 2006. 47

[65] 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 (Suppl. 1), S7, 2006. 47

[66] 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. 47

[67] 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, no. 3. pp. 377-383, 2005. 47

[68] van Berlo, J.P., van Someren, E.P., Reinders, M.J. (2003) Studying the Conditions for Learning Dynamic Bayesian Networks to Discover Genetic Regulatory Networks. *Simulation* **79**(12), 689-702. 47, 51

[69] Schfer J, Strimmer K: A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Statist Appl Genet Mol Biol 2005, 4:32. 47

[70] Faith *et al.* (2007) Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles. *PLoS Biol*, **5** doi:10.1371/journal.pbio.0050008. 47

[71] Opgen-Rhein, R., Strimmer, K., 2007. From-correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. BMC Syst. Biol. 1, 37. 50

[72] Wichert, S., Fokianos, K., and Strimmer, K. (2004) Identifying periodically expressed transcripts in microarray time series data. *Bioinformatics* **20** 5–20. 34, 57

[73] Ahdesmaki, M., Lahdesmaki, H., Pearson, R., Huttunen, H., and Yli-Harja O. (2005) Robust detection of periodic time series measured from biological systems. *BMC Bioinformatics* **6**: 117. 57

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

[75] 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. 58

[76] T. Kohonen, Self-Organizing Maps, Third extended edition, Springer, 2001. 58

[77] 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. 58

[78] http://graphviz.org/ 47

[79] P. Shannon P et al., Cytoscape: a software environment for integrated models of biomolecular interaction networks, Genome Research 13(11) (2003). 47

[80] Beirlant J, Dudewicz E, Gyorfi L, van der Meulen E: Nonparametric entropy estimation: An overview. Int J Math Stat Sci 1997, 6(1):17-39. 48

[81] Cover TM, Thomas JA: Elements of Information Theory. New York: John Wiley & Sons; 1991. 48

[82] Butte AJ, Kohane IS (2000) Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements. Pac Symp Biocomput: 418429. 49

[83] 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, Issue 1, pp. 65–73, ISSN 1097-2765, DOI: 10.1016/S1097-2765(00)80114-8. 52

[84] Cover, T., Hart, P., "Nearest neighbor pattern classification", *IEEE Transactions on Information Theory*, vol.13, no.1, pp. 21–27, 1967. 55

[85] Dudani, S.A. "The distance-weighted k-nearest-neighbor rule", *IEEE Trans. Syst. Man Cybern.*, SMC-6:325-327, 1976. 55

[86] http://dip.doe-mbi.ucla.edu/ 66, 82

[87] http://www.helmholtz-muenchen.de/en/mips/ 66

[88] Y. Ho, A. Gruhler, A. Heilbut, G.D. Bader, L. Moore, S.L. Adams, A. Millar, P. Taylor, K. Bennett, K. Boutilier, Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry, Nature, 415, (2002), 180-183. 66

[89] D. Eisenberg, E. M. Marcotte, I. Xenarios, T. O. Yeates Protein function in the post-genomic era Nature, 405, (2000). 65

[90] M. M. Maslon, T. R. Hupp, Drug discovery and mutant p53, Trends in Cell Biology, 20(9), (2010), 542-555. 66

[91] D. Ucar, S. Parthasarathy, S. Asur, C. Wang, Effective Pre-Processing Strategies for Functional Clustering of a Protein-Protein Interactions Network. BIBE, (2005), 129–136. 66

[92] G. Sabidussi, The centrality index of a graph, Psychometrika, 31(4), (1966), 581-603. 66

[93] L. C. Freeman, A set of measures of centrality based on betweenness, Sociometry, 40, (1977), 35-41. 66

[94] M. A. Bayir, T. D. Guney, T. Can, Integration of topological measures for eliminating non-specific interactions in protein interaction networks. Discrete Applied Mathematics, 157, (2009), 2416–2424. 66, 78

[95] H. Yu, A. Paccanaro, V. Trifonov, M. Gerstein, Predicting interactions in protein networks by completing defective cliques, Bioinformatics, 22(7), (2006), 823–829. 66, 74

