UNIVERSITÀ DEGLI STUDI DI PALERMO

DOTTORATO DI RICERCA IN INGEGNERIA INFORMATICA

DIPARTIMENTO DI INGEGNERIA INFORMATICA

DEVELOPMENT OF A DECISION SUPPORT SYSTEM FOR BIOINFORMATICS. EXTRACTION OF PROTEIN COMPLEXES FROM A PROTEIN-PROTEIN INTERACTION NETWORK: A CASE STUDY.

Tutor
Ch.mo Prof. Salvatore Gaglio

Candidato
Ing. Antonino Fiannaca

Coordinatore
Ch.mo Prof. Salvatore Gaglio

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

Development of a Decision Support System for Bioinformatics. Extraction of Protein Complexes from a Protein-Protein Interaction Network: a case study.

Antonino Fiannaca

Department of Computer Science (DINFO)

University of Palermo

A thesis submitted for the degree of

PhilosophiæDoctor (PhD)

February 15, 2011

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 the thesis, 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.

A case study about extraction of protein complexes from protein-protein interaction networks is presented, in order to show how the system faces a problem and how it interacts with the user.

To my father who, when I was a child, told me:
"*It doesn't matter what your job will be;*
to be truly happy, you have to love your job!"
Beloved dad, I think I love this work so much.

Acknowledgements

My sincere thanks to all the people who have contributed to this work during the last three years.

My special thanks to:
Prof. S. Gaglio, because he believed in me since I started my thesis of academic degree, in 2005;
Dr. Urso and Dr. Rizzo, for bearing me and bearing with me, during my PhD work;
Prof. G. Di Fatta, who inclined me towards the world of scientific research.

Contents

List of Figures

List of Tables

Glossary

PPI	Protein-Protein Interaction is the interactions between two proteins that bind for a biological functions.
DSS	Decision Support System is a computer-based information system that supports business or organizational decision-making activities.
WFMS	Workflow Managment System is a computer system that allow organizations to define and control the various activities associated with a business process.
GUI	Graphic User Interface is a type of user interface that allows users to interact with electronic devices with images rather than text commands.
BORIS	Bioinformatics Organized Resources - an Intelligent System. It is the project belonging to the research work order "Analisi intelligente dei dati per la bioinformatica" of the ICAR-CNR.
MRL	Meta-Reasoning Level. The proposed decision making process is arranged in a stack of meta-reasoning, according to the entity of a problem.

1

Introduction

In recent years, the science of bioinformatics is increasingly being used to improve the quality of life as we know it. Bioinformatics has developed out of the need to understand the code of life, DNA, that is the basic molecule of life and directly controls the fundamental biology of life. DNA codes for genes which, in turn, code for proteins which determine the biological makeup of humans or any living organism.

The ultimate goal of bioinformatics is to uncover the wealth of biological information hidden in the continuous growing amount of sequence, structure, literature and other biological data, such as proteomic sequences and structures or protein-protein interaction data, in order to obtain a clearer insight into the fundamental biology of organisms. For this reasons, continued development of new databases, new methodologies and analytical tools is critical for the health-related quality of life.

1.1 Motivation and Goals

Researchers in bioinformatics aim at applying well established artificial intelligence approaches and machine learning algorithms to biological issues, in order to discover and explain biological phenomena *in silico*, rather than *in vitro*, helping this way the experimentalists in their activities.

Up to the present, several methodologies have been developed for each biological issue, and a lot of tools or online services have been proposed for implementing these methodologies. For instance, the prediction of the three-dimensional structure of a protein from its amminoacid sequence has been faced with more than 70 softwares citecasp. Obviously each technique has its proper indications, advantages, and disadvantages.

Another approach in bioinformatics aims to develop decision support systems that can help the experimentalists to find, among all the available techniques, the better solution to solve a biological problem; this way users can handle the growing amount of available data in a simple way.

This PhD thesis introduces a new hybrid architecture that integrates not only a decision support system for bioinformatics, but also a workflow management system that is able to set and run a set of algorithms suggested by the system, assisting the user during the whole biological process, from input data to final results.
The proposed architecture exploits three different perspective:

1. It shows how to achieve a specific result for an input problem at different abstraction layer, dealing with the direct execution of each task and subtask.

2. It decides about what to do with the knowledge of the system, using strategies and heuristics in the knowledge base to generate some consistent models.

3. It allows reconfiguration of each selected tool or service, selecting alternative paths or restart the workflow from a process selected by the user and tracing step by step, the workflow evolution of the system.

The basic idea of the 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. This way, the user can see behind the workflow of operation what is the conceptual scheme at the basis of the simple succession of tasks.

The proposed system is integrated in a project belonging to the research work order “Analisi intelligente dei dati per la bioinformatica” of the CNR, called

"Bioinformatics Organized Resources - an Intelligent System" (BORIS). This project aims to help biological researchers that are not bioinformatician and, consequently, are not able to handle available bioinformatics tools. The basic idea is to separate the user from the details of the tools or the on-line services used in research work, in analysis of biological data and to build a cognitive path that takes the user from raw data to knowledge and helps him to navigate this path.

1.2 Background

The system proposed in this Ph.D. 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) (33) 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 (7), 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 (8).

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 (9) 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 (10). 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 (22). KDSS are composed by three components (23):

- 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) (24), 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 (11).

MYCIN (25) 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” (26). 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 (27), 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 (28), 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 (29), implementing guidelines for hypertension using Stanford Medical Informatics EON architecture (30); LISA (31) that is a clinical information system for supporting collaborative care in the management of children with Acute Lymphoblastic Leukaemia (ALL); TherapyEdge (32) 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 PhD work, have Some typical features (34):

- 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 (35), 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) (36), 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 (38); 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 adaptative.

1.3 Dissertation outline

The rest of the thesis will be organized as follows:

In Chapter 2 BORIS project the research work order related to this thesis, will be introduced. The proposed hybrid architecture for DSS is introduced in Chapter 3. Chapter 4 contains an application scenario related to the protein complex extraction in PPI Networks. In Chapter 5 a brief explanation of all tools and methods used in order to develop the proposed system will be provided. Appendix A reports some information about the BORIS user interface.

1.4 Publications

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

- FIANNACA A, LA ROSA M., PERI D, RIZZO R (2011). An Intelligent System for decision Support in Bioinformatics. ERCIM NEWS, vol. 84.
- FIANNACA A, GAGLIO S, LA ROSA M., PERI D, RIZZO R, URSO A (2010). A Proposed Knowledge Based Approach for Solving Proteomics Issues. LECTURE NOTES IN COMPUTER SCIENCE, vol. LNCS 6160; p. 304-318, ISSN: 0302-9743, doi: 10.1007/978-3-642-14571-1
- FIANNACA A, GAGLIO S, LA ROSA M., PERI D, RIZZO R, URSO A (2010). A Knowledge Based Decision Support System for Bioinformatics and System Biology. In: Proceedings of CIBB 2010. Palermo, Italy, 16-18 September 2010, ISBN/ISSN: 978-88-95272-87-0
- FIANNACA A, GAGLIO S, LA ROSA M., PERI D, RIZZO R, URSO A (2009). A Proposed Knowledge Based Approach for Solving Proteomics Issues. In: Proceedings of CIBB 2009. Genova, Italy, 15-17 Ottobre 2009, vol. 3, ISBN/ISSN: 978-88-903537-2-7
- FIANNACA A, GAGLIO S, LA ROSA M., PERI D, RIZZO R, URSO A (2009). A Decision Support System for Reverse Engineering Gene Regulatory Networks. In: Proceedings of SysBioHealth Symposium 2009. Milano, Italy, November, 25-27 2009, p. 81-83.
- FIANNACA A., DI FATTA G, RIZZO R, URSO A, GAGLIO S (2009). Clustering Quality and Topology Preservation in Fast Learning SOMs. NEURAL NETWORK WORLD, vol. 19; p. 625-639, ISSN: 1210-0552
- FIANNACA A., DI FATTA G, URSO A, RIZZO R, GAGLIO S (2009). A New Linear Initialization in SOM for Biomolecular Data. Computational Intelligence Methods for Bioinformatics and Biostatistics, vol. 5488, p. 177-187, ISBN/ISSN: 978-3-642-02503-7

- FIANNACA A., DI FATTA G, GAGLIO S, RIZZO R, URSO A (2008). Clustering Quality and Topology Preservation in Fast Learning SOMs. 18th International Conference on Artificial Neural Networks. Prague, Czech Republic, 3-6/09/2008, BERLIN: Springer, vol. LNCS 5163, p. 583-592, ISBN/ISSN: 978-3-540-87535-2, doi: 10.1007 / 978-3-540-87536-9

- FIANNACA A., DI FATTA G, URSO A, RIZZO R, GAGLIO S (2008). A New Linear Initialization in SOM for Biomolecular Data. Proceedings of the CIBB 2008. Vietri sul Mare, Salerno (Italy), 3-4 October 2008, SALERNO: Masulli, vol. 2, ISBN/ISSN: 978-88-903537-1-0

- FIANNACA A., RIZZO R, URSO A, GAGLIO S (2008). A New SOM Initialization Algorithm for Nonvectorial Data. Knowledge-Based Intelligent Information and Engineering Systems. Zagreb, Croatia, 3-5/09/2008, HEIDELBERG: Springer, vol. LNAI 5177, p. 41-48, ISBN/ISSN: 978-3-540-85562-0

2

Bioinformatics Organized Resources - an Intelligent System

This Chapter introduces the BORIS project, acronym for “Bioinformatics Organized Resources - an Intelligent System”, that is the framework which this PhD work has been developed for. It is a project belonging to the research group “Analisi intelligente dei dati per la bioinformatica”, related to the project “Bioinformatica” of the CNR. The BORIS project manager is Dr. A. Urso, CNR researcher at ICAR-CNR of Palermo.

2.1 BORIS Project

The aim of BORIS project is to provide an helpful and effective support to researchers or experimentalists, that have to solve problems in the field of system biology, such as the prediction of protein structures, the design of gene regulatory network, the extraction of protein complexes, and so on.

Since several biological researchers are not bioinformatician, that is they are not able to handle available bioinformatics tools, the basic idea of BORIS project is to separate the user from the details of the tools or the on-line services used in research work, in analysis of biological data and to build a cognitive path that takes the user from raw data to knowledge and helps him to navigate this path.

Moreover, the system have to provide to the users 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. Another feature of the BORIS project is the use of a knowledge, that is the heuristics and strategies that can be extracted from bioinformatics papers and experiments representing the expertise on the application domain. Finally, all the tools/services and heuristics/strategies must be easy-to-use, as well as each operation must be traceable, through a workflow with expertise that shows succession of tasks and the conceptual scheme at the basis of that workflow.

2.2 BORIS Guideline

Some guidelines of the whole project have been provided in order to define the features of the system.

In the following there is a list of technical specifications that have to be implemented in BORIS project development.

- The system is based on a decision support system, with a rule-based expert system engine.
- The system application domain is the bioinformatics field, including all the -omics science.
- The system help the experimentalist suggesting, for a given problem, the right tool at the right moment.
- The system generate a navigable interactive workflow, that is visualized at different abstraction layers.
- The system free the user from implementation details and assists him in the correct configuration of algorithms/services, allow users to define different workflows for different types of jobs or processes.

These guideline indicates that the system not only have to propose the tools able to resolve a problem, but also the knowledge used in order to justify the choice of those specific tools and strategies. Since the system must be able to handle tools/services and to interface their inputs, outputs and parameters, and it must manages and defines a series of tasks within an organization to produce a final outcome, then BORIS can be seen as a crossover between classical decision support systems (DSS) and the most recent workflow management systems (WFMS).

2.3 BORIS software architecture

The software architecture of BORIS is arranged in a three layer structure. As shown in Figure 2.1, the architecture is inspired by its main goal: separate the researcher from the tools in order to let him focus on the problem.

Many times biology researchers do not have a precise idea of the flow of task that they need to reach a goal and just want to explore many available options; in the same way, they are not interested in algorithm details or in web services configurations. For this reason, this architecture sets these objects in the "Object Layer": the system decides how to use them in order to accomplish the users goal. The components of the Object Layer are not part of the system and can change in time, in fact an algorithm can be substituted by a more efficient one, a web service can be unavailable. Of course they can communicate with the system, according to technical specifications, by means of a protocol that ensure inputs/outputs of each tool/web service can interact with the rest of the system.

The Object Layer is accessed by the "Controller Layer" that is the system core. In this layer it is contained the "Reasoner" and the "Knowledge Base" of the system: the former decides which operations to perform on the basis of the available knowledge and the users request. The knowledge is organized by means of an ontology, that encapsulates all the informations, called facts, encoding the expertise of the system about a biological application domain. The main idea at the basis of the BORIS ontology is shown in Figure 2.2. It is composed by three main sub-tree:

- the **Domain**, that is the application field, the data we want to analyse.
- the **Tasks**, that represent what we can do according to the domain.

