Universidade do Minho
Escola de Engenharia

José Pedro Basto Gouveia Pereira Pinto

Computational tools for large-scale biological network analysis

Junho de 2012

Computational tools for large-scale biological network analysis
José Pedro Basto Gouveia Pereira Pinto
UMinho | 2012

Universidade do Minho
Escola de Engenharia

José Pedro Basto Gouveia Pereira Pinto

Computational tools for large-scale biological network analysis

Tese de Doutoramento em Informática

Trabalho realizado sob a orientação do
Doutor Miguel Francisco de Almeida Pereira da Rocha
e da
Doutora Isabel Cristina de Almeida Pereira da Rocha

Junho de 2012

Autor
José Pedro Basto Gouveia Pereira Pinto

Email: josepedr@di.uminho.pt/ josepedr@gmail.com
Telefone: +351 914992366
BI: 12405427

Título da tese
Computational tools for large-scale biological network analysis

Orientadores
Doutor Miguel Francisco de Almeida Pereira da Rocha
Doutora Isabel Cristina de Almeida Pereira da Rocha

Ano de conclusão 2012
Doutoramento em Informatica

Universidade do Minho, Junho de 2012

Agradecimentos/ Acknowledgments

Nesta secção gostava de agradecer a todos o que me ajudaram a terminar este trabalho.

Primeiro aos meus orientadores, Doutor Miguel Rocha e Doutora Isabel Rocha, por toda a ajuda que me deram durante estes anos.

A todos os meus colegas do grupo de investigação BisBII por me terem aturado durante este tempo.

Ao Departamento de Informática da Universidade do Minho por ter disponibilizado e espaço e equipamento para a realização do meu trabalho.

À Fundação para a Ciência e Tecnologia por me terem atribuído uma bolsa de doutoramento (ref:SFRH/BD/41763/2007).

Aos meus amigos pelo suporte que me deram.

É por fim à minha família. Não teria conseguido fazer isto sem o vosso apoio.

Abstract

The surge of the field of Bioinformatics, among other contributions, provided biological researchers with powerful computational methods for processing and analysing the large amount of data coming from recent biological experimental techniques such as genome sequencing and other omics. Naturally, this led to the opening of new avenues of biological research among which is included the analysis of large-scale biological networks.

The analysis of biological networks by itself is not new, but until recently researchers were limited to small-scale networks, due to the complexity inherent to biological systems. Recently, Bioinformatics provided researchers with the tools and methodologies needed to create and study large-scale networks. So, progressively larger networks have been built and more biological complex systems have been represented as networks.

Since the study of large-scale biological networks is a relatively recent field, there are still few software tools focused in this research area. The main objective of this work was to contribute to this field, through the development of methodologies and computational tools for the creation and analysis of large-scale cellular networks.

One of the major contributions was the development of InBiNA, an open-source user-friendly application for the analysis of biological networks. InBiNA is a generic tool that can be used with most kinds of cellular networks, being focused in the analysis of integrated networks potentially representing metabolic, regulatory and/or signalling sub-systems. The usefulness of InBiNA has been shown by a case study including some pathways of *Escherichia coli*'s metabolism, together with different types of regulatory systems controlling these pathways.

Also, TNA4OptFlux, a plug-in for the metabolic engineering software platform OptFlux, was created. Using the methodologies developed during this work, this plug-in is capable of combining the model-based phenotype simulation methods of OptFlux with network-based topological analysis methods, giving the user a new way of analysing the metabolism. One of the major applications is the comparison of the networks corresponding to wild-type and mutant strains, designed by strain optimization algorithms to overproduce interesting compounds. This brings interesting tools for the analyses of the strategies followed by mutant strains, as compared to the original ones. A case study, also using *E. coli*, for the production of succinate shows the usefulness of the tool.

In this thesis the capabilities of InBiNA and TNA4OptFlux are presented, confirming their validity and utility as novel tools in the portfolio of Systems Biology research.

Resumo

O aparecimento do campo da Bioinformática trouxe, entre outras contribuições, ferramentas computacionais poderosas para o processamento e a análise das grandes quantidades de dados provenientes das recentes técnicas experimentais de alto débito em Biologia, tais como a sequenciação de genomas e outra ómicas. Naturalmente, isto conduziu à abertura de novas áreas na investigação biológica, entre as quais se inclui a análise de redes biológicas em larga escala.

A análise de redes biológicas, por si só, não é uma novidade, mas até muito recentemente os investigadores da área limitavam-se ao estudo de redes em pequena escala, dada a complexidade inerente aos sistemas biológicos. Recentemente, a Bioinformática veio fornecer as ferramentas e as metodologias necessárias para criar e estudar redes em larga escala. Assim, redes progressivamente maiores têm sido construídas e cada vez mais sistemas biológicos complexos têm sido representados como redes.

Dado que a análise de redes biológicas em larga-escala é ainda um campo recente, existem ainda poucas ferramentas focadas nesta área. O principal objetivo deste trabalho é o de contribuir para este campo, através do desenvolvimento de metodologias e ferramentas computacionais que permitam a criação e a análise de redes celulares em larga-escala.

Uma das principais contribuições deste trabalho foi o desenvolvimento da aplicação InBiNA, uma aplicação aberta com uma interface amigável e que permite a análise de redes biológicas. Trata-se de uma ferramenta genérica que pode ser usada para analisar diversos tipos de redes celulares, sendo focada na análise de redes integradas, potencialmente representando sub-sistemas metabólicos, regulatórios e/ou de transdução de sinal. A utilidade da aplicação foi demonstrada através de um caso de estudo que envolveu a criação de uma rede incluindo algumas vias metabólicas da bactéria *Escherichia coli*, em conjunto com diferentes tipos de regulação controlando estas vias.

Adicionalmente, o plug-in TNA4OptFlux foi desenvolvido, sendo um plug-in para o OptFlux, uma plataforma de software de Engenharia Metabólica. Usando as metodologias desenvolvidas durante este trabalho, este plug-in é capaz de combinar métodos de simulação de fenótipos baseados em modelos metabólicos com métodos de análise topológica de redes biológicas, fornecendo aos utilizadores uma forma distinta de analisar o metabolismo. Uma das principais aplicações passa pela comparação de redes metabólicas correspondentes a estirpes selvagens e mutantes desenhadas por algoritmos de otimização de estirpes que procuram a sobre-produção de compostos com interesse industrial. Assim, conseguem-se produzir ferramentas com interesse para a análise das estratégias seguidas pelas estirpes mutantes, quando comparadas com as originais. Um caso de estudo usando *E. coli* para a sobre-produção de succinato demonstra a utilidade das ferramentas.

Neste trabalho, as capacidade das aplicações InBiNA e TNA4OptFlux são demonstradas, confirmando a sua validade e utilidade como novas ferramentas no portfólio da investigação na Biologia de Sistemas.

Contents

List of Figures

List of Tables

Acronyms

BC - betweenness centrality

BFS - breadth-first search

BPCY - Biomass-Product Coupled Yield

CC - closeness centrality

CSV - Comma-separated values

DEFL - double extended feedback loop

DOR - dense overlap regulon

EA - Evolutionary Algorithm

EFL - extended feedback loop

EFM - Elementary Flux Mode

FBA - Flux Balance Analysis

FFL - feed forward loop

GP - gene-protein network

GPC - gene-protein-compound network

GPR - gene-protein-reaction network

GUI - Graphical User Interface

HITS - Hyperlink-Induced Topic Search

InBiNA - Integrated Biological Network Analysis

indegree - incoming degree

JUNG - Java Universal Network/Graph Framework

ME - Metabolic Engineering

MVC - model-view-controller

nRGs - non-regulatory genes

outdegree - outgoing degree

PC - protein-compound network

PR - protein-reaction network

PRC - protein-reaction-compound network

RC - reaction-compound network

RG - regulatory genes

RSIM - reversed single input module

SA - Simulated Annealing

SBBFS - set-based breadth-first search

SBML - Systems Biology Markup Language

SIM - single input module

TF - transcription factor

TNA4OptFlux - Topological Network Analysis for OptFlux

TRN - transcriptional regulatory network

UML - Unified Modeling Language

VANTED - Visualization and Analysis of Networks containing Experimental Data

XML - eXtensible Markup Language

Chapter 1

Introduction

In this chapter, the motivation for the work is given, providing a context for the remaining work. Then, the objectives of the work are put forward. The chapter closes with an outline of the thesis organization.

1.1 Motivation

There are a vast number of systems of variable complexity in all areas of science that can be classified as networks, i.e. systems composed by several discrete individual entities with different types of interactions among them. Consequently, the analysis of networks is an activity occurring in research across multiple fields, from social studies to natural and computer sciences. Naturally, this fact has led to the development of numerous metrics, algorithms and methodologies for analysing networks, as well as to adaptations to handle specific networks in certain fields.

Many biological systems, from ecosystems to molecular interactions, can be modelled as networks, a fact that is not altogether surprising if one ponders at the nature of life's organization. Indeed, for instance, a large-scale ecosystem consists in a set of interacting living organisms, while at a cellular scale the most basic systems, which make life possible, also have a network-like organization. For instance, the metabolism of any organism is ultimately a network made up of hundreds or thousands of interconnected reactions and metabolites. While scientists have studied biological networks for several years, until recently these efforts were led by isolated teams and were focused in small subsets of biological networks [1][2].

The main obstacle for the analysis of large-scale biological networks is their inherent complexity. At all levels of analysis, living organisms are very complex systems composed of several individual components, typically themselves other complex systems. This fact made obtaining the data necessary to model large-scale biological networks beyond the capabilities of most individual research groups. Even when relevant data was available, it was inevitably distributed over several repositories, in different formats.

This reality is changing, however, with the advances brought by the rapid development of the Bioinformatics field. Suddenly, researchers are starting to have both the means for quickly exchanging data and the computational methods for integrating them. This new reality has recently led to the creation of several large-scale networks of biological systems and their subsequent analysis.

The efforts on biological network analysis started with the application of previously developed methods for other fields, from social sciences to web page analysis [3][4][5]. Repurposing methods from other areas into biological networks also uncovered the existence of motifs [6][7], patterns which are overrepresented in real networks when compared to random networks with similar properties. Further study into motifs revealed that they appear to correspond to specific functions of the system the network represents. Consequently, it is currently believed that motifs are the basic building blocks of most complex networks [6], being a major focus of biological network analysis.

Since the analysis of large-scale biological networks is a relatively recent field, there are still no generally agreed standards or methodologies. Consequently, there are also few software tools specifically geared towards biological networks analysis, when compared with other related disciplines.

While it is possible to use generic network analysis tools, or those developed for other specific areas, this is usually not an ideal solution. Indeed, in the former case, the tools available often are created with users whose academic

background is focused in Mathematics, making them very user unfriendly to most biological researchers. In the latter, the networks are assumed to possess characteristics specific to other fields, which probably differ from those of biological networks. It is, thus, easy to infer that specific tools for the study of biological networks are necessary.

1.2 Objectives

The main objective of the work described in this thesis was to contribute to the field of large-scale biological network analysis, through the development of novel algorithms and specific open-source software tools. Although keeping a certain degree of generality in the tools developed, the aim was to address the representation and analysis of biological networks at the cellular level, modelling processes such as the metabolism, regulatory systems and signal transduction.

Two main concerns were central in the development of this work. The first was to be able to represent and analyse networks at a genome-scale level, i.e. being able to cope with networks that can involve a few thousand nodes. The second major concern was to enable the analysis of integrated networks, i.e. to address the study of cellular systems as a whole, being able to work with networks with several distinct types of nodes and interactions. Together, these capabilities will make the developed tools able to work with large-scale cellular networks that can include the metabolic, regulatory and signalling systems, as well as their inter-connections, allowing a truly integrated analysis.

More specifically, the work addressed the following scientific/ technological aims:

- To create a core software library containing several methods for creating and analysing biological networks, serving as the backbone for the

development of the other software tools and enabling a software basis for future component-based software development in the field.

- To develop an open-source application for the analysis of large-scale biological networks based on the previous core. This tool should allow the calculation of several distinct topological metrics (including, among others, degree and shortest path analysis, node rankers/ centralities, clustering, connectivity, etc.) and the detection of network motifs over large-scale integrated cellular networks, including distinct types of vertices and interactions.

- To develop a plug-in for the *in silico* Metabolic Engineering (ME) tool OptFlux, an application created by the host research group allowing for phenotype simulation of microbial organisms using mathematical models, as well as to address strain optimization tasks. This plug-in would share basic network analysis tools with the previous application, but also include specific tools to analyse metabolic networks under an ME perspective. The main aim of these specific tools is to make easier the analysis of the strategies found in mutants provided by strain optimization algorithms, by comparing their metabolic networks with those of wild-type original strains.

- To validate both tools with selected case studies that can highlight their major features and show their usefulness in biomedical research. In these, *Escherichia coli* is used as the model organism, given its importance in Biotechnological applications, the large amount of data available and the know-how existent in the group.

It is important to mention that all software developed in this work is open-source, being fully available for the community both in the form of freely downloadable applications and in the form of its source code. Also, the development methodology follows a plug-in based architecture that makes it easy to allow for new contributions from other authors.

1.3 Thesis organization

The following text is organized as follows:

Chapter 2 includes a review of the main concepts related to biological network analysis and related graph theory definitions, including its main applications and available software tools. It provides definitions on the main topics of this work, including basic algorithms and metrics.

Chapter 3 presents the InBiNA application, the central software tool developed in this work for the analysis of integrated biological networks. It presents its main capabilities and illustrates its use with two selected case studies using networks for the *Escherichia coli* bacterium, the first being used to highlight the functionalities of InBiNA and the second to present a large-scale study where the tools developed show to be useful for the analysis of regulatory mechanisms.

Chapter 4 describes the tools and methods developed to aid in ME efforts through topological analysis over metabolic networks. The OptFlux plug-in TNA4OptFlux is presented, a tool that implements those features with a user-friendly interface. The main functionalities are described and a practical case study is presented. The chapter closes with an example of a large-scale study over sets of strain optimization solutions.

Chapter 5 collects important information regarding the implementation of the tools presented in the last two chapters. It describes the software development methodologies; explains the architecture and options followed and addresses the main algorithms developed during the work.

Chapter 6 closes the text with the main conclusions of the work, summarizing the main achievements and contributions, addressing the limitations of the work and providing clues for future work.

References

[1] X. Zhu, M. Gerstein, and M. Snyder, "Getting connected: analysis and principles of biological networks.," *Genes & development*, vol. 21, no. 9, pp. 1010-24, May 2007.

[2] J. van Helden, L. Wernisch, D. Gilbert, and S. J. Wodak, "Graph-based analysis of metabolic networks.," *Ernst Schering Research Foundation workshop*, no. 38, pp. 245-74, Jan. 2002.

[3] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821-6, Jun. 2002.

[4] D. Koschützki and F. Schreiber, "Centrality analysis methods for biological networks and their application to gene regulatory networks.," *Gene regulation and systems biology*, vol. 2, pp. 193-201, Jan. 2008.

[5] O. Mason and M. Verwoerd, "Graph Theory and Networks in Biology," Systems Biology, IET In Systems Biology, IET, Vol. 1, No. 2. (March 2007), pp. 89-119.

[6] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks.," *Science (New York, N.Y.)*, vol. 298, no. 5594, pp. 824-7, Oct. 2002.

[7] A.-L. Barabási and Z. N. Oltvai, "Network biology: understanding the cell's functional organization.," *Nature reviews. Genetics*, vol. 5, no. 2, pp. 101-13, Feb. 2004.

Chapter 2

Biological Network Analysis: concepts, applications and software tools

In this chapter, the basic concepts underlying this work are presented. The chapter starts by reviewing basic concepts of networks and graphs. It proceeds by presenting the main types of biological networks studied and highlighting their major properties. The following section addresses network motifs, presenting the relevant definitions and describing the main types of motifs relevant in biological networks. Finally, existing software in biological network analysis is reviewed and analyzed.

2.1 Basic concepts of networks and graphs

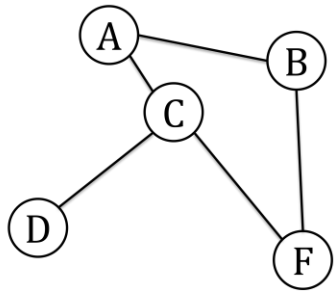
A network is basically a set of entities that are connected by some kind of relationship. There are many systems in all fields of science and engineering, including the biological fields, which according to this definition can be viewed as networks. Due to the prevalence of networks, many methods were developed for the analysis of their properties. This chapter presents the main methods for network analysis that form the basis of this work. In many cases, these methods are not specific for biological networks, being general methods that can be used for the analysis of most types of networks.

In order to analyze a network, it is necessary to translate it into a mathematical object. For this purpose, networks are normally modelled as graphs. Thus, the discipline that studies the properties of graphs, called *graph theory*, is deeply intertwined with network analysis. In this context, this chapter should also be seen as an introduction to some relevant concepts of graph theory.

2.1.1 Basic graph definitions

The first fact that should be defined about graphs is that, while both terms are commonly used interchangeably, a graph is not a network even if it can represent one. A *network* is an informal designation given to a system composed by distinct entities, which are interconnected by some type of relationship. A *graph*, on the other hand, is a mathematical object composed by two sets of distinct objects, containing vertices and edges, where each edge defines a relationship between two vertices. Usually, the term *graph* is used when general properties of the mathematical representation are being described, while the term *network* is used when referring to properties associated with a system (or systems) that can be represented by graphs; however, both terms are also frequently used interchangeably.

Typically, a graph is represented as $G = (V, E)$, where V and E are the sets of vertices and edges, respectively. An *edge e* is represented as $e = \{s, t\}$ where s and t are vertices connected by e, i.e. the edge e represents a relationship between the pair of vertices. A graph can be visually represented drawing a point for each vertex and a line connecting the vertices connected by each edge (see Figure 2.1 for an example).



V = {A,B,C,D,F}

E = {(A,B),(A,C),(B,F),(C,D),(C,F)}

Figure 2.1 - Representation of a graph $G = (V,E)$ composed by 5 vertices and 5 edges.

In a graph, when two vertices are connected by an edge they are called *neighbours*. The *degree* of a vertex is defined as the number of edges that connect to a vertex. On the other hand, a sequence of vertices and edges (v_1, e_{12}, v_2, e_{23}, …, v_{n-1}, e_{n-1n}, v_n), where each vertex is connected to the next by the edge between

them, is called a *walk*. If all edges are distinct, the walk is further called a *path*, while a path in which the vertices are also distinct is a *simple path*. When there is a walk between two vertices, they are *connected*, and this connectivity usually means that there is some kind of relationship between the vertices even if an indirect one. The *length* of a walk is the number of intermediate edges it contains. A graph where there is path between all pairs of vertices is called a *strongly connected* graph.

Several metrics can be derived from the analysis of shortest paths of a graph. The *distance* between two vertices is the length of the shortest path between them, while the longest shortest path between any two vertices in the graph is called the *diameter*. Unless the graph is strongly connected the diameter will have a value of ∞ (infinity). The *characteristic path length* is the mean distance between all pairs of vertices and the *efficiency* is the inverse of the mean path length. All of these metrics give an idea of how connected the graph is.

Another important concept related with the edges is the notion of directionality. An edge *e = (s,t)* is classified as *directed* if the relation it represents holds only from *s* to *t*, while it is *undirected* if it holds in both directions. For example, in a metabolic network, the relation between an irreversible reaction and the compounds that participate in it should be represented as a directed edge. On the other hand, a reversible reaction will be represented by an undirected edge. Directed edges are represented by ordered vertex pairs *(s, t),* while undirected edges are represented by unordered pairs *{s, t}*. A graph containing only directed or undirected edges is called a *directed* or *undirected graph,* respectively, while a graph with both kinds of edges is called a *mixed graph*.

The presence of directed edges affects the concept of degree. In the case of directed or mixed graphs, two other types of degree can be considered: the incoming degree, or *indegree*, gives the number of edges that end in the vertex and the outgoing degree, or *outdegree*, gives the number of edges that start in the vertex. Also the concept of walk is changed, since in a directed graph only the walks taking the direction of the edges into account, called strongly connected

walks, are meaningful (see Figure 2.2. for an example). Those directed edges are connected only "one-way".

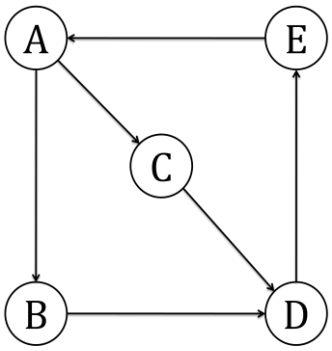


Figure 2.2 - Paths in a directed graph: there are two paths for A→D: (A,C,D) and (A,B,D), but only one path for D→A: (D,E,A).

When it is possible to partition the set of vertices V from a graph $G = (V, E)$ into two distinct sets A and B, in a way that $V = A \cup B$ and $A \cap B = \varnothing$ and all edges start in a vertex from A and end in vertex from B, G is called a *bipartite graph*. Usually, in a system represented by a bipartite graph, there are two types of entities that are represented by the vertices in each group. A logical extension of the concept of bipartite graphs is the definition of *k-partite graphs*, where the vertices set can be divided into k independent sets and no pair of vertices from the same set can be neighbours.

When all vertices of a graph are directly connected to one another, in other words when the shortest path between any two vertices has a length of one, a graph is said to be *complete*. It is unusual for graphs that represent real networks to be complete. Occasionally, however, certain sub-graphs within a graph are complete, being called *cliques*. These usually represent a sub-network that, for some reason, has to maintain its connectivity regardless of any system failures.

To enrich their representation capabilities, graphs often also have attributes associated with their components (vertices and/ or edges). These can be as simple as names (identifiers of the entities the vertices/ edges represent) or numeric weights (e.g. stoichiometric coefficient of a reaction). Graphs with numeric weights on their edges are quite common in many fields. The attributes

of a graph can be either related with the system the graph represents or derived from the structure of the graph itself. For instance, weights can be assigned depending of the importance of the vertices or edges computed according to some metric. These attributes can influence the analysis processes, sometimes requiring alterations in the used algorithms and methods to accommodate the attributes.

Many other graph properties could have been included in this section, but those are either beyond the scope of this work or will be presented in specialized sections in the remaining text.

2.1.2 Shortest paths

In many situations, identifying the relationships between the entities that make up the system is an important, if not the main objective of graph analysis. As mentioned before, if two vertices are connected, the entities they represent usually share some kind of relationship and consequently the analysis of the connections, walks and paths of a graph are a major part of graph theory.

Most studies using this type of analysis have been based in the analysis of shortest paths. This limitation is a natural consequence of the complexity associated with the determination of walks and paths other than the shortest one (typically NP-hard problems). This restriction can bring important limitations when other longer paths are of interest. An example in biological research occurs in the analysis of regulatory networks, where it is sometimes observed that a regulator has an effect different than predicted, probably because of the effect it has on the regulated gene through paths other than the shortest one.

Sometimes, the attributes of the graph may affect the paths and the walks. A common situation is when, instead of assuming that all edges are equivalent, they have a numerical cost (weight) associated. In this case, the sum of the weights of the edges in a path can be more meaningful than its length and

consequently the shortest path is considered as the one with the smallest sum (Figure 2.3).

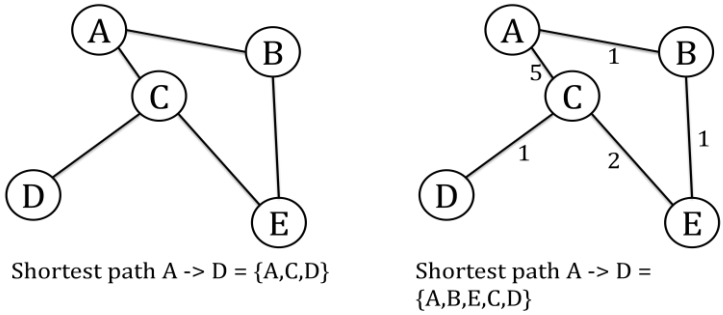


Figure 2.3 - Example of shortest paths in non-weighted (left) and weighted graphs (right).

Calculating the shortest path between a pair of vertices is by no means a trivial operation, being quite computationally heavy, although efficient algorithms have been proposed for different variants of the task. In this section, we will present two algorithms: breadth-first search (BFS) and the Dijkstra algorithms.

The breadth-first search (BFS) [1] is a graph traversal algorithm, i.e. used to visit all vertices in a graph, that can be applied to determine shortest paths. The typical BFS works as shown in Algorithm 2.1, where the inputs are a graph G and a starting node s.

Algorithm 2.1– Breadth first search

```
BFS(G, s)

Create a queue Q, initially with s
WHILE Q not empty
       Remove the head n of Q
       IF n not yet visited
              Mark n as visited
              Add all neighbors of n to queue Q
```

Unless all shortest paths are desired, BFS is normally modified to determine the path between a pair of vertices *(s, t)*. In this situation, the BFS starts with the vertex *s* as before, but stops when the vertex *t* is found. An example is shown in Figure 2.4.

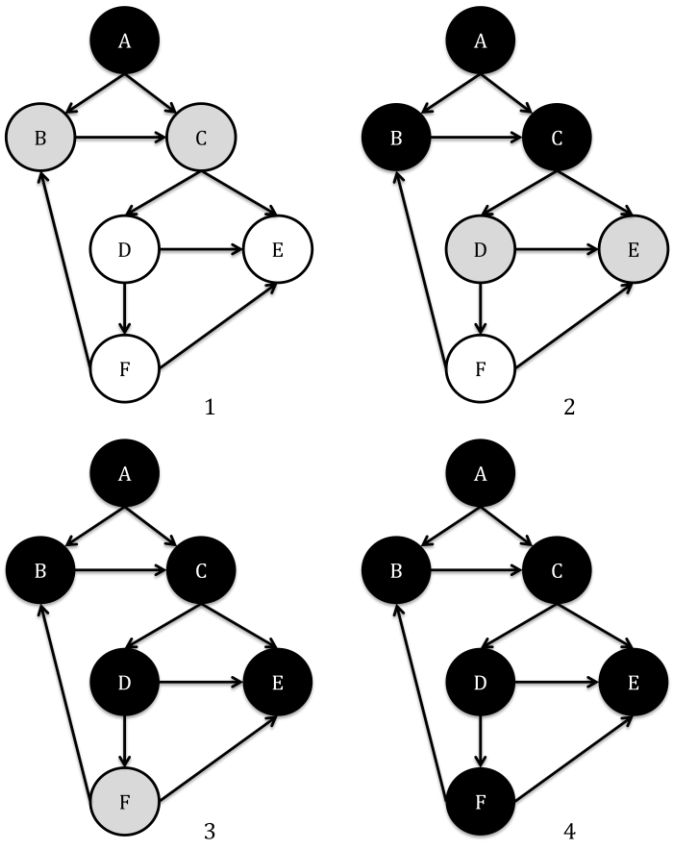


Figure 2.4 - Example of the BFS algorithm. Here, the starting vertex is A and the steps of the algorithm to visit all the vertices of the graph are shown in 1) to 4), where grey vertices are the next ones to be visited and black ones have already been visited.

The BFS does not take into account weights of the edges and as such it is usually only used for graphs with unweighted edges. It is possible to modify this algorithm to take edge weights in account. However, in this situation, the modified BFS is usually less efficient than the Dijkstra algorithm. The Dijkstra algorithm [1] is capable of finding all shortest paths between a source vertex and all the vertices connected to it. Unlike the BFS, this algorithm can be used in a weighted graph with no modifications, but requires all weights to be positive. Algorithm 2.2 shows the main steps for calculating all shortest paths from *s* to all other reachable vertices in graph *G*.

Algorithm 2.2– Dijkstra algorithm

```
Dijkstra(G, s)

FOR every vertex V in G
       Dist(V) = infinity
       Visited(V) = false
Dist(s) = 0
v = s
WHILE it is possible to visit more vertices from v
       FOR all neighbors n of v
              IF Dist(n) > Dist(v) + weight of edge (v,n)
                     Dist(n) =  Dist(v) + weight of edge (v,n)
       Visited(v) = true
       Select nv as the vertex where Visited(nv) = false and dist(nv) is
       minimal
       v = nv
```

The result of the Dijkstra algorithm is a shortest path tree with the source vertex *s* as the root and composed by the vertices connected with *s* and the set of edges that belong to shortest paths from *s*. As before, this algorithm can be altered if only one shortest path is required by stopping once the destination is reached. When dealing with unweighted graphs, the Dijkstra algorithm is less efficient than BFS, but it excels in weighted graphs. Biological networks are typically unweighted and consequently in this work the BSF was the mostly used algorithm.

2.1.3 Centrality metrics

When analyzing graphs, it is commonly necessary to determine how important individual vertices are for the graph as a whole. The act of ranking all vertices in a network is called centrality analysis and the methods used to calculate the ranks are named centrality measures or metrics. The concept of vertex importance, or centrality, is a fundamental concept, which at a first sight may seem simple to grasp. However, exactly what makes a vertex more important than others depends largely on the characteristics and purpose of the system the graph represents. For example, in Figure 2.5, vertex 1 has more neighbors than vertex 2, but if vertex 2 is removed, the graph will be divided into two non connected sub-graphs. In this situation, which vertex is more important? Well, if the system is a local computer network, the vertices are routers and the purpose

of the analysis is to find bottlenecks, then most certainly vertex 2 is more important. However, if the graph represents stations in a subway network and it the best place to build a new mall is being analyzed, then probably vertex 1 is more important.

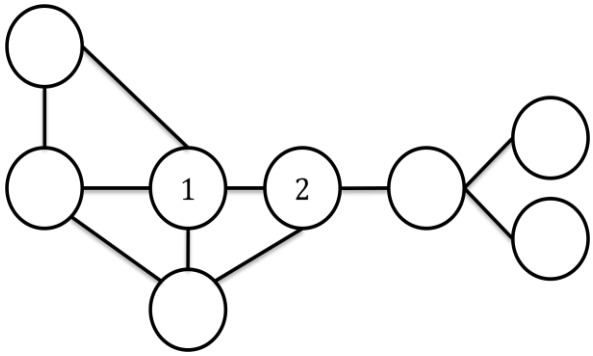


Figure 2.5 - Illustration of different criteria to assign importance to vertices (detailed explanation in the main text).