[96] G.D. Bader, C.W. Hogue, An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics. 4(1) ,(2003). 68

[97] A.D. King, N. Przulj, I. Jurisica, Protein complex prediction via cost-based clustering. Bioinformatics, 20(17), (2004), 3013-3020. 68

[98] S. Van Dongen, Graph clustering by flow simulation, Ph.D. thesis, Centers for Mathematics and Computer Science (CWI), University of Utrecht, (2000). 68

[99] T. Aittokallio , B. Schwikowski, Graph-based methods for analysing networks in cell biology. Briefings in Bioinformatics, 7(3), (2006), 243–255. 68

[100] J. A. Papin, T. Hunter, B.O. Palsson, S. Subramaniam, Reconstruction of cellular signalling networks and analysis of their properties. Nat. Rev. Mol. Cell. Biol., 6, (2005), 99–111. 68

[101] H.N. Chua, K. Ning, W.K. Sung, H.W. Leong, L. Wong, Using Indirect Protein-Protein Interactions for Protein Complex Prediction Journal of Bioinformatics and Computational Biology, 6(3), (2008) , 435–466. 78

[102] L. Gao, P.G. Sun, J. Song, Clustering algorithms for detecting functional modules in protein interaction networks, Journal of Bioinformatics and Computational Biology, 7(1), (2009), 217-242. 78

[103] S. Asur, D. Ucar, S. Parthasarathy, An ensemble framework for clustering protein-protein interaction networks Bioinformatics, 23, (2007), 29-40.

[104] V. Arnau, S. Mars, I. Martn, Iterative cluster analysis of protein interaction data, Bioinformatics, 21(3), (2004), 364-378. 71

[105] B.L. Drees, B. Sundin et al. A protein interaction map for cell polarity developmen, Journal of Cellular Biology, 154, (2001), 549-571. 71

[106] A.L. Barabsi, Z.N. Oltvai, Network biology: understanding the cell's functional organization, Nature Reviews Genetics, 5, (2004), 101–113. 73

[107] C. von Mering et al. Comparative assessment of large-scale data sets of protein-protein interactions, Nature, 417, (2002), 399–403. 73

[108] P. Legrain, How Useful Will Functional Proteomics Data Be? Comp Funct Genomics, 2(5), (2001), 301-303. 73

[109] J. Chen, W. Hsu, M. L. Lee, S. Ng, Increasing confidence of protein interactomes using network topological metrics, Bioinformatics, 22(16), (2006), 1998–2004. 73

[110] S. Brohe, J. van Helden, Evaluation of clustering algorithms for protein-protein interaction networks, BMC Bioinformatics, 7(488), (2006). 78

[111] P. Shannon P et al., Cytoscape: a software environment for integrated models of biomolecular interaction networks, Genome Research, 13(11), (2003). 80

[112] The Protege Ontology Editor and Knowledge Acquisition System, *http://protege.stanford.edu.* 86

[113] 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. International Journal of Human-Computer Studies, 58:89123, 2003. 86

[114] Sandia National Laboratories, "Jess: The rule engine for the JavaTM platform", Available at http://herzberg.ca.sandia.gov/jess/, 2003. 85

[115] C. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", *Artificial Intelligence*, vol. 19, pp 17-37, 1982 85

[116] L. Brownston, R. Farrell, E. Kant, N. Martin, Programming Expert Systems in OPS5, Addison-Wesley, 1985. 86

[117] B.Clayton, "ART Programming Tutorial", Vol 1–3 Department of Artificial Intelligence, University of Edinburgh. 86

[118] J. C. Giarratano, G. D. Riley, Expert Systems: Principles and Programming, Third Edition, Course Technology, 1998. 86

[119] SHriMP research group, Jambalaya - Information Browser for Protege, Department of Computer Science University of Victoria. 87

[120] OntoViz: a visual browser for ontologies, http://ontoviz.sourceforge.net/. 87

[121] Henrik Eriksson. Using JessTab to Integrate Protg and Jess. IEEE Intelligent Systems, 18(2):43-50, 2003. 87

[122] *http://www.jgraph.com/* 87

[123] *http://www.eclipse.org/* 88