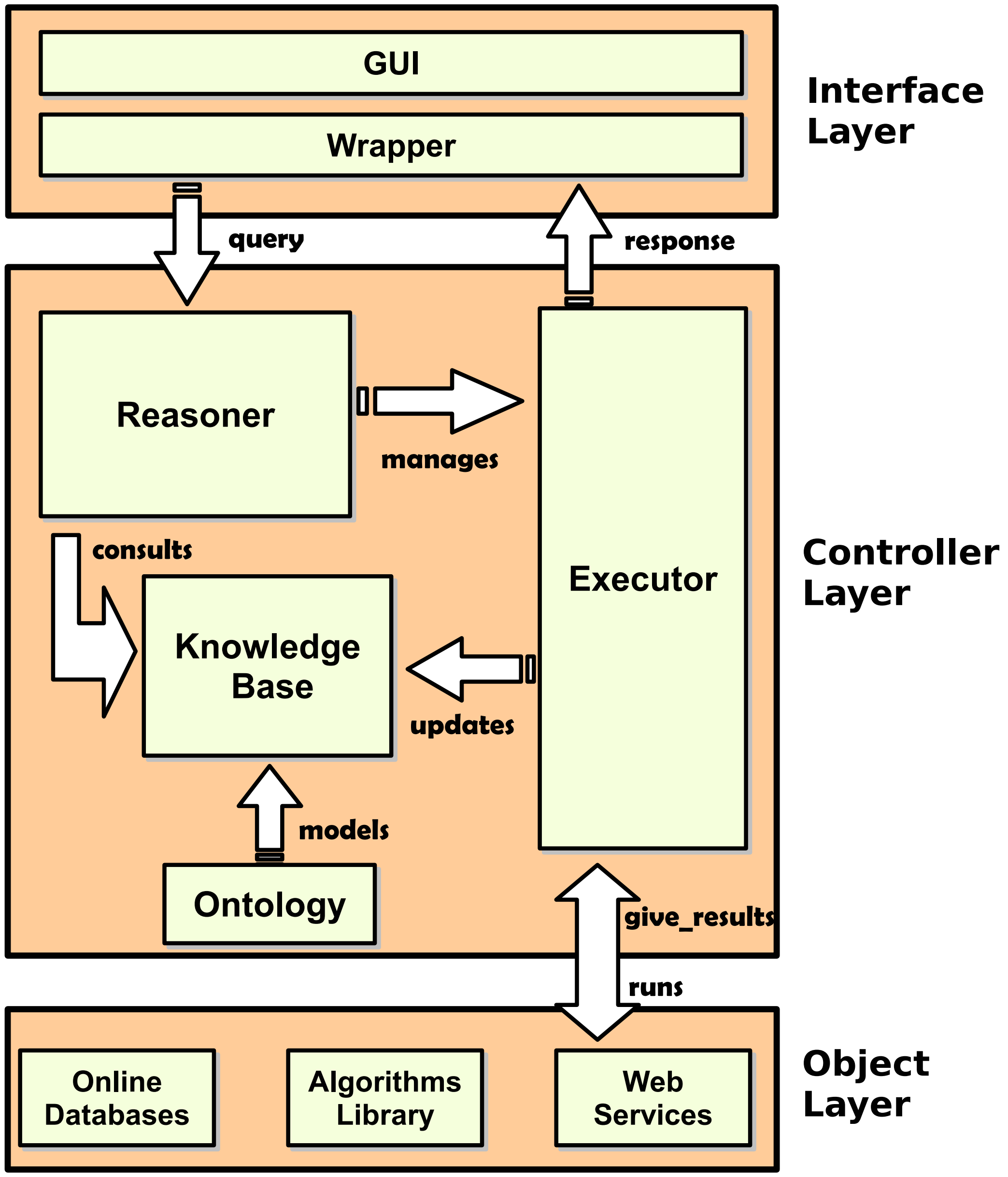


Figure 2.1: BORIS Software Architecture. It is composed by three layer: the interface, the controller and the object one.

- the **Tools**, i.e. what are the instruments, algorithms and services we can use to accomplish the tasks.

Using these definitions, the only relationships are then between elements of the Tasks class and the Domain on one hand; elements of the Tasks class and

Tools on the other one. Furthermore possible changes in one or more of the three main classes do not affect the rest of the ontology, since it is enough modifying or adding the relationships.

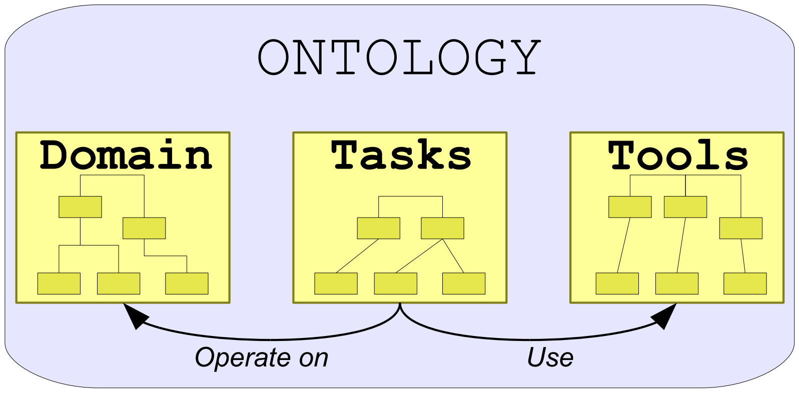


Figure 2.2: BORIS Ontology is composed by three sub-tree: Domain, Tasks and Tools.

The domain of BORIS ontology, according to bioinformatics topics classification in (16), is divided into four classes of problems:

1. ***Structure Prediction***:

   The structure of a protein represents a key feature in its functionality (17). 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, aiming at improving prediction techniques in the years, has been instituted;

2. ***Function Prediction***:

   The prediction of protein function is a challenge at the proteomics scale. Although 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 (19): 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.

3. ***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 (20), but they are time and cost consuming, thus a computational technique able to screen possible candidates for further analyses, appears a desirable solution.

4. ***Protein Annotation***:

   A correct organization of available databases and technical information on proteins form the raw material 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 (18), 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.

Facts are given a rigorous and organized structure by means of the ontology of concepts (41). Apart from the facts, the Knowledge Base also has a set of rules, in the typical form **IF** <*precondition*> **THEN** <*action*>. The rules, acting on facts, have to be considered as the coding for heuristics, guidelines and strategies adopted by an expert of the domain. Both facts and rules can be provided by one or more experts of the domain or can be extracted from bioinformatics papers, experimental papers and, in case, from domain experts. These rules also describe which are the conditions that should be satisfied in order to run a specific task or algorithm present in the Object Layer: in other words rules code the strategies and heuristics that the system can provide to the user.

In the Controller layer there is also the executor module that is the part of the Controller Layer that runs the tools in the Object Level according to the input data. The executor is controlled by the Reasoner and updates the Knowledge Base with the intermediate results, moreover it will send the final results to the user. The user looks at the system operation using the GUI and the wrapper 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 that is the last interface level.

The user interacts with the GUI that sends message to the wrapper, the wrapper formats this messages in the right way for the executor module, and sends query to the reasoner. This allows to easily change the GUI without interferences to the other parts of the system.

The inference mechanism related to a running process in Boris, is performed by a continuous loop reported in Fig. 2.3. As previously said, the engine of Boris is a rule-based expert system, that exploit the expertise for the problem solving process; for this reason, the Knowledge Base plays a fundamental role in the inference mechanism, together with the Executor. Inside the Executor there are two phases: Planning and Execution; both of them contains some processes of rule engine. In fact, Planning phase concerns the activation of a list of rules, that satisfy some criteria involved with Input phase, and takes care of conflicts resolution, in the event that more than a strategy could satisfy the same problem. Otherwise, Execution phase is responsible for rules firing, when an active rule is ready to run, and for tools activation, when an instance of a tool/service should be produced. During this phase, the Knowledge Base can be update, according to inference engine results. Finally the user can evaluate the result of input query in the Outputs phase, interacting with Boris and producing a new Input for the inference mechanism that, in turns, will generate a Knowledge Base update.

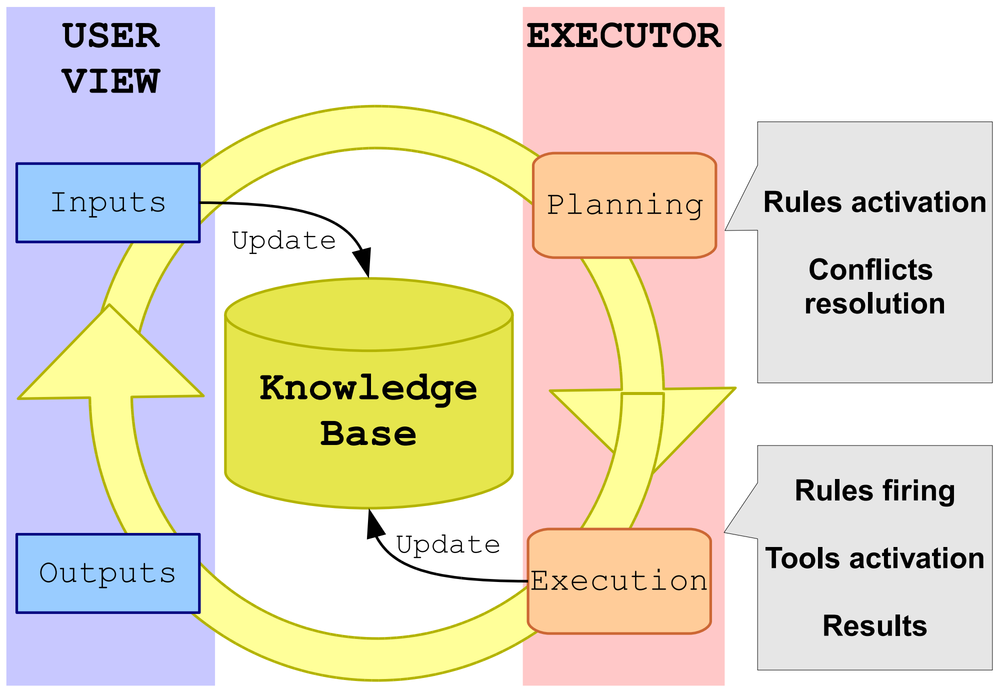


Figure 2.3: Boris Inference Mechanism. The expert system exploits a rule-based engine.

2.4 Contribution to BORIS

Management of the previous software architecture requires a solid Decision Support System that can help a bioinformatics researchers to deal with the plenty of tools and services currently available. Main guidelines of BORIS project are oriented to design and implementation of a system that should propose different strategies according to the selected experiment, the type of input data and other user's requirements; moreover it has to provide and run all the instruments necessary to execute the proposed strategies. In this context, BORIS system follows a new paradigm: it uses a declarative approach for deciding "what to do"; a procedural approach for realizing "how to do" something and a process approach to manage a workflow of operation.

The contribution of this Ph.D. thesis consists, according to guidelines of Boris Project, in doing an hybrid architecture (declarative/procedural/process) for the DSS, that is able to:

1. Easily handle the management of the strategies/heuristics, exploiting the rule-based engine of the system to provide a valid support to the user, also during the configuration of each available tool/service;

2. Provide an overview of software development along different abstraction layers, from the main goal to the implementation (object layer), in order to give to the user a decomposition in sub-systems and some different perspectives of the problem;

3. Trace a workflow of all the operations during the experiment, ensuring the navigability of all procedures, so that this workflow can be saved and eventually shared with other users.

This hybrid architecture will be introduced in next section. A case study will be discussed in section 4, where the problem of protein complex extraction from a protein-protein interaction network is faced.

Identification and extraction of protein complexes represents an hard task for machine learning algorithms (21), because uncertain information about interconnection and functionality of each protein could lead to erroneous interpretation. Moreover several tools have been developed in order to preprocessing a protein networks, as well as to extract protein complexes with a biological significance.

The proposed 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. In addiction, during the description of the experiment, the status of the system will be shown and the decision making process will be explained.

3

Architecture of Decision Support System

In this Chapter the major contribution of this PhD thesis work is presented. 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 (44) 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) (45), 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 (46), where the agent acts in a complex dynamic environment, having only an unstructurated and broken knowledge about this environment.
- Homer by (47) 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,

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.

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 (42). 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 (43), on the design of the ATLANTIS architecture for mobile robots. Based on the observation that an 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 PhD thesis, according to the software architecture of the BORIS project (see section 2.3), 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 (37), 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 PhD thesis 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.