Regardless of how one defines the centrality metric, ranking vertices based on their importance is frequently an important step in graph analysis, since it allows identifying key elements of the system that could be the focus of future studies. These elements may be bottlenecks, influential individuals or groups in social networks, important proteins in biological networks or really any other type of object or structure that for some reason is central for the system represented by the graph.

In this context, there are several metrics and algorithms to determine the importance of vertices in graphs. The use of centrality measures is especially common in social networks where they are used to rank the actors according to their position in the network that is interpreted as their prominence in a social structure [2]. However, the utility of the centrality metrics is not limited to social networks and, in fact, they have been successfully used for the study of multiple types of networks, including citation networks, computer networks and, with the dawn of Bioinformatics, metabolic networks [3] and other types of biological networks.

It should be noted that some centrality measures have prerequisites regarding the type of graphs they may be applied to, varying from metric to metric. One of the most common limitations of some metrics is that they can only be applied to strongly connected graphs. Nevertheless, most restrictions can be overcome with minor alterations. For instance, the last restriction can be overcome if the individual components of the graph are treated as independent graphs. In this situation, however, the resulting centrality values must be interpreted in the context of the component and not in the context of the whole graph.

Degree centrality

The degree can be used as the most simple centrality metric by ranking vertices in descending order of their degree. Indeed, there are many systems where components with many relationships can be considered the most important or even vital for the integrity of the system. Furthermore, when dealing with directed networks, this metric can be broken down into two, the indegree and the outdegree centralities. The degree is a local centrality measure: only the immediate neighborhood of the vertex of interest is considered [1]. Because of its universality, this metric can be applied to any graph.

Betweenness centrality

The betweenness centrality (BC) is a metric whose basic idea is that an important vertex will lie on a high proportion of paths between other pairs of vertices in the network [4]. The BC of a vertex n can be calculated using the expression:

$$BC(n) = \sum_{s \neq n \neq t \in V} \frac{\sigma_{st}(n)}{\sigma_{st}} \qquad (2.1)$$

where σ_{st} is the total number of shortest paths between distinct vertices in the network and $\sigma_{st}(n)$ is the number of those paths that pass through the vertex n. Usually, when BC is used, there is some kind of flow (be it data or matter) in the network and, if the shortest paths are preferentially used, the amount of the flow that will pass though a vertex will be proportional to its BC value.

Closeness centrality

The closeness centrality (CC) is another centrality metric, which similarly to the BC, also assumes a flow in the network. However, unlike the BC, it assigns ranks according to the distances of vertices to other vertices. Essentially, the idea behind CC is that an important vertex is typically "close" to, and can communicate quickly with, the other vertices in the network [4]. In other words, if there is a flow in the network and if the shortest paths are favoured**,** then the more important vertices will be the ones to receive flux from all the other vertices in the least amount of time.

There are several variations of the closeness centrality (see [4] and [1]), mainly dealing with the impossibility of calculating it for networks that are not fully connected. In some cases, it is proposed that when calculating the metric for a given vertex, it should make use only of the shortest paths to all reachable vertices from that vertex. These variants turn the CC values into a local metric. However, assuming a well-connected network, the results are not much different from the standard CC.

The CC, as it was used in this work, is calculated as follows:

$$CC(n) = \frac{1}{\sum_{t \in V} d(n,t)} \qquad (2.2)$$

where *d(n,t)* is the distance between nodes *n, t* (the length of the shortest path) and *V* is the set of reachable nodes from *n*.

HITS

The hubs-and-authorities (also known as Hyperlink-Induced Topic Search or HITS) algorithm calculates two different metrics for each vertex in a network: the hubness and the authority. It was originally developed to rate web pages based in their content, splitting them into pages with good content on the topic, called authorities, and directory-like pages with many hyperlinks to sites on the topic, called hubs [5].

Unlike the previous metrics, the hubness and the authority values are not calculated by simply applying an expression to each vertex of the network. Instead, HITS is calculated by a recursive algorithm, in which all vertices initially have the same values of hubness and authority and iteratively these values are recalculated based in the values of their neighbours, until they converge [5]. While we are not aware of any attempt of applying HITS to biological networks, other web page analysis algorithms have been used to analyze biological networks (e.g. PageRank centrality is used in regulatory networks [6]). Also, as it will be shown later, on the case study presented in this work, HITS can return interesting biological results with certain networks.

2.1.4 Clusters

Many networks present strongly connected clusters that in many cases have a biological significance. The clustering coefficient is a metric that quantifies the inherent tendency to clustering of a vertex [7] and seen globally can provide an idea of the clustering level of an network. The clustering coefficient of a vertex provides a measure of the strength of connectivity of its neighbours, i.e. if it belongs to a connected cluster or not. This value can be calculated for a directed network as:

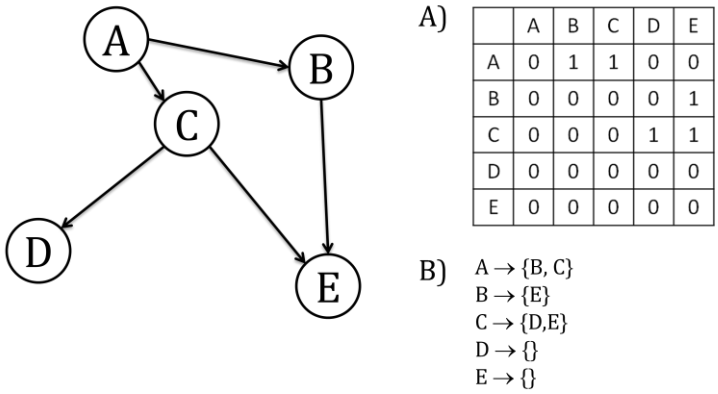
$$C(v) = \frac{e}{k\,(k-1)} \qquad (2.3)$$

where *e* is the number of edges between the neighbours of *v* and *k* is the number of neighbours of *v*. This metric essentially calculates how close a vertex and its neighbours are of forming a clique.

A graph whose vertices are joined together in tightly knit groups (clusters), between which there are only looser connections is said to have a community or hierarchical structure, being the case of several important biological networks [8].

2.1.5 Graph representations

When a graph is represented in a human readable format, normally the chosen representation is either the common graphical format, where the vertices are dots (or other symbols) and the edges are lines which connected them, or the mathematical format, *G = (V, E).* However, when a graph must be represented in a computer readable format, both these options are obviously a poor choice. There are two common types of representation, which are used to store graphs in computer readable format: *adjacency matrices* and a*djacency lists*. These formats are illustrated in Figure 2.6.



	A	B	C	D	E
A	0	1	1	0	0
B	0	0	0	0	1
C	0	0	0	1	1
D	0	0	0	0	0
E	0	0	0	0	0

Figure 2.6 - Example of a graph represented by an adjacency matrix (a) and an adjacency list (b).

Adjacency matrices are a very intuitive representation, where the graph is stored as an *n* x *n* matrix *M* of binary values, where *n* is the number of vertices. Here, if a pair of vertices *a* and *b* are connected by an edge, then M_{ab} will have a value of 1 (or true); otherwise M_{ab} will be 0 (or false). In some variants, the adjacency matrix, instead of binary values, contains some information about the edges, such as the weight associated to the edge, if it is a weighted graph.

The main problem with adjacency matrices is that they always need a lot of memory, proportional to n^2, regardless of the number of edges in the graph. This makes this format a good way for storing *dense graphs* (graphs with a large number of edges). However, when dealing with graphs with a large number of

vertices but relatively few edges (*sparse graphs*) the memory wasted is usually unacceptable. Additionally, the use of adjacency matrices results in several graph search algorithms having a longer running time.

On the other hand, in an adjacency list, each vertex of the graph has an associated array containing the list of all its neighbours. When dealing with sparse graphs, this format is quicker to search and consumes less memory than an equivalent adjacency matrix. However, in many programming languages maintaining a large set of arrays, especially if dealing with dynamic arrays, can be computationally heavier than using a matrix. Thus, in many cases, this is an unsuitable choice for dense graphs. Ultimately, the correct format to use depends on the network (number of vertices and edges) and the programming language being used.

2.2 Biological networks

There are many different types of known (and possibly unknown) biological systems that can be studied as networks and, consequently, there is a vast number of possible biological networks. While a thorough characterization of all existing types of biological networks is way beyond the scope of this project, we review in this section some of the most commonly used biological networks, focusing on cellular networks, i.e. networks that represent the behavior of a single cell. These will be the main focus of this work. Note that since the attention is focused in a type of system and not graphs in general, the term *network* will be used from now on instead of graph.

2.2.1 Metabolic networks

Cell metabolism is mainly characterized by two entities: the chemical compounds (or metabolites) that cells obtain from the environment or are capable of producing and the chemical reactions that either convert the metabolites into the necessary cellular building blocks (anabolism) or that

decompose compounds to obtain energy and other necessary compounds (catabolism).

Of all the cellular processes, metabolism is probably the one that is more intuitively represented by a network, for two reasons: firstly, it is a system composed by elements connected by shared objects (in this case reactions connected by shared compounds or compounds connected by shared reactions, depending on the point of view); secondly, the study of series of successive or tightly associated biochemical reactions for a specific metabolic purpose (metabolic pathways) is an important concept in Biochemistry [1]. Classic metabolic pathways can, themselves, be represented as small networks and, thus, it is not a great leap to combine all the pathways into a single network that contains data relevant to all the cell's metabolism.

Many of the modern developments in biological research, from the use of microorganisms to create desired chemicals to the treatment of several diseases, depend directly on the understanding (and manipulation) of metabolic processes, making metabolic networks among the most studied networks in biological research.

A complete metabolic network has to contain all the metabolic information about a cell, including all possible reactions and metabolites. Given the large amount of data involved, large-scale metabolic networks have only started to be used recently. In fact, while biochemical researchers have identified metabolic reactions and pathways for a long time, it was only in 1995 that a complete bacterial genome was sequenced [9]. Since the first genome was fully sequenced, technological advances have allowed for hundreds of other genomes to follow, and currently this information is available in databases allowing an increase in the creation and use of metabolic networks.

A metabolic network is created through the use of Bioinformatics tools and manual curation, in a process called metabolic network reconstruction [10][11]. These methods typically extract the annotated genome from information

repositories and use it to create the portfolio of reactions for a given organisms, deriving the desired metabolic network.

There are several possible variations of metabolic networks, none of which can be said to be the best, since each kind of network used depends on the type of information the researchers are interested, and the data sources used. A particularly important point of consideration when creating a metabolic network is complexity: more complex networks will contain more information, but their creation requires more detailed data, which may not always be available.

Also, since many network analysis metrics are computationally heavy, more complex networks are more difficult to analyze. Of all the variants put forward for metabolic networks, we will focus on three major alternatives of interest for this work (illustrated in Figure 2.7).

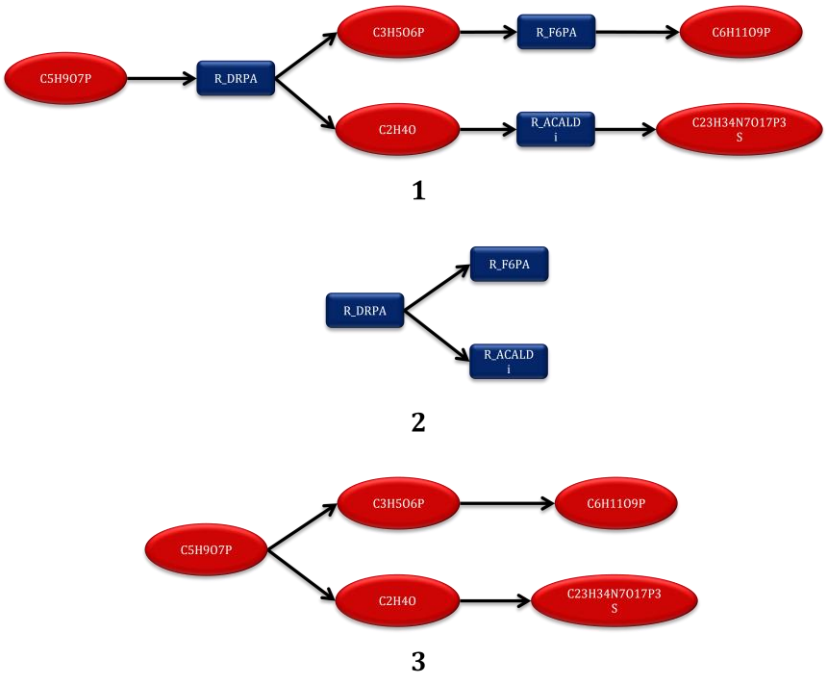


Figure 2.7 - Illustration of different metabolic networks: 1) reaction-compound network; 2) reaction-reaction network; 3) compound-compound network.

Reaction-compound networks

The most complete representation of a metabolic network is a directed bipartite graph. Here, reactions and metabolites are represented as vertices and the relationships of consumption and production are given by directed edges that either start (in case of production) or end (in case of consumption) in the reactions. This is the most complex network type, but this added complexity results in a graph that contains the maximal amount of information. Also, it does not present any of the difficulties the other models have in the calculation of paths [12].

Compound-compound networks

In this case, a network is represented by a directed graph where vertices represent the metabolites and edges represent the reactions. Here, edges connect one metabolite to another if a reaction consumes the first metabolite and produces the second. This representation directly provides information about relationships between compounds, but it is not straightforward to use it for path finding [12], because the paths found by search algorithms may have been obtained by passing through edges which represent the same reaction, thus resulting in mathematically correct but biologically unfeasible paths.

Reaction-reaction network

This representation is similar to the previous, but with reactions represented by vertices and edges pointing from reactions producing metabolites to reactions using them as a substrate. This model is useful when the main point of focus is the study of the relationships between reactions. However, it has similar problems for calculating paths as the former one [12].

All the previous alternatives are based in directed graphs. Reversible reactions in all of the three models use parallel directed edges pointing in opposite directions. This solves the representation problem but creates some complications in path calculations that will be addressed later in this work. An alternative is to use undirected graphs that can help in making easier the

application of certain topological metrics (e.g. clustering coefficients) but means losing some of the information.

Some researchers consider that the effect of the enzymes on the reactions should also be considered in the reconstruction of metabolic networks. Since enzyme data is not always necessary for all metabolic research and the inclusion of enzymatic data leads to a significant increase in the network complexity, this is not a unanimous opinion. If enzymatic data is included, these more detailed metabolic networks are directed and weighted tripartite graphs, whose three types of vertices are metabolites, reactions and enzymes with two types of edges representing mass flow and catalytic action [13].

2.2.2 Regulatory networks

Cell's metabolism is a complex web of chemical reactions that, to function correctly in a dynamic environment, must be able to adapt itself to the changing conditions. This adaptation is partially achieved thanks to the regulation of the metabolic reactions, through the presence or absence of enzymes. The synthesis of enzymes (and of all proteins) is regulated, among other processes, by the concentrations of other proteins that affect the degree of expression of the corresponding genes. It has been observed that the number of regulators grows even faster than the number of genes [14], which suggests that the alterations of the regulation process are more significant to the process of evolution than the appearance of new metabolic genes.

Gene regulation is attained by the interaction of several processes that determine which genes are expressed or inhibited and, ultimately, which proteins are synthesized. These systems can be represented as a single network or, due to the complexity involved, as a series of individual networks. It is common to treat separately the systems of signal transduction (see next section) and transcriptional regulation. Other more complex networks can also represent other types of regulation, such as post-transcriptional regulation or metabolic regulation.

Transcriptional regulation is one of the major regulatory systems, controlling the expression of genes, i.e. their transcription to mRNA and consequently the production of proteins. This mechanism is mediated by specific proteins, such as transcription factors (TFs), which bind to specific DNA sequences in the promoter regions of the genes (i.e. before the transcription start sites) to either activate or repress their expression. A single TF can regulate several genes and a gene can be regulated by more than one TF. Also, as TFs are proteins, produced by gene expression, this means that the transcriptional regulation system can be visualized as a web of interconnected genes and proteins that can be easily translated into a network.

The more direct representation of the transcription regulation system into a *transcriptional regulatory network* (TRN) is to consider both the TFs and the genes associated with them as vertices and to represent the relationships as directed edges, with associated metadata that describes the type of relationship they represent (encoding, inhibition, activation) [15]. This type of network gives a complete view of the transcriptional regulation process and it allows researchers to represent the connections between complex proteins acting as TFs and the genes that encode the simple proteins making them up. It should be noted that, while it has two types of vertices, this network is not a bipartite graph since there are connections between the simple and complex proteins.

Often, to reduce the complexity of the TRN, the transcription factors are represented by the genes encoding them [13]. In this case, the vertices only represent genes and the edges connect genes that encode transcription factors to the regulated genes. This kind of network will contain only interactions that represent the transcriptional regulation events. One of the particularities of these networks is that, due to the prevalence of transcriptional self-regulation (a TF regulates the gene that encodes either it or one of its subunits), they tend to present more self loops. Naturally, this network excludes information related with the composition of the proteins and consequently a decision must be made regarding which genes should be associated with transcription factors that are

protein complexes. In this work, it was decided to associate complex proteins with all the genes coding their components. Figure 2.8 illustrates both approaches described above to represent TRNs.

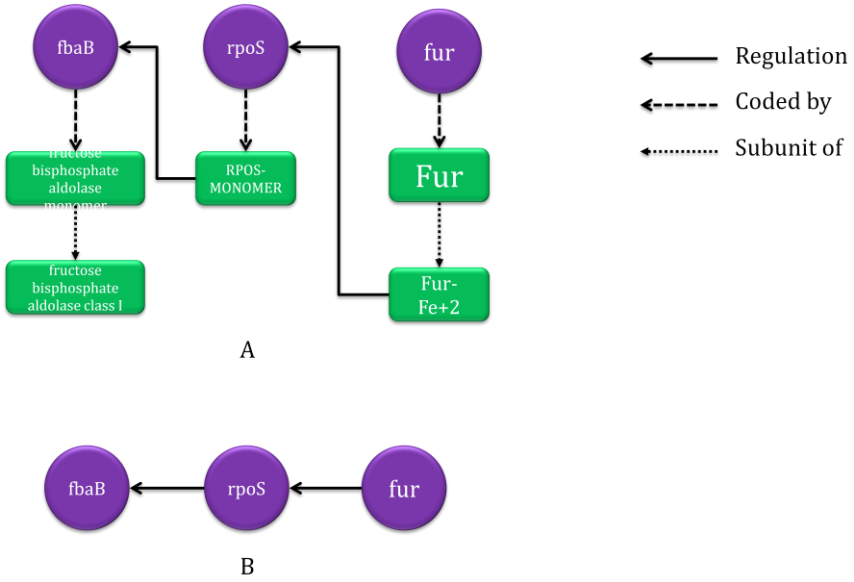


Figure 2.8 - Illustration of different transcriptional regulatory networks: a) with proteins as vertices; b) with genes only.

The analysis of several regulatory networks has shown that they share some properties with metabolic networks, as will be discussed in section 2.3. Consequently, tools and methods developed for the analysis of metabolic networks can in many cases be used on regulatory networks with minimal changes. Among biological networks, regulatory networks and specifically TRNs stand out as the most prominent targets of motif related research (see section 2.4).

2.2.3 Signalling networks

Typically, regulatory networks usually omit any information on how the cell reacts to external factors, whose presence requires adjustments in the cellular behaviour. Cells "perceive" the extracellular environment through a series of receptors. When an external ligand binds itself to a compatible receptor, that information is passed to the correct cellular elements, which alter their

behaviour as necessary. The group of processes through which a cell receives and interprets information regarding external stimuli is called signal transduction.

Signal transduction can be thought as a flux of data from the cell's receptors to the cellular elements (whose type can vary depending on the situations), which must act according to this information. Between receptors and the data's final destination there are multiple steps in which the data is progressively converted from a format to another until it is in a state which can be perceived by the target. These steps are termed signal transduction pathways and are mostly composed by reversible chemical reactions and the interaction between proteins in order to form complexes.

This heterogeneous combination results in a system in which, unlike metabolic pathways, there is only a limited mass flow, but rather it is information flowing from the receptors to their destinations. In fact, normally after they are no longer needed, most of the changes a signal transduction pathway creates are reverted and the involved molecules can be reused. By taking multiple signal transduction paths, it is possible to create signalling networks based in the signal transduction processes.

However, currently, and despite the fact that there is a large amount of information regarding signal transduction pathways, there is little knowledge on the organization and properties of signalling networks and few methods have been proposed for their analysis in a large scale [1][16][17]. The main reasons for the reduced number of studies in signalling networks, when compared to the previous examples, is the lack of defined methodologies for their reconstruction and analysis, a problem which originates from the heterogenic and fragmented nature of the available information regarding these systems and from the slow progress of the understanding on how signal pathways interact with each other as a whole [1] and from the heterogeneity of the signalling systems themselves [17].

Despite these limitations, some work has been made to attempt to determine the properties of these networks [1]. In these cases, the network obtained kept some properties of the other cellular networks explored above (see next section). Assuming that the development of these networks follows the trend set by the metabolic networks, genome-scale signalling networks should be reconstructed within the next decade [17].

2.2.4 Integrated cellular networks

All biological network types presented so far contain information about specific systems of the cell. However, it must not be forgotten that these individual components make up a larger integrated system whose comprehension requires not only an understanding of the individual parts, but also of how they are connected together and of the interactions occurring between different levels.

In fact, the Systems Biology field has been gaining importance in biological research sustaining an integrated and holistic view of the cell [18]. Thus, cellular biology is progressively moving to a research view centred on the study of the cell as a whole. Network analysis needs to keep up with these developments and new types of biological integrated networks have been considered that combine the information from several sub-systems. While the use of biological integrated networks is still in its infancy, some researchers consider that when the knowledge of properties of the biological networks which represent individual systems and of the methods for their reconstruction is consolidated, then the next logical step it to create larger integrated networks which fully describe a cell [17].

Due to the lack of standards, the complexity inherent to biological networks and the fact that the research of biological networks is still not mature, currently there has still been little research into the analysis of these integrated networks. In fact, there is no generally accepted format to these networks and their properties are mostly unknown.

Most of the current work with integrated networks is not focused in the analysis of the networks themselves, but instead the integrated systems are used to simulate the behaviour of the organisms as an alternative to the use of only metabolic systems [19][20][21][22]. Another common use of integrated biological networks, more in line with the work of this project, is the identification of composite motifs [23][24], patterns whose components are drawn from different systems.

In this work, the focus will rely on integrated networks including vertices and edges of different types (the metadata includes variables defining the kind of biological entity they represent), such as the one represented in Figure 2.9. These integrated networks can be thought of different biological networks "glued" through the interactions between their components, but in some cases the results of analyzing such networks are well beyond the sum of the analysis performed in the simpler networks.

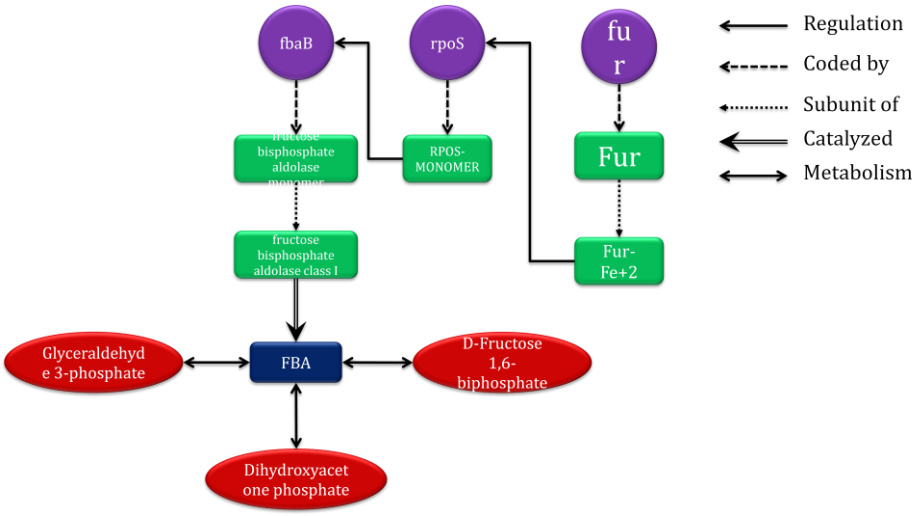


Figure 2.9 - Illustration of a possible integrated network. Genes are shown as purple circles; proteins as green rectangles; reactions as blue rectangles and metabolites as red ellipses.

2.3 Properties of biological networks

This section will be focused in some topological properties that are characteristic of biological networks and, more specifically, the ones presented in the previous section. This does not mean that these properties are not shared with many other networks in other fields. In fact, many of the properties referred next are common on several types of complex networks, ranging from areas such as social networks, computer networks, urban networks, Physics and, of course, Biology.

2.3.1 Scale-free networks and robustness

Most systems are subject to external perturbations, also called noise or attacks, depending on the domain, which impair their function and can result in a total loss of the system's functionality if they are severe. Robustness is the capacity of a system to maintain its functionalities in the face of these disruptions. When dealing with networks, one of the main disruptions is the removal of edges and/ or vertices. When a network loses its functionality it has either suffered a topological breakdown, due to having lost key vertices or edges which resulted in its fragmentation into unconnected sub-networks, or it has suffered a functional breakdown because the new configuration of the network simply does not allow it to fulfil essential actions to its function (for instance the path between two important vertices is too long or nonexistent).

Biological networks are known to show remarkable capabilities of adaptation and robustness when subject to random disruptions [25][26][13], being able to maintain their functionality, even when several vertices are removed. This makes perfect sense from an evolutionary point of view, since a severe "system failure" often means death.

The robustness of biological networks greatly derives from their structure, since biological networks are characterized by a scale-free organization [7]. In scale-free networks, the degree distribution – *P(k)* - follows a power-law, i.e. $P(k) = k^{-\gamma}$. Here, *P(k)* is the probability of a vertex having a degree *k* and γ is the degree

exponent. In practical terms, this means that these networks possess relatively few highly connected vertices (hubs) and many vertices have only few connections. As a consequence of their heterogeneity, random perturbations usually do not lead to major network breakdowns, since they have a low chance of affecting hubs.

It has been shown that networks naturally adopt a scale-free structure when they are subject to preferential attachment. In other words, when adding a new vertex, the likelihood of it connecting to another vertex *v* depends of the vertex degree of *v* [7]. This property is thought to characterize the evolution of biological systems [26].

While scale-free networks are highly resistant to random perturbations, the removal of hubs quickly results in their breakdown into isolated clusters [13], and consequently they are very vulnerable to deliberate attacks. Because of this trade-off, biological systems are sometime said to be robust, yet at the same time fragile.

The scale free property was observed it practically all cellular networks including metabolic networks [13][26][27], regulatory [13] and signaling networks [1]. These results are not surprising, considering that all of these systems exhibit great robustness, which, as explained above, is consistent with a scale-free organization.

It is interesting to note that in metabolic networks, the scale-free organization was not only consistent in all domains of life, but also when networks corresponding to the metabolism of several different organisms were compared, it was observed that the metabolites which acted as hubs were shared by most of them [26]. As for regulatory networks, in both prokaryotic and eukaryotic organisms, a scale-free distribution was observed for their outdegrees, but not in the case of the indegrees [13]. Finally, while there is relatively little information about the signalling networks properties, they appear to have a scale free distribution for both the indegrees and outdegrees [1][13].

2.3.2 Modularity and hierarchy

It is assumed that many complex networks are built up from (interacting and possibly overlapping) modules or communities [1]. These are groups of nodes that share some kind of similarity or have a close association. Biological networks are among the ones exhibiting a modular structure. Since most biological systems are composed by smaller systems (for instance the metabolism is composed by several pathways) this seems like a logical organization.

While detecting modules is an important aspect of network analysis, there is no consensus on the definition of modularity and consequently there is no standard method of identifying if a network is modular. A common definition of modularity relates with the concept of clustering, being a network considered modular if it is characterized by the presence of densely connected groups (the modules) of vertices, with only sparser connections between them [28]. Another definition is borrowed from engineering and considers a module to be a set of vertices that have strong interactions and a common function [29].

Regardless of the definition used, the identification of modules can be made in two ways: by grouping the vertices of a network according with some common feature (agglomerative methods) or by iteratively splitting the network into smaller sub-networks (divisive methods). So far, the most efficient algorithms for module identification are standard ways of clustering data, which reinforces the validity of associating the concepts of modularity and clustering.

When it was determined that biological networks are both modular and scale-free, this discovery appeared to be contradictory, because the existence of hubs with a large number of links should create a single, integrated web in which the existence of disconnected modules is not common [30]. On the other hand, the high clustering coefficient common in biological networks pointed for a modular structure.

Currently, it is believed that biological networks manage to be both modular and scale-free by having a hierarchical topology (Figure 2.10), a network organization which allows these two properties to coexist [30][31][32]. Hierarchical networks are characterized by the presence of low clustering coefficient hubs that connect the smaller clusters and of vertices with a low degree but high clustering coefficients.

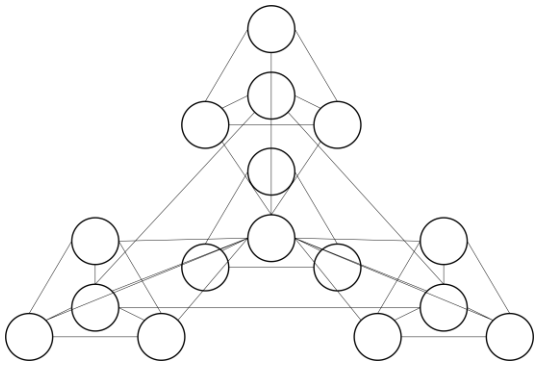


Figure 2.10 - Example of a hierarchical network.