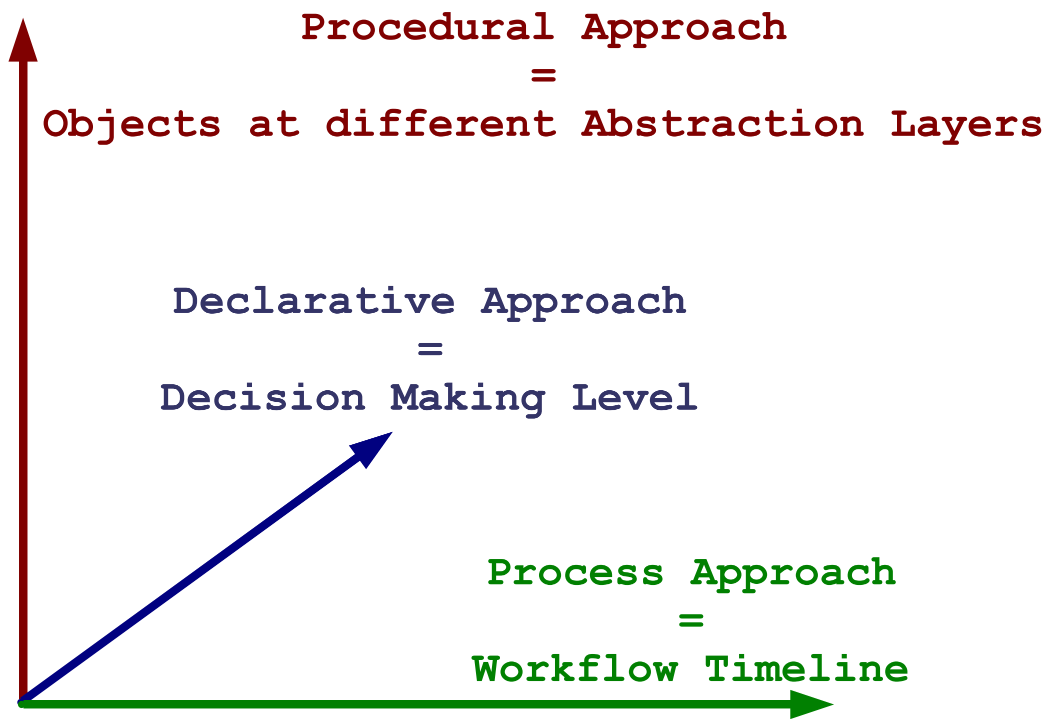


Figure 3.1: Space of Decision Support System. The hybrid architecture introduces tree point of view for the problem, i.e. abstraction layers ((based on Procedural Approach), decision making levels (based on Declarative Approach) and workflow timeline (based on Process Approach).

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 PhD work uses a procedural reasoning system that have to operate with BORIS software architecture described in section 2.3. 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 (49), "*Decision making is the study of identifying and choosing alternatives based on the values and preferences of the decision maker. Making a 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 (50), the decision making process used in this PhD 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 (50), 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

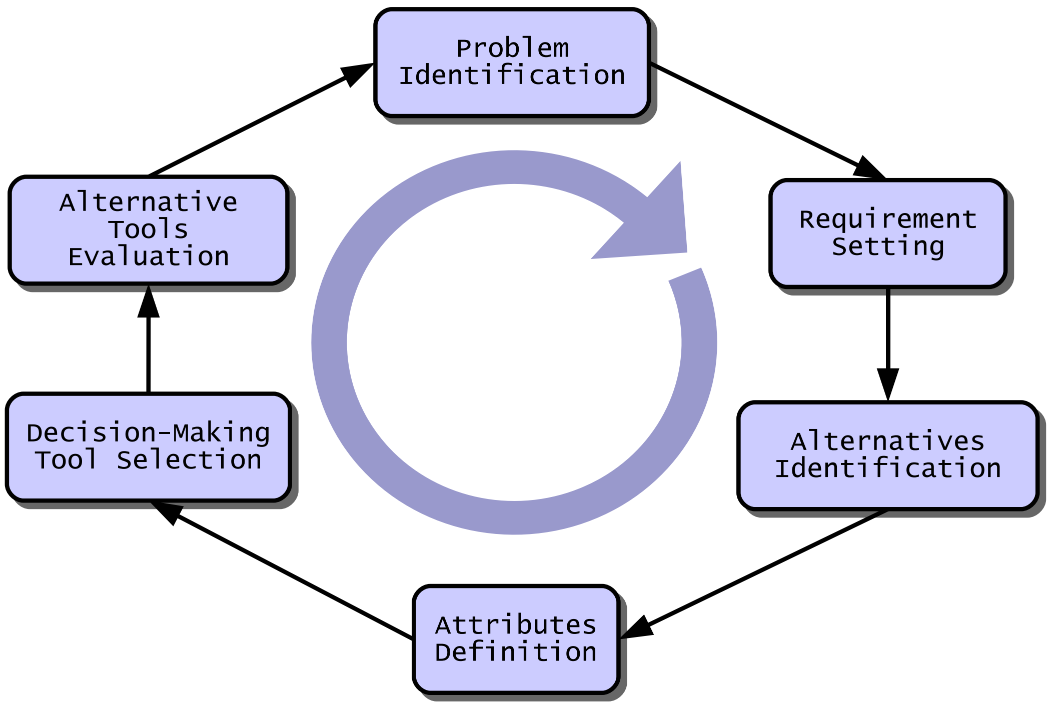

Problem Identification
Requirement Setting
Alternatives Identification
Attributes Definition
Decision-Making Tool Selection
Alternative Tools Evaluation

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

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.

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

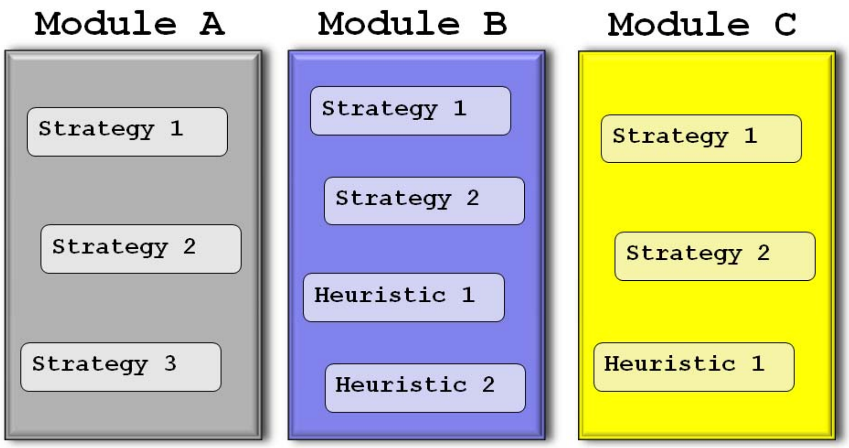


Figure 3.3: Decision-Making Modules. Each module contains all the strategies and/or heuristics for a well defined task. In addiction some modules are also responsible for the management of directives related to tools and services.

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 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 pro-

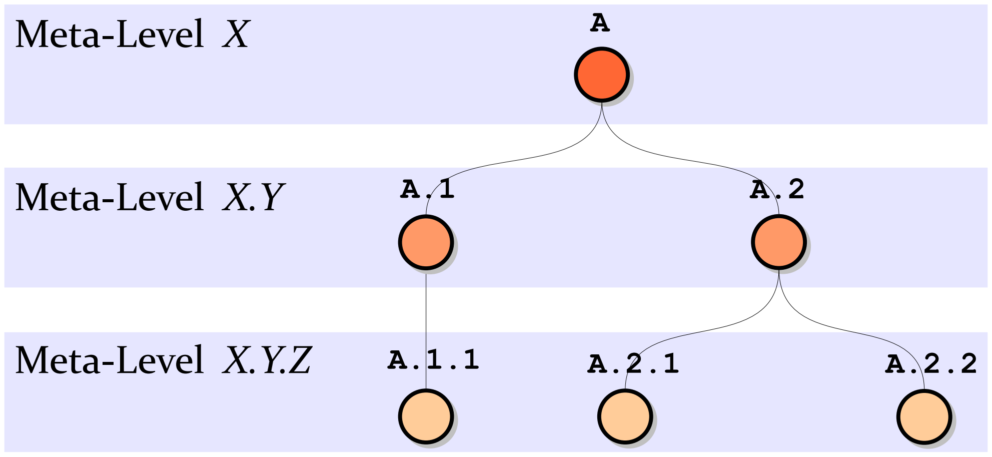


Figure 3.4: Meta reasoning tree. Decision making modules are distributed into some different meta-reasoning levels, according to problem/sub-problem hierarchy.

pose 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 introduced 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 (52) 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, used in this PhD work, 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 results.

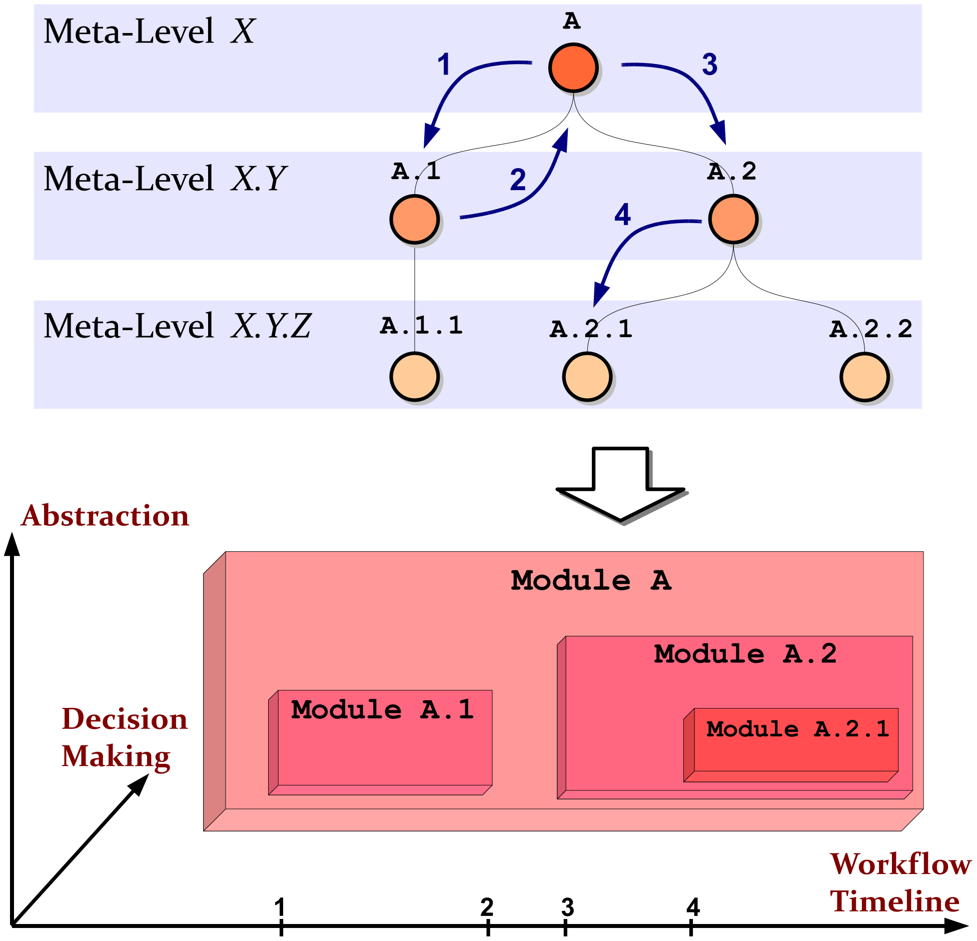


Figure 3.5: From meta reasoning tree to hybrid architecture space. Decision making modules are represented by means of a dynamic treemap.

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

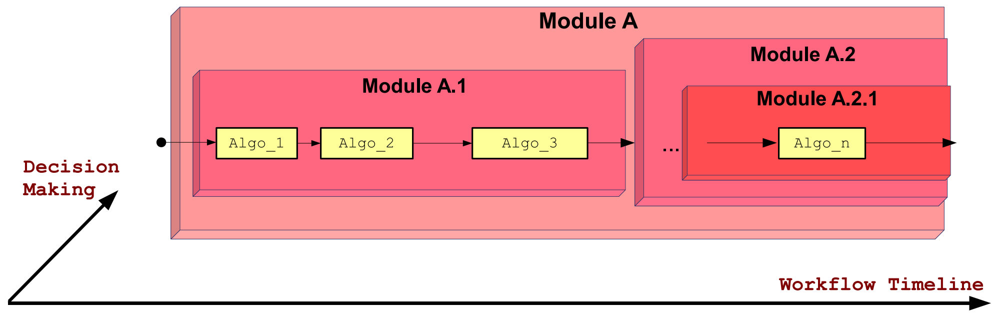


Figure 3.6: Representation of a simple sequence of algorithms. The workflow is projected into the 3D space; no information about abstraction layer is reported, because only the object layer of the system is considered.

`A.2.1`. Bounding boxes representing modules at different meta-reasoning levels 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 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

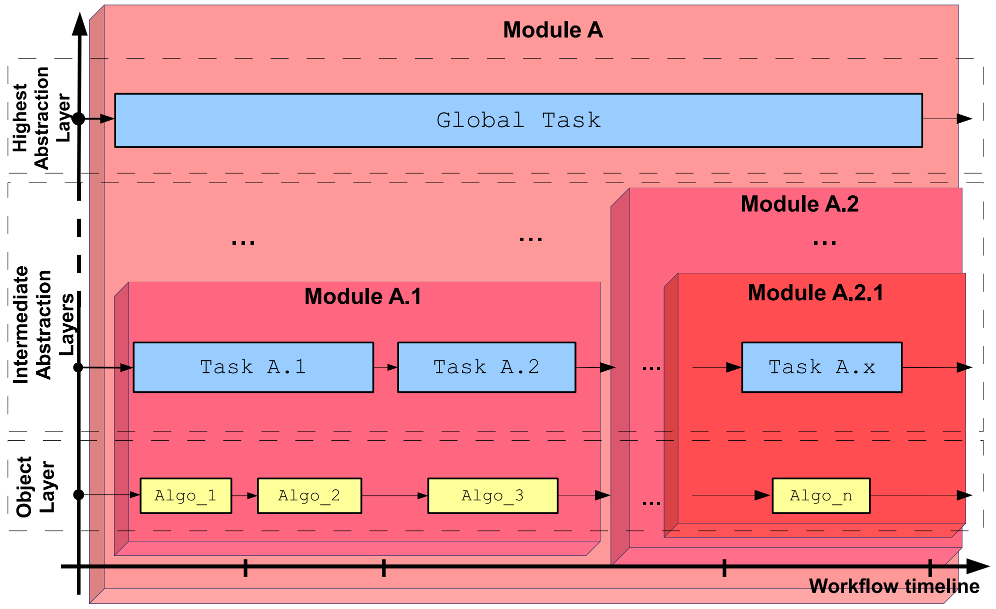


Figure 3.7: An overview of the DSS 3D space. There is a workflow for each abstraction layer, according to the user's point of view.

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

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.

4.1 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* (4), 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 4.1 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).

4.2 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 (1), MIPS (2), etc..), but unfortunately available datasets are still incomplete and contain non-specific (false positive) interactions (3), 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 (5), 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

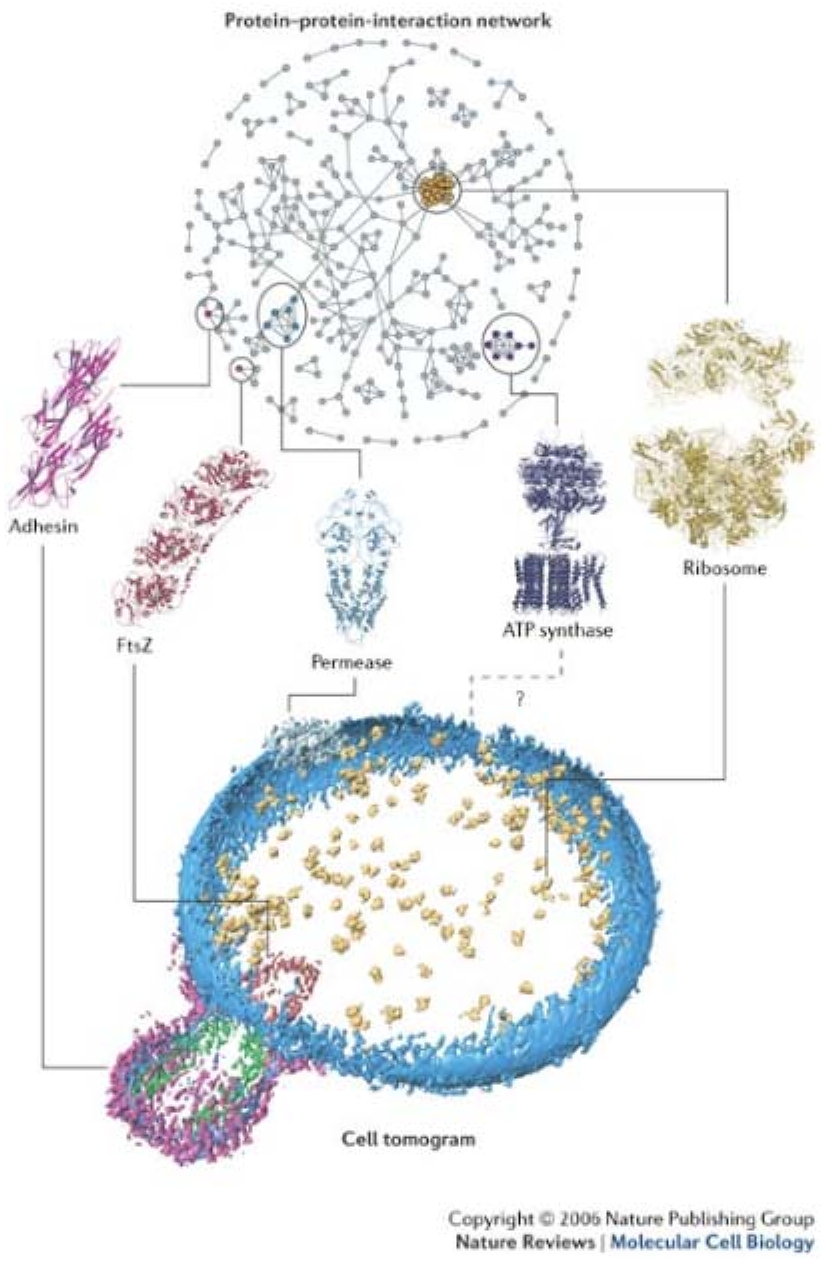

Protein–protein-interaction network
Adhesin
FtsZ
Permease
ATP synthase
Ribosome
?
Cell tomogram
Copyright © 2006 Nature Publishing Group
Nature Reviews | Molecular Cell Biology

Figure 4.1: 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).

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 (53) 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(54) and Centrality(55). Also (56) 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 (57), that attempts to detect 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 (58), 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 (59). 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)(60), 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.

4.2.1 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 (61), 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 (62). 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.

4.2.2 Algorithms and Tools for Complex Extraction

4.2.2.1 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.

4.2.2.2 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.

4.2.2.3 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.

4.2.2.4 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.

4.3 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 4.3. 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 (66, 67) with small scale experiments (*in vitro*) at biological

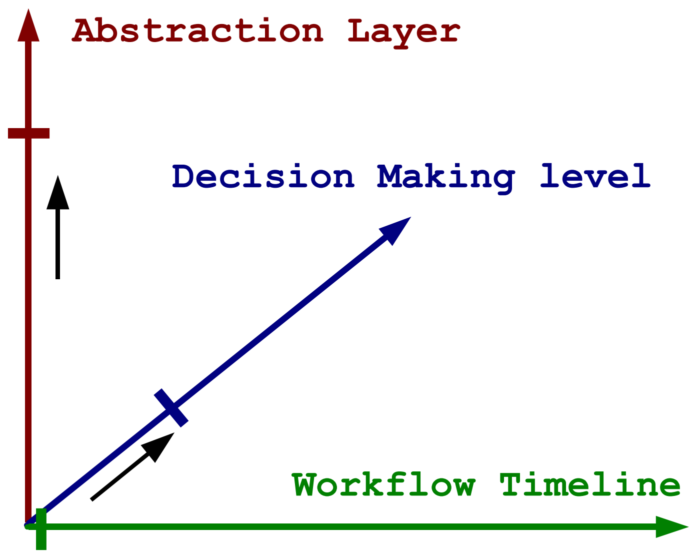


Figure 4.2: Projection of the system state over the hybrid architecture space at the first step of the protein complex extraction scenario.

interaction level. *DIP* also provides a subset of PPIs curated manually by experts, that are called core PPIs.

4.4 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 4.2. The transitions from start position (when the system state is at the point 0,0,0) to the current system state are highlighted in the figure 4.2 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 4.3.

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:

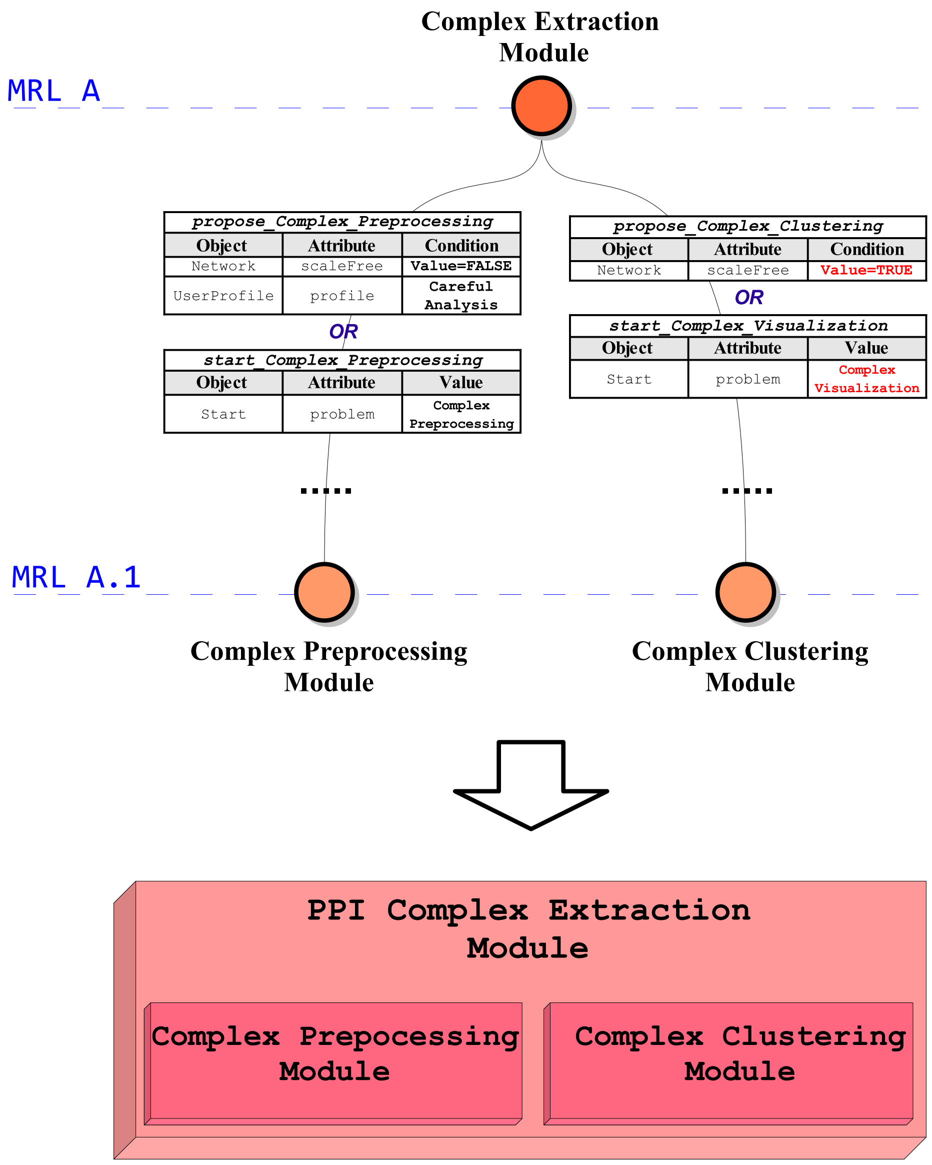
- `Complex Extraction`, the parent module that gives directives to two children modules at the bottom, that could be activate in order to deal with more specific tasks;
- `Complex Preprocessing`, the child module that contains the reasoning about strategies and tools able to face the PPI Network preprocessing 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 4.3 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 4.2, 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


Complex Extraction Module
MRL_A
OR
OR
.....
.....
MRL_A.1
Complex Preprocessing Module
Complex Clustering Module
PPI Complex Extraction Module
Complex Prepocessing Module
Complex Clustering Module

propose_Complex_Preprocessing

Object	Attribute	Condition
Network	scaleFree	Value=FALSE
UserProfile	profile	Careful Analysis

start_Complex_Preprocessing

Object	Attribute	Value
Start	problem	Complex Preprocessing

propose_Complex_Clustering

Object	Attribute	Condition
Network	scaleFree	Value=TRUE

start_Complex_Visualization

Object	Attribute	Value
Start	problem	Complex Visualization

Figure 4.3: Decision making modules responsible for the protein complex problem and related treemap representation. Some transitions for the activation of child modules are reported.