In biochemistry, it is well established that modules consisting of several interacting reactions or metabolic pathways build discrete functional units of metabolism [1], which would point for a modular architecture for the metabolic networks. This is confirmed by network analysis, because metabolic networks normally have a high mean clustering coefficient, which suggests that they have a high degree of redundancy and cohesiveness [13] and a modular architecture [32]. As mentioned before, metabolic networks also possess a scale-free degree distribution, which combined with a modular architecture, implies a hierarchical organization. A hierarchical modular organization was also observed in mammalian, yeast and bacterial transcriptional networks [1].

A possible explanation for the modular architecture of biological networks is that modular networks can be readily reconfigured to adapt to new conditions, providing them with an advantage in environments that change over time [29].

2.3.3 Small world networks

The observation of several distinct empirical networks has revealed an unexpected property: in the majority of these networks, regardless of their size, the average shortest path length is unexpectedly small when compared to random networks of the same size. This characteristic is maintained even when network size increases. These are then said to be *small world networks* [33]. For example, within cellular metabolism, represented by a network of metabolites (vertices) linked by biochemical reactions (edges), the average path length between two metabolites is approximately d ≈ 3, an approximate value that holds independently of the specific organism [1].

The small world property was identified, initially, in the study of social networks and its name is derived from the assertion that within networks of social acquaintances (or friendships), all people (vertices) are separated from each other by just a small number of intermediate acquaintances. Subsequent studies reveled that the small world property appears to be shared by most complex networks including biological ones [13][32].

Despite being present in many complex networks, strictly speaking, the term small world is not a genuine network property, i. e., there is no measure or statistical test that allows to check whether a given specific empirical network belongs to the class of small world networks [1]. The small world can nevertheless be applied to network models whose average path length d increases slower or equal than the logarithm of the network size $d \sim \log NV$ for $NV \rightarrow \infty$[1]. Another definition of a small world network is that of a sparse graph much more highly clustered than an equally sparse random graph whose characteristic path length L is close to the theoretical minimum shown by a random graph [34].

There is also a second classification that derives from the small world networks, called the ultra-small networks, where the mean shortest path increases less or equal than the logarithm of the logarithm of the number of vertices [1]. Scale-

free networks are known to be ultra-small [35][32]; consequently, this property is shared by several biological networks. In biological networks, the ultra-small property was originally identified for metabolic networks where paths containing only three to four reactions can link most pairs of metabolites. Curiously, the same mean path length was shared by complex multicellular organisms and parasitic bacterium, which indicates that there are evolutionary mechanisms maintaining the ultra-small property [32].

Additionally, the small world property has also been observed in signal transduction networks [1].

2.4 Network Motifs

2.4.1 Definitions

In all complex networks, there are repeated patterns, called *motifs*, whose analysis provides information about the typical local interconnection patterns in the network [32]. Motifs have been described as basic building blocks and design patterns of complex networks [36], and several motifs have been shown to be functionally relevant in biological networks [1]. Essentially, a motif is a sub-graph that is overrepresented in a given network. Often, motifs are related to some kind of function that must occur in several points of the network and thus their analysis can reveal insights into the functional properties of the system.

There is no standard way of determining which patterns should *a priori* be considered as possible motifs. One way is simply to search the network for all possible combinations of *n* vertices. Alternatively, knowledge of the system can be used to attempt to guess patterns that should be present in the network.

Finding motifs in a network is typically a two step process: firstly, frequent patterns have to be identified and characterized; then, the patterns have to be validated in order to determine if their presence is statistically significant or

simply a quirk of the network construction process. Before starting to search a network for patterns, a concept of frequency must be selected, i.e. how the patterns are counted. While it may seem like a simple notion, different concepts will return different results, which in turn will influence the motifs found. There are three commonly used concepts of frequency: F_1 considers all instances of the pattern being searched regardless of shared vertices and edges; F_2 allows for vertices to be shared between patterns but not edges; finally, F_3 disallows both the sharing of vertices and edges [37].

After calculating the frequency of the possible motifs in the network, the results have to be compared with the frequency distribution of the pattern in networks following a random null model to determine if they are statistically relevant. Generating valid and meaningful null model networks is not a trivial process, as these random networks must have a structure similar to the original network. A common algorithm used to create random networks maintains the degree values of the vertices and only randomizes the connections, but sometimes more complex algorithm are used. For instance, when dealing with metabolic networks, it might be necessary for the random networks to be chemically valid.

2.4.2 Common motifs in biological networks

Each network is characterized by its own set of distinct motifs [32]. However, some motifs are typically found in certain types of networks. The examples presented in this section are some of the most commonly identified motifs in biological networks.

Feed-forward loop

The feed-forward loop (FFL) motif is one of the most common patterns identified in different types of directed networks, both in Biology and in other fields [1]. It is composed of three vertices (denoted by X, Y and Z in Figure 2.11 a) and three edges connecting those.

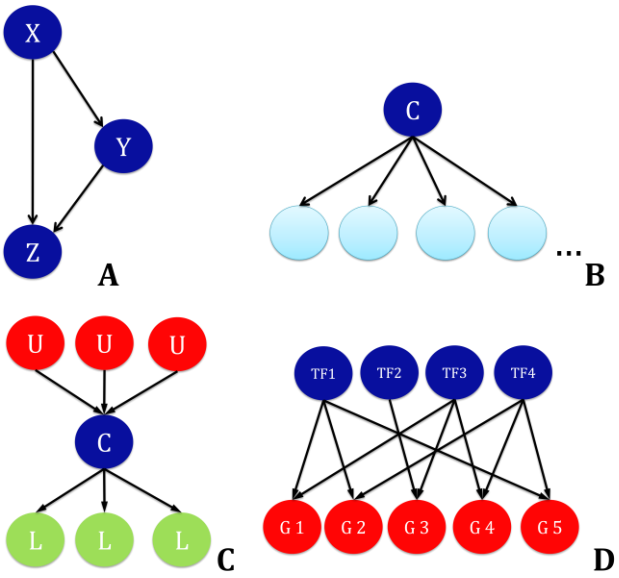


Figure 2.11 - Illustration of the configuration of different motifs: a) feed forward loop (FFL); b) single input module (SIM); c) bowtie; d) dense overlap regulon (DOR); e) extended feedback loop (EFL); f) doubly extended feedback loop (DEFL).

Each network is characterized by its own set of distinct motifs [32]. However, some motifs are typically found in certain types of networks. The examples presented in this section are some of the most commonly identified motifs in biological networks.

Feed-forward loop

The feed-forward loop (FFL) motif is one of the most common patterns identified in different types of directed networks, both in Biology and in other fields [1]. It is composed of three vertices (denoted by X, Y and Z in Figure 2.11 a) and three edges connecting those.

This motif is common in gene regulatory networks where the vertices A and B are a pair of regulators of gene C and A also regulates B. In gene regulatory networks, FFL motifs are classified according with the effect of the regulation their vertices have in each other: if the direct regulation of A has the same effect in C as the indirect regulation through B, the motif is classified as coherent; otherwise, it is classified as incoherent (Figure 2.12).

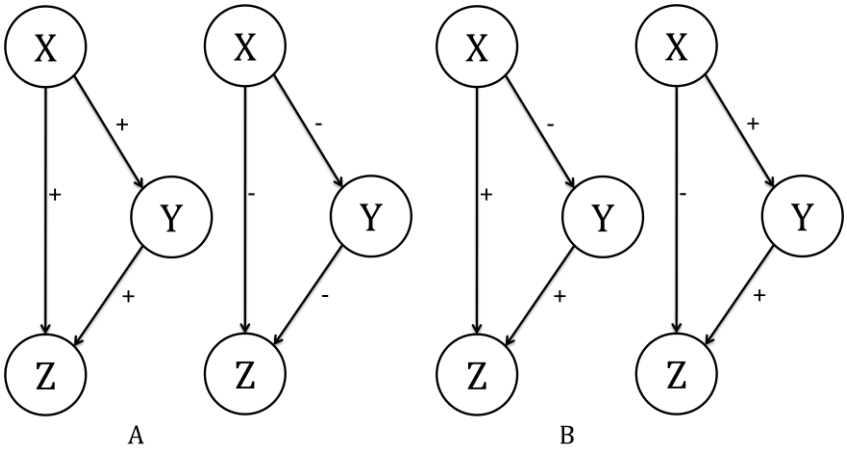


Figure 2.12 - Examples of FFL motifs: a) two examples of coherent motifs; b) two examples of incoherent motifs.

Single input modules

The single-input module (SIM) pattern is composed of a single central vertex (vertex C in Figure 2.11 b) connected to at least k other vertices by directed edges starting at C. The direction of the edges can be in the opposite direction leading to the reversed SIM pattern. This motif has been identified in TRNs [38], where the central vertex is a transcription factor controlling several operons.

Bowtie

A bowtie pattern is basically a combination of a SIM and a reversed SIM, being composed by a central vertex (vertex C in Figure 2.11 c), that receives at least k_u edges from vertices of group U and is the origin of at least k_l edges connecting C to vertices of group L.

Dense overlapping regulons

The dense overlapping regulon (DOR) pattern is composed by a series of overlapping interactions between two groups of vertices (TF and G in Figure 2.11 d), where all the edges that connect both groups start in a vertex from TF and end at a vertex from G. This motif was found in transcriptional regulation networks, where the vertices from the first group are transcription factors regulating genes/ operons. In these networks, a DOR motif is a layer of

overlapping interactions much more dense than corresponding structures in random networks [38].

2.5 Existing software for biological network analysis

There are many available applications for network analysis; some are generic and can be used in any kind of network, others were created for a specific kind of system and consequently their functionalities are optimized for a particular type of network. Also, network analysis is a vast area of study and, as such, applications tend to be specialized in a particular type of analysis that can be anything from network visualization to motif identification.

While a compressive list of all network analysis software is beyond the scope of this project, in this section some applications that can be used for analyzing biological networks and a description of their main capabilities will be presented.

2.5.1 Cytoscape

Cytoscape is an open source application for visualizing molecular interaction networks and biological pathways and integrating these networks with annotations, gene expression profiles and other data [39]. This tool is characterized by its user-friendly user interface and its plug-in based architecture, which allows users with the right technical know-how to expand its core functionalities.

The Cytoscape core distribution is a good network visualization and editing tool, which allows users to either draw new networks or modifying existing ones with ease. Every element in a Cytoscape network, including the network itself, can have associated metadata attributes, which gives users great flexibility when customizing their networks.

Cytoscape also has a functionality called "visual styles" of great interest since it allows users to customize the appearance of the networks based in the metadata attributes associated with its elements. A visual style is created by defining a series of parameters determining the colour, shape and size of the vertices and edges based in their metadata attributes. A network can have several associated visual styles and it is possible to freely switch between them. Cytoscape supports several layout algorithms, but these tend to produce bad results when dealing with large-scale networks.

By itself, Cytoscape is a good tool for drawing and visualizing networks. However, the main strength of this application lies in the large number of plug-ins developed within its framework, dramatically expanding its functionalities. To list all available Cytoscape plug-ins would be beyond the scope of this work. However, some were found to be more related with this work, namely:

- CentiScaPe [40] - Adds the capability to calculate six different centrality metrics (eccentricity, closeness, betweenness, stress, centroid and radiality); these values are determined for each vertex and for the network itself, with the results being associated as metadata to the respective objects. This plug-in can also be used to calculate the diameter and average path length of the network. Additionally, it can also present its results through a graphical output, either by highlighting vertices based in their centrality values or by generating plots.

- NetworkAnalyzer [41] - Adds the capability of calculating a large number of network topological parameters, including vertex rankers and general network metrics. This plug-in is capable of calculating network metrics treating the network being analyzed as directed or undirected. Analysis results are both presented graphically through a combination of graphs and plots or as metadata associated with the vertices and edges of the network. An interesting feature is that it is capable of generating Cytoscape visual styles based in the metadata it adds to networks, which further helps the visual analysis of the results it generates.

- ClusterViz [42], MCODE [43] and AllegroMCODE [44] - These plug-ins add the capability of cluster location. MCODE and AllegroMCODE use the same algorithm, the Molecular Complex Detection or MCODE algorithm [43], which was created for detecting densely connected regions in large protein-protein interaction networks, while ClusterViz can use two other algorithims (FAG-EC [45] and EAGLE [42]). MCODE was the first of the three to be developed, ClusterViz is essentially a version of the initial plug-in with more algorithms incorporated while AllegroMCODE is a faster and more sophisticated version which supports GPU acceleration. In all, after running, the user is presented with a graphical representation of the clusters found which can be then analyzed in more detail or exported as independent networks.

- ShortestPath [46]- A very simple plug-in for calculating shortest paths. The user must only highlight a pair of vertices and run the plug-in that finds the shortest path between them.

- EnhancedSearch [47] - Improves the search functions of Cytoscape by allowing the user to make queries to the network based in multiple attributes.

- netMatch [48] - Can be used to locate network patterns defined by the user. The patterns are defined using a graphical interface, which allows them to be drawn; specific edge and vertex attributes can also be defined. If desired, the results can be exported as independent networks.

- RandomNetwork [49] - A plug-in that adds the capability of generating random networks. It supports three random network models: Erdos-Renyi, Watts-Strogatz and Barabasi-Albert. As an alternative to generate completely new random networks this plug-in can be used to randomize existing ones.

The main limitation of Cytoscape lies in the fact that it is in its core a visualization tool, which limits its use when dealing with large networks.

2.5.2 Pajek

Pajek [50] is an application for the analysis and visualization of large generic networks, not specialized in any field. It is a very powerful tool, capable of supporting vast networks (with up to hundreds of millions of vertices as of version 2.05) and with an array of many different analysis functionalities, ranging from centrality metrics to network partition tools. This application is also capable of generating random networks using some of the main network models. Finally, it can connect with the applications SPSS and R, allowing users to easily use the capabilities of these applications to complement Pajek's functionalities.

The visualization capabilities of Pajek are less impressive than its analysis tools, and this was probably a deliberate choice from the developers. In Pajek, there is no layout necessarily associated with a network, but instead, when a network is visualized, the application draws a representation using a standard layout. This becomes unreadable if the network has many vertices or edges. The main weakness of Pajek is probably its user interface, based in a series of menus and sub-menus with limited friendliness, which combined with the large number of options available results in an application with a steep learning curve.

2.5.3 Cell Designer

CellDesigner [51] is a tool for the visualization and design of biochemical networks, being popular among biological researchers due to its user friendly interface and the fact that it supports several versions of the Systems Biology Markup Language (SBML), a popular XML format for the representation of models and networks. CellDesigner offers a series of design tools, which allows the creation of nice readable networks with comments associated with any

object. Multiple forms are available to draw edges and vertices and hypervertices are supported. Furthermore, if a user is working with a network created from a source with no layout information, CellDesigner supports multiple layout algorithms, which can be used to give a first organization of the network. Because of all these tools, a skilled user can create human readable networks with this application.

It should be noted that CellDesigner allows the creation of models (e.g. based in ODEs), therefore allowing simulations of the network in the software or connecting to more powerful simulation applications such as Copasi. Another noteworthy feature of CellDesigner is its capability to connect to existing databases to extract networks. This is not an unusual capability in network tools, but CellDesigner stands out because of the sheer number of databases it supports. Unlike the applications already presented, CellDesigner is mainly a network/ model design tool, lacking analysis features.

2.5.4 Visone

Visone [52] is an application developed for the visualization and analysis of social networks, but despite the fact that it was not developed with that objective, many of its functionalities can be used for the analysis of biological networks as well. Visone's interface follows a structure combining the visualization and analysis functionalities in an easy to use interface that allows users to easily edit the network. This structure is easy to use but it is not very efficient when dealing with large networks, because these tend to overload the visualization interface.

This application supports several centrality and clustering metrics and it can be used to create sub-networks, usually based in the values of the metrics associated with the edges and vertices. For the visualization part, it supports several layout algorithms.

2.5.5 Mfinder and mDraw

Mfinder [53] is a software tool for network motif detection, detecting frequent patterns in a network and comparing their frequency with those of random networks. Two algorithms are used in this analysis: full enumeration of sub-graphs [36] to detect frequent patterns and a sampling of sub-graphs for estimation of sub-graph concentrations [54] to verify if the patterns are motifs. It is a powerful tool that identifies all motifs of a size selected by the user allowing the user to define the parameters of the motif sampling and random network generation process. It employs methods of random network generation that preserve the vertex degree.

While Mfinder runs only in command line, its developers have also made a network visualization tool, mDraw [53], which is capable of communicating with Mfinder, thus acting not only as a network viewer but as an interface to Mfinder. MDraw is for most part a simple application, which gives the users little customization of the network appearance. However, it supports good layout algorithms and, if connected with Mfinder, it can enlighten found motifs in the network giving a visual support to the found data.

2.5.6 Vanted

VANTED[55][56] (Visualization and Analysis of Networks containing Experimental Data) is an application for both network creation and visualization and experimental data analysis. It was created to be used with biological pathways or functional hierarchies. VANTED is characterized by solid graphical capabilities and robust data integration methods. In VANTED, networks can either be manually created or imported from files in one of the supported formats (which include SBML) and a large number of layout algorithms are supported. As for the customization of the network's appearance, several options are available for the shape and colour of the vertices and edges, which although not as extensive or user friendly as those of Cytoscape or CellDesigner, are better than those of most applications available.

One of the most interesting features of VANTED is that pathway networks can be directly imported by the application from external data repositories. In the current release (2.0), the repositories supported are KEGG [57], RIMAS [58] and MetaCrop [59], which gives users a direct access to a good number of already made networks without the usual work of data extraction and network reconstruction.

VANTED becomes really useful when there is experimental data to associate with a network. The process of association itself is quite sophisticated, since the data is assigned to the vertices and edges they belong, taking into account synonymous and if the necessary elements are not present they are added into the network. This means that a whole network can possibly be constructed only from experimental data. Once a vertex or edge has associated experimental data, a graphical representation of that information (usually a chart) is added to it. After adding experimental data to a network, the statistical functions of VANTED can be used to evaluate them, using t-test or correlation analysis.

Similarly to other applications mentioned, VANTED's functionalities can also be extended through plug-ins (referred in VANTED's site as add-ons). Currently, there are a few but useful add-ons of which three in particular are related with this work:

- HIVE - An add-on extending the types of biological networks supported by VANTED. It adds the capability of handling omics data and provides tools for the integration of new information and networks. These new types of networks brought by HIVE include gene regulatory, metabolic, signal transduction and protein interaction networks.

- FluxMap - A graphical add-on allowing the visualization of measured or simulated fluxes in the network by adjusting the appearance of the edges (usually the thickness).

- FBASimVis - An add-on which allows VANTED to perform simulations with the models represented by the networks, in this case using Flux Balance Analysis. FBASimVis allows both wild type and knockout simulations and after a simulation, it modifies the graphical representation of the network to identify the fluxes and the lethal reactions.

With excellent graphical capabilities, data association methods and statistical functions, VANTED is an excellent tool for researchers that must handle both metabolic networks and simulated or experimental data. Its only flaw is that it depends on the graphical representations, which can be a problem when dealing with large networks.

2.5.8 CentiBiN

CentiBiN [60] is a software tool created for centrality analysis in biological networks based in the JUNG library. The graphical capabilities of CentiBiN are very limited, since it simply draws networks using a default format which cannot be customized by the user. Several layout algorithms are supported. The only metadata that can be associated with networks are the vertices identification.

CentiBiN can be used to generate random networks using several algorithms commonly used in biological networks. This functionality is useful in centrality analysis, where it is common to compare the values obtained with those of random networks.

As for centrality metrics, the application supports a large number of vertex centralities, 16 for directed networks and 17 for undirected networks, plus three graph centralities (metrics which return values relative to the network as a whole).

Additionally, users can also use CentiBiN to prepare networks for centrality analysis by applying a small number of pre-programmed functions which

transform networks in ways that allow metrics which would be otherwise incompatible to be used (for example, by removing loops or converting a directed network into an undirected one).

While CentiBiN is a solid tool for those interested in centrality based network studies, it offers few functionalities for other types of network analysis.

2.5.8 Limitations of current applications

The main problem with most applications aimed for the analysis of biological networks (e.g. Cytoscape) is that they were developed with an emphasis in network visualization. The classic user interface of a biological network analysis application includes a window with a central panel where the network is visualized and a group of buttons or menus allowing users to perform analysis operations. This structure is common in all kinds of network analysis applications, but brings some problems when dealing with large-scale networks, because of the difficulty in creating acceptable layouts when a large number of vertices and edges are present. In fact, while there are automatic layout algorithms, the ones currently known have proven to be incapable of solving this problem. This fact, combined with the increase of computational power necessary to deal with the representation of a large-scale networks, means that when dealing with these networks, these applications can waste a significant quantity of memory (possibly enough to cause slow down if the networks are sufficiently large) to produce an output which is essentially useless.

Some applications (e.g. Mfinder) avoid the trap of network visualization by separating the visualization from the main tool or completely omitting it. However, these tend to be specialized tools with a narrow range of functionalities. Naturally, this limits their use to only researchers engaged in specific fields.

Generic applications created for analyzing large-scale networks from a pure mathematical point of view (e.g. Pajek) offer a possible way to avoid the

visualization problem. Using generic applications brings its own set of problems, however. Firstly, most are intended for use with a scientific background significantly different from most biological researchers, which can result in a steep learning curve. Even if the initial difficulties are overcome, generic applications are naturally not well adapted to the characteristics of the biological networks, and consequently the metrics they support may not be valid for biological networks.

It can be concluded that many of the limitations of the tools for analysis of biological networks result from the fact that they were not developed with large-scale networks in mind. Consequently, while there are currently good applications to analyze small and medium sized biological networks, new applications are necessary for the study of biological large-scale networks,

References

[1] B. H. Junker. and F. Schreiber, Ed., Analysis of Biological Networks. Wiley & Sons, 2008.

[2] U. Brandes, "A faster algorithm for betweenness centrality," Journal of Mathematical Sociology, vol. 25, no. 2, pp. 163-177, 2001.

[3] M. Newman, "A measure of betweenness centrality based on random walks," Social networks, vol. 27, pp. 39-54, 2005.

[4] O. Mason and M. Verwoerd, "Graph Theory and Networks in Biology," Systems Biology, IET In Systems Biology, IET, Vol. 1, No. 2. (March 2007), pp. 89-119.

[5] M. Henzinger, "Hyperlink analysis for the web," Internet Computing, IEEE, vol. 5, pp. 45-50, 2001.

[6] D. Koschützki and F. Schreiber, "Centrality analysis methods for biological networks and their application to gene regulatory networks.," Gene regulation and systems biology, vol. 2, pp. 193-201, Jan. 2008.

[7] R. A. and A.-L. Barabási, "Statistical mechanics of complex networks," Rev. Mod. Phys. 74, 47–97 (2002).

[8] M. Girvan, "Community structure in social and biological networks," Proceedings of the National Academy of Sciences of the United States of America, Vol. 99, Issue 12 pp. 7821-7826, 2002.

[9] RD. Fleischmann, MD. Adams, O. White, RA. Clayton, EF. Kirkness, AR. Kerlavage, CJ. Bult, JF. Tomb, BA. Dougherty, JM. Merrick, , "Whole-genome random sequencing and assembly of Haemophilus influenzae Rd," Science, no. 28, pp. 496-512, 1995.

[10] H. Ma and A. P. Zeng, "Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisms," Bioinformatics, vol. 19, no. 2, pp. 270-277, 2003.

[11] C. Francke, R. J. Siezen, and B. Teusink, "Reconstructing the metabolic network of a bacterium from its genome.," Trends in microbiology, vol. 13, no. 11, pp. 550-8, Nov. 2005.

[12] J. van Helden, L. Wernisch, D. Gilbert, and S. J. Wodak, "Graph-based analysis of metabolic networks.," Ernst Schering Research Foundation workshop, no. 38, pp. 245-74, Jan. 2002.

[13] R. Albert, "Scale-free networks in cell biology.," Journal of cell science, vol. 118, no. Pt 21, pp. 4947-57, Nov. 2005.

[14] E. V. Nimwegen, "Scaling laws in the functional content of genomes: Fundamental constants of evolution?," Mol. Biol. Evol. 1998;15:583-589.

[15] T. I. Lee et al., "Transcriptional regulatory networks in Saccharomyces cerevisiae.," Science (New York, N.Y.), vol. 298, no. 5594, pp. 799-804, Oct. 2002.

[16] S. Klamt, J. Saez-Rodriguez, J. a Lindquist, L. Simeoni, and E. D. Gilles, "A methodology for the structural and functional analysis of signaling and regulatory networks.," BMC bioinformatics, vol. 7, p. 56, Jan. 2006.

[17] D. R. Hyduke and B. Ø. Palsson, "Towards genome-scale signalling-network reconstructions," Nat Rev Genet, vol. 11, no. 4, pp. 297-307, Feb. 2010.

[18] S. Klamt and J. Saez-Rodriguez, "Structural and functional analysis of cellular networks with CellNetAnalyzer," BMC Systems Biology, vol. 13, pp. 1-13, 2007.

[19] Covert, M.W., Knight, E.M., Reed, J.L., Herrgard, M.J., and Palsson, Acknowledgments B.O. (2004). Integrating high-throughput and computational data elucidates bacterial networks. Nature 429, 92–96.

[20] M. J. Herrgård, B.-S. Lee, V. Portnoy, and B. Ø. Palsson, "Integrated analysis of regulatory and metabolic networks reveals novel regulatory mechanisms in Saccharomyces cerevisiae.," Genome research, vol. 16, no. 5, pp. 627-35, May 2006.

[21] E. Klipp, B. Nordlander, R. Krüger, P. Gennemark, and S. Hohmann, "Integrative model of the response of yeast to osmotic shock.," Nature biotechnology, vol. 23, no. 8, pp. 975-82, Aug. 2005.

[22] J. F. Moxley et al. Moxley, Joel F, J.C. Michael, M.R. Antoniewicz, S.G. Villas-boas, H. Alper, R.T. Wheeler, L. Tong, A.G. Hinnebusch, T. Ideker, J. Nielsen, G. Stephanopoulos, "Linking high-resolution metabolic flux phenotypes and transcriptional regulation in yeast modulated by the global regulator Gcn4p," Proceedings of the National Academy of Sciences of the United States of America vol. 106, no. 16, pp. 6477-6482, 2009.

[23] E. Yeger-Lotem et al., "Network motifs in integrated cellular networks of transcription-regulation and protein-protein interaction.," Proceedings of the National Academy of Sciences of the United States of America, vol. 101, no. 16, pp. 5934-9, Apr. 2004.

[24] X. Zhu, M. Gerstein, and M. Snyder, "Getting connected: analysis and principles of biological networks.," Genes & development, vol. 21, no. 9, pp. 1010-24, May 2007.

[25] N. Barkai and S. Leibler, "Robustness in simple biochemical networks to transfer and process information," Nature, pp. 913-917, 1997.

[26] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and a.-L. Barabasi, "The large-scale organization of metabolic networks," Nature, pp. 651-654, Oct. 2000.

[27] R. Tanaka, "Scale-Rich Metabolic Networks," Physical Review Letters, vol. 94, no. 16, p. 168101, 2005.

[28] M. Newman, "Modularity and community structure in networks," Proceedings of the National Academy of, vol. 103, no. 23, pp. 8577-82, Jun. 2006.

[29] U. Alon, "Biological networks: the tinkerer as an engineer.," Science (New York, N.Y.), vol. 301, no. 5641, pp. 1866-7, Sep. 2003.

[30] A.-L. B. E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, "Hierarchical organization of modularity in metabolic networks," science, pp. 1-18, 2002.

[31] E. Ravasz, "Hierarchical organization in complex networks," Physical Review E, vol. 67, no. 2, p. 026112, 2003.

[32] A.-L. Barabási and Z. N. Oltvai, "Network biology: understanding the cell's functional organization.," Nature reviews. Genetics, vol. 5, no. 2, pp. 101-13, Feb. 2004.

[33] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks.," Nature, vol. 393, no. 6684, pp. 440-2, Jun. 1998.

[34] A. Wagner and D. a Fell, "The small world inside large metabolic networks.," Proceedings. Biological sciences / The Royal Society, vol. 268, no. 1478, pp. 1803-10, Sep. 2001.

[35] R. Cohen, "Scale-free networks are ultrasmall," Physical Review Letters, 2003.

[36] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks.," Science (New York, N.Y.), vol. 298, no. 5594, pp. 824-7, Oct. 2002.

[37] F. Schreiber, "Frequency concepts and pattern detection for the analysis of motifs in networks," Transactions on Computational Systems Biology, 3 (LNBI 3737):89–104, 2005

[38] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, "Network motifs in the transcriptional regulation network of Escherichia coli," Nature genetics, vol. 31, no. 1, pp. 64-68, May 2002.

[39] M. E. Smoot, K. Ono, J. Ruscheinski, P.-L. Wang, and T. Ideker, "Cytoscape 2.8: new features for data integration and network visualization.," Bioinformatics (Oxford, England), vol. 27, no. 3, pp. 431-2, Feb. 2011.

[40] G. Scardoni, M. Petterlini, and C. Laudanna, "Analyzing biological network parameters with CentiScaPe.," Bioinformatics (Oxford, England), vol. 25, no. 21, pp. 2857-9, Nov. 2009.

[41] M. Assenov, Y., Ramírez, F., Schelhorn, S.E., Lengauer, T., Albrecht, "Computing topological parameters of biological networks," Bioinformatics, vol. 24, no. 2, pp. 282-284, 2008.

[42] "ClusterViz Webpage." [Online]. Available: http://code.google.com/p/clusterviz-cytoscape/.

[43] G. Bader, "An automated method for finding molecular complexes in large protein interaction networks," BMC bioinformatics, vol. 27, pp. 1-27, 2003.

[44] "AllegroMCODE webpage." [Online]. Available: http://www.allegroviva.com/allegromcode.php.

[45] J. C. Min Li, Jianxin Wang, "A fast agglomerate algorithm for mining functional modules in protein interaction networks.," BMEI, pp. 3-7, 2008.