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(68)

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 (69, 70, 71), 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* algorithm, created by (57). 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 4.4. 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 4.2, 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

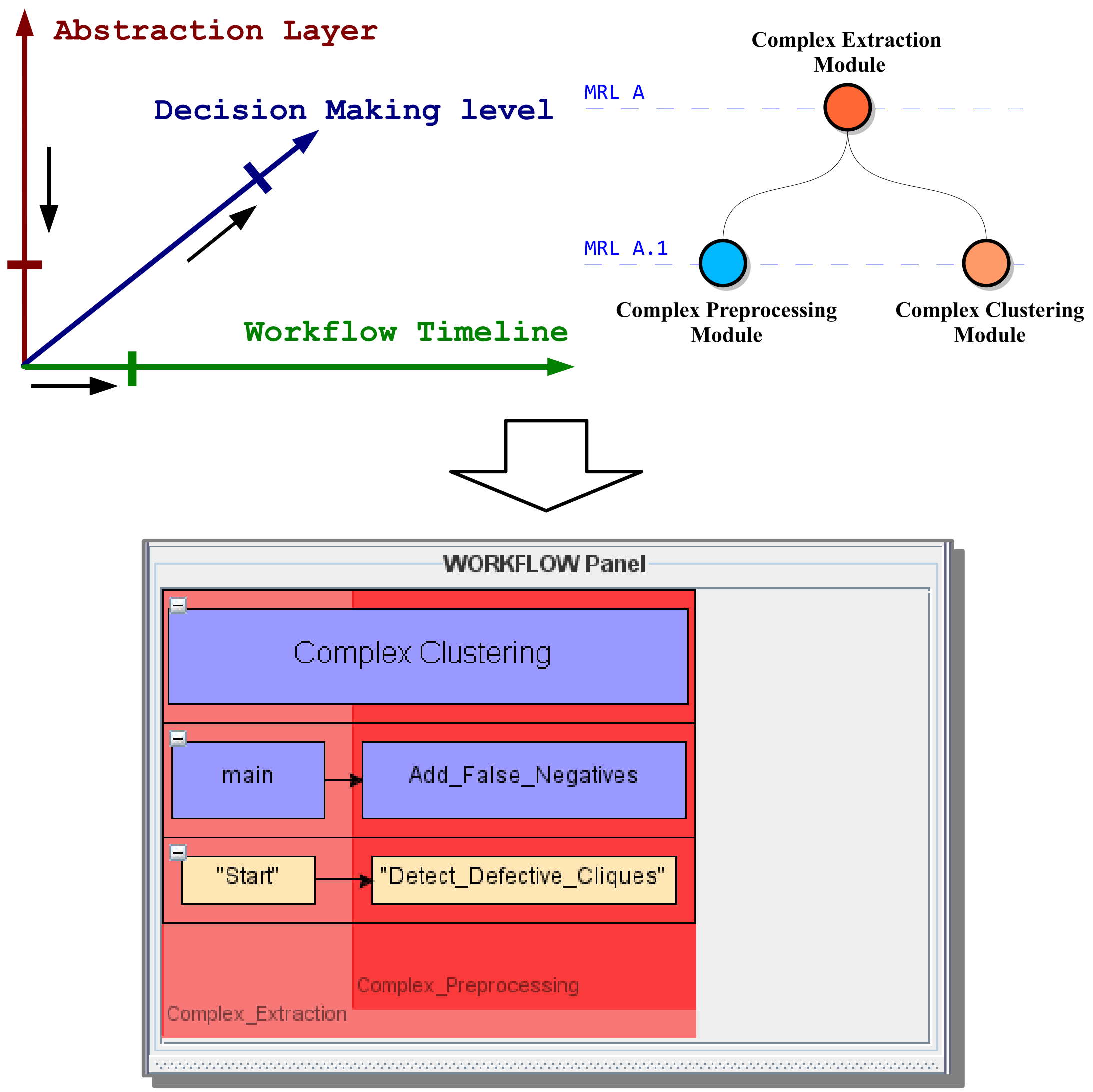


Figure 4.4: Case study at the preprocessing phase. The projection of the state of system over the tree axis is reported, the active module is highlighted and the multi-level workflow representing the system output is shown.

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 *Ap-*

pendix 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 4.5 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 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

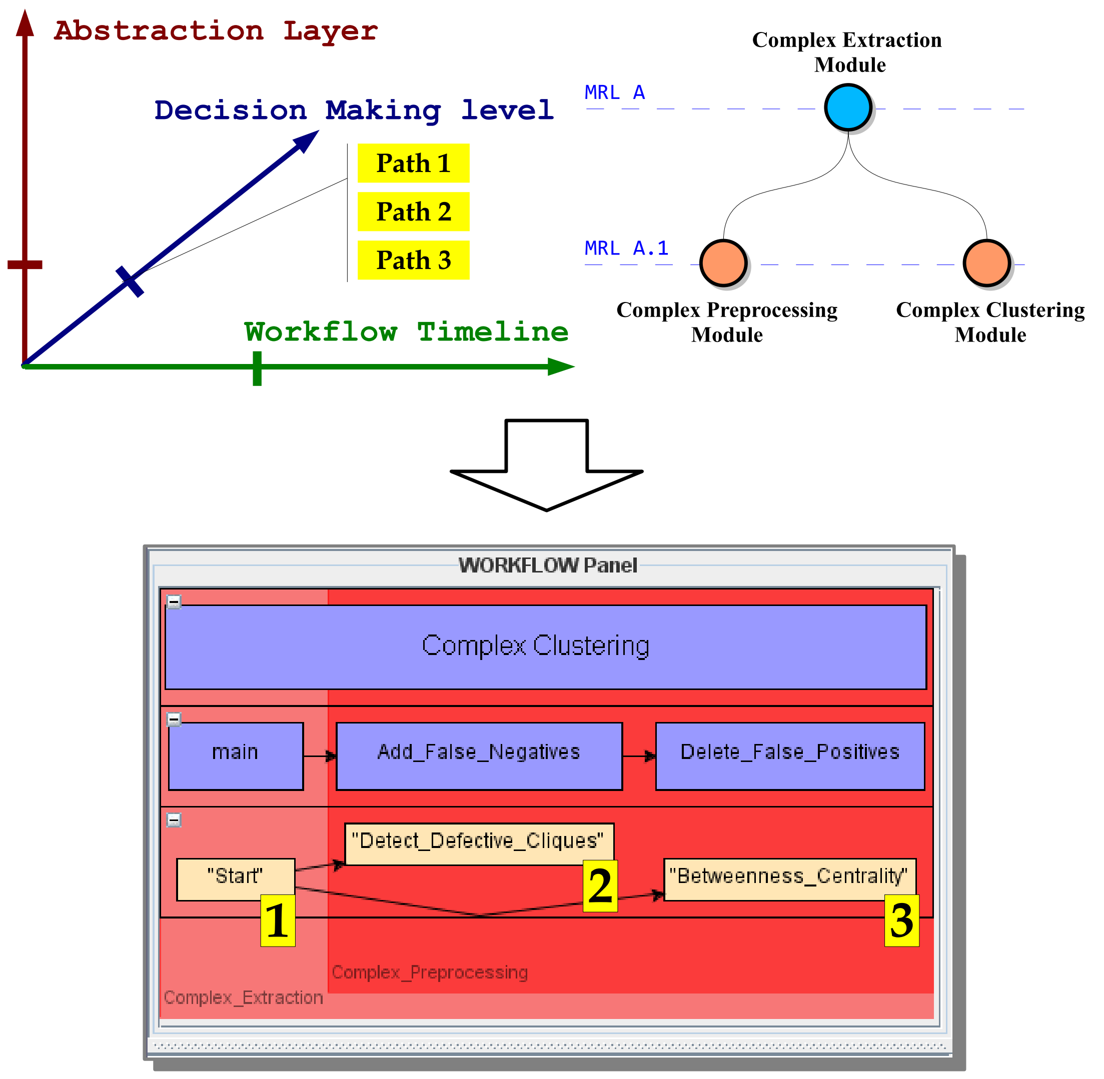


Figure 4.5: Selection of the preprocessing tool. After the execution of tree different tools, the system proposes to the user the available outputs for data analysis.

system integrated in hybrid architecture offers to the user, that agree with the 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, 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 (56, 72) demonstrated *MCODE* is sensitive to noise in the PPIN and the preprocessing phase can increase the algorithm performance. Other authors (63, 64) 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 4.1, in order to highlight some their characteristics.

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

Table 4.1: 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
...	...	...	...

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. 4.6. At the intermediate abstraction layer are depicted all the strategies within the boundaries of their respective 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 (73). Visualization of clustering results, obtained through Cytoscape, are shown in Fig. 4.7. Finally, the user can choose between two outputs shown in Table 4.2, according to its knowledge about the protein complex domain and/or using external evaluation parameters.

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

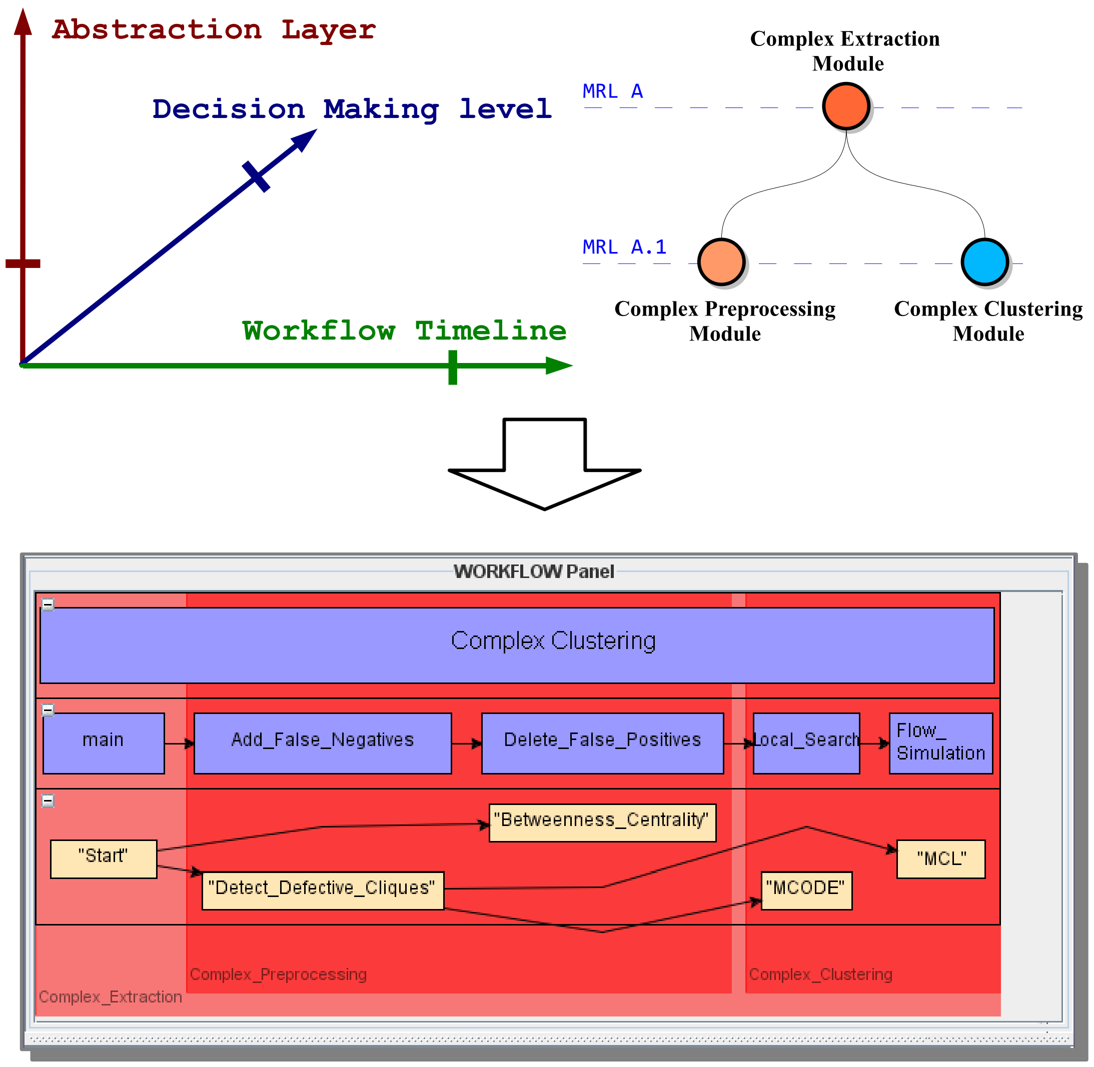
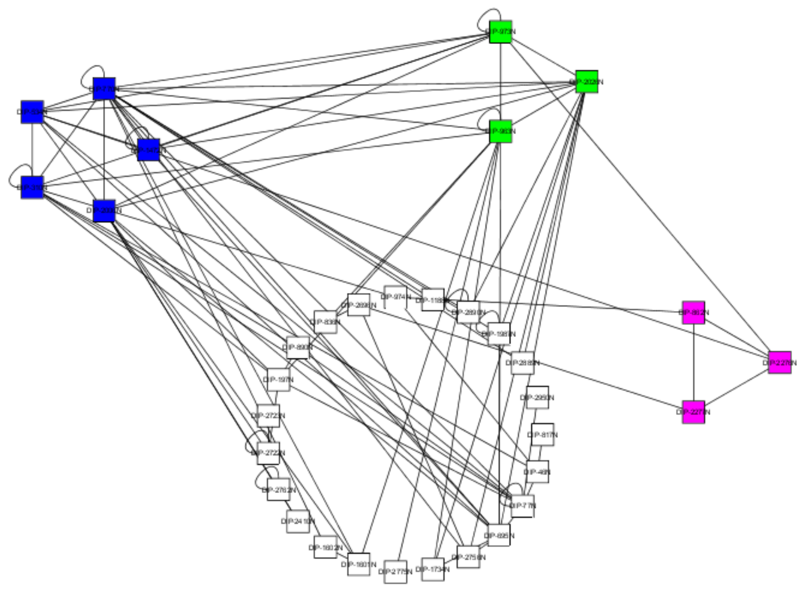
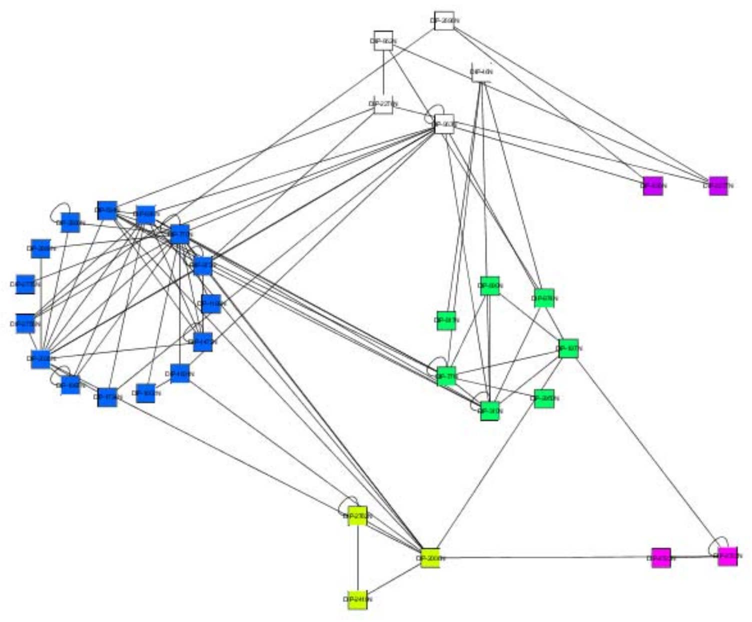