[46] "Cytoscape plug-in list." [Online]. Available: http://chianti.ucsd.edu/cyto_web/plugins/.

[47] M. Ashkenazi, G. D. Bader, A. Kuchinsky, M. Moshelion, and D. J. States, "Cytoscape ESP: simple search of complex biological networks.," Bioinformatics (Oxford, England), vol. 24, no. 12, pp. 1465-6, Jun. 2008.

[48] "NetMatch webpage." [Online]. Available: http://ferrolab.dmi.unict.it/netmatch.html.

[49] "RandomNetworks webpage." [Online]. Available: http://sites.google.com/site/randomnetworkplugin/.

[50] "Pajek's website." [Online]. Available: http://pajek.imfm.si/doku.php.

[51] "Celldesigner's website." [Online]. Available: http://www.celldesigner.org.

[52] "Visone's website." [Online]. Available: http://visone.info.

[53] U. Alon, "Uri Alon Lab - Design principles of protein networks." [Online]. Available: http://www.weizmann.ac.il/mcb/UriAlon/.

[54] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, "Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs.," Bioinformatics (Oxford, England), vol. 20, no. 11, pp. 1746-58, Jul. 2004.

[55] B. H. Junker, C. Klukas, and F. Schreiber, "VANTED: a system for advanced data analysis and visualization in the context of biological networks.," BMC bioinformatics, vol. 7, p. 109, Jan. 2006.

[56] "VANTED's webpage." [Online]. Available: http://vanted.ipk-gatersleben.de/.

[57] "KEGG's website." [Online]. Available: http://www.genome.jp/kegg/.

[58] "RIMAS' webpage." [Online]. Available: http://rimas.ipk-gatersleben.de/.

[59] "MetaCrop's webpage." [Online]. Available: http://pgrc-35.ipk-gatersleben.de/pls/htmldb_pgrc/f?p=269:111:3792815449513584.

[60] B. H. Junker, D. Koschützki, and F. Schreiber, "Exploration of biological network centralities with CentiBiN.," BMC bioinformatics, vol. 7, p. 219, Jan. 2006.

Chapter 3

InBiNA – an open-source tool for the analysis of integrated biological networks

In this chapter, the core software tool developed in this work, the Integrated Biological Network Analysis (InBiNA) application, will be presented. The focus will be on the full presentation of the functionalities implemented in the tool and the demonstration of its usefulness in selected case studies for the analysis of integrated cellular networks. The methodologies followed in the implementation and related details are left to chapter 5.

3.1 Functionalities of the tool

InBiNA handles large-scale integrated biological networks, allowing different types of vertices and edges to co-exist in a single network. Networks in InBiNA can be created from distinct sources and also exported in different formats. It supports a number of tools for network topological analysis including vertex degree, shortest path, clustering and sub-graph analyses, distinct types of vertex rankers based on centrality metrics and pattern detection.

Also, the tool enables the application of different filters to the network, creating new networks for additional analyses, while also allowing the comparison of different networks in several perspectives. All these will be described in detail in the next sub-sections.

The tool is made freely available together with extensive documentation in the web site (http://darwin.di.uminho.pt/inbinawiki). The main operations of the tool are explained in a set of "How To's" with simple examples and screenshots.

The complete source code is also available in the Sourceforge web site (http://sourceforge.net/projects/inbina/files).

3.1.1 Network definitions

All networks in InBiNA share some common properties:

- All vertices and edges must have a type, normally used to identify the kind of biological entity or relationship it represents.
- Every vertex or edge type can have associated metadata fields (for example the vertex type "protein" has the metadata field "EC number").
- Every vertex must have a name and a unique id.
- Edges are directed (bilateral relationships are represented though parallel edges pointing in opposite directions).

Typical vertex types include biological entities like genes, RNA, proteins, compounds, reactions, among others, while edges usually represent interactions between these entities, such as protein encoding by genes, transcriptional or metabolic regulation of proteins, metabolites consumed/produced in reactions, enzymes catalysing reactions, etc.

3.1.2 Network creation and export

In InBiNA, networks are created though importation of the necessary data from files in one of the supported formats, namely:

- SBML level 2 – The Systems Biology Markup Language (SBML) [1][2] is a free and open XML format developed for the representation of biological processes. InBiNA can extract the information present in SBML files and

use it to create metabolic networks. SBML is a popular format for representing networks used by many applications and, consequently, it has become the closest thing to a de facto standard among researchers.

- SIF - A popular ASCII network file format, which was designed for biological networks. SIF is a generic network file format containing in the main file the network. Any metadata associated with the vertices or edges are stored in additional files (NA and EA files respectively, described in the documentation of the tool), with each file containing data related with one vertex or edge attribute. InBiNA can read both the SIF files containing the networks structure and the attribute definitions.

- MBNF (Multiple Biological Network Format), a flat file ASCII format developed for InBiNA, it was designed to be a format whose direct observation can give users an idea of the network's structure. In this format only a single file is needed to represent the network and metadata that can easily be associated to the vertices or edges. The full description of this format is included in supplementary material, being also available within the documentation available in the web site.

When using the application, networks and the results of the analyses performed can be stored in binary files. Also, InBiNA allows the exportation of the networks created into the SIF, MBNF and XGMML (an XML based format supported by various applications) file formats.

Since InBiNA was designed to work mainly with large-scale networks and due to the difficulties involved in representing them graphically, it was opted for not including an in-application visualization tool. However, the exportation tools allow to easily use other applications, such as Cytoscape, to visualize the networks, provided that they are small enough for the visualization to be possible.

3.1.3 Network filters and bypasses

Biological networks tend to be quite large and this can be a problem if only part of the network is of interest to the analysis. To handle these situations, InBiNA includes filtering functionalities, which can be used to obtain sub-networks by selectively removing parts of the original network.

The filtering tools of InBiNA can be divided into regular or bypass filters, according to how they generate sub-networks. The former creates sub-networks through the simple removal of vertices and/or edges from the original network based in a series of parameters selected by the user, while bypass filters provide added flexibility by re-arranging the network through the redefinition of the connections between the remaining vertices. This allows applying transformations of the original network using transitive operations.

Regular filters

Regular filters can be used to remove parts of the network that are not considered important for a set of analyses. Illustrative examples are the removal of currency metabolites [3] from metabolic networks through the removal of vertices with a high degree or the removal of isolated vertices (i.e. vertices with a degree equal to zero) in the network. InBiNA currently supports five types of regular filtering operations:

- Removal of vertices whose degree is above a maximum user defined threshold value (it can discriminate between in and out degree).

- Removal of the N vertices in the network with the higher degree (N is a user defined parameter).

- A variation of the first filter that can use any ranker (see below) to select the vertices to remove.

- Removal of vertices and edges of selected types, allowing the creation of sub-networks including only some biological entities or interactions.

- Removal of all vertices and/ or edges from a user defined list. It can be used to make custom tailored changes to the network.

Bypass filters

Bypass filters work by applying a series of user selected operations to the network which, similarly to the regular filters, remove selected vertices or edges, but also redirect existing edges or create new ones. A bypass filter can contain as many bypass operations as the user considers necessary. The current version of InBiNA supports two possible bypass operations:

- Normal bypass operations – Here, the user selects three vertex types, the start (S), middle (M) and end (E), and two edge types which should connect S to M (T1) and M to E (T2). The user also defines a new edge type R. The operation eliminates all edges of type T1 that connect vertices of types S to M and all edges of type T2 connecting vertices of type M to E, creating new edges of the type R connecting S to E. Figure 3.1a) illustrates the method.

- Replacement bypass operations: Here, the user selects two types of vertices, the replacement (R_v) and original (O_v) and a type of edge (Ed). The operation removes all vertices of type O_v that are connected to vertices type R_v by edges of type Ed, redirecting all existing edges which were connected to the original vertices to the replacement ones. If there are multiple potential replacement vertices for an original vertex the altered edges will be replicated so that they connect to all replacement vertices. The process is illustrated in Figure 3.1b).

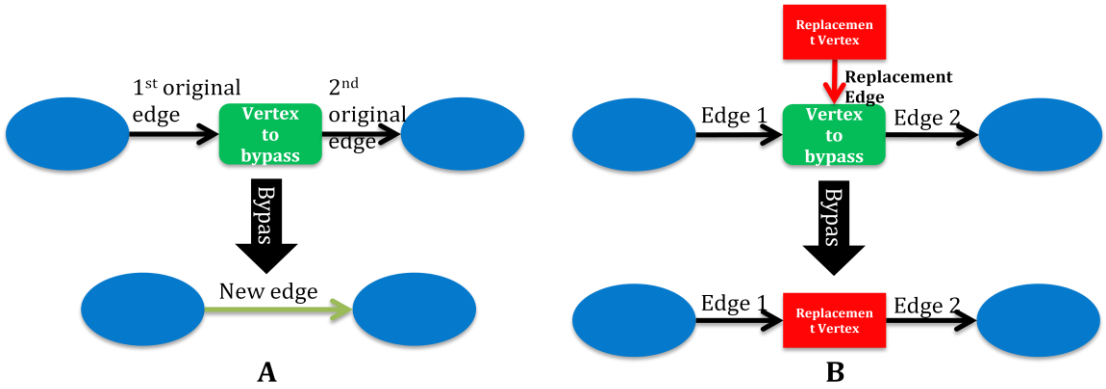


Figure 3.1 - Depiction of the bypass operations implemented: a) regular bypasses; b) replacement bypasses.

Additionally, when using bypass filters, the user must also select how the different bypass operations will be applied to the network: simultaneously or sequentially. This choice will determine the computational weight of the filter. In simultaneous bypass filters, all operations defined by the user are applied simultaneously, being an alternative that is more efficient in terms of computational power. However, if conflicting operations exist, the resulting network can be distinct depending on the order of the operations. Sequential bypass filters allow the user to specify the correct order of the transformations, guaranteeing consistency of the results regardless of existing conflicts, but they are computationally more demanding, since intermediate networks are generated after each operation.

InBiNA includes some predefined bypass filters which can be useful for users studying biological networks. These predefined filters can be applied to any network that contains the edge and vertex types defined, also helping to illustrate common uses of bypass filters. The available predefined bypass filters are the following:

- Connect reactions with “shared” compounds: it executes a normal bypass filter with start and end vertex types as “reaction” and the intermediate vertex as “compound”, while the edge types are “produced metabolite” and “consumed metabolite”; this filter connects reactions that produce

and consume the same metabolite, converting a metabolic network with reaction and compound vertices into a reaction-reaction network.

- Connect metabolites with "shared" reactions: similar to the previous one, but the roles of compounds and reactions are reversed, thus creating a compound-compound network.

- Replace reactions by their catalyzing enzymes: this is a replacement bypass, where reactions are replaced by the enzymes that catalyze them, i.e. proteins that are connected to the enzymes through edges of type "Catalyzes".

In both regular and bypass filters, after creating a new network all disconnected vertices (with a zero degree) are automatically removed from the resulting network.

3.1.4 Basic topological metrics: degrees, shortest paths and sub-graphs

InBiNA can be used to perform the basic topological analysis operations over a network: degree, shortest path and sub-graph analysis.

Degree analysis

The degree analysis functionality provides the user a table view containing the degree of all vertices in a network, discriminated by in- and out-degree. Additionally, the same view can also be used to identify the list of neighbours of a selected vertex through a pop-up window, providing an idea of that vertex relative position in the network. A global analysis is also provided in the form of degree histograms, either in table format or drawn as a chart. Finally, InBiNA also calculates the degree distribution of the whole network, a simple metric that allows to identify scale-free networks [4] (see section 2.3.1).

Shortest path analysis

Besides being able of calculating the length of the shortest paths between pairs of selected vertices, InBiNA supports several other functionalities based in the shortest paths, including global network metrics, vertex specific metrics and a shortest path histogram. Global network metrics include:

- Network diameter: the longest shortest path of the network (if the network is not fully connected it will have a value of infinity);
- Longest shortest path between two vertices that are connected (if the network is fully connected it will be equal to the diameter);
- Mean shortest path length, calculated over all pairs of connected vertices in the network.

At the level of a single vertex, InBiNA can be used to determine all the other vertices connected to it and show the distance between them in a specific table view. Additionally, a few metrics related to the individual vertex can also be calculated:

- Longest shortest path that starts/ends in the vertex;
- Number of vertices that the vertex is connected to/ are connected to the vertex;
- Average shortest path length that starts/ ends in the vertex.

Subnetwork identification

Often, networks are fragmented, which can be a problem when using analysis metrics that classify vertices based in their relationship with others. When these metrics are applied to fragmented networks, the vertices will be ranked within the modules they belong to and not the whole network.

While simple observation is often enough to determine if a small network is fragmented, it can be hard to manually identify fragments in large-scale networks. InBiNA is capable of determining the independent modules in a network automatically and presents the user with tables identifying the vertices and edges belonging to each component. Independent modules identified can be turned into independent networks for further analysis using other functionalities.

3.1.5 Ranking algorithms: centralities and clustering coefficients

When developing InBiNA, ranking algorithms was a term used to group all methods that provide a ranking of the vertices in the network, based in a numerical value. Many methods of network analysis can be considered within this class. The ones that were selected to be included in InBiNA were those that return values that can be considered to be biologically meaningful when applied to some kind of biological network. Currently, InBiNA supports the following types of ranking algorithms: degree centrality, betweenness centrality, closeness centrality, HITS and clustering coefficients. These metrics have been previously described in sections 2.1.3 and 2.1.4.

3.1.6 Finding motifs/ patterns

In section 2.4, network motifs were defined and their basic concepts were put forward. Also, a number of common motifs in biological networks have been identified and described.

Identifying new motifs whose frequency is statistically significant is not InBiNA's focus. However, the application is capable of searching networks for the presence of user-defined patterns that can be statistically significant motifs or not. Indeed, these can correspond to previously identified motifs in other works or to structures that, while not verified as significant in statistical terms, are considered relevant to the user. It is important to notice that the user can define

the types of the network elements, which should compose the motifs being searched for. This feature is useful when dealing with integrated networks, because in these cases similar motifs/patterns composed of different elements can have different biological meaning.

InBiNA supports the search for a number of pre-defined patterns that include the ones described in section 2.4.2, namely: the feed-forward loop (FFL), the single input module (SIM), the bowtie and the dense overlap regulons (DOR). In all these cases, InBiNA allows the user to restrict the search by specifying the types of the vertices and of the edges involved in each case. In the case of the SIM motif, the user can invert the direction of the edges, thus searching for a reversed SIM motif. In the case of the bowtie and the DOR, the user can also restrict the search by specifying the cardinality of the sets of vertices involved.

Additionally, InBiNA also includes the search for other types of patterns of interest, named *feedback cycles*. The extended feedback loop (EFL) and the double extended feedback loop (DEFL) (Figures 3.2a and 3.2b, respectively) are cyclic patterns that were not identified as significant motifs in any previous study. However, these are structures which were found to be important during this work, in collaboration with other members of the host research group. These patterns can be though as extensions of the FFL motif.

The EFL pattern is composed by at least two vertices S and E, connected by an edge from E to S and by a path in the network in the reverse direction (from S to E), which can contain an arbitrary number of intermediate vertices.

The DEFL is a natural extension of EFL, where the connection of E to S does not have to be a direct edge but can instead also be a path in the network with a variable number of vertices.

One of the applications of the EFL patterns in metabolic networks is to identify cases where one of the compounds in the end of a metabolic chain acts as a metabolic regulator for an enzyme catalysing a reaction in the start of that chain.

The DEFL pattern was found in similar situations, but in this case the effect is indirect. Due to the computational weight involved in locating instances of either EFL or DEFL, users must always select specific types of the S and E. The selection of the edges, on the other hand, is more flexible, since the user can select any number of edge types.

While the presented motifs and patterns are among the most commonly searched for in biological networks, they are by no means the only ones that can be of interest. In fact, it is impossible to create an extensive list of all the possible patterns that can be of interest to researchers, much less to create an application which implements the search from them all as individual functionalities. For this reason, InBiNA contains a functionality that gives users the capability of detecting instances of any pattern which they can define. The user must first write the pattern in an ASCII file according to a given syntax, the file is then loaded and processed by InBiNA and its data is used for detecting the pattern. The full description of the format for pattern files is included in the documentation available in the web site.

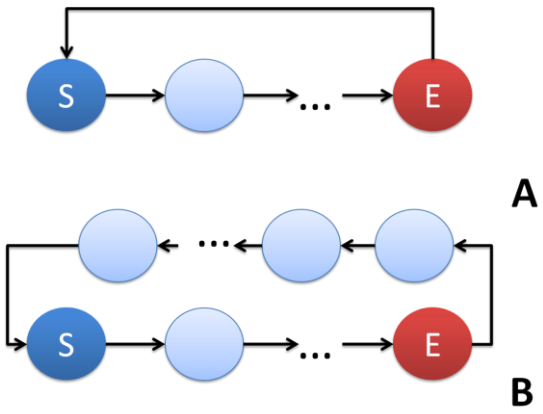


Figure 3.2 - Illustration of the feedback patterns identified by InBiNA: a) extended feedback loop; b) double extended feedback loop. S and E are the start and endpoint vertices, respectively, while the light blue vertices are the intermediary vertices in the paths between S and E.

3.1.7 Network comparison

Frequently, biological entities are shared by different networks, for instance because the networks represent the same biological system but were derived using different data or software, or because the entities themselves are part of many different systems. Regardless of the reason, knowledge about shared elements of distinct networks and about the networks themselves can be obtained comparing them and determining how the roles of the common entities vary.

InBiNA possesses functionalities to compare two networks based in common vertices, topological metrics and ranking algorithms. Some of the comparisons analyse the networks as a whole, while others work only at the vertex level. When comparing networks, InBiNA assumes that vertices that have the same id and type represent the same biological entity, regardless of any other associated metadata. Many of the comparison functionalities are based in tools previously described. To use those relevant tools, they must be first applied to both networks. The comparison works this way to reduce the computational cost, since otherwise all related analysis functionalities would have to be executed during the comparison.

When two networks are compared, InBiNA assumes that all metrics, from the ones that were applied to both networks, will be used as comparison parameters. The full list of results available to a comparison operation includes:

- Exclusive vertices and edges, those appearing in only one of the networks.

- Overall degree comparison, including the average, lower, higher degree discriminated by in- and out-degree, and comparison of the degrees of shared vertices. Also, a degree percentage analysis can be performed: this metric is calculated by first determining between which values the

vertices' degree of both networks vary and then determining the percentage of vertices which share the same degree.

- Overall shortest path metrics, including the diameter, the longest shortest path and the mean shortest path. Additionally, the comparison tool can also be used to determine the shortest path between vertices common to both networks;

- Mean values from ranking algorithms and comparison of ranking values for each vertex (the results are normalized to ensure that they can be compared despite the different structure of networks).

All the results of network comparisons are presented as tables that can be saved as flat files (CSV format). As a final note, if new metrics become available for comparison (i.e. if they are applied to both networks), the results will be automatically updated.

3.2 Case study: integrated network for core pathways in *E. coli*

With the objective of demonstrating most of the functionalities described earlier, a case study is proposed. An integrated biological network was created, containing metabolic, transcriptional and regulatory information about the glycolysis/ gluconeogenesis and citric acid cycle pathways of *Escherichia coli*. Data for this network were extracted from the database Ecocyc [5]. In the analysis, smaller networks are derived though filtering operations and those, together with the original network, are then subject of a series of tests using the provided tools. Meaningful biological conclusions are drawn from this analysis.

3.2.1 Network creation process

The original network (also called base network) was built according to the following steps:

1. Data regarding all reactions in the glycolysis/gluconeogenesis and citric acid cycle pathways, their substrates and products, were used to create a metabolic network, where the vertices represented reactions and compounds, while edges stand for relationships of consumption and production.

2. The metabolic network was enriched adding enzymatic information, as well as the regulatory effects of the compounds already present over the newly added enzymes.

3. The network is finished through an iterative process: firstly, the subunits of any complexes are added to the network; next, genes coding for all proteins are added; finally, proteins regulating these genes are also added. This process is repeated until no other information can be added using these steps.

The base network created using the previous process contains four types of vertices, each representing a different type of biological entity and nine different types of edges representing different kinds of interactions between the entities. The four types of the entities are evident by their names (gene, proteins, reactions and compounds). However, the meaning of each edge type requires a longer explanation:

- Coded by: edges pointing from genes to the peptide chains they encode (i.e. protein subunits);

- Consumed metabolite/Produced metabolite: represent the metabolism, pointing from substrates to the reactions that consume them and from the reactions to their products, respectively;

- TF Regulation: starting in regulator proteins (transcription factors) and ending in the genes they regulate (this type of edges is simply identified as Regulation);

- Sigma regulation: from regulator proteins (sigma factors) to the genes they regulate.

- Catalyzes: from proteins (enzymes) to the reactions they catalyze.

- Subunit of: from peptide chains to the complex proteins that they are part of.

- Transcription effectors: from compounds acting as metabolic effectors of proteins (transcription factors).

- Metabolic regulations: point from compounds acting as metabolic regulators to the proteins (enzymes) they act upon.

It should be noted that in this network, transcription factors, sigma factors and enzymes are all represented as protein vertices, whose function in the network is defined by the edges connected to them.

3.2.2 Sub-networks: filters and bypasses

After obtaining the base network, the remaining networks used in the analysis were derived by applying filters (regular filters and bypasses). Most of the simpler networks were created through the use of vertex filters used to create sub-networks containing only two or three of the four types of vertices. After discarding networks that are not meaningful, seven networks remained, named according with the types of vertices they contain: gene-protein (GP), protein-reaction (PR), protein-compound (PC), reaction-compound (RC), gene-protein-reaction (GPR), gene-protein-compound (GPC) and protein-reaction-compound

(PRC) networks. It should be noted that the filters used do not remove edges directly and consequently these sub-networks contain all the edges of the base network that connect the vertices present.

Other sub-networks were constructed through the use of bypass replacement filters, resulting in networks where certain types of vertices were not only filtered out, but also their role was transferred for other neighbouring vertices: Three bypass networks were built for this case study, whose details are given next.

The bypass 1 network was obtained removing compound vertices and transferring edges that were connected to them to the reactions producing them. This resulted in a network where the reactions were connected to each other through "Consumed metabolite" edges (the "Produced metabolite" edges were removed by the filter) and the metabolic regulation edges now start in the reactions. The use of the bypass filter does not remove compounds not produced by any reaction (usually external metabolites) and these had to be removed with a normal filter applied afterwards.

The bypass 2 network was obtained by removing the proteins and redirecting edges which connect to them to their encoding genes. When dealing with complexes, all edges linking to them are copied and connected to the genes that encode all simple proteins that compose the complex. The creation of this network required the use of several bypass filters and the generation of intermediary networks until the desired network was obtained, through three steps:

1. A bypass filter replaced all simple proteins with the genes that encode them.

2. Several bypass filters were use to replace the complexes with the genes which encode the simple proteins. This process could not be made with a single bypass, since a complex can be composed of other complexes. Each

bypass was used to create a network that was "lower in the chain of simple-complexes" until all the protein complexes were handled.

3. The type of edges identifying the composition of the complex proteins ("Subunit of"), which after step 2 connected genes to genes, were removed with a simple filter.

This network thus retains all the information of the original network except for protein composition.

The bypass 3 network was obtained by combining the bypass filters used to create the previous two bypass networks, thus creating a network with only gene and protein vertices. This network retains a good deal of the information from the base network, while having significantly less vertices.

In many analyses, the presence of currency metabolites causes difficulties and results with limited meaningfulness. This problem affected the base network and all the filtered networks which contained data related to the metabolism (RC, GPC, PRC and the bypass networks). To address this issue, the base network was filtered out of currency metabolites and the affected networks were remade. The results obtained before and after this change are analysed next.

3.2.3 Results for the topological analysis

The basic topological analysis and ranking metrics were applied to the networks created for the case study. The results are given next organized by network.

Base network

In the base network, only one kind of edge ("Subunit of") can connect vertices of the same type and consequently there are relatively few triangles, which results in low clustering coefficients. Observation of the few vertices with a high value revealed that most correspond to proteins that are involved in situations of auto-regulation.

Regarding the betweenness centrality (BC), most of the vertices with high values have paths to RPOD-MONOMER, the regulatory protein that affects more genes in the network. Since RPOD-MONOMER is connected to so many other vertices, it is logical that many shortest paths pass through this vertex. Other vertices with high BC values are reactions that both consume and produce currency metabolites (e.g. H_2O, Pi, etc).

An initial analysis of the closeness centrality (CC) results shows that all vertices with high values were either proteins regulating several genes, the genes which encode them and currency metabolites. While identifying these nodes can be useful, this information could also be obtained by simply looking at the degree. However, the CC proved to be a useful metric when the vertices with low (but not null) values were observed, as these were determined to be biological entities at the beginning of long chain (both regulatory or metabolic).

In the past, when the HITS algorithm was applied to biological networks, often it was observed that one type of biological entities would take the role of authorities, while another would take the role of hubs and the other vertices would receive neither classification. This case was no exception, since all authorities were genes and all hubs were proteins.

By the definition of the algorithm, vertices are authorities if hubs point to them and vertices are hubs if they point to authorities. Following this line of thought, the proteins which act as hubs have a regulatory effect on the genes which act as authorities and the value of hubness of a protein will depend both on the number of genes it regulates and the number of regulators which affect the genes it regulates. The value of authority of a gene will depend on the number of enzymatic regulators that affect it and the number of genes these regulators affect. Because of the double rank it produces, HITS can be used to give an idea of both the importance of the regulatory proteins and the regulated genes.

The network obtained after removing the currency metabolites presented, for most analyses, similar results. The exception was the betweenness centrality, where, while the RPOD-MONOMER and vertices related to it still present a high value, a new set of high BC values emerges. After studying these vertices, it was verified that these were reactions and compounds belonging to important sections of the metabolism (for example the TCA cycle). This makes sense, since a great deal of the metabolic fluxes should pass through these important sections and, consequently, it is expected that the metabolic network has evolved in a way that minimizes the paths in which these central vertices are involved.

GP network

Most of the results obtained over the base network were related with gene-protein interactions (mostly regulatory). Since this type of connections was maintained in this network, the results obtained are very similar.

PR network

Unlike the base network, this protein-reaction network was highly fragmented, being composed by small independent modules, formed by complex proteins, their subunits and the reactions they catalyse. Here, the removal of both the genes and compounds eliminated all instances of auto-regulation, leading to all vertices having a clustering coefficient of zero.

The betweenness and closeness centralities were not very useful because these metrics rank vertices according to their relation to the rest of the network and in a fragmented network they only give an idea of the local importance of the vertices. Vertices with high BC values were proteins composed by several subunits that regulate several genes. The closeness centrality is only useful when there are long chains in the network and these are absent from this network making it useless in this case.

Using HITS, only proteins are identified as hubs, while both proteins and reactions acted as authorities. These results can be explained by the fact that no edges start in the reaction and, thus, they cannot be hubs. On the other hand,

proteins can catalyse reactions, be subunits of complexes or be complexes, which means that edges will both start and end in them giving the ability to be both hubs and authorities. Here, good hubs are proteins that act both as enzymes and subunits and good authorities are proteins and reactions composed or catalysed by the hubs, respectively.

PC network

The only edges in the protein-compound network represent relationships between complex proteins and their subunits and instances of metabolic regulation. Thus, this network is even more fragmented than the protein-reaction one and the modules are even smaller (there is only one shortest path in the network with a length greater than one) which makes both betweenness and closeness centralities useless. Similarly to the previous case, the removal of reaction and gene vertices also eliminated all instances of auto-regulation and again this means all vertices have a clustering coefficient of zero.

When HITS was applied, all the good authorities were proteins and the good hubs were compounds that regulated the good authorities and their subunits. While initially this looked like an interesting result, a careful observation revealed that due to the fragmented nature of the network, similar results could be obtained by simply consulting the degrees. Overall, this network proved to be too fragmented and the fragments too small for it to be worthwhile to apply any but the simpler network analysis methods to it.

RC network

The reaction compound network is a classic minimalist metabolic network, where vertices represent compounds and reactions and the edges the consumption and production of metabolites. In these networks, all edges either start in a reaction and end in a compound or vice versa. This structure makes the formation of triangles impossible consequently the clustering coefficient is not applicable.

No interesting results were obtained with the betwenness centrality, while the currency metabolites were present. After their removal, the vertices with high betwenness centrality were identified as the ones belonging to central parts of the metabolism, which is consistent with the results obtained in the base network.

The closeness centrality was as useful in this network as it was in the base network. Naturally, all the chains found in this network were metabolic.

The results of HITS were again dominated by the presence of currency metabolites. After their removal, an unexpected distribution of the values of authority and hubness was observed: vertices with higher values of authority and hubness were the same and they shared a very similar value in both ranks. A careful analysis revealed that in the RC network the vertices with high rank in both HITS metrics were either reversible reactions or compounds involved in reversible reactions, explaining the similar values of authority and hubness.

GPR network

Unlike the PR network, this network was only slightly fragmented with a large central module and seven small secondary modules. Apparently, the presence of genes connects the otherwise separated protein-reaction modules through links of genetic codification and regulation. Consequently, it was possible to apply the clustering coefficient metric to this network and, unsurprisingly, vertices with high clustering coefficients were associated with situations of auto-regulation.

Similarly to the base network, most vertices with high BC values have paths to RPOD-MONOMER. The elimination of the compound vertices leads to the enzymes having a lower value of BC due to the fact that all reactions become dead-ends.

Once again, closeness centrality was useful for detecting the start of long chains, though in this case most chains were regulatory.

HITS metrics produced results similar to the base network, which was expected because neither the vertices which were good hubs and authorities (proteins and reactions) neither the edges which connected them were removed in this network.

GPC network

With the exception of the presence of metabolic regulation edges (which are relatively few) and the absence of edges related to enzymes catalysing reactions (which make only a small fraction of the edges connected to proteins), the gene-protein-compound network is very similar to the previous network, which was to be expected because removing the compounds or reactions from the base network has comparable impact. Consequently, results obtained were extremely similar to the previous ones.

PRC network

The PRC network contains both protein composition and metabolic regulation information. This means that despite the absence of gene vertices, situations of auto-regulation occur. Consequently, the clustering coefficient can be applied to the PRC network. The results obtained for BC were similar to the ones of the base network. One major difference was, however observed, due to the fact that this network contains no genetic information. The vertices with high BC are for most part the ones belonging to important parts of the metabolism. Once again, the closeness centrality could be used to identify the beginning of long chains.

When HITS was applied, the usual division between hubs and authorities was observed. However, it was less clear than in most previous cases. Here, all good authorities were either reactions or proteins and all good hubs were compounds and proteins. This organization was maintained both in the presence and absence of currency metabolites. After comparing the HITS values of this network with the ones of the RC network, it was concluded that it is the presence of proteins, more specifically enzymes, which changes the distribution of the values of HITS. Due to their catalytic effects, enzymes are neighbours of reactions through outgoing edges and due to the metabolic regulators they are also

neighbours of the metabolites, although in this case through incoming edges. Consequently, their presence leads to a general increase of the values of authority of the reactions and of hubness of the metabolites.

Bypass 1 network

The structure of this network is, in many ways, similar to the base network. However, reaction vertices tend to replace the functions that the compound vertices had in the original structure. The values of the clustering coefficients were partially similar to the ones of the base network, since the same proteins and genes were highly ranked. However, new situations of high coefficients appeared as a consequence of reactions now being directly connected, which occurs when two reactions share compounds and they are catalyzed by the same proteins and/or produce metabolites, which regulated that same proteins. Either way, a triangle is formed and consequently the associated protein's clustering coefficient increases.

For most parts, vertices tend to have equivalent betwenness and closeness centralities values in this network, as compared to those obtained in the base network. In fact, the only noteworthy changes observed was an increase in the values of betwenness centrality of reactions that produce or consume currency metabolites. Also, there is a general increase in the values of closeness centrality due to the reduction of size of the reaction chains.

HITS was the metric whose results diverted more from the base network, due to the significant increase on the number of edges starting from reactions. The best hubs in the network are now reactions and consequently the better authorities are a mix of proteins and reactions. From a biological point of view, hubs are reactions that produce currency metabolites and metabolic regulators and authorities are the proteins regulated and the reactions which consume the products of the hubs. Despite the initially observed divergences, when HITS was applied to the bypass network obtained from the base network with no currency metabolites, the results were nearly identical to those obtained in the base network.

Bypass 2 network

In this network, genes not only replace the simple proteins they encode, but also the complexes composed by them. Consequently, many indirect auto-regulatory interdependencies are now represented as simple edges between genes, leading to an increase in the values of clustering coefficients. Thus the *rpo*D, the coding gene of RPOD-MONOMER, and other coding genes took the place of related reactions in the results obtained with the betwenness centrality. The closeness centrality is less useful here, since in certain situations during the transition from the base network to the bypass 2 network paths which make sense from a graph theory point of view but have no biological meaning can be created, as a consequence of concentrating all the vertices connected to the complex networks in the genes.

For most part, the results obtained with HITS in this network, despite the fact that the coding genes take the place of proteins, are consistent with the ones of the base network. However, probably due to the elimination of the simple-complex relations, which resulted in all relationships of metabolic regulation to be redirected to the gene vertices, a few compounds (all of which act as metabolic regulators) are now good hubs.

Bypass 3 network

The combinations of the two bypass filters, each causing an increase in the number of triangles because of different reasons, resulted in a general increase in mean clustering coefficients. The results obtained with BC were consistent with the ones obtained in bypass 1 and 2 networks: reactions which produced or consumed currency metabolites had an increase in rank and the coding genes took the ranks of the proteins they coded. As for the closeness centrality, this network has the same false path problems as bypass 2 network.

Finally, in this network, the usual separation of authorities and hubs by vertex type completely disappears with both classifications being given to a mix of proteins and genes.

Discussion

This case study showed that InBiNA can be used to analyse complex biological networks either by looking at them as a whole or by isolating a part of them in a subnetwork, as long as the final network is not too fragmented. The usefulness and viability of the topological metrics presented in the application were methodically demonstrated. As expected, the results obtained are dependent on the properties of the network, explaining the creation of the different variants of the same network.

An interesting fact observed was that the integrity of the base network appeared to be maintained by the gene-protein relations and the reaction-metabolite relations. This was noted when the networks PR and PC, in which these relations were removed, were analyzed and it was observed that they were highly fragmented. The protein codification combined with the gene regulation and the metabolism interactions appear to be two independent networks connected by the metabolic regulation and the enzymatic catalysis systems.

3.2.4 Results from pattern finding algorithms

The networks created within the case study were examined with the InBiNA's pattern detection tools previously described. This allowed to identify the occurrence of potentially interesting patterns, facilitated by the tool’s capability to restrict the types of the individual components of the searched patterns, since with was possible to limit the search only to patterns with known biological significance.

In this section, the most interesting patterns identified over all networks will be presented and interpreted. It should be noted that, since regular filters only remove vertices, networks obtained in this way do not show any patterns that are not present in the original network. The situation is different for the bypass networks, given that they rearrange edges. Thus, this analysis was conducted

over the base and the three bypass networks. The results are organized into sub-sections for each pattern type.

Feed-forward loops

Figure 3.3a) shows two examples of FFL patterns found that have a biological meaning. Case 1 was found in bypass network 1 and illustrates a metabolic self-regulatory circuit, where a reaction (TPI) produces a metabolic regulator of an enzyme (Fructose biphosphate aldolase class I) that catalyzes another reaction (FBA) that, on its hand, consumes one product of the original reaction.

The second example was observed in both the original and bypass 2 networks, where a metabolite (D-Fructose 6-phosphate) is a metabolic regulator of the enzyme (6-phostphofructokinase I) that catalyzes a reaction (PFK) that consumes it. This biological circuit appears to correspond to a situation where a metabolite acts as a catalytic switcher, either by promoting or preventing the activity of the reaction that consumes it. In general, this kind of patterns can be found in pathways where the level of metabolites being consumed in a reaction influences the activity of the enzyme catalyzing that same reaction. The first case is a special case of the second, where the product of the first reaction is also the metabolic regulator.

SIM and reverse SIM

When looking for these patterns in the original network, the results were very predictable, since the pattern served merely to highlight the vertices known to be the start or end of many edges, respectively. A much more interesting use of the SIM or reversed SIM was observed when these patterns were searched in the bypass networks, where new instances of the SIM or reversed SIM emerge. These reveal vertices that have an indirect effect difficult to identify in the base network.

Figure 3.3b) to d) show some examples of SIM and RSIM patterns found in the bypass networks. This list is by no means exhaustive, containing only some significant examples. In each case, the pattern in the bypass network is shown,

together with a view of the same entities and relationships in the original network.

The first example (Figure 3.3b)) was identified in the bypass 2 network, where the *rcs*B gene regulates a set of other genes through a complex formed partially by the subunit protein it encodes which acts as a transcriptional regulator. A second example (Figure 3.3c)) found in the bypass 3 network highlights the importance of the reaction GLGC, both in terms of the metabolism and metabolic regulation, i.e. it produces a metabolite (ADP-glucose) that is directly involved in the glycogen biosynthesis via GLCS1 reaction and its control via the enzyme-coding genes, *glg*C and *glg*P. Finally, an example of RSIM is shown in Figure 3.3d), found in bypass network 1 that exemplifies the indirect association between metabolic regulators and enzyme coding genes.

Bowtie

Searching for instances of the bowtie pattern in the base network returned uninteresting results with the few patterns found not being biologically relevant. However, with the use of bypasses the results were more interesting. When the proteins were bypassed by the genes, the occurrences of the bowtie motif revealed central regulatory genes, confirming the results of previous work [6].

The instances of the bowtie pattern found when the reactions were bypassed were quite unexpected: since most reactions are reversible, many of the patterns found had vertices which acted both as lower and upper vertices. While initially these results seemed to point to an error in the detection algorithm, further analysis revealed that in these situations the central vertex appeared to be acting as a "control point" for the direction of the metabolic flux.

Two particular cases were especially interesting: in the first (Figure 3.4a) the FBA reaction appears to act as a switch between glycolysis and gluconeogenesis with its direction determining which pathway is active; in the second (Figure 3.4b) the direction of MDH appears to be an key point in controlling if either the normal TCA cycle or the reductive TCA cycle will occur. This highlights the

importance of metabolic switches under different circumstances, especially in central pathways that require a tight control to respond to the energy needs of the cells. While this analysis is based only on topological analysis of the network, and as such has no experimental data to back it up, it points nonetheless to points of the metabolism which would be worthwhile of further study.

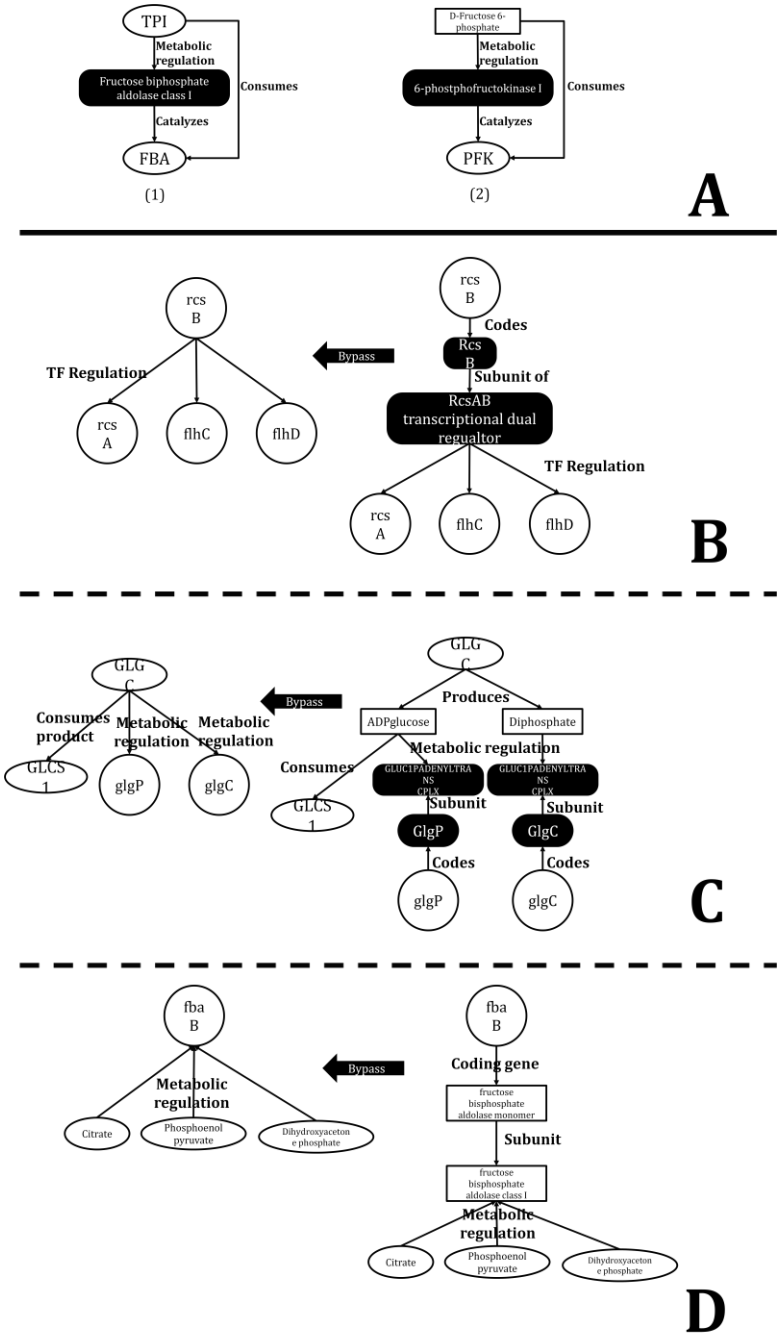


Figure 3.3 - Selected examples of FFL and SIM patterns found in the analysis of the networks from the *E. coli* case study: a) FFL examples; b) to d) SIM and reversed SIM examples.

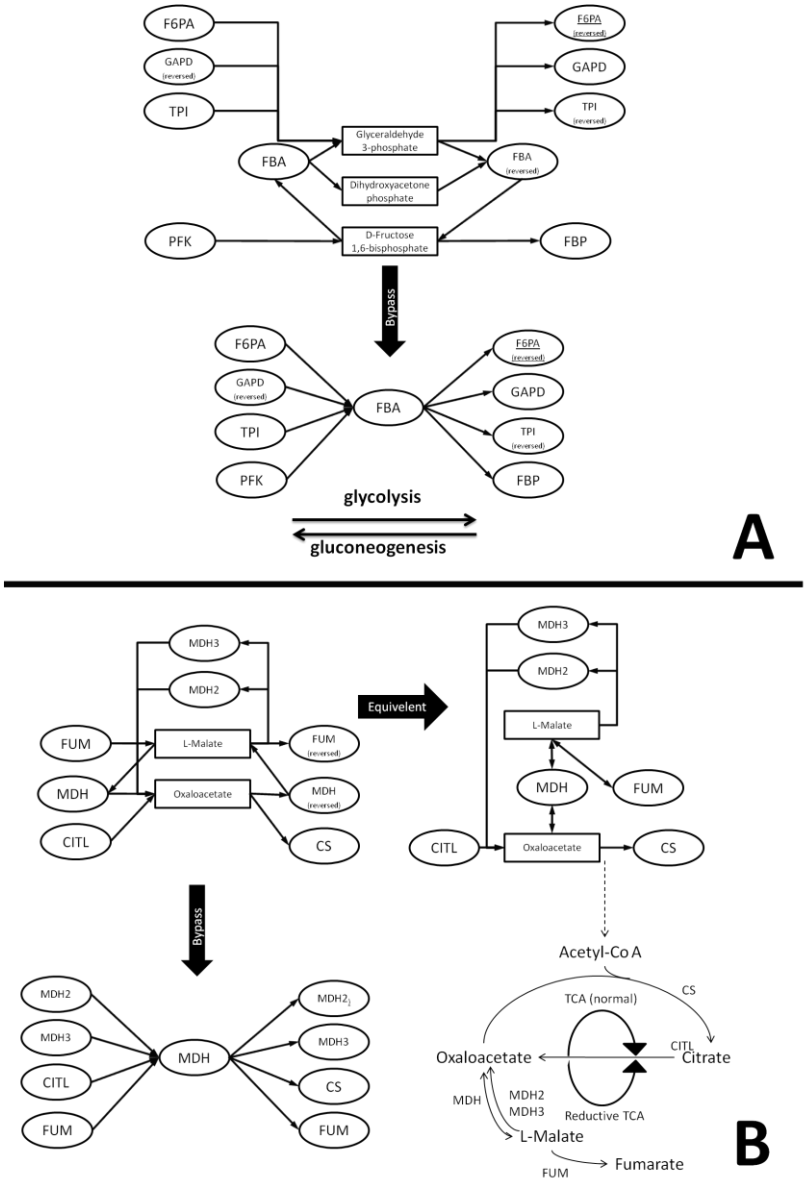


Figure 3.4 - Selected examples of bowtie patterns found in the analysis of the networks from the *E. coli* case study.

Extended feedback loops

In the case study, the EFL and DEFL patterns proved to be an excellent way of detecting cycles. Of particular use, was the capability of selecting the edge types when searching these patterns providing a user friendly way of detecting multiple different types of biological cycles. A very useful characteristic of the EFL and DEFL patterns is that InBiNA does not just find the shortest cycle connecting a pair of vertices, instead it finds all cycles even if the paths that make

the alternative cycles share components. This capability leads to finding cycles with alternative connections, which were designated as bifurcate cycles.

After searching the base network for EFL and DEFL patterns, it was determined that it contained three main types of cycles:

1. Metabolic cycles - These are metabolic chains, where the first reaction consumes one of the products of the last. These are interesting because they point to metabolic chains where part of their product is possibly later reused.

2. Metabolic regulation cycles - These are composed by a metabolic chain where the last reaction produces a metabolic regulator of an enzyme which catalyses the first reaction. This is a noteworthy situation of regulatory feedback.

3. Genetic regulation cycles - These are composed by a gene that encodes a simple protein that belongs a one complex regulating its transcription. These cycles represent a situation of indirect genetic self-regulation.

It should be noted that, since bypass operations do not lead to the creation of cycles not present in the base network but only reduce the length of the existing ones, it was decided to only search for EFL and DEFL patterns in the base network. Figures 3.5a) to c) show examples of cycles of the three types identified above.

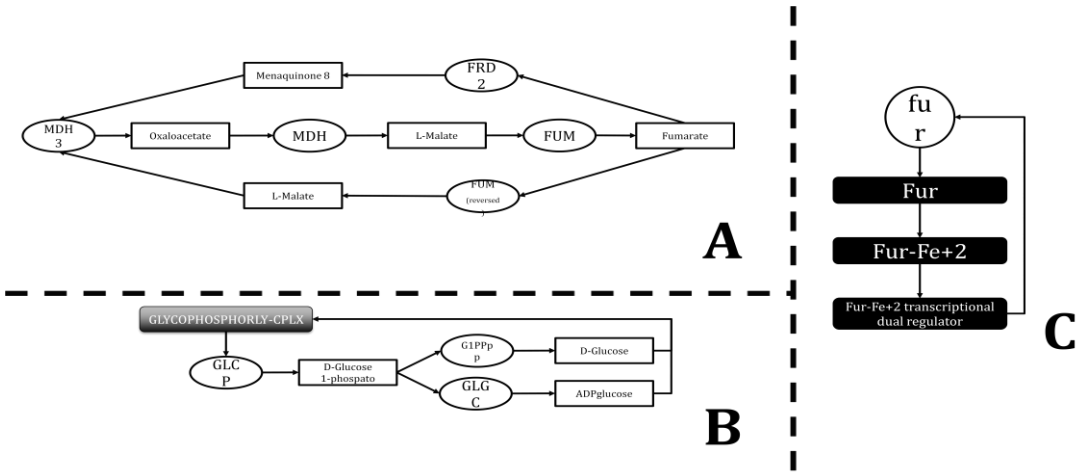


Figure 3.5 - Cycles found in EFL/ DEFL pattern analysis for the case study showing, respectively, a bifurcate metabolic cycle (a), a bifurcate metabolic regulation cycle (b) and a genetic regulation cycle without bifurcations (c).

There were several examples of bifurcate metabolic and metabolic regulation cycles, but there was only one genetic regulatory cycle (Fig. 3.5c). It is still unclear if this happens because the network used in the case study is too small or if genetic regulation cycles are simply rare.

3.3 A Study of the Short and Long-term Regulation of *E. coli* Metabolic Pathways[1]

In this section, a study performed on the regulation of metabolic pathways for *Escherichia coli* is presented. This work was conducted using some of InBiNA's functionalities, together with other complimentary tools programmed as command line scripts to help in processing data at a large scale. This study addresses the regulatory network of *E. coli* and offers a global view of the short and long-term regulation of its metabolic pathways.

It is known that different mechanisms are recruited for regulation, either on a long-term basis by changing the expression level of genes or a short-term basis by changing the activity of enzymes. Long-term regulation is crucial for cell

[1] This work has been previously published in the *Journal of Integrative Bioinformatics*, 8(3):183, 2011, being co-authored by Anália Lourenço, Sónia Carneiro and Eugénio Ferreira, together with the supervisors of this thesis.

survival and serves to adapt to environmental challenges that require extensive cellular alterations at the transcriptional level. In contrast, short-term regulation is used to control specific enzymatic activities and hence, balance metabolite pools to avoid metabolic and energetic losses.

Gene expression is mostly controlled by transcriptional regulation. In turn, the activity of enzymes can be controlled by some effector molecules binding at the allosteric site, by substrate inhibition or by alteration of some environmental condition (e.g. pH or ionic strength). In recent years, the interest in understanding the coordination of these regulatory mechanisms (i.e. co-regulation) to modulate the enzymatic activity of specific biosynthetic steps in pathways has increased significantly. The manipulation of the activity of key enzymes, either by transcriptional regulation or allosteric control of the activity of the enzyme itself, can benefit the adaptation of cells to specific conditions or enhance the production of valuable metabolites.

Here, the focus is on the study of how transcriptional regulation couples with the regulation of the activity of metabolic pathways via enzymatic regulation, and how similar regulatory mechanisms are used across different metabolic pathways. The goal is to obtain a global view of the regulatory interplay affecting the metabolism of the bacterium *E. coli K12*.

3.3.1 Data integration

Different information needs to be integrated when constructing a genome-scale regulatory network. For this purpose, validated metabolic and regulatory networks are taken into account, together with information retrieved from publicly available repositories, and we need to perform the necessary data integration.

This study required the construction of a genome-scale network involving known regulation on *E. coli* metabolic pathways. This network is intended to encompass various information:

- gene coding or genetic information, i.e. the genes that code for regulatory and non-regulatory proteins;

- gene regulation, i.e. the control of gene transcription by TFs encoded by regulatory genes, usually called long-term regulation;

- gene-reaction association, i.e. non-regulatory genes (nRGs) coding for enzymes catalyzing metabolic reactions;

- enzymatic regulation, i.e. control of the activity of biochemical reactions by metabolites (Ms) that bind directly to enzymes, usually called short-term regulation.

Since *E. coli* is a model organism, there are a number of data resources that can be used for this purpose. A previously validated metabolic network was used and complemented with regulatory information from publicly available repositories. For simplicity purposes, all information was reverted into vertices representing genes, i.e., TFs, enzymes and reactions are represented by vertices representing the corresponding coding genes.

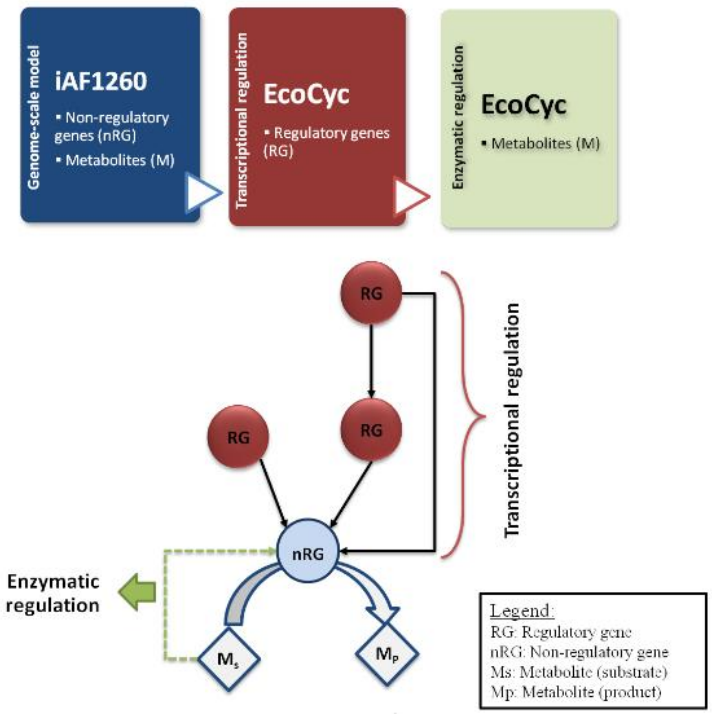


Figure 3.6 - An integrated regulatory network for *E. coli* combining the manually curated iAF1260 model with regulatory data from the EcoCyc repository. Gene-reaction associations consisting on non-regulatory genes (nRGs), which code for enzymes that catalyse the conversion of metabolic substrates (Ms) into products (Mp), were obtained from the iAF1260 model. EcoCyc information on transcriptional and enzymatic regulation was further associated to non-regulatory genes (nRGs). All regulatory genes (RGs) that directly or indirectly affect the transcription of enzyme coding genes were included in the integrated network. In the case of enzymatic regulation, all metabolites that affect the activity of an enzyme included in the metabolic model were associated with the respective enzyme-coding genes or non-regulatory genes (nRGs). In addition, metabolites that are not a substrate or a product in any reaction of the model were filtered.

The genome-scale metabolic network of *E. coli K12* (*iAF1260*) [7] was at the core of the data integration. This is a manually curated and experimentally validated model, which means that all its information is interconnected and *in silico* simulations have shown that the set of reactions and enzymes described is sufficient to characterize the metabolism of *E. coli*. The model encompasses reaction stoichiometry (the identification and quantification of substrates and products), enzyme-reaction associations (grouping reactions into pathways) and

enzyme-coding genes (identified by *bnumbers*, which are Locus identifiers specific for *E. coli*).

The enzyme-coding genes were then integrated with gene transcriptional regulation derived from the organism-specific EcoCyc database [5]. Specifically, TF-encoding or regulatory genes (RGs) were included, regulating the expression of metabolic or non-regulatory genes (nRGs), i.e. genes associated with a reaction from the iAF1260 model, and regulating other TF-encoding genes linked to metabolic genes. As shown in Figure 3.6, RGs can regulate metabolic genes (or nRGs) directly or indirectly through other RGs creating a cascade of regulatory interactions between genes. Information on the control of enzyme activity by metabolic regulators was also extracted from EcoCyc. However, only metabolic regulators that intervene as substrates (Ms) or products (Mp) in our model were considered.

Since the *iAF1260* model was found to be incomplete in terms of enzyme-coding gene information, a semi-automatic integration process was implemented where manual curation was issued to complete enzyme-coding information and retrieve the corresponding regulatory information whenever needed.

3.3.2 Network analysis

The graph that represents the network has two types of nodes, genes and metabolites, and two types of edges: one that connects metabolites (M) to the enzyme-coding genes (or nRGs) they regulate; and another that connects pairs of genes that are linked by transcriptional regulation.

The frequency of occurrence of different types of edges was analysed, associated with nRGs, specifically: RGs directly connected to nRGs, cascades of RGs that end up connected to nRGs, and metabolites (M) that are part of the cell's metabolism (i.e. are being processed by the cell or are somehow going into the cell) and act as regulators of enzymatic activities. Additionally, a search for regulatory patterns (or motifs) was performed to get a deeper understanding on the

interplays taking place on particular pathways. The motifs used in this work were the SIM, reversed SIM (RSIM), bowtie and FFL, as described in section 2.4.2.

To be able to assess the importance of different regulatory motifs, in particular metabolic pathways, the prevalence of motif types per pathway was calculated, as well as the average number of genes in the pathway participating in such motifs, as follows:

- the absolute frequency of each motif type ***T*** for every pathway ***P***, i.e. the number of motifs that affect at least one gene $\mathbf{G}_{\boldsymbol{P}}$ in pathway ***P***:

$$\mathbf{Abs\_Freq\_T\_P} = number\ of\ motifs\ of\ type\ \boldsymbol{T}\ in\ pathway\ \boldsymbol{P}$$

- the relative frequency of motif type ***T***, i.e. the number of times the motif type ***T*** occurs in the pathway (i.e. affects at least one of its genes) divided by the number of times it occurs in the network:

$$\mathbf{Rel\_Freq\_T\_P} = \frac{Abs\_Freq\_T\_P}{number\ of\ motifs\ of\ type\ \boldsymbol{T}\ in\ the\ network}$$

- the relative frequency of genes from pathway $\boldsymbol{P(G_P)}$ that are included in motifs of type ***T***:

$$\mathbf{Rel\_Freq\_G_{\boldsymbol{p}}\_P} = \frac{number\ of\ genes\ in\ motif\ type\ \boldsymbol{T}\ in\ pathway\ \boldsymbol{P}}{number\ of\ genes\ in\ \boldsymbol{P}}$$

3.3.3 Types of regulation

The distribution of different types of regulation over the metabolism of *E. coli* is illustrated in Figure 3.7 showing that 53% of genes are regulated by at least one TF, but co-regulation (i.e. the combination of transcriptional and enzymatic regulation) and enzymatic regulation are also present (16% and 29% of genes, respectively). This suggests that metabolic activities do not necessarily correlate

proportionally with the gene expression levels of the corresponding enzymes, but are also dependent on immediate enzyme control by metabolites.

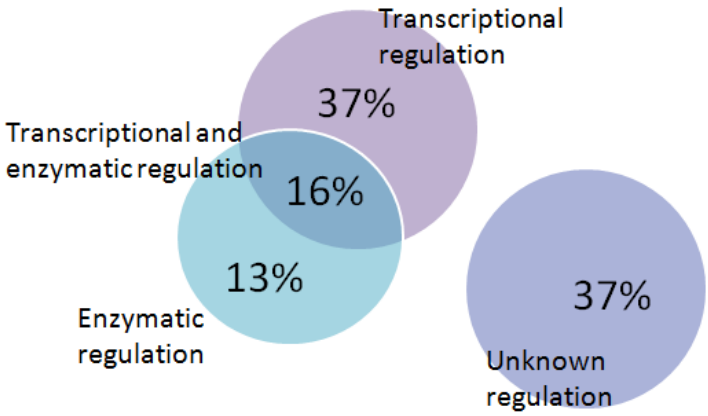


Figure 3.7 - The different types of regulation in the *E. coli* network.

The high percentage of unknown regulation (37%) can be explained by the insufficiency of knowledge about enzyme activities (it has not yet been possible to assign a function to approximately one third of the proteins identified in *E. coli* and many details are still missing from its biochemical characterization [8]) and the exclusion of some regulatory mechanisms of this study (e.g. post-translational modifications and ribosome-mediated transcriptional attenuations).

Some additional hypotheses about the mechanisms coordinating key metabolic processes can be formulated by detailing the regulation per pathway (Table 3.1). The transcriptional regulation is dominant in pathways like "Nitrogen Metabolism", "Citric Acid Cycle", "Inorganic Ion Transport and Metabolism" and other transport pathways (87%, 83%, 76% and 72-81%, respectively). These pathways require modifications at gene expression level to respond to environmental inputs, such as availability of nitrogen and carbon sources. In contrast, enzymatic activities that are only dependent on enzymatic regulation are not so representative within *E. coli* pathways.