Figure 4.6: Workflow of the whole experiment. The system shows in a tree-like structure all the strategies and algorithms have been used, according to abstraction layers.



(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 4.7: 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
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 4.3: 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(1).

5

Materials & Methods

BORIS implementation is based on Java technology. The Java programming language is a high-level object-oriented language used in every major industry segment; it has a presence in a wide range of devices, computers, and networks. Grace to its features, such as platform and location independence, portability, OS independence, Java represents a good support for this project work.

The rest of this section aims to describe briefly the tools used to develop BORIS: Jess, the rule-based engine, used to menage the knowledge base; Protege, the powerful ontology editor, used for modeling the knowledge base; JGraphX library, used for generate and visualize an interactive workflow; Eclipse Platform, the integrated development environment that allows to bind all these technology in an embedded Java environment.

Figure 5.1 shows the relations between adopted technologies: Java, Jess and Protege. In addiction, the JGraphX library is highlighted, because it is responsible of visualization and user interaction of process workflow.

5.1 Jess: the Rule Engine for the Java Platform

The BORIS DDS implements a Rule-Based system to manage the knowledge-base. The rule based engine adopted is Jess (77), the Rule Engine for the Java Platform. Jess supports the BORIS declarative approach, acting at the Decision Making level in the tree-dimensional space. A declarative approach is well suited above all for solving problems without a clear algorithm solution, like for instance

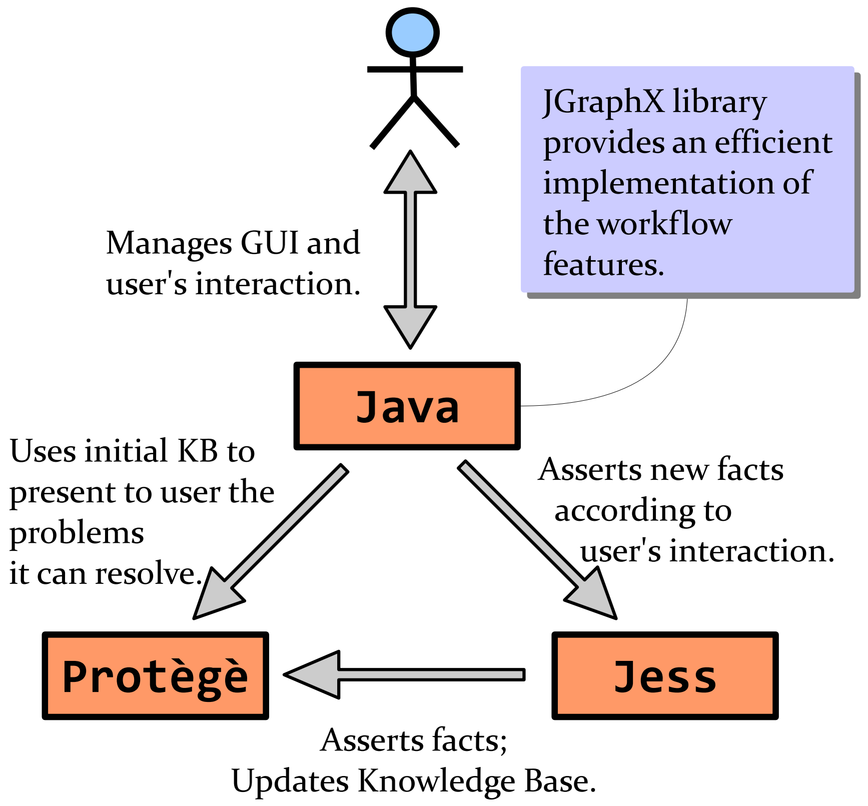

JGraphX library provides an efficient implementation of the workflow features.
Manages GUI and user's interaction.
Java
Uses initial KB to present to user the problems it can resolve.
Asserts new facts according to user's interaction.
Protègè
Jess
Asserts facts; Updates Knowledge Base.

Figure 5.1: BORIS implementation. The main programming language is Java; the other languages, such as Jess and Protege, are also based on Java.

classification, prediction, diagnosis that have some heuristics or guidelines rather than a predefined set of instructions.

A Rule-Based system can be defined as an intelligent system that is able to make inferences from a set of initial knowledge, called *facts*, by means of *rules*, representing reasoning activity. Rules are organized according to the paradigm predicate-action or premises-conclusion; they code typically the expertise, the skill and the heuristics typical of human experts.

Both the BORIS framework and Jess are written totally in Java: for this reason they can be easily integrated. Jess inference engine uses RETE algorithm (78) as pattern matcher. 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.

As well as the decision making process, 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. Each module can receive the focus from a "parent" module, when a shift action is 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. For these reasons, each decision module of hybrid architecture is implemented as a Jess module, where all the strategies and the heuristics related to a specific tasks, are coded as facts and the reasoning about the problem domain is coded as rules. Using Jess, it is possible to implement the different meta-level reasoners through a set of decision-making modules. For example, high level decision modules make a reason at meta-levels, with a set of rules for deciding what are the main phases to solve a request, and then it give the focus to lower level decision modules that is responsible to select and suggest a specific strategy.

5.1.1 Architecture of Jess

Main components of Jess are the Knowledge-Base (KB) and the inference engine.

KB contains both the pieces of information, called facts, and some constraints on the values of facts' attributes. 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; the set of all facts is also known as working memory. The inference engine is made of three elements:

1. The **Pattern Matcher** is 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. It is important to remember that activated rules are not immediately executed, or "fired".

2. The **Agenda** contains all activated rules. It 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.

3. The **Execution Engine** can actually execute the right part of the rules. This way it can produce new knowledge, in the sense of new facts to be added to the KB; moreover, 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 architecture of a typical rule-based system is shows in figure 5.2. The Jess inference engine works in a reasoning loop; first of all, the pattern matcher checks

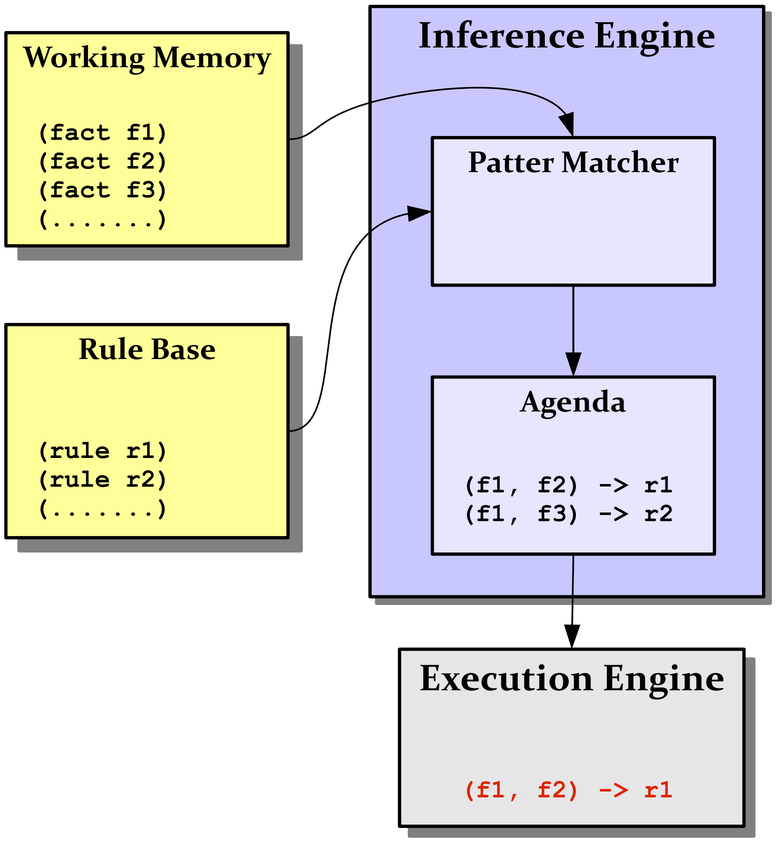


Figure 5.2: Jess Architecture is based on the architecture of a typical rule-based system.

the KB for activated rules and stores them into the agenda; then the agenda decides the firing scheduling of the rules; finally the execution engine runs the right part of rules, obtaining eventually new information that updates KB and that can trigger the activation of other rules. The cycle restarts until the working memory is empty.

5.2 Protege Ontology Editor

The BORIS knowledge base have been modeled using one of the largest adopted tool for building ontologies, that is Protege' (74, 75).

Proteg is useful for represent the knowledge used by the proposed architecture, because it implements a methodology for creating ontologies based on declarative knowledge representation systems.

There are several features that distinguish Protg from other knowledge base editing tools. In the following a list of some characteristics:

- It has an intuitive and easy-to-use graphical user interface.
- It is scalable, in facts Proteg's database back-end loads frames only on demand and uses caching to free up memory when needed.
- It has an extensible plug-in architecture. For example some plug-ins tailored for a some domain and task are implemented, such as small user-interface components or custom back-end plug-ins that use storage mechanisms of the host system.

The figure 5.3 shows a Proteg view of ontology representation of the BORIS knowledge base related to the protein complex extraction. In this figure it is possible to see the three main branch of the ontology for PPI analysis: `Proteomics (Domain)`, `Graph Analysis (Tool)` and `Protein Complex Extraction (Task)`. The last one contains information (facts and rules) about the decision making module “Complex Extraction” representing the root (at the highest meta-level) in decision making tree used into the section 4.