Most pathways have less than 40% of their gene products regulated at this level and 10 out of the 37 pathways considered in the model have no genes associated with this regulatory mechanism. The only exception is the pathway "Folate Metabolism", which has approximately half of its genes regulated at the enzymatic level. This indicates that few pathways include metabolic functions governed exclusively by enzymatic regulation and only specific metabolic functions, such as energy-generating reactions, seem to be coupled with this type of regulation.

Table 3.1 - Effect of transcriptional, enzymatic and co-regulation in *E. coli* pathways. For each gene associated with a particular pathway it was determined if there are transcriptional regulations (i.e. a link between genes), enzymatic regulations (i.e. a link between a metabolite and a non-regulatory gene (nRG)) or both types of regulation (co-regulation). Unknown regulation denotes the absence of regulatory associations.

Pathway	Number of Genes	Transcriptional regulation	Metabolic regulation	Transcriptional and metabolic regulation	Unknown regulation
Alanine and Aspartate Metabolism	11	27%	9%	27%	36%
Alternate Carbon Metabolism	159	48%	6%	21%	26%
Anaplerotic Reactions	10	20%	40%	20%	20%
Arginine and Proline Metabolism	40	23%	18%	40%	20%
Cell Envelope Biosynthesis	53	17%	17%	11%	55%
Citric Acid Cycle	18	83%	0%	11%	6%
Cofactor and Prosthetic Group Biosynthesis	136	20%	16%	7%	57%
Cysteine Metabolism	20	35%	15%	20%	30%
Folate Metabolism	4	25%	50%	25%	0%
Glutamate Metabolism	11	0%	9%	64%	27%
Glycerophospholipid Metabolism	20	15%	35%	0%	50%
Glycine and Serine Metabolism	15	27%	7%	40%	27%
Glycolysis/Gluconeogenesis	32	31%	13%	28%	28%
Glyoxylate Metabolism	4	25%	0%	25%	50%
Histidine Metabolism	9	0%	22%	11%	67%
Information Transfer	167	56%	0%	0%	44%
Inorganic Ion Transport and Metabolism	67	76%	1%	0%	22%
Lipopolysaccharide Biosynthesis / Recycling	46	7%	9%	2%	83%
Membrane Lipid Metabolism	14	50%	14%	7%	29%
Methionine Metabolism	13	31%	0%	23%	46%
Methylglyoxal Metabolism	10	0%	20%	10%	70%
Murein Biosynthesis	10	20%	0%	0%	80%
Murein Recycling	34	29%	6%	0%	65%
Nitrogen Metabolism	15	87%	0%	7%	7%
Nucleotide Salvage Pathway	57	19%	26%	19%	35%
Oxidative Phosphorylation	77	65%	8%	5%	22%
Pentose Phosphate Pathway	13	15%	31%	23%	31%
Purine and Pyrimidine Biosynthesis	22	41%	18%	27%	14%
Pyruvate Metabolism	20	50%	15%	15%	20%
Threonine and Lysine Metabolism	19	11%	37%	21%	32%
Transport, Inner Membrane	216	72%	0%	1%	26%
Transport, Outer Membrane	21	81%	0%	0%	19%
Transport, Outer Membrane Porin	4	75%	0%	0%	25%
tRNA Charging	24	0%	17%	4%	79%
Tyrosine, Tryptophan, and Phenylalanine Metabolism	24	29%	4%	29%	38%
Unassigned	23	39%	4%	13%	43%
Valine, Leucine, and Isoleucine Metabolism	17	76%	0%	18%	6%

The existence of co-regulation in amino acids biosynthesis pathways, like "Glutamate Metabolism", "Tyrosine, Tryptophan", and "Phenylalanine Metabolism" and "Arginine and Proline Metabolism", can be explained by the need to ensure the fine-tuning of metabolic activities and the rapid response to over-accumulation of end products. Co-regulated pathways present advantages in terms of functional changeability and reorganization. For instance, 64% of the genes coding for enzymes involved in the "Glutamate Metabolism" are simultaneously controlled at the transcriptional and enzymatic levels, while only 9% are exclusively regulated at the enzymatic level and none is exclusively dependent on the transcriptional control. The combination of regulatory mechanisms allows a tight control of glutamate biosynthesis (the principal amino donor to other amino acids), manipulating protein expression levels and restraining enzymatic activity whenever needed.

3.3.4 Regulatory motifs

Many metabolic pathways are dependent on the activity of transcriptional regulators, often organized into regulatory structures or motifs, such as SIMs, RSIMs, Bowties and FFLs, acting as specific functional modules (Table 3. 2). Therefore, the association of certain structures with particular metabolic activities can be hypothesized as a consequence of specific information processing.

SIM and RSIM motifs are simple regulatory structures that coordinate the activity of multiple genes. While SIMs represent patterns where a set of genes are controlled by a single TF (i.e. a one-to-many relationship), RSIMs display structures where a single gene is being controlled by multiple regulators (i.e. a many-to-one relationship). The occurrence of these two-gene patterns, particularly SIMs, among the transcriptional regulatory network of *E. coli* has been previously evaluated [9]. These regulatory structures are commonly associated with the regulation of genes that belong to the same operon, and usually form a protein assembly (such as flagella), or are part of a metabolic

pathway (such as the amino acid biosynthesis). Most genes in *E. coli* are regulated by several TFs, either negatively or positively.

Table 3.2 - Statistical evaluation of the occurrence of regulatory motifs per pathway. Abbreviations: # is the number of motifs of type T found in pathway P; T_P is the relative frequency of a motif type T in pathway P; and G_P_T is the frequency of genes from the pathway P (G_P) involved in motif type T (see section 3.3.2 for details). The values of T_P indicate the prevalence of motif type T in metabolic pathway P, while the values of G_P_T indicate the number of genes in pathway P that are affected by the regulation of motif type T.

Pathway	FFL			SIM			RSIM			BowTie		
	#	T_P	Gp_T	#	T_P	Gp_T	#	T_P	Gp_T	#	T_P	Gp_T
Alanine and Aspartate Metabolism	3	0%	0.27	6	5%	1.00	4	1%	0.36	2	10%	0.18
Alternate Carbon Metabolism	144	13%	0.91	60	46%	1.74	82	17%	0.52	9	45%	0.21
Anaplerotic Reactions	6	1%	0.60	7	5%	1.70	3	1%	0.30		0%	
Arginine and Proline Metabolism	10	1%	0.25	15	12%	1.33	12	2%	0.30	3	15%	0.15
Cell Envelope Biosynthesis		0%		9	7%	0.38	4	1%	0.08	1	5%	0.02
Citric Acid Cycle	61	6%	3.39	17	13%	4.11	17	3%	0.94	6	30%	0.89
Cofactor and Prosthetic Group Biosynthesis	45	4%	0.33	27	21%	0.63	23	5%	0.17	5	25%	0.10
Cysteine Metabolism	6	1%	0.30	6	5%	0.85	4	1%	0.20	1	5%	0.05
Folate Metabolism	2	0%	0.50	5	4%	1.25	1	0%	0.25		0%	
Glutamate Metabolism	41	4%	3.73	13	10%	3.18	6	1%	0.55	5	25%	1.18
Glycerophospholipid Metabolism	7	1%	0.35	8	6%	0.45	2	0%	0.10	1	5%	0.05
Glycine and Serine Metabolism	7	1%	0.47	12	9%	1.40	6	1%	0.40	2	10%	0.13
Glycolysis/Gluconeogenesis	15	1%	0.47	10	8%	1.06	8	2%	0.25	3	15%	0.09
Glyoxylate Metabolism		0%		1	1%	0.50		0%			0%	
Histidine Metabolism		0%		1	1%	0.11		0%			0%	
Information Transfer	96	9%	0.57		0%		57	11%	0.34	16	80%	0.36
Inorganic Ion Transport and Metabolism	38	4%	0.57	28	22%	1.54	26	5%	0.39	5	25%	0.39
Lipopolysaccharide Biosynthesis / Recycling		0%		4	3%	0.13	2	0%	0.04	1	5%	0.04
Membrane Lipid Metabolism	2	0%	0.14	7	5%	1.21	7	1%	0.50	1	5%	0.14
Methionine Metabolism	3	0%	0.23	6	5%	1.00	6	1%	0.46	1	5%	0.08
Methylglyoxal Metabolism	1	0%	0.10	2	2%	0.20	1	0%	0.10		0%	
Murein Biosynthesis		0%		1	1%	0.10		0%			0%	
Murein Recycling	2	0%	0.06	8	6%	0.59	6	1%	0.18		0%	
Nitrogen Metabolism	49	5%	3.27	15	12%	5.53	11	2%	0.73	4	20%	1.53
Nucleotide Salvage Pathway	16	1%	0.28	18	14%	0.82	12	2%	0.21	2	10%	0.14
Oxidative Phosphorylation	218	20%	2.83	24	18%	3.35	52	10%	0.68	8	40%	0.65
Pentose Phosphate Pathway	3	0%	0.23	8	6%	0.77	3	1%	0.23	2	10%	0.15
Purine and Pyrimidine Biosynthesis	4	0%	0.18	8	6%	1.27	8	2%	0.36	3	15%	0.23
Pyruvate Metabolism	42	4%	2.10	17	13%	2.50	13	3%	0.65	3	15%	0.50
Threonine and Lysine Metabolism	5	0%	0.26	11	8%	0.63	4	1%	0.21	3	15%	0.16
Transport, Inner Membrane	229	21%	1.06	86	66%	1.93	111	22%	0.51	18	90%	0.39
Transport, Outer Membrane	12	1%	0.57	18	14%	1.76	7	1%	0.33	2	10%	0.33
Transport, Outer Membrane Porin	8	1%	2.00	9	7%	3.50	2	0%	0.50	1	5%	0.25
tRNA Charging		0%		1	1%	0.04		0%			0%	
Tyrosine, Tryptophan, and Phenylalanine Metabolism	2	0%	0.08	7	5%	0.79	4	1%	0.17		0%	
Unassigned	17	2%	0.74	16	12%	1.26	7	1%	0.30	4	20%	0.35
Valine, Leucine, and Isoleucine Metabolism	15	1%	0.88	10	8%	1.76	7	1%	0.41	2	10%	0.18

This stringent control of gene expression ensures that cells are able to respond to several different stresses through the activation of rescue activities, such as anaplerotic reactions or, more commonly, by spreading this information from global regulators to more specialized regulatory modules. Pathways like "Alternate Carbon Metabolism" and "Transport, Inner Membrane" have several genes whose expression is dependent on these forms of regulation to respond to external stimuli (in particular, to alterations in the nutrient carbon sources). As presented in Table 3.2, the "Alternate Carbon Metabolism" and "Transport, Inner Membrane" are coordinated by multiple regulatory structures encompassing about 46% and 66% SIMs and 17% and 22% RSIMs, respectively. RSIMs are also widely represented in "Information Transfer" with a percentage of almost 11%. Clearly, there is an overlap of regulatory structures, which are essentially composed by RGs, repeated across pathways. Nevertheless, this information is still relevant to have an overview of the complexity of the regulatory mechanisms behind certain metabolic functions.

The bowtie motif can be interpreted as the coupling of SIM and RSIM motifs through a single central element, suggesting that these structures share a similar conceptual and architectural design. TFs that regulate more than one operon (i.e. genes involved in more than one metabolic function that are scattered over different operons) and are regulated by more than one TF are frequently part of these patterns as central nodes. Their main role is to spread information from different stimuli to the adequate functional components, behaving like decision points where a TF elects the most appropriate response to a certain stimulus. For example, the transcriptional activator *GadE* controls the transcription of genes involved in the maintenance of pH homeostasis, glutamate dependent (GAD) system and multidrug efflux, among others, in response to the transcriptional control of two-component systems, like *EvgS*/*EvgA* and *PhoQ*/*PhoP*, and other regulators involved in cellular response to acid resistance or in the catabolism of secondary carbon sources [10].

The capacity of central nodes to admit variability of input information (i.e. regulation from other genetic elements), confers high flexibility and robustness

to the system, while supporting the modulation of multiple pathways simultaneously [11]. Pathways like "Transport, Inner Membrane" and "Information Transfer", which support inherently complex information exchange processes, are in need of these regulatory structures to guarantee the adequate propagation of the information throughout the network (90% and 80%, respectively). However, the number of genes that are regulated by this type of structure is higher in pathways like "Glutamate Metabolism" and "Nitrogen Metabolism" (Gp_T values equal 1.18 and 1.53, respectively), probably as means to guarantee the activation of these metabolic functions in a more effective way.

The FFL motif represents a simple form of regulation where the activities of two RGs regulate the expression of the target gene both directly and indirectly. These motifs are likely to occur when a rapid response to an external signal is required, such as shifts in carbon sources or availability of oxygen [9]. The abundance of these patterns suggests that many metabolic functions might respond favorably to persistent rather than transient signals. This means that activation of the target gene (node Z) via direct control is more effective if the signals from node X are transient, which would delay the conveyance of information via the indirect control (i.e. through node Y).

For this reason, pathways like "Oxidative Phosphorylation", "Transport, Inner Membrane" and "Alternate Carbon Metabolism" present higher abundance of these motifs (20%, 21% and 13%, respectively). Also, most of the genes associated with these pathways are controlled by this regulatory pattern (Gp_T values of approximately 1 or larger), which allows a rapid functional switch in response to stimulus, e.g. deficiencies in the availability of exogenous electron acceptors like nitrate, nitrite, fumarate [12]. Although few FFL motifs were found in pathways like "Glutamate Metabolism" (4%), "Citric Acid Cycle" (6%) or "Nitrogen Metabolism" (5%), the number of genes that were repeatedly included in this type of structure was considerable (Gp_T values greater than 3). This indicates that the density of GRs overlapping is higher in these pathways and the flow of regulatory information crossing these GRs is decisive. This feature can be also observed in other regulatory structures for the same pathways, for example

the regulation of "Nitrogen Metabolism" comprises 15 SIMs (around 12% of the SIMs found in the network), but the pathway only has 15 genes, which means that some genes must be included more than once in these structures explaining the Gp_T value larger than 5. These properties provide evidences that the entanglement of interactions and the density of regulatory structures coupled to metabolic networks define largely the complexity behind the coordination of cellular metabolism.

A number of detailed examples of regulatory mechanisms governing some of the metabolic activities in *E. coli* can be found in [6].

3.3.5 Discussion

Considering that TFs and metabolic regulators have similar functional purposes, i.e. to ultimately regulate the activity of enzymatic reactions, the integrated analysis of their activities provides a new perspective over the capacity of modulation of *E. coli* metabolic pathways. TF-based regulation is meant to perform system adaptation whereas enzymatic regulation is chosen when a rapid shift in a given metabolic activity is needed. For example, the shift from aerobic respiration to anaerobic respiration requires gene expression adjustments as the cells have to adapt their entire metabolism to a new environment. In turn, enzymatic regulators are needed to balance certain metabolic pools and thus, maintaining the concentration of end products within acceptable ranges. Moreover, most of the observations gathered for one pathway and its regulation via a given TF or metabolic regulators show that the cooperation of both regulatory mechanisms is fundamental to coordinate the activity of the many enzymes working in a pathway.

The full networking between TFs, by which cells can adapt their metabolic states, is known to be supported by various structures of regulation capable of responding to one or more environmental/ internal inputs. Each structure has a unique way to process information (it may receive multiple inputs and/or it may affect multiple gene targets) and its relevance over a pathway is given by the

number of affected genes belonging to it. This study discriminates pathways that are heavily regulated from those where TF regulation is scarce. Additionally, the study has shown that certain regulatory structures are characteristic of a subset of pathways.

References

[1] J. Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A. , Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofm, "The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524-531, 2003.

[2] M. Hucka, A. Finney, S. Hoops, and S. Keating, "Systems biology markup language (SBML) Level 2: structures and facilities for model definitions," *Nature*, 2008.

[3] P. Gerlee, L. Lizana, and K. Sneppen, "Pathway identification by network pruning in the metabolic network of Escherichia coli.," *Bioinformatics (Oxford, England)*, vol. 25, no. 24, pp. 3282-8, Dec. 2009.

[4] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and a.-L. Barabasi, "The large-scale organization of metabolic networks," pp. 651-654, Oct. 2000.

[5] P. . Keseler, I.M., Collado-Vides, J., Santos-Zavaleta, A., Peralta-Gil, M., Gama-Castro, S., Muniz-Rascado, L., Bonavides-Martinez, C., Paley, S., Krummenacker, M., Altman, T., Kaipa, P., Spaulding, A., Pacheco, J., Latendresse, M., Fulcher, C., Sarker, M., S, "EcoCyc: a comprehensive database of Escherichia coli biology," *Nucleic Acids Research*, vol. 39, pp. 583-590, 2011.

[6] S. Carneiro, J. Pinto, and M. Rocha, "A Study of the Short and Long-term Regulation of E. coli Metabolic Pathways," *Regulation*, vol. 8, no. 3, 2011.

[7] A. M. Feist et al., "A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information.," *Molecular systems biology*, vol. 3, no. 121, p. 121, Jan. 2007.

[8] P. Hu et al., "Global functional atlas of Escherichia coli encompassing previously uncharacterized proteins.," *PLoS biology*, vol. 7, no. 4, p. e96, Apr. 2009.

[9] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, "Network motifs in the transcriptional regulation network of Escherichia coli," *Nature genetics*, vol. 31, no. 1, pp. 64-68, May 2002.

[10] F. Hommais, "GadE (YhiE): a novel activator involved in the response to acid environment in Escherichia coli," *Microbiology*, vol. 150, no. 1, pp. 61-72, Jan. 2004.

[11] H.-W. Ma, X.-M. Zhao, Y.-J. Yuan, and A.-P. Zeng, "Decomposition of metabolic network into functional modules based on the global connectivity structure of reaction graph.," *Bioinformatics (Oxford, England)*, vol. 20, no. 12, pp. 1870-6, Aug. 2004.

[12] N. P. Stenmark P, Gurmu D, "Crystal Structure of CaiB, a Type-III CoA Transferase in Carnitine Metabolism," *Biochemistry*, vol. 43 (44), pp. 13996–14003, 2004.

Chapter 4

Software tools for the analysis of strain optimization strategies in Metabolic Engineering

In this chapter, the methods and software tools developed for the use of network analysis to aid in Metabolic Engineering (ME) efforts are described. TNA4OptFlux, a plug-in, which adds to the open-source platform OptFlux the capability of creating and performing topological analysis over metabolic networks, is described. The capabilities of the tool are illustrated by using it to aid the interpretation of a four knockout *E. coli* strain designed in OptFlux for the overproduction of succinate. A larger scale study for the same target is also put forward, where solution sets from strain optimization algorithms are analysed using these tools to study common patterns used by successful mutant strains, as compared to the original wild type strains. The plug-in is available in the OptFlux web site (http://www.optflux.org).

4.1 Metabolic Engineering

Metabolic Engineering deals with the identification of rational modifications that improve the production capabilities of microbes in target compounds. One of the major challenges created by algorithms used in ME problems is the interpretation of the changes that lead to a given overproduction. Often, one single knockout induces changes in the fluxes of dozens of reactions, as compared with the wild type, and it is therefore difficult to evaluate the physiological differences of the *in silico* mutant. In this chapter, a number of tools to aid in the analysis of these mutants are proposed, based on concepts and tools from network analysis. Before addressing those in detail it is important to address basic concepts in the ME field that will be needed for the full

understanding of the remaining of this chapter. Also, OptFlux, a software platform for ME that served as a basis for the development of these tools, is described.

4.1.1 Phenotype simulation methods

Phenotype simulation methods allow the calculation of the flux values for the whole set of reactions in the model. It is possible to perform simulations of the wild type and mutant strains. In the first case, the original model is considered with no additional constraints, while in the latter a number of user selected reactions (or genes if the model includes gene-reaction associations) are inactivated in the original model before simulation.

Flux Balance Analysis (FBA) is the most popular method for phenotype simulation, where the flux of a particular reaction is typically optimised using linear programming [1]. FBA is based on a steady state approximation to concentrations of the internal metabolites, which reduces the corresponding mass balances to a set of linear homogeneous equations. For a network of M metabolites and N reactions:

$$\sum_{j=1}^{N} S_{ij} v_j = 0, \qquad i = 1, \ldots . M \tag{4.1}$$

where v_j corresponds to the rate of reaction j, or to the j^{th} metabolic flux and the stoichiometric coefficient, S_{ij}, stands for stoichiometric coefficient of metabolite i in reaction j. For most metabolic networks, the number of fluxes is greater than the number of mass balance constraints, resulting in an underdetermined system. Also, thermodynamic and capacity constraints can be added as inequalities:

$$\alpha_j \le v_j \le \beta_j, \quad j = 1, \ldots N \tag{4.2}$$

FBA uses linear programming to determine the optimal flux distributions using a specified linear objective function:

$$\begin{aligned} &\text{Maximize } Z \\ &\text{Subject to } \sum_{j=1}^{N} S_{ij} v_j = 0, \quad i = 1,\ldots M \\ &\alpha_j \le v_j \le \beta_j, \quad j = 1,\ldots N \end{aligned} \tag{4.3}$$

For metabolic applications, the objective function (Z) to be maximized can correspond to different objectives ranging from a particular ME design objective to the maximization of cellular growth. Since studies in several different organisms have demonstrated that their metabolic networks have evolved for the optimization of the specific growth rate under several carbon source–limiting conditions [2], the most commonly used objective function is the maximization of the biomass formation reaction rate.

Other alternatives to FBA have been proposed, including some more recent variants such as Parsimonious FBA [3], where a second optimization problem follows the first to guarantee that among all possible optimal solutions, it is chosen the one minimizing the total sum of fluxes.

4.1.2 Strain optimization algorithms

Based on the phenotype simulation layer, it is possible to develop algorithms that aim to find the best set of genetic modifications to apply to a given organism in order to achieve a mutant strain that is able to overproduce a given compound. In previous work, the host group for this thesis has been active in proposing meta-heuristics, such as Evolutionary Algorithms (EAs) [4] and Simulated Annealing (SA) [5] for these tasks.

These algorithms are able to identify sets of reaction deletions (or gene deletions if gene-reaction associations are available) that maximize a given objective function related with a desired industrial objective. The ultimate purpose of the

implemented algorithms is to identify genetic modifications that force the microorganism to produce a particular metabolite, while still obeying the physiological aim of maximizing biomass production.

In both algorithms, the encoding of a solution is achieved by a variable size set-based representation, where each solution consists of a set of reactions from the model that will be deleted. For all reactions deleted, the flux will be constrained to 0, therefore disabling them from the metabolic model. The process proceeds with the simulation of the mutant using the chosen phenotype simulation method (e.g. FBA). The output of these methods, i.e. the set of fluxes for all reactions, is then used to compute the fitness value, given by the objective function.

In this work, the objective function used is the Biomass-Product Coupled Yield (BPCY) [4], given by: $BPCY = PG/S$, where P stands for the flux representing the excreted product; G for the organism's growth rate (biomass flux) and S for the substrate intake flux. Besides optimizing for the production of the desired product, this function also allows to select for mutants that exhibit high growth rates.

A more detailed description of these algorithms, including their overall structure, the reproduction operators used and other important configuration issues are fully described in [5].

4.1.3 OptFlux

OptFlux [6] is an open-source software platform for Metabolic Engineering (ME) based on the use of stoichiometric models and constraint-based approaches to metabolic modelling. It has been developed with the aim to expand the user base of these methods beyond bioinformaticians and expert researchers, being characterized by its user-friendly interface.

OptFlux is able to create metabolic models by importing data in different formats including flat files (in an application specific format), comma separated text files, Systems Biology Markup Language (SBML) or Metatool files. The models used in the platform are stoichiometric models and therefore include information on the set of reactions (their equations and reversibility), metabolites and, if available, gene-reaction associations.

OptFlux includes operations to run *in silico* phenotype simulations of the wild type or mutant strains in different environmental conditions (e.g. media), using several methods. These include the popular approach of FBA and its variants, as well as specific methods for the simulation of mutants.

Users can also perform strain optimization, using EAs or SA. These algorithms return several solutions, each consisting of a set of gene/ reaction knockouts.

To ensure that it could be adapted and extended to the user needs, both by its original developers and by third-party players, OptFlux possesses a modular component-based architecture that allows an expansion of its functionalities through the use of plug-ins, such as the one described in the next section. Some relevant details on the implementation of OptFlux are included in the next chapter.

OptFlux is an open-source project, being the application freely available for the community in the web site: http://www.optflux.org, where there is also abundant documentation including detailed “how to’s” on the major operations and use cases.

4.2 TNA4OptFlux

This section presents a software tool, the Topological Network Analysis for OptFlux (TNA4OptFlux), a plug-in that adds to OptFlux the capability of creating metabolic networks from the OptFlux stoichiometric models and performing topological analysis over them. One of the major advantages is the possibility of

using its functionalities in the analysis and comparison of simulated phenotypes, namely those coming from the results of strain optimization algorithms. This plug-in is available in OptFlux's web site, together with extensive documentation.

One of the major challenges created by algorithms that search for non-intuitive ME solutions such as knockouts that improve the production of target compounds, is the interpretation of the changes that lead to that overproduction. Often, one single knockout induces changes in the fluxes of dozens of reactions, as compared with the wild type, and it is therefore difficult to evaluate the physiological differences of the *in silico* mutant. This is aggravated by the fact that genome-scale models *per se* are difficult to visualize, given the high number of reactions and metabolites involved.

Besides adding network analysis functionalities to OptFlux, TNA4OptFlux was also created with the objective of providing a bridge between the model-based phenotype simulation methods and network-based topological analysis methods. By combining these two methodologies, TNA4OptFlux gives users a new way of studying the results of simulations obtained with OptFlux, broadening the application capabilities for analysing ME solutions. This feature distinguishes TNA4OptFlux from other similar works. In fact, with the exception of the area of pathway analysis, including the calculation and analysis of Elementary Flux Modes (EFMs) [7] these fields have traditionally remained separated.

4.2.1 Creating, visualizing and exporting networks

TNA4OptFlux supports three types of networks that can be generated from the same metabolic network information, namely those defined in section 2.2.1: *reaction-compound*, *compound-compound* and *reaction-reaction* networks. These three types can be easily created with minimal human input. All the previous alternatives are based in directed graphs. Reversible reactions in all of the three representations use parallel directed edges pointing in opposite directions.

Networks in TNA4OptFlux are always created based on an existing model loaded in OptFlux. Regardless of the type, the metabolic networks obtained are typically quite complex graphs with a large number of vertices and edges. This is expected, since OptFlux typically works with genome-scale models that have a few thousand/ hundreds of reactions and metabolites. Unfortunately, this places these metabolic networks well beyond the capability of even the most sophisticated current graph layout algorithms. For this reason, TNA4OptFlux, like InBiNA (chapter 3), does not provide the graphical visualization of full networks. Instead, a set of tables is used to represent and characterize the network. These tables contain data regarding both the network's structure and associated metadata.

While a visual representation of the full graph is not supported, TNA4OptFlux offers users some visual support, using a radial graph panel associated with each network. Here, it is possible to select a vertex and the plug-in draws a small graph with that vertex in the centre and its neighbours. The network can then be navigated by clicking in one of the vertices in the radius, which is redefined as the central vertex and the graph is redrawn. Additionally, if for a given situation visualizing only the immediate neighbours of the central vertex is insufficient, the distance of the vertices from the central vertex to be included in the graph (which is by default 1) can be adjusted up to 5, creating circles of growing radius. This feature is illustrated in the two screenshots shown in Figure 4.1 below.

TNA4OptFlux is capable of exporting networks into several different kinds of file formats that are used by other network analysis applications. This allows users to complement the analysis conducted in this plug-in with the functionalities of other software tools. Currently, the following file formats are supported:

- SBML [8][9] - standard format for the representation of biological models, being supported by several network analysis applications including Cytoscape [10] and CellDesigner [11].

- Pajek [12] file format- one of the most complete applications for network analysis, possessing a large number of methods of network analysis and manipulation.

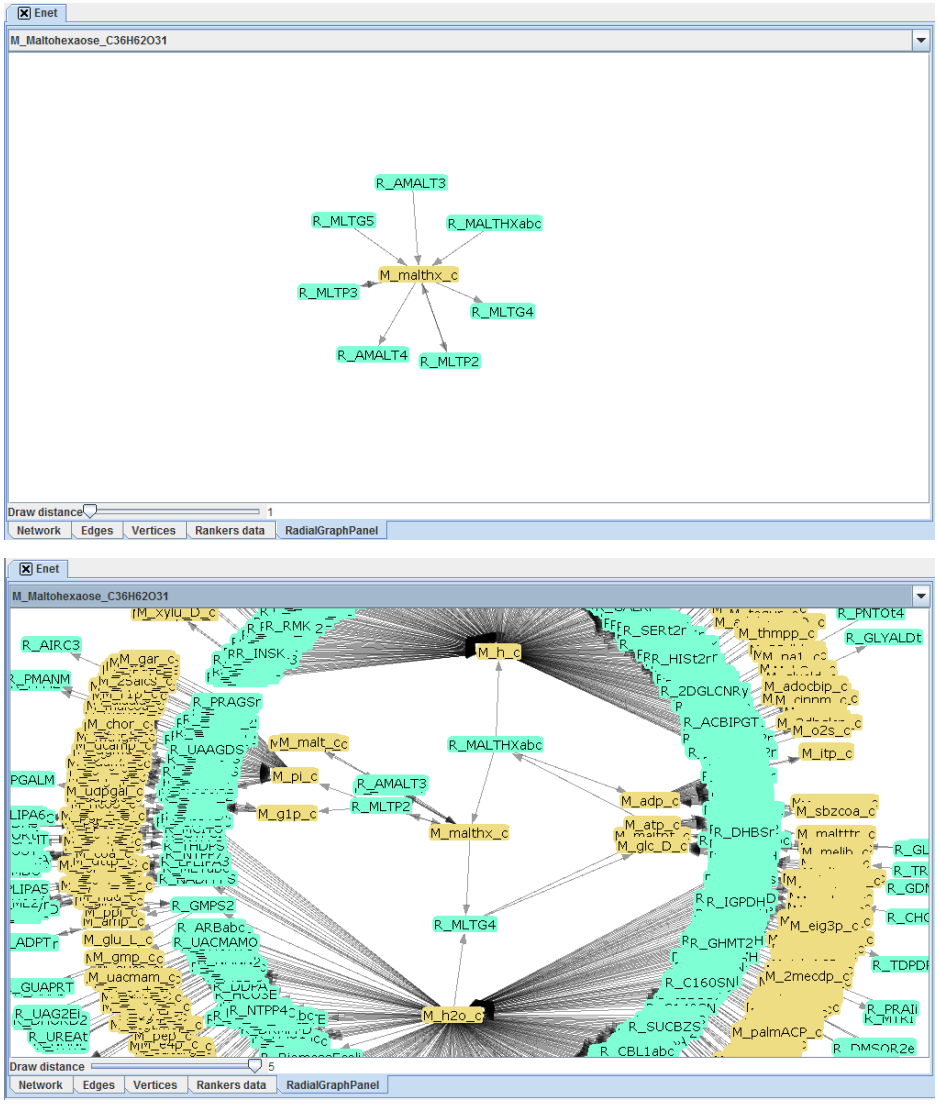


Figure 4.1 - An example of visualization functionalities of TNA4OptFlux; notice how the complexity increases with the radius (also called draw distance).

- XGMML format - An XML based network format, capable of storing all the metadata present in a network, facilitating the interchange between applications. The support for XGMML was included mostly for allowing the visualization of variation networks in other applications (see below).

Unlike full metabolic networks, variation networks are typically composed by only a few independent modules, allowing their easier visualization.

- MBNF – a flat file ASCII format developed for InBiNA (see chapter 3), thus allowing the integration of both tools.

4.2.2 Topological analysis tools

Often, regardless of the system it represents, the topological analysis of a network is the first step in its study, revealing interesting properties of its structure. So, naturally, some topological analysis functionalities are provided by TNA4OptFlux. The topological analysis tools included are the degree analysis, the shortest path analysis and the calculation of distinct rankings based on centrality metrics.

Degree analysis

The degree analysis functionality provides the users a table view containing the degree of all vertices, discriminated by in- and out-degree. It can also be used to identify the list of neighbours of a selected vertex through a pop-up window, providing an idea of that vertex's relative position in the network. A global analysis is provided in the form of degree histograms, either in table format or as a chart. Finally, it also calculates the degree distribution of the network.

Shortest path analysis

The determination of shortest paths can be used, for example, to give an idea, in a metabolic network, of the transformation an initial compound goes through until a desired substance is obtained. The plug-in supports the calculation of shortest paths between two selected vertices and can also be used to determine all the other vertices that a source vertex is connected to and the respective distances. Also, it includes a few global network metrics, such as the network diameter and the mean shortest path calculated over all pairs of connected

vertices in the network. This last metric can be useful to evaluate if the network is a small world network.

Ranking algorithms

This plug-in also includes a number of ranking algorithms, i.e. methods that provide a metric for each vertex, allowing them to be ranked. Besides the basic degree ranker, other more elaborate methods have been included, including three known centrality metrics: betweenness centrality (BC), closeness centrality (CC) and the hubs and authorities method (also known as Hyperlink-Induced Topic Search or HITS). All these were described previously in section 2.1.3.

4.2.3 Locating active vertices

A reaction-compound network in TNA4OptFlux has a structure very similar to a Petri Net [13]. Taking inspiration from this similarity, a functionality called the "location of active vertices" was implemented. It uses an algorithm that starts with a set of "seed" metabolites (the active metabolites set) and from it determines the full set of reactions that can be active when this set of metabolites is present. This functionality works through an iterative process that, at each step, adds the metabolites that can be produced by the reactions in the model by using the current set of active metabolites as substrates to the active metabolites set. Additionally, at each step, all the reactions where all substrates are present in the active metabolites set are added to the active reaction set (which starts empty). The process continues until no further metabolites can be added to the active metabolites set. The final result are sets of reactions which will be active and of the metabolites which will exist in the system assuming an unlimited supply of the "seed" metabolites and sufficient time for the reactions to occur.

This functionality can help users determine if some metabolites are related with the production of other seemingly unrelated metabolites. A practical use is to define as active a set of external metabolites representing the components of the

growth medium (nutrients) and determining all metabolites that can be potentially produced based in these initial seed metabolites.

4.2.4 Filters

One of the difficulties of working with metabolic networks is the presence of the so-called 'currency metabolites' (or ubiquitous compounds) such as ATP, NAD and protons, which take part in a large number of reactions [14]. However, despite their large presence, currency metabolites often cannot be considered as valid intermediates for path finding, or for that matter, for establishing biologically meaningful network connections [15]. Thus, currency metabolites are often removed from metabolic networks before their analysis. However, since there is no commonly agreed definition of currency metabolites, often their removal is not as simple as it may seem and can vary from study to study.

To address this problem and other similar tasks, filtering capabilities, which allow the removal of selected vertices, based in user defined criteria were included in TNA4OptFlux. The possible parameters for the removal of vertices include among others: a user defined list, degree values and minimum thresholds for the available ranking algorithms.

Since the degree is often the metric used to identify currency metabolites, most of the filtering operations within TNA4OptFlux are based in the degree, either by defining a degree threshold for vertex removal or by permitting the removal of the top **n** vertices with higher degree (where **n** is an user defined value). All these filtering operations can use either the value of degree or discriminate between out or in degree.

4.2.5 Linking network analysis and phenotype simulations

As mentioned in earlier sections, TNA4OptFlux was created with a more ambitious objective besides simply allowing the analysis of metabolic networks

associated with stoichiometric models: to serve as a bridge between model-based and network-based analysis methods. Thus, a series of methods for the integration of simulation results with metabolic networks and processes for their subsequent analysis have been developed. These are presented in this section.

Simulation filtering

In a typical phenotype simulation (e.g. using FBA) only a few of the metabolic reactions will take place, i.e. have a flux different from zero. This makes sense, since the bulk of the metabolism is composed by redundant or complimentary metabolic capabilities whose occurrence depends on the conditions the organism is subject to and the available substrates.

To this effect, network filtering features were added to TNA4OptFlux, based on the results of phenotype simulation processes. These methods take as inputs a complete metabolic network (typically the original network created from the whole model) and the results of a phenotype simulation (made over the same model), and produces as output a sub-network containing only the parts of the metabolism that are active on that simulation. The obtained filtered network can be considered as a "snapshot" of the metabolism in the simulated conditions.

The process of simulation filtering is composed of two steps, as follows:

1. The first step is to remove from the network all vertices corresponding to reactions with a predicted flux below a given threshold; these are considered to be inactive reactions.

2. After removing the inactive reactions from the network, some vertices will have a zero degree, i.e. they are isolated from the rest of the network. This happens with metabolites that were only involved (produced and consumed) in reactions that were removed. The second step consists in removing every isolated vertex from the network.

These results of the simulation filtering can be seen as a "map", a sub-network containing all parts of the metabolism that are predicted to be used in a given condition. The study of the properties of this network can give insights into the behaviour of the cell and the mechanisms involved in the metabolic responses in the conditions of the simulation. Also, the comparison of different sub-networks of the same network, induced by the results of simulations under distinct genetic/ environmental conditions can highlight the major differences in the metabolic behaviour of the two scenarios.

Network comparison

Besides the analysis of a single network, the plug-in allows comparing the structure of different networks. This feature can be especially useful when analysing several networks obtained from the same model through simulation filtering. Some examples of fruitful application of this tool are the comparison of mutant strains with selected gene deletions versus wild type strains and the comparison of simulations from distinct environmental conditions.

As in InBiNA, most of the network comparison functionalities are based in the provided network analysis metrics. These are used to identify the differences between the networks. Once again, to make a good use of computational resources, when two networks are compared, the plug-in checks which analysis metrics were applied to both and uses those in the comparison. Thus, to have a metric included in this comparison, the calculation needs to be previously executed for both networks. The comparison results are, however, updated when new metrics are calculated for the networks.

The results of a process of network comparison are presented as a series of tables where the values of the metrics applied to individual vertices and for the networks as a whole can be observed. It should be noted that, when comparing networks, TNA4OptFlux assumes that vertices sharing the same id and type represent the same biological entity. This assumption is always true when the networks compared are derived from the same model. Some care should be taken however when comparing networks obtained from different models.

When a phenotype simulation is performed for a model in OptFlux, flux values are kept in an object in the clipboard. In this plug-in, we have added a feature that, while not directly related to network analysis, can be of help in the comparison of different simulation results and complement the topological analysis tools. This feature allows a set of simulations to be selected and the results of the comparison between them to be presented both as a table and as bar plots that show the variation of flux values.

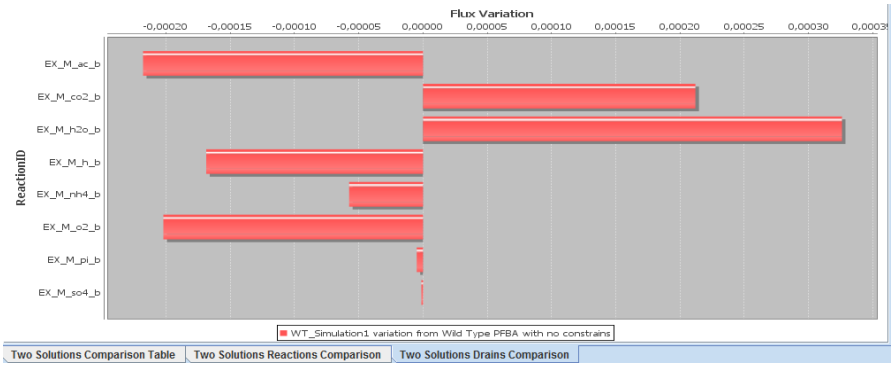


Figure 4.2 - A bar plot of the drain reactions flux variations generated by TNA4OptFlux flux comparison feature.

Variation networks

A variation network is obtained by comparing a pair of networks obtained through simulation filtering, using the same base network and simulation results obtained under different conditions. The result is the creation of a third network containing the elements that differed between the compared networks (i.e. the variations).

The idea behind the use of variation networks is to identify the components of a given metabolic system that change when the conditions inherent to the simulations change. Assuming the dimension of these variations to be significantly smaller than the networks themselves, variation networks should be easier to analyse than a full network.

The most important decision in this process is to select how the variations between the networks should be identified, i.e. what criteria should be used to decide on the components to be selected. Two methods are included in TNA4OptFlux:

1. *Exclusivity*: this is the simplest method but it has proven useful in analysing phenotype simulation results (see the results from section 4.3). The criterion used is based on the identification of the exclusive reaction vertices in each network, i.e. those existing in one of the networks but not the other. After the identification of these reactions, the respective vertices are added to the variation network, together with the vertices corresponding to metabolites that they consume or produce, as well as the edges connecting them.

2. *Flux variation*: This second method was developed after it was verified that methods based purely on network topology can be, in some cases, insufficient to capture more subtle metabolic flux variations. The flux variation is based in the comparison of flux values of the reactions present in both networks: if the absolute value of the difference in fluxes exceeds a user defined threshold (which can be an absolute flux value or a percentage) then the vertex corresponding to that reaction, as well as the ones corresponding to the metabolites which participate in it and the edges which connect them, are added to the variation network.

These two methods can be used independently or combined. After a variation network is created, it can then be analysed as any other network using any of the tools available in the plug-in. There is, however, a point in which variation networks differ from other TNA4OptFlux networks: they have metadata associated with their vertices that identify why each vertex was added to the network (values related to exclusivity, flux change, and flux values for each of the reactions). These tags are saved with the rest of the metadata when a variation network is exported into an XGMML file and can be used to visualize the type of variations associated with each vertex when using other network analysis tools.

Indeed, some visualization applications can change the appearance (e.g form, colour) of the vertices based in these tags. For instance, in the case study described next, these tags were used to customize the colour of the vertices when using Cytoscape [16] to visualize the variation network.

Besides the comparison of pairs of networks, the concept of variation networks was expanded to work with simulation sets, i.e. sets of phenotype simulations. For creating a variation network from a simulation set, one network not belonging to the set (e.g. obtained from an wild type simulation) is considered the reference network. This network is compared to all networks of the set using the same methods as before. In this case, vertices are only included in the final variation network if they have shown a significant variation in at least K comparisons (where K is a user defined parameter).

The creation of variation networks from simulation sets was designed as a method to identify common changes in mutants (as compared to a wild type strain) in multiple gene knockout mutants resulting from the application of a strain optimization algorithm, i.e. strains that were optimized for the same objective. The idea was that those common changes might contain key alterations, highlighting common features occurring in the metabolism of a host organism to adopt the desired behaviour.

4.2.6 Case study: succinic acid production with *E. coli*

In order to evaluate the usefulness of the developed tool on a practical example, strain optimization results previously obtained using OptFlux were used. The mechanism behind a set of knockouts for the production of succinic acid, using *E. coli* as a host, was analysed. The aim was to show that this plug-in can prove extremely valuable, especially when combined with network visualization software, such as Cytoscape.

We began by picking a solution of reasonable complexity, that was neither too obvious nor had too many knockouts. In Table 4.1, a brief description is given of

the set of knockouts elected as our case study. Apparently, there is nothing in common between the reactions in the table, and with the exception of succinate dehydrogenase, it is difficult to relate the inactivation of these reactions to succinate production.

Table 4.1 - Set of knockouts and the corresponding chemical reactions used as case study.

Reaction	Stoichiometric equation
Serine hydroxymethyltransferase	L-serine + tetrahydrofolate <=> glycine + 5,10-methylenetetrahydrofolate + H_2O
pyridine nucleotide transhydrogenase	H^+_e + NADH + $NADP^+$ => H^+_c + NAD^+ + NADPH
Succinate dehydrogenase	FAD^+ + Succinate => $FADH_2$ + Fumarate
Transketolase I	D-erythrose-4-phosphate + D-xylulose-5-phosphate <=> D-fructose-6-phosphate + D-glyceraldehyde-3-phosphate

To analyse this case study, the first step was to simulate both the wild-type and selected mutant using Parsimonious FBA [3], one of the methods available in OptFlux. Then, the reactions with zero flux were filtered out and some currency metabolites (H_2O, H^+, AMP, ADP and ATP) were removed from the network, to simplify the graphical representation. Then, the variation network between the wild-type simulation and the succinate producing mutant was computed. The options were set to include the reactions exclusive to each simulation and the reactions where the flux had changed above 2 mmol/gCDW.h. The resulting variation network was exported to an XGMML file and within Cytoscape we customized the visual parameters of the network to help in the analysis. Reactions exclusive to the wild-type were coloured in red, the ones exclusive to the mutant in blue, the ones with increased flux in the mutant in green and with reduced flux in the mutant in orange.

After generating the variation it was manually filtered to remove a few reactions known to have no significance to the solution based in the available knowledge about the system. The final result was the network shown in Figure 4.3. This simplified view allowed to infer that NADPH balance was the key factor behind succinate production.

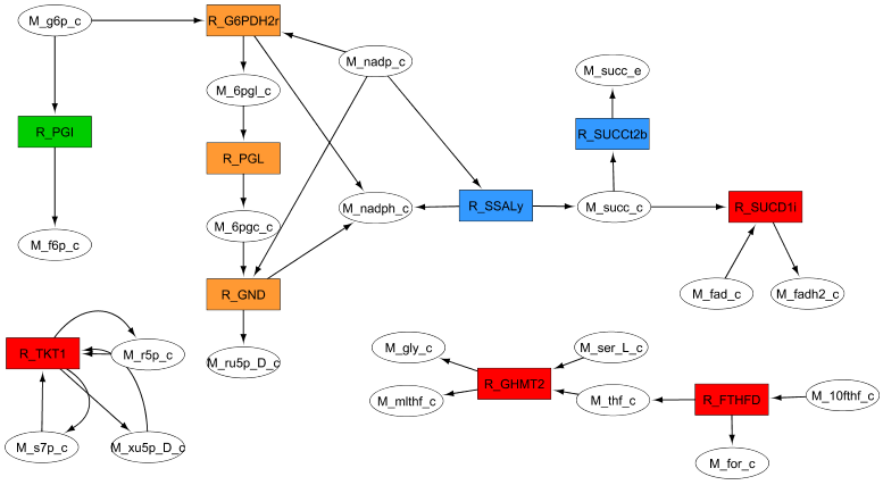


Figure 4.3 - Manually curated variation network between the wild-type and the succinate producing mutant described in Table 1. Wild-type exclusive reactions are shown in red, the ones exclusive to the mutant are in blue, while green and orange show reactions with increased or reduced flux in the mutant, respectively.

Abbreviations: M_10fthf_c – 10-Formyltetrahydrofolate, M_6pgc_c- 6-Phospho-D-gluconate, M_6pgl_c- 6-phospho-D-glucono-1,5-lactone, M_f6p_c- D-Fructose-6-phosphate, M_fad_c- Flavin adenine dinucleotide oxidized, M_fadh2_c- Flavin adenine dinucleotide reduced, M_for_c- Formate, M_g6p_c- D-Glucose-6-phosphate, M_gly_c- Glycine, M_mlthf_c- 5,10-Methylenetetrahydrofolate, M_nadp_c- Nicotinamide adenine dinucleotide phosphate, M_nadph_c- Nicotinamide adenine dinucleotide phosphate reduced, M_r5p_c- alpha –D-Ribose-5-phosphate, M_ru5p_D_c- D-Ribulose-5-phosphate, M_s7p_c- Sedoheptulose-7-phosphate, M_ser_L_c- L-Serine, M_succ_c- Succinate, M_thf_c- 5, 6, 7, 8-Tetrahydrofolate, M_xu5p_D_c- D-Xylulose-5-phosphate; R_FTHFD- formyltetrahydrofolate deformylase, R_G6PDH2r- glucose 6-phosphate-1-dehydrogenase, R_GHMT2- Serine hydroxymethyltransferase, R_GND- 6-phosphogluconate dehydrogenase, R_PGI- phosphoglucose isomerase, R_PGL- 6-phosphogluconolactonase, R_SSALy- succinate-semialdehyde dehydrogenase, R_SUCCt2b- succinate transporter, R_SUCD1i- succinate dehydrogenase, R_TKT1- transketolase.

Firstly, the deletion of Transketolase I causes a decrease of flux through the pentose phosphate pathway, which means that much less NADPH will be produced here. Together with the inactivation of pyridine nucleotide transhydrogenase, it creates a shortage of this metabolite. In the figure, we can see that in order to compensate for this shortage, the model predicts the use of succinate-semialdehyde dehydrogenase, which produces succinate while reducing $NADP^+$ to NADPH in the process. In order to promote the excretion of succinate outside of the cell, it is also vital to inactive succinate dehydrogenase.

The final deletion was not as easy to understand, because through the analysis of the variation network there was no apparent connection to the mechanism described above. Therefore, we tried to compare the distribution of fluxes between the succinate producing mutant and the triple mutant with an active Serine hydroxylmethyl-transferase reaction (R_GHMT2). We repeated the procedure followed before and obtained the network shown in Figure 4.2. The conclusion was that if only three of the deletions shown in Table 1 are implemented, a NADPH producing cycle is formed in the glycine production pathway in order to compensate for the shortage of this metabolite. This cycle is eliminated with the last knockout.

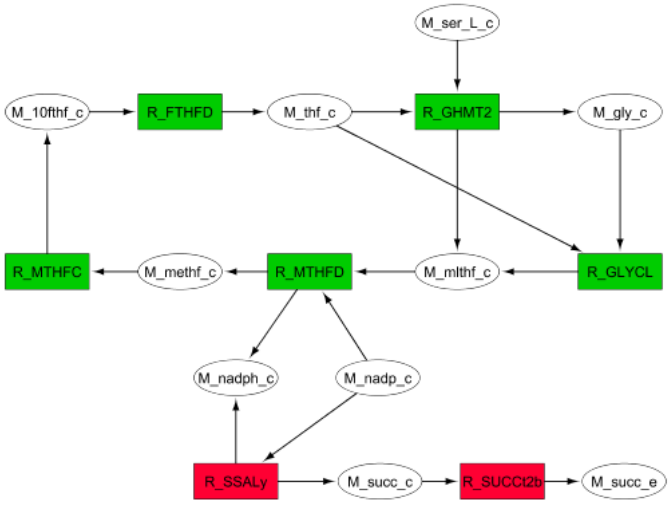


Figure 4.4 - Manually curated variation network between the succinate producing mutant and the triple mutant (R_TKT1, R_SUCD1i and R_THD2) described in Table 1. Wild-type exclusive reactions are shown in red, the ones exclusive to the mutant are in blue, while green and orange show reactions with increased or reduced flux in the mutant, respectively.

Abbreviations: M_10fthf_c – 10-Formyltetrahydrofolate, M_gly_c- Glycine, M_mlthf_c- 5,10-Methylenetetrahydrofolate, M_methf_c- 5,10-Methenyltetrahydrofolate, M_nadp_c- Nicotinamide adenine dinucleotide phosphate, M_nadph_c- Nicotinamide adenine dinucleotide phosphate reduced, M_ser_L_c- L-Serine, M_succ_c- Succinate, M_thf_c- 5, 6, 7, 8-Tetrahydrofolate; R_FTHFD- formyltetrahydrofolate deformylase, R_GHMT2- Serine hydroxymethyltransferase, R_GLYCL- glycine cleavage system, R_MTHFC- 5,10-methylene-tetrahydrofolate cyclohydrolase, R_MTHFD- 5,10-methylene-tetrahydrofolate dehydrogenase, R_SSALy- succinate-semialdehyde dehydrogenase, R_SUCCt2b- succinate transporter.

In conclusion, TNA4 OptFlux proved to be a valuable tool to help uncover non-obvious mechanisms obtained from *in silico* simulations. When used together with visualization software such as Cytoscape, it proved to reduce the analysis time needed to understand how the redirection of fluxes leads to the accumulation of the target product.

4.3 Large-scale analysis of strain optimization results[1]

In this section, automatic methods are proposed to analyse large sets of mutant strains, by taking the phenotypes of a large number of possible solutions and identifying shared patterns, using methods from network topology analysis. The topological comparison between the networks provided by the wild type and mutant strains highlights the major changes that lead to successful mutants. The methods are applied to the same case study presented in the last section, but now considering large sets of solutions (mutants) found by strain optimization algorithms: Simulated Annealing and Evolutionary Algorithms.

This study made use of many of the core tools that are described in the previous section. However, given the large-scale nature, in this case, a set of scripts to run over a command line were developed and the user interface was not deemed necessary. It should be noted, however, that the implementation of the tools is based on the same core source code. These implementation issues will be further explored in the next chapter.

It should be noted that this case study was made before the current methodology for the creation and analysis of variation networks in TNA4OptFlux was finalized. Therefore, the methods used differ in some minor points from the ones supported by the plug-in. In fact, besides validating the concept of variation networks, this case study also served to consolidate the knowledge of topological analysis processes within ME. The current version of the plug-in reflects some of the lessons learned.

[1] The work reported in this section was published in the proceedings of the international conference *Pattern Recognition in Bioinformatics 2011 (PRIB 2011)*, *Lecture Notes in Bioinformatics*, under the title "Highlighting Metabolic Strategies using Network Analysis over Strain Optimization Results"

4.3.1 Overall workflow

In this study, the workflow used can be summarized in the following steps:

Inputs: A genome scale metabolic model of a host organism; a set of currency metabolites; a metabolite of interest to be overproduced;

Step 1: The strain optimization algorithms (EA and SA) are executed with the provided configuration; each algorithm is executed a given number of runs and the result from each run is a set of solutions (mutant strains) of interest; the optimization algorithms are run according to the configuration given in [5]. The implementation of the original EA and SA methods was only modified to keep not only the best solutions obtained during the run, but rather the whole set of solutions deemed to be interesting for analysis. This does not change the optimization process, but stores more intermediate results. Therefore, each run of the algorithms generated a *solution set* containing all solutions where both the value of product and biomass fluxes where larger than 0. This set includes only simplified solutions, i.e. solutions where the removal of a given knockout would reduce the fitness function value. All solutions are simplified by removing unnecessary knockouts before entering this set.

Step 2: Solution pre-processing: the solution sets from the previous step are merged in a single set and filtered (see details in section 4.3.2).

Step 3: Each solution in the set from step 2 is simulated using FBA and the corresponding network is created according to the methods described in section 4.2.5 (simulation filtering).

Step 4: Each of the networks from step 3 is compared to the wild type network, as described in section 4.3.3;

Step 5: The comparisons from step 4 are analyzed for common patterns of variability analysis (see details in section 4.3.4);

Step 6: The results from the previous step are compiled in a sub-network that can be also visualized and manually analyzed.

4.3.2 Solution pre-processing

The first step in the pre-processing is to merge all the solutions coming from each individual run of the optimization. In this task, all duplicate solutions are removed. Also, the final solution set is checked for the existence of solutions where the set of knockouts is a superset of other solutions and these are only kept if its fitness value is higher.

The next step is to filter this solution set, since using all the solutions can be an undesirable option because in many cases they do not provide acceptable results from the biological standpoint. Also, the comparison process can be computationally heavy if the solution sets are too large.

So, the number of solutions used was reduced by filtering solutions: (i) with a low growth, by setting a minimal threshold for the biomass production flux; (ii) with a low production of the desired flux, by setting a minimal threshold for the flux associated with the excretion of the metabolite; (iii) that lead to the same set of reactions in the network after simulation filtering (see next section). In the experiments, the thresholds for (i) and (ii) correspond to 40% of the maximum value (this value was empirically set to keep solutions near to the best values obtained).

4.3.3 Network representation, filtering and comparison

The metabolic networks used are reaction-compound networks, based on directed bipartite graphs as described above. The first step was to create a

network with all reactions and metabolites contained in the metabolic model, thus creating a network that can function as a map of the organism's metabolism, a network with all the possibilities which can occur in a simulation. This network is called the *base network*.

All other networks were derived from the base network using simulation filtering as described in section 4.2.5 above. This essentially allows keeping in the network the reactions carrying flux and the metabolites used by those reactions.

The network comparison process is basically a series of operations in which each mutant network is compared with the wild type network used as a reference. These operations are typically followed by some global variation analysis to identify the most common patterns of network variation.

During the development of the methods used in this study, it was noticed that the majority of the viable mutant networks are very similar to the wild type network, which limits the use of many global metrics for graph topology, such as centrality values, shortest path analysis or clustering coefficients as comparison metrics. This led us to define some novel network comparison metrics adapted to the purposes of the work and focused on the identification of local patterns of interest in metabolic networks. The set of analyses conducted and the metrics used are defined next,

Exclusivity

This comparison metric is based on the set of exclusive nodes, i.e. those existing in one of the networks but not the other. In this case, for each mutant-wild type comparison two lists are created containing the set of nodes exclusive to each of the networks. After all mutants are compared to the wild type these lists are used to determine the frequency of each node in the exclusive lists.

Decision points

When analyzing metabolic networks it is important to look not only at the topology but also to understand the flows over the network. For instance, in the case of linear pathways with no splits, the existence of a flow in a reaction may be determined by another upstream reaction that can be distant. The decision point concept was thought as a way to determine the upstream network metabolite of pathways that exist on one of the networks and not the other. The first step in identifying decision points is to identify the decision metabolites, i.e. that are common to both networks, but that are consumed by different sets of reactions. The next step is to identify which reactions consuming these metabolites are present in one of the networks and not in the other.

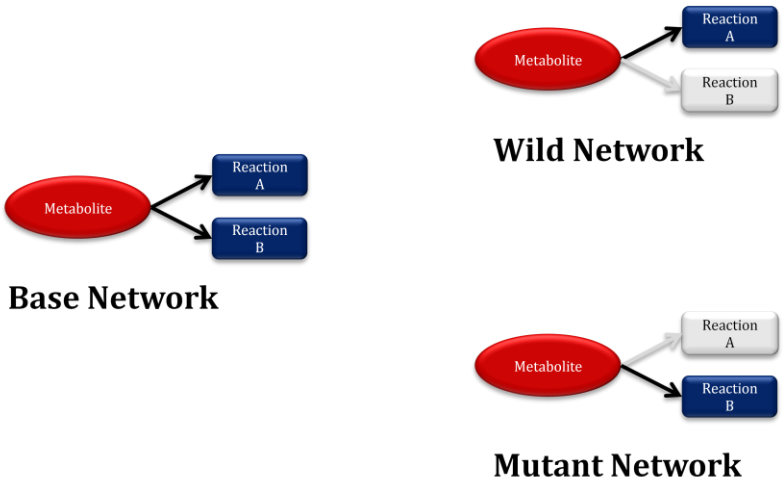


Figure 4.5 - Example of a decision point. Since the same metabolite is consumed by different reactions in the wild and mutant networks this probably corresponds to a point where the flow of matter diverges.

Inversions

Many reactions in the metabolic model are classified as reversible, meaning that they can occur on both directions. The inversion metric is based in the fact that manipulations of the metabolism can result in a flux changing signs, meaning the reaction changes its direction. All the inversions that occur in the mutants compared to the wild type are identified and their frequency is calculated.

4.3.5 Variation analysis

After all network comparisons between the mutants and the reference are conducted, the results obtained are used to identify common patterns. As a first step, the reactions that are exclusive in a significant part of the comparisons are identified. In this case, all reactions that are in the mutant exclusive lists with a frequency exceeding a threshold (in this work 80%) are identified. This group includes reactions typically used in mutant phenotype but not on the wild type strains. The same process is conducted for the wild type exclusive lists, thus identifying reactions used in the wild type but typically absent from the mutant strains resulting from the strain optimization algorithms.

Each of these sets of reactions is used to create a network also including the metabolites involved in those reactions. This typically creates several independent modules not connected by any metabolite. This network can be visualized using, for instance, the Cytoscape tool.

In some cases, the networks obtained are manually modified to include nodes and data, which could not be identified by purely computationally process, but that are nonetheless important for the analysis. The final result is a network that contains the parts of the metabolism that are more commonly altered when the organism is manipulated to produce the metabolite of interest (a variation network as defined above}.