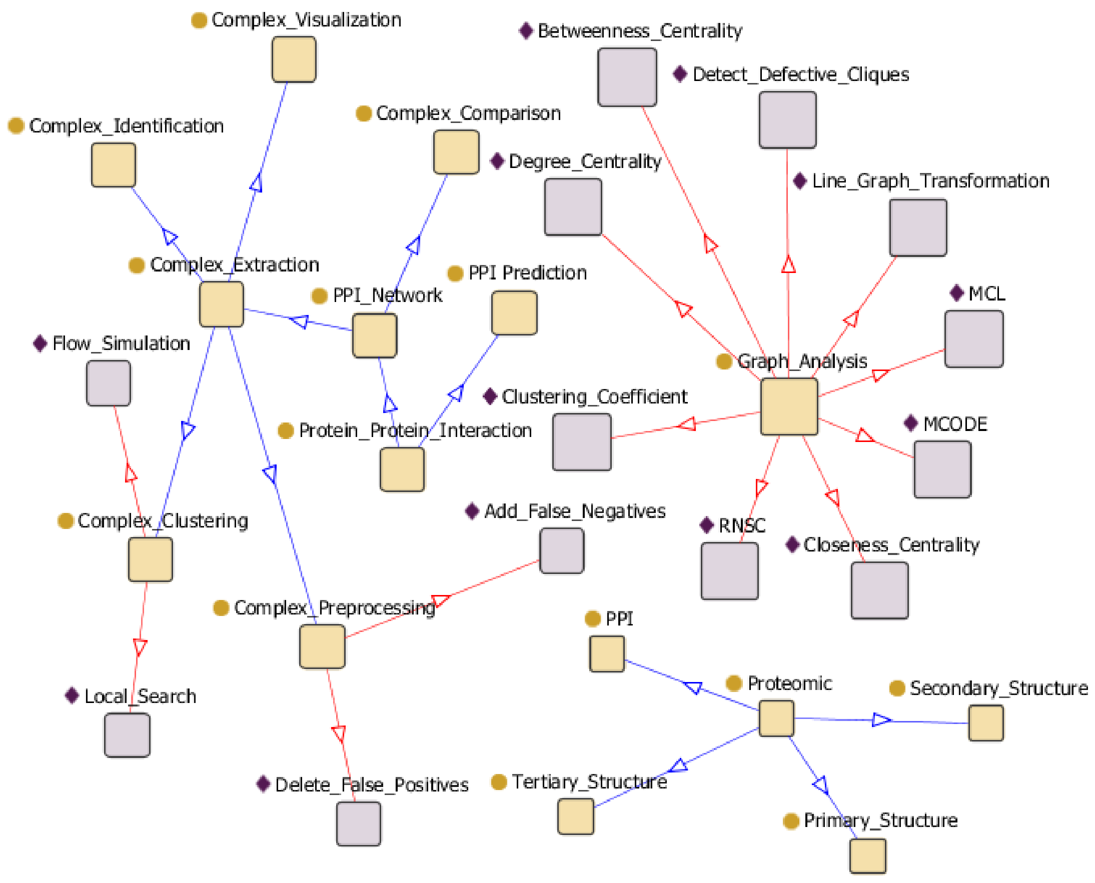


Figure 5.3: Proteg representation of the BORIS knowledge base related to PPI analysis. The tree main concepts for this scenario are: "Graph Analysis", "Proteomic" and "Complex Extraction".

5.3 JGraphX Library

JGraphX is the Java Swing library version of mxGraph (82), 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 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 5.4 shows an example of JGraphX visualization.

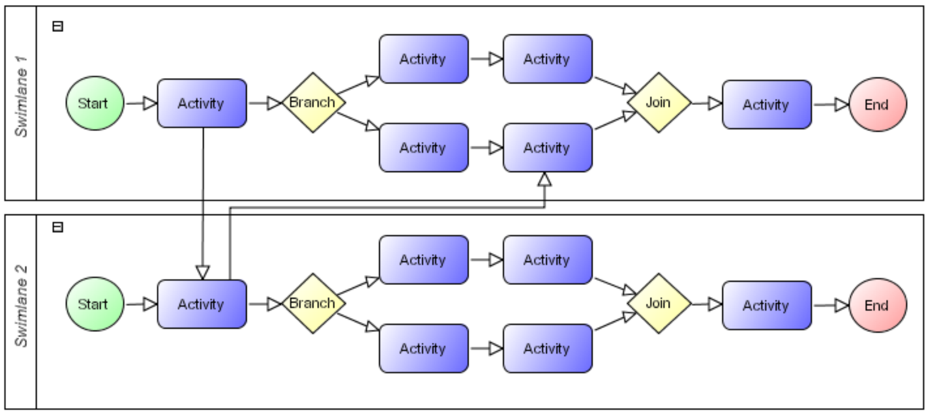


Figure 5.4: JGraphX: an example of the workflow layout. Figure from "*JGraphX User Manual. Copyright (c) David Benson, Gaudenz Alder 2006-2010.*"

5.4 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 (83). 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 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.

6

Conclusions and Future Works

In this PhD thesis a new hybrid architecture for decision support system in bioinformatics has been introduced, the presented work has been developed as a part of the "Bioinformatics Organized Resources - an Intelligent System" (BORIS) project, belonging to the research group "Bioinformatica" of the CNR.

The proposed hybrid architecture has been designed in order to include some features of three different approaches (procedural, declarative and process approach): this way the system can offer to the user different viewpoints on the same problem. In facts, a new 3D-space for decision support systems has been defined, composed by abstraction layer axis, decision making axis and workflow timeline axis: the overall vision of the problem, in terms of abstraction of tasks/sub-tasks, decision-making process and the workflow building, makes the proposed system an ideal joint between classical decision support systems and more recent work?ow management systems.

The major advantages of this work are the capability of facing a problem at different abstraction layers and reasoning levels, handling a workflow management with expertise that execute tools/services and exploiting a modular organization for strategies/heuristics.

In addiction a novel approach for the extraction of protein complexes (responsible for many biological mechanism of a cellular process) based on the proposed DSS has been presented. With a knowledge base created by more than 50 scientific papers about protein-protein interaction network, the system is able to suggest the most useful strategies and algorithms that are suitable for the problem and, moreover, helps him providing the description, pros and cons of each available technique for the complex clustering problem. Finally the system also runs the selected tools, showing intermediate results and, eventually, suggesting to the user what are appropriate values of parameters for the specific situation. During the experiment the system builds a workflow of executed operations in real-time, allowing the user to see what operations are being executed, having the chance of backtracking for exploring alternative paths.

The next step of the entire BORS project will be the migration of the system from the desktop platform to a web based client application.

This way, it will be available to the whole bioinformatics community, that could both use the system and improve its performance and features; in facts new knowledge could be introduced by enabled user in order to increase the capability of the system in terms of both strategies and tools. For this reasons, the hybrid architecture will be provided with formal guidelines that will allow developers to organize its problem in different abstraction layers and to structure the knowledge about the specific problem into a set of decision making modules that will be located into proper meta-reasoning levels. Moreover, all the new implemented decision modules could be integrate with old available modules, such as a node in the reasoning tree.

Make the hybrid architecture able to handle the new knowledge and expertise in a very simple way is a main goal of next BORIS generation.

7

Appendix A

7.1 BORIS User Interface

The software interface between the DSS and the user represents a critical point that often determines whether an application will be successful or will be hard to use.

Sever authors, among which (23, 51), have examined the characteristics that make a GUI the winning horse of a DSS. According to guidelines suggested to previous authors, the BORIS GUI is provided by an aesthetic and minimalist interface design in order to reduce information load. It has a layout balanced and proportional to the information it can visualize about the decision making process and workflow management. In addiction the UI offers informative feedback about system status, allowing the user both to understand control mechanisms behind the process flow and to extend its expertise about a problem.

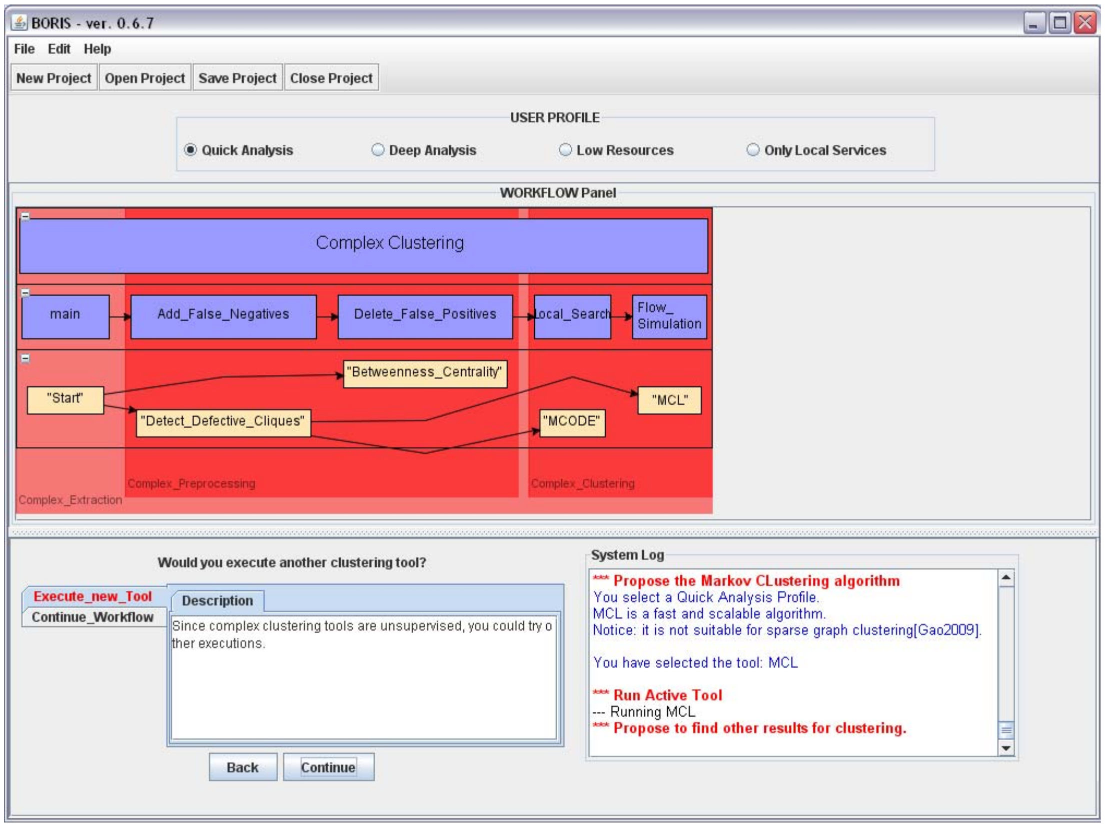
The GUI also offers the possibility to load and save generated workflow, in order to allow the user to solve a problem at different moments or to exploit a previously done project as reference for further analysis and reporting.

A caption of the BORIS GUI during the execution of an experiment is reported in figure 7.1. The GUI is composed by four main components:

- Profile Panel
- Workflow Panel

- Strategy Panel
- System Log Panel

The functionality of each component is analyzed in the following subsections.



USER PROFILE: [✓ Quick Analysis] Deep Analysis Low Resources Only Local Services

Would you execute another clustering tool?

Since complex clustering tools are unsupervised, you could try other executions.

```
*** Propose the Markov CLustering algorithm
You select a Quick Analysis Profile.
MCL is a fast and scalable algorithm.
Notice: it is not suitable for sparse graph clustering[Gao2009].

You have selected the tool: MCL

*** Run Active Tool
--- Running MCL
*** Propose to find other results for clustering.
```

Figure 7.1: An overview of the entire program. It is composed by four main panels: user can interact with all of them.

7.1.1 Profile Panel

The panel, locate in the top of the figure 7.1, allows the user to choose one of available profiles that will be taken into account during decision making process.

It provide an accelerator for users, in facts setting of user profile imply that the user can reduce the number of interactions and to increase the pace of interaction.

This way the system can respond to the differing needs of its users, speeding up the interaction for the expert user.

A list of available profiles, considered by the system in the choice of strategies and tools for the selected problem, is shown in the following:

1. **Quick Analysis**: the system aims to select tools with a low computational complexity algorithm;

2. **Deep Analysis**: the system prefers the most accurate tools, without time or resources constraints;

3. **Low resources**: the system prefers tools that require low resources to run.

4. **Only local services**: the system prefers the execution of local tools and software.

7.1.2 Workflow Panel

This panel shows the building of the workflow. As shown in figure 7.1, it is located in the center of the application with a large percentage of the user interface, because it maintains the most of information content of the system.

It visualizes the hierarchy of tasks and subtasks used to solve the problem organized in different abstraction layers according to their complexity level. Strategies and corresponding algorithms are shown in rectangular boxes. Reasoning levels are depicted as pink bounding boxes. This way, the sequences of actions/steps are organized into groups according to tasks, decision modules or abstraction layer; each group contains informative feedback that allow the user to run a proper set of operations.

Right-clicking any box of the workflow, a context-sensitive pop-up menu shows allowed operations, such as saving results, showing results (when possible) or using external tools, i.e. Cytoscape, to further process the results. In addiction, it permits easy reversal of user actions, providing the chance to restart the reasoning from any point inside the workflow in order to explore alternative paths, if any, or to select an algorithm block for changing input parameters and re-run it.

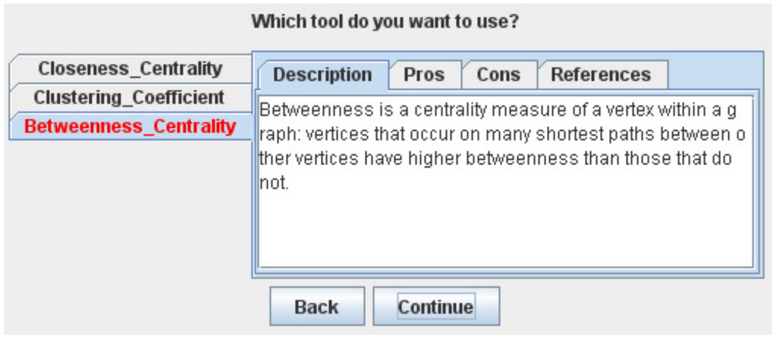
7.1.3 Strategy Panel

Perhaps the Strategy Panel, depicted in the figure 7.2, is the most important panel of the GUI, because it displays the user perspective of the current state of decision making process; In other words, this panel reports the operation flow for the user and it is responsible for the principal interaction with the user.

It can contain a list of suggested strategies and algorithms, that satisfy the reasoning process of the hybrid architecture. It shows a general description of the strategy/algorithm proposed by the system, a list of pros related to the evolution of workflow, some cons (if there are) and a list of bibliographic references related to the selected strategy/algorithm.

Moreover, if the reasoning process leads the workflow to some possible forks, this panel shows all the alternative pathways and get the user to make a decision showing him all the information about the current context.

Each user decision is confirmed with the button “continue”, showed in the bottom of the figure 7.2; if the user want to change decision on workflow design, he can exploit the back-tracking feature by means of “back” button, restoring the state of decision making process at previous step.


Which tool do you want to use?

Betweenness is a centrality measure of a vertex within a g raph: vertices that occur on many shortest paths between o ther vertices have higher betweenness than those that do not.

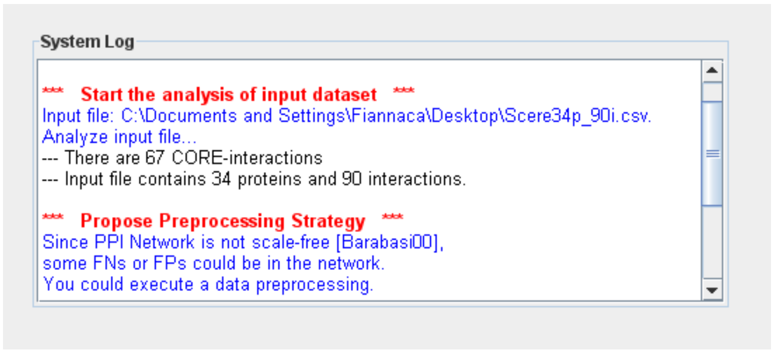
Figure 7.2: BORIS Strategy Panel. It suggests strategies/tools for a specific task and it offers information about them.

7.1.4 System Log

The System Log panel contains events that are logged by the system components. These events are obtained by the user-BORIS interaction, and contain

information about operations, intermediate process, experiments and more.
The BORIS's system log is a formatted text with no interactive features; the text is represented as the following:

- `System Strategy`: red colour and boldface style. It reports each active rule that has been accepted by the user by means of "continue" button in the Strategy Panel. Typically, it instances a new task in the Workflow Panel.
- `Reasoning Process`: blue colour and normal style. It shows the reasoning behind a rule and gives the user some suggestions related to strategy/tool has been executed.
- `Execution Result`: black colour and normal style. It displays result of external processes, such as algorithms used by selected tools, outputs of web services, and so on.



```
***   Start the analysis of input dataset   ***
Input file: C:\Documents and Settings\Fiannaca\Desktop\Scere34p_90i.csv.
Analyze input file...
--- There are 67 CORE-interactions
--- Input file contains 34 proteins and 90 interactions.

***   Propose Preprocessing Strategy   ***
Since PPI Network is not scale-free [Barabasi00],
some FNs or FPs could be in the network.
You could execute a data preprocessing.
```

Figure 7.3: BORIS System Log panel. It reports information about executed operations, intermediate process, experiments and more.

References

[1] http://dip.doe-mbi.ucla.edu/ viii, 40, 59

[2] http://www.helmholtz-muenchen.de/en/mips/ 40

[3] 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. 40

[4] D. Eisenberg, E. M. Marcotte, I. Xenarios, T. O. Yeates Protein function in the post-genomic era Nature, 405, (2000). 39

[5] M. M. Maslon, T. R. Hupp, Drug discovery and mutant p53, Trends in Cell Biology, 20(9), (2010), 542-555. 40

[6] D. J. Power, Brief History of Decision Support Systems, DSSResources.COM, http://DSSResources.COM/history/dsshistory.html

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

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

[9] 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). 5

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

[11] Perreault L., Metzger J. A pragmatic framework for understanding clinical decision support. Journal of Healthcare Information Management, 13(2), (1999), 5–21. 6

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

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

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

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

[16] P. Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armananzas, G. Santafe', A. Perez, V. Robles, Machine learning in bioinformatics. Briefing in bioinformatics, 7(1), (2005), 86–112. 17

[17] 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, 31, (2004), 117–136. 17

[18] D. Hanisch, K. Fundel, H.T. Mevissen, R. Zimmer, J. Fluck, ProMiner: rule-based protein and gene entity recognition, BMC Bioinformatics, (6), (2005), Suppl 1:S14. 18

[19] J.C. Whisstock, A.M. Lesk, Prediction of protein function from protein sequence and structure, Quarterly Reviews of Biophysics, Cambridge University Press, 36(3), (2003), 307–340. 18

[20] 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, 8, (2007). 18

[21] H. Yamakawa, K. Maruhashi, Y. Nakao, Predicting Types of Protein-Protein Interactions Using a Multiple-Instance Learning Model, LNCS, 4384, (2007), 42–53. 21

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

[23] D. J. Power, Decision Support Systems: Concepts and Resources for Managers, Westport, CT Greenwood/Quorum, (2002). 5, 70

[24] J. Wyatt , D. Spiegelhalter, Field trials of medical decision-aids: potential problems and solutions. Clayton P (ed). Proc. 15th Symposium on Computer Applications in Medical Care, Washington 1991. New York: McGraw Hill Inc., (1991), 3–7. 6

[25] B. G. Buchanan, E. H. Shortliffe, Editors, Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, AAAI, (1984). 6

[26] M.A. Musen, Stanford Medical Informatics: uncommon research, common goals. MD Comput, 16(1), (1999). 6

[27] 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

[28] 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. 7

[29] 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. 7

[30] M. A., Musen, S. W. Tu, A. K. Das, 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. 7

[31] 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). 7

[32] R. Boulme, D. Gonzalez, JC. Schmit, Storing genotypic resistance data and linking to other clinical information, XV International AIDS Conference, (2004) Bangkok, Thailand. 7

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

[34] A. DiCaterino, K. Larsen, M.H. Tang, W.L. Wang. An Introduction to Workflow Management Systems, (1997). 8

[35] 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). 9

[36] P. Romano, E. Bartocci, G. Bertolini, F. De Paoli, D. Marra, G. Mauri, E. Merelli, L. Milanesi, Biowep: a workflow enactment portal for bioinformatics applications, BMC Bioinformatics, 8, (2007). 9

[37] 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. 25

[38] E. Bartocci, F. Corradini, E. Merelli, L. Schortichini, BioWMS: a Web-based Workflow Management System for Bioinformatics, BMC Bioinformatics, 8(1), (2007). 9

[39] A. Fiannaca, S. Gaglio, M. La Rosa, D. Peri, R. Rizzo, A. Urso, A Proposed Knowledge Based Approach for Solving Proteomics Issues, Computational Intelligence Methods for Bioinformatics and Biostatistics II, LNCS 6160 (2010), 304–318.

[40] P. Jackson, Introduction to Expert Systems, Addison-Wesley, (1998).

[41] B. Chandrasekaran, J. R. Josephson, V. R. Benjamins, What Are Ontologies, and Why Do We Need Them?, IEEE Intelligent Systems, 14(1), (1999), 20–26. 18

[42] S. Srinivasan, D. Kumar, V. Jaglan, Agents and their knowledge representations Journal of Ubiquitous Computing and Communication, 5(1), (2010). 24

[43] E. Gat, Integrating planning and reacting in a heterogeneous asynchronous architecture for mobile robots, SIGART Bulletin, 2, (1991), 70–74. 25

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

[45] 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). 23

[46] B. Hayes-Roth, Dynamic Control Planning in Adaptive Intelligent Systems, Proceedings of the DARPA Knowledge-Based Planning Workshop, (1987). 23

[47] S. Vere, T. Bickmore, A basic agent. Computational Intelligence, 6, (1990), 41–60. 23

[48] R. Harris, Introduction to Decision Making, VirtualSalt, (1998).

[49] J. Flp, Introduction to Decision Making Methods, Workshop on Biodiversity & Ecosystem Informatics,(2005). 28

[50] 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). 28, 29

[51] B. Shneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, (2nd edition) Reading, MA: Addison-Wesley, (1992). 70

[52] 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. 33

[53] 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. 42

[54] G. Sabidussi, The centrality index of a graph, Psychometrika, 31(4), (1966), 581-603. 42

[55] L. C. Freeman, A set of measures of centrality based on betweenness, Sociometry, 40, (1977), 35-41. 42

[56] 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. 42, 54

[57] H. Yu, A. Paccanaro, V. Trifonov, M. Gerstein, Predicting interactions in protein networks by completing defective cliques, Bioinformatics, 22(7), (2006), 823–829. 42, 50

[58] G.D. Bader, C.W. Hogue, An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics. 4(1) ,(2003). 42

[59] A.D. King, N. Przulj, I. Jurisica, Protein complex prediction via cost-based clustering. Bioinformatics, 20(17), (2004), 3013-3020. 42

[60] S. Van Dongen, Graph clustering by flow simulation, Ph.D. thesis, Centers for Mathematics and Computer Science (CWI), University of Utrecht, (2000). 42

[61] T. Aittokallio , B. Schwikowski, Graph-based methods for analysing networks in cell biology. Briefings in Bioinformatics, 7(3), (2006), 243–255. 43

[62] 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. 43

[63] 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. 54

[64] 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. 54

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

[66] V. Arnau, S. Mars, I. Martn, Iterative cluster analysis of protein interaction data, Bioinformatics, 21(3), (2004), 364-378. 46

[67] B.L. Drees, B. Sundin et al. A protein interaction map for cell polarity developmen, Journal of Cellular Biology, 154, (2001), 549-571. 46

[68] A.L. Barabsi, Z.N. Oltvai, Network biology: understanding the cell's functional organization, Nature Reviews Genetics, 5, (2004), 101–113. 49

[69] C. von Mering et al. Comparative assessment of large-scale data sets of protein-protein interactions, Nature, 417, (2002), 399–403. 50

[70] P. Legrain, How Useful Will Functional Proteomics Data Be? Comp Funct Genomics, 2(5), (2001), 301-303. 50

[71] J. Chen, W. Hsu, M. L. Lee, S. Ng, Increasing confidence of protein interactomes using network topological metrics, Bioinformatics, 22(16), (2006), 1998–2004. 50

[72] S. Brohe, J. van Helden, Evaluation of clustering algorithms for protein-protein interaction networks, BMC Bioinformatics, 7(488), (2006). 54

[73] P. Shannon P et al., Cytoscape: a software environment for integrated models of biomolecular interaction networks, Genome Research, 13(11), (2003). 55

[74] The Protege Ontology Editor and Knowledge Acquisition System, *http://protege.stanford.edu*. 64

[75] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubzy, H. Eriksson, N. F. Noy, S. W. Tu, The evolution of protege: An environment for knowledge-based systems development. International Journal of Human-Computer Studies, 58,(2003), 89-123. 64

[76] CASP: Critical Assessment of Techniques for Protein Structure Prediction. *http://predictioncenter.org/index.cgi*

[77] Sandia National Laboratories, Jess: The rule engine for the JavaTM platform, Available at http://herzberg.ca.sandia.gov/jess/, (2003). 60

[78] C. Forgy, Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, 19, (1982), 17-37. 61

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

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

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

[82] *http://www.jgraph.com/* 65

[83] *http://www.eclipse.org/* 66

Declaration

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

The thesis work was conducted from January 2008 to February 2011 under the supervision of Prof. S. Gaglio at University of Palermo.

Palermo - February 15, 2011.