The last step of our methodology is the analysis of the variation network. During the analysis, the variation network was also compared with pathway maps obtained from KEGG and EcoCyc to determine the relationship of the variations with the organism metabolism as a whole and if they were related with any known important metabolic cycles.

4.3.6 Experimental setup

The case study considered uses the microorganism *E. coli* and the aim is to produce succinate with glucose as the limiting substrate, as previously described above.

The genome-scale model includes a total of N = 1075 fluxes and M = 761 metabolites. A number of pre-processing steps were conducted to simplify the model and reduce the number of targets for the optimization (see [5] for details) leaving the simplified model with N = 550 and M = 332; 227 essential reactions are identified, leaving 323 variables to be considered when performing strain optimization.

In this study, we have used both EA and SA executing each algorithm for 30 runs. Each solution has 6 knockouts, since this was the minimum number of knockouts able to provide high quality solutions and an increase in this value does not provide significant gains. After the pre-processing was done we had a total of 4949 distinct networks from an initial batch of 8018 solutions.

4.3.7 Results

The results of the first stage of the analysis, i.e. the results of the comparison between mutant networks and the wild type network is provided in a spreadsheet, provided as supplementary material in http://darwin.di.uminho.pt/prib2011.

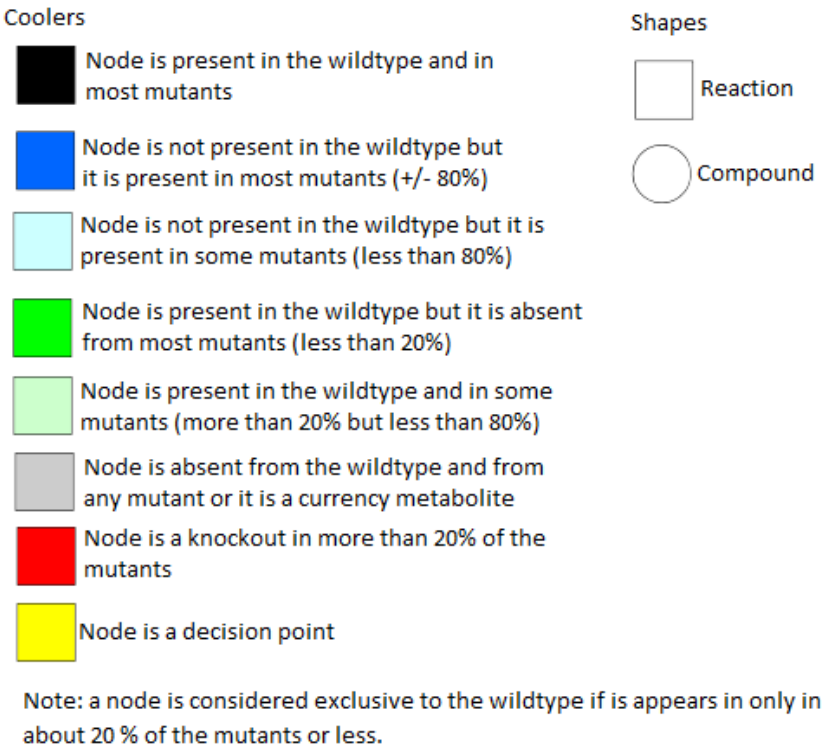


Figure 4.6 - Colour code used in the analysis of the variation network.

After conducting the variation analysis, the final sub-network was composed by three apparently independent modules. Subsequent observations revealed that these were not in fact independent changes but in fact were all directly or indirectly related to a common set of alterations, which occur in practically all mutant organisms. To show the results and address the discussion we will analyse these three modules in more detail. Figure 4.6 shows the meaning of the colour code used in the following figures to identify the different nodes coming from the analysis.

Main knockouts and succinate production

This module, shown in Figure 4.7, is particularly interesting because it contains two of the most frequently knocked out reactions (SUCD4 and SUCD1i) and it is directly related with the production of succinate. Of all the modules in the variation network this is the one whose analysis is the most straightforward, since the reaction SUCD1i, central of this module, is the only consumer of succinate in the wild type. Thus, to achieve the production of succinate, the most direct method is to remove it. SUCD1i is one of the reactions with the higher

value of wild type exclusivity and also of frequency of selection as a knockout in the solutions. SUCD1i is also the first fumarate production reaction mentioned in the dGDP Consumption Module (shown next). Its indirect effect on the former module illustrates how alterations in the metabolism can have unexpected effects.

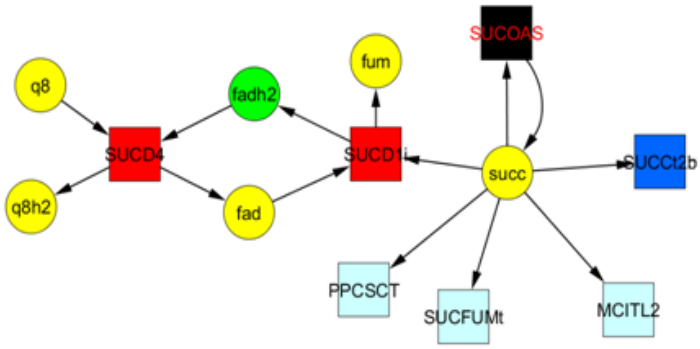


Figure 4.7 - Sub-network for the main knockouts in succinate production.

Besides the direct removal of SUCD1i, another simple way of suppressing this reaction is the knockout of SUCD4. This reaction is the main way in which FADH2 is consumed and SUCD1i is the only producer of FADH2. The removal of SUCD4 will necessarily lead to the suppression of SUCD1i.

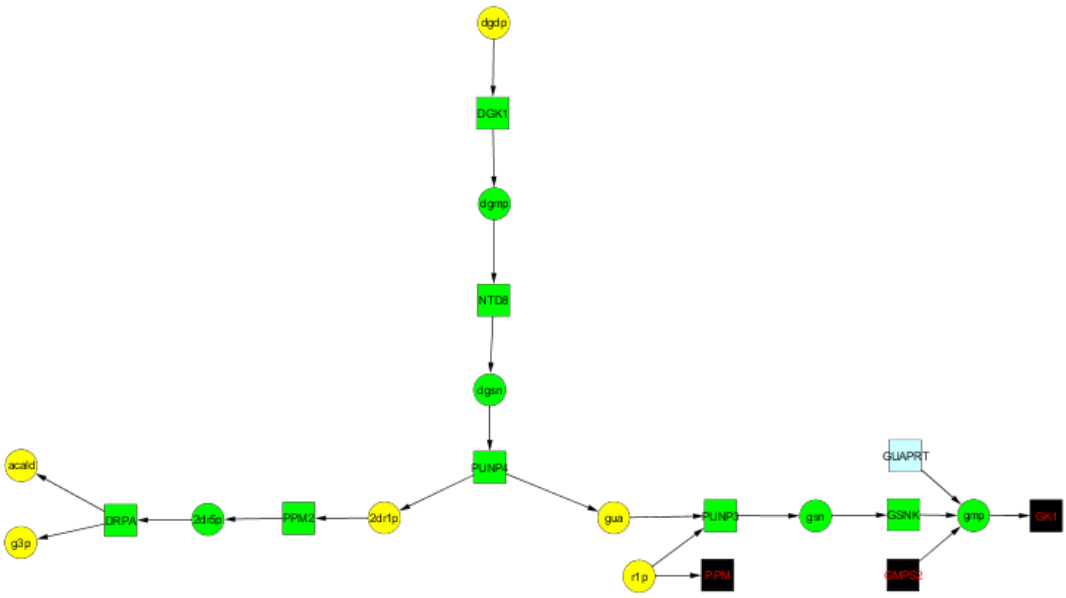


Figure 4.8 - Sub-network for dGDP consumption.

Another reaction of note is the transporter SUCCt2b which excretes succinate from the cell naturally. This is a mutant exclusive reaction for obvious reasons, since in the wild type no succinate is excreted. There are other reactions that only occur in the mutants, but the only one that appears in a significant number of mutants is SUCOAS. This reaction is also present in the wild type in its reversed form.

dGDP Comsumption

Originally, this module (Figure 4.8) appeared as two independent reaction chains exclusive to the wild type:

- Guanine (gua) + alpha-D-Ribose 1-phosphate (r1p) => PUNP3 => Guanosine (gsn) => GSNK

- 2-Deoxy-D-ribose 1-phosphate (2dr1p) => PPM2 => 2-Deoxy-D-ribose 5-phos-phate (2dr5p) => DRPA

The analysis of the variation network revealed that these two chains were actually a consequence of the wild type exclusive chain:

- dGDP (dgdp) => DGK1 => dGMP (dgmp) => NTD8 => Deoxyguanosine (dgsn) => PUNP4 => Guanine (gua) + Deoxy-D-ribose 1-phosphate (2dr1p)

This chain was not initially identified because its value of exclusivity was slightly below the defined threshold. It was determined that this module is wild type exclusive because the reaction DGK1 rarely appears on mutants. However, neither DGK1 or any of the reactions which produce the metabolites it consumes is a common knockout, which means that some alteration in the mutants' metabolism provokes the redirection of the flux of some of the compounds used by DGK1. Initial observations of the flux values of the reactions which compose this cycle and their immediate neighbors revealed that the reason for its wild

type exclusivity is a consequence of the significant reduction of dGDP in the mutants. This module does not produce any essential metabolite that can not be obtained by other reactions. Also, dGDP is necessary for the production of dGTP which is a biomass precursor. The reduction of the concentration of dGDP leads to all dGDP being channeled to the production of dGTP to ensure the survival an growth of the organism.

A more thorough analysis of the reactions revealed that the reduction of the dGDP production in the mutants is ultimately due to the reduced production of a compound used in its synthesis: 3-phosphohydroxypyruvate. This compound is obtained from a reaction which uses 3-phospho-D-glycerate as a substrate, while the production of 3-phospho-D-glycerate is not being reduced in the mutants (in fact it is somewhat increasing). Most of it is being used to produce D-glycerate 2-phosphate, which in turn is used for the production of phosphoenolpyruvate.

Continuing the analysis of the reactions which use phosphoenolpyruvate, it was determined that the increased production of phosphoenolpyruvate in the mutants is a consequence of the change in the TCA cycle due to a reduction in the production of L-malate. This, in turn, leads to a need for an increase of the production of phosphoenolpyruvate in order to maintain the cycle.

The alterations in the production of L-malate are due to the reduction of the production of fumarate, a metabolite used by a reaction external to the TCA cycle which produces L-malate. Its reduction is a consequence of the flux reduction of the two major fumarate production reactions in the mutant:

1. The main fumarate production reaction is also the main succinate consuming reaction; since the objective is to maximize succinate production this means that this reactions is a frequent knockout and even when it is not, it tends to be inactive.

2. The other reactions which produce fumarate compete with for the consumption of L-Aspartate 4-semialdehyde with the alternative L-

threonine production chain which how it will shown later is essential for the survival of most mutants

It is interesting to note that the flux reduction of the fumarate production reactions is related with two other modules which indicates that the modules of the variation network are not as independent as the initial observations of the network implied.

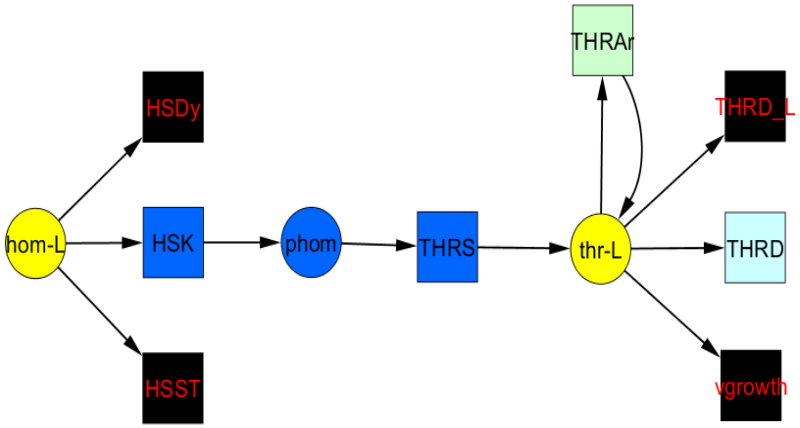


Figure 4.9 - Sub-network for alternative L-threonine production.

Alternative L-threonine production

This module, shown in Figure 4.9, is characterized by a mutant exclusive reaction chain:

- HSK => O-Phospho-L-homoserine (phom) => THRS => L-Threonine (thr-L)

Our analysis revealed that the reason for this chain being mutant exclusive lies in the reaction THRAr which normally produces L-Threonine, which is inverted in most mutants. This chains is necessary to compensate this inversion. The inversion of reaction THRAr occurs because in the inverted form it produces glycine and the alternative reaction for the production of glycine (GHMT2) is wild type exclusive.

Initially, the fact that GHMT2 was wild type exclusive appeared strange. However, we eventually concluded that this reaction is a major producer of NADPH and its removal forces the mutants to compensate by using a reaction which produces succinate as a byproduct, thus increasing the production of succinate. This fact was difficult to determine because NADPH is a currency metabolite removed from the networks in the pre-processing stages.

It should be noted that the second fumarate production reaction mentioned in the dGDP Consumption Module is the producer of L-Homoserine which is at the beginning of the reaction chain central to this module. Again, this shows the unity of the metabolism and it gives further evidence to the idea that the variations are not distinct modules but a closely related group of metabolic changes.

References

[1] J. M. Lee, E. P. Gianchandani, and J. a Papin, "Flux balance analysis in the era of metabolomics.," *Briefings in bioinformatics*, vol. 7, no. 2, pp. 140-50, Jun. 2006.

[2] I. Rocha, J. Förster, J. Nielsen, "Design and Application of Genome-Scale Reconstructed Metabolic Models," *Methods in Molecular Biology*, vol. 416, pp. 409-431, 2008.

[3] N. E. Lewis, K. K. Hixson, T. M. Conrad, J. A Lerman, P. Charusanti, A. D. Polpitiya, J. N Adkins, G. Schramm, S.O. Purvine, D. Lopez-Ferrer, K.K. Weitz, R. Eils, R. König, R.D. Smith, B. Ø. Palsson., "Omic data from evolved E. coli are consistent with computed optimal growth from genome-scale models.," *Molecular systems biology*, vol. 6, p. 390, Jul. 2010.

[4] K. R. Patil, I. Rocha, J. Förster, and J. Nielsen, "Evolutionary programming as a platform for in silico metabolic engineering.," *BMC bioinformatics*, vol. 6, p. 308, Jan. 2005.

[5] M. Rocha, P. Maia, R. Mendes, J.P. Pinto, E.C. Ferreira, J. Nielsen, K.R. Patil, I. Rocha., "Natural computation meta-heuristics for the in silico optimization of microbial strains.," *BMC bioinformatics*, vol. 9, Jan. 2008.

[6] I. Rocha, P. Maia, P. Evangelista, P. Vilaça, S. Soares, J.P. Pinto, J. Nielsen, K.R. Patil, E.C. Ferreira, M. Rocha, "OptFlux: an open-source software platform for in silico metabolic engineering.," *BMC systems biology*, vol. 4, p. 45, Jan. 2010.

[7] S. Schuster, D. a Fell, and T. Dandekar, "A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks.," *Nature biotechnology*, vol. 18, no. 3, pp. 326-32, Mar. 2000.

[8] J. Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A. , Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofm, "The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524-531, 2003.

[9] M. Hucka, A. Finney, S. Hoops, and S. Keating, "Systems biology markup language (SBML) Level 2: structures and facilities for model definitions," *Nature*, 2008.

[10] M. E. Smoot, K. Ono, J. Ruscheinski, P.-L. Wang, and T. Ideker, "Cytoscape 2.8: new features for data integration and network visualization.," *Bioinformatics (Oxford, England)*, vol. 27, no. 3, pp. 431-2, Feb. 2011.

[11] H. Funahashi, A.; Matsuoka, Y.; Jouraku, A.; Morohashi, M.; Kikuchi, N.; Kitano, "CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks," *IEEE*, vol. 96, no. 8, pp. 1254-1265, 2008.

[12] A. M. V. Batagelj, "Pajek – Analysis and Visualization of Large Networks," *Springer (series Mathematics and Visualization)*, pp. 7-103.

[13] C. Chaouiya, "Petri net modelling of biological networks.," *Briefings in bioinformatics*, vol. 8, no. 4, pp. 210-9, Jul. 2007.

[14] P. Gerlee, L. Lizana, and K. Sneppen, "Pathway identification by network pruning in the metabolic network of Escherichia coli.," *Bioinformatics (Oxford, England)*, vol. 25, no. 24, pp. 3282-8, Dec. 2009.

[15] J. van Helden, L. Wernisch, D. Gilbert, and S. J. Wodak, "Graph-based analysis of metabolic networks.," *Ernst Schering Research Foundation workshop*, no. 38, pp. 245-74, Jan. 2002.

[16] M. Ashkenazi, G. D. Bader, A. Kuchinsky, M. Moshelion, and D. J. States, "Cytoscape ESP: simple search of complex biological networks.," *Bioinformatics (Oxford, England)*, vol. 24, no. 12, pp. 1465-6, Jun. 2008.

Chapter 5

Software development methodology and implementation issues

In this chapter, the software development strategy followed during this work is presented, the global architecture of the software modules is described, important implementation options are justified and the main algorithms developed in this work are detailed.

5.1 Overall software development methodology

Within the overall software development process, a number of principles were followed, namely:

- Free availability – all tools are made freely available for the users in the respective web sites, together with appropriate documentation for its use;
- Open-source – all source-code written during the development of this work was made available (in the web sites), inviting its use and extensions by other researchers;
- User-friendly – the final applications were developed having in mind its use by users with no/little background in informatics and no programming knowledge;
- Modular – all software has been developed in a plugin-based approach, in a highly modular way, facilitating the addition of specific features by computer scientists;
- Compatible with standards – whenever possible the tools maintain compatibility with existing standars (e.g. SBML, XGMML, etc).

During the work described in this thesis two main pieces of software were developed for biological network analysis purposes: the application InBiNA and the OptFlux plug-in TNA4OptFlux that were described in the previous two chapters. While both tools were created to be used in different circumstances, it was evident since the start that they would inevitably share some core functionalities and that at some time during their development, capabilities of one would have to be migrated to the other which would entail the reuse of code.

In order to avoid the unnecessary repetition of code and the incoherence problems that come with developing in parallel similar pieces of code, a core library was created to serve as a backbone for the software development needed for this work. This allowed to isolate the main interfaces of the software developed and to increase modularity. The organization and specific implementation issues related to this core library are given in section 5.2.

One important feature in the software development process was the separation of the implementation of the main network handling functionalities from the user interface related code. To fully achieve this independence, the model-view-controller (MVC) design pattern was adopted, by resorting to the use of the *AIBench* platform [1] to implement the final applications and to support the user interface design and implementation. This paradigm leads to units of work with high coherence that can easily be combined and reused.

Also, *AIBench* is plug-in based: applications are developed adding components, called plug-ins, each containing a set of *AIBench* objects. This allows reusing and integrating functionality of past and future developments based on *AIBench*. These aspects are further detailed in section 5.3, where the AI Bench platform is described in more detail and implementation details of the InBiNA and TNA4OptFlux applications are put forward.

5.2 Development of a core library for biological networks

The *BiologicalNetsCore* library defined a network representation structure and the basic operations performed over it, thus allowing the sharing of functionalities between InBiNA and TNA4OptFlux. Also, it acted as an intermediary between the graph library used (JUNG) and the developed software, thus concentrating the JUNG related software development in *BiologicalNetsCore*. This section explains the architecture of this library and the main methods implemented.

5.2.1 Network representation

In order to streamline the software development one of the first objectives was to develop a network structure that could be used in the final applications. This called for a structure flexible enough to represent multiple types of biological networks, but that at same time clearly identified each individual biological entity. The key for the desired network structure was found in the handling of metadata, more specifically by forcing all network elements to possess certain kinds of metadata.

In the *BiologicalNetsCore* network structure, vertices represent biological entities and the edges stand for interactions between them. All the vertices and edges have an associated metadata tag called "type" that identifies the kind of biological entity or relationship they represent, respectively. Besides allowing a quick identification of the network elements, the inclusion of the "type" tag also allows the modification of generic network analysis methodologies in a way that takes biological data specific features into account. For instance, a shortest path algorithm can be modified to find only paths that are metabolic chains or motif searches can be limited to only a few selected types of vertices and edges. All edges in a *BiologicalNetsCore* network are necessarily directed. If it is necessary to represent a relationship which is undirected by nature (for example a reversible reaction) parallel edges pointing in opposite directions are used.

As for the metadata, it was also decided that each network element should be able to store optional generic properties in the form of maps, i.e. pairs of strings standing for keys and values that can be used to identify important features of biological entities (e.g. EC numbers for enzymes, chemical formulas for compounds, etc). To facilitate the observation of the metadata in table representations, all network elements of the same type share the same set of metadata fields, even if in some cases they are empty.

Writing methods capable of efficiently analysing or manipulating biological networks requires a way of uniquely identifying each biological entity. While this information could be obtained from associated metadata fields, this would require situation specific solutions, which would increase the necessary work and would prevent the development of generic analysis functionalities. For this reason it was decided that all vertices should have a second mandatory metadata identification field (the "id" tag), which would uniquely identify each vertex of a given type (i.e. repeated ids are allowed in distinct types of vertices).

5.2.2 Architecture of the BiologicalNetsCore library

The *BiologicalNetsCore* library is divided into two main packages:

- The ***core*** package defines the network structure and contains general classes and interfaces for network manipulation and analysis in a way that is designed to be independent of specific implementations;

- The ***jung*** package contains an implementation of the network structure defined in the *core* package based in the JUNG library. Essentially, all the interfaces of the core package have a corresponding implementation in this package.

Ultimately, the *core* package can be considered as containing the core structure of the code in the library, while the *jung* package is a connecter that links *BiologicalNetsCore* with the JUNG 2.0 library. This organization allows further

developments of *BiologicalNetsCore* either by increasing its functionalities expanding the core package or by adding to it the capability to link to other graph libraries by creating new packages similar to JUNG.

Figure 5.1 shows the main packages of the *BiologicalNetsCore* library and its organization, using Unified Modelling Language (UML) schemes. In supplementary material, given in the web site of the thesis (http://darwin.di.uminho.pt/tese-jpp), a more detailed set of UML schemes is provided including also the set of classes developed. The main packages and classes are described in more detailed in the next sub-sections.

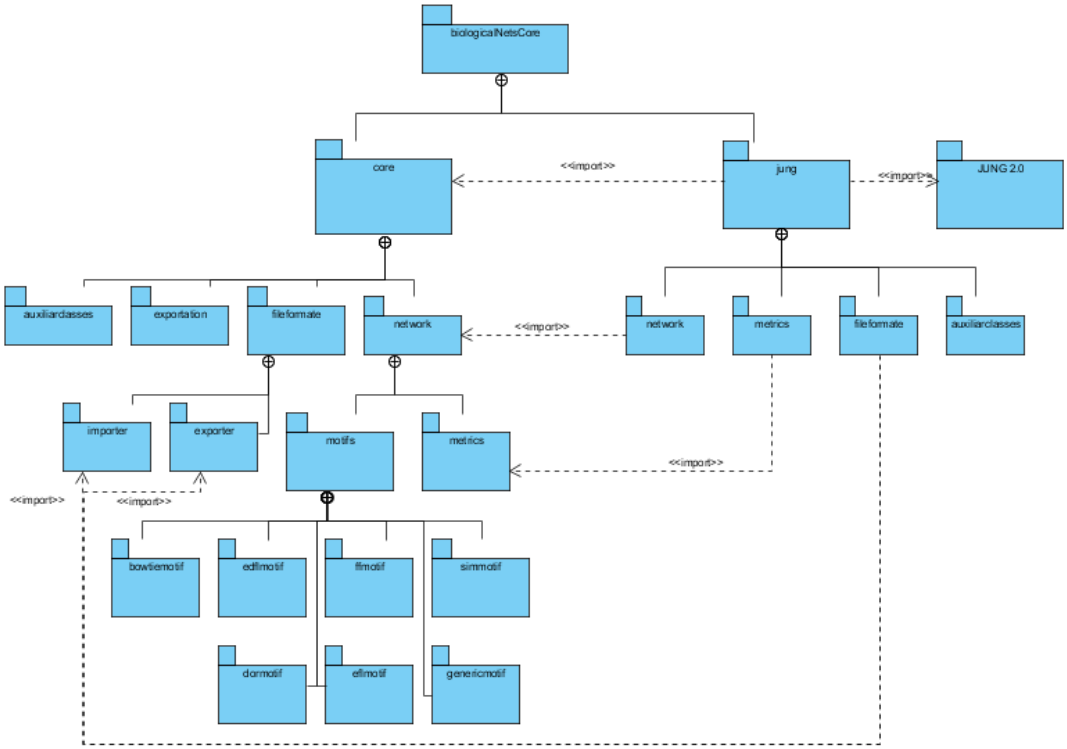


Figure 5.1 - Organization of *BiologicalNetsCore*'s packages

5.2.3 The package *core*

The *BiologicalNetsCore* network structure is defined in this package, together with generic classes that manipulate or analyse networks and can be programmed independently of a specific implementation. The package *core* is

divided into four distinct sub-packages: *network*, *fileformat*, *exportation* and *auxiliarclasses*.

The *network* package is probably the most important of the sub-packages, containing the interfaces defining the network structure and the classes and interfaces for the analysis and manipulation of networks. The network defining interfaces are *INetwork*, *INode* and *IEdge*, representing objects that implement biological networks, vertices and edges respectively. Early during the development, it was decided that the organization of the network would be stored solely in the *INetwork* object instead of being distributed by the vertices and edges objects. Consequently, *INetwork* defines methods mostly related to network access and organization while *INode* and *IEdge* define identification and metadata related methods. This organization is followed by the *JUNG* library and facilitated the connection with *BiologicalNetsCore*.

For network analysis and manipulation, the *network* sub-package contains four classes:

- ***Union*** and ***Intersection*** - these two classes are capable of applying the mathematical operations of union and intersection to pairs of networks and return the results as a new network. These classes assume that vertices sharing the same identification and type values represent the same biological entity.
- ***ActiveNodeLocater*** - this class executes the "active node location" operation used by TNA4OptFlux. It is an operation that takes a network and a list of seed vertices, determining which vertices and edges are reachable through a series of recursive functions.
- ***NetworkComparison*** - a class that allows comparing biological networks using a series of static methods each capable of a specific kind of comparison.

Additionally, the *core.network* package also contains two sub-packages of its own: *metrics* and *motifs*. The first stores classes and interfaces for network analysis, namely:

- ***IRanker*** - this interface is used to define a series of analysis methods classified as ranking algorithms, as described in Section 2.1.3. Since there are many possible ranking algorithms, *IRanker* was deliberately written in a generic way. In fact, its only operation is used to return a real value when given a vertex. Two more classes, related with *IRanker*, were written: *RankerData* and *RankingData*, two data storage objects for the results of one or several implementations of *IRanker,* respectively.
- ***DegreeData*** - a class oriented towards the visualization of the vertex degrees. Given a network, it returns a table with the degree (including indegree and outdegree) values of each of its vertices.
- ***DegreeDistribution*** - another degree analysis centered class that produces histograms of the degree distribution of a network.
- ***ShortestPathMetrics*** - a class that extracts, from a given network, metrics related to its shortest paths. This class can be used to obtain both shortest path data of the network as a whole or only pertaining to individual vertices. The supported network shortest paths metrics are: the diameter, average shortest path length, maximal shortest path length, number of vertices who are the end of shortest paths and number of vertices who are the start of shortest paths. The supported vertex shortest paths metrics are the identification of the longest shortest path, which start and end in the vertex, the number of shortest paths which start and end in the vertex and their average length. To calculate the shortest path metrics, the class *ShortestPathMetrics* implements the interface *IShortestPathCalculator*, also present in the same package.
- ***BFSPathsShortestPathCalculater*** – this class is an implementation of the *IShortestPathCalculator* that uses the breadth first shortest path algorithm. It is a generic class and, consequently, it can be less efficient than an *IShortestPathCalculator* implementations created with an specific network structure in mind.

- ***ClusteringMetrics*** - this class uses an implementation of the *IClusteringCoefficientsCalculator* interface, also present in the *core.network* package, to calculate the average clustering coefficient of both the whole network and of its individual vertices.

The localization of motifs was a major part of this project. Initially, motif location was handled by a set of heterogeneous unrelated classes. However, it became evident that it was more practical to standardize the location operations, which resulted in the creation of the *core.network.motifs* package. This package defines generic processes to locate, store and visualize motifs using three interfaces:

- ***IGetMotifs*** - This interface defines classes which search a network for instances of a specific pattern.
- ***IMotif*** - This interface defines classes for the storage of one pattern located by an implementation of *IGetMotifs*.
- ***IMotifs*** - This interface defines classes that store multiple patterns and several methods for the table based visualization of the stored motifs.

The main advantage of using standard interfaces is that it considerably reduces the code necessary to utilize motif detection at the application level. Besides these three interfaces, this package also contains several sub-packages that have implementations of the interfaces for the location of specific motifs or patterns. Currently, the supported motifs are: bowtie, DOR, FFL and SIM as described in Section 2.4.2. Also, the patterns extended feedback loop and double extended feedback loop that are implemented in InBiNA are implemented through the use of these interfaces.

Additionally, a more complex sub-package is also included called *genericmotif,* which instead of detecting pre-defined motifs can be used to find any user-defined pattern (as implemented in InBiNA). Naturally, using *genericmotif* is computationally heavier than using a motif specific implementation, but this sub-package greatly extends the pattern detection capabilities of the library.

The sub-packages *fileformat* and *exportation* define methods to convert networks from the *BiologicalNetsCore* format to and from ASCII file formats. During this work these packages were used not only for creating methods to store networks but also as a way to convert networks into formats that could be read by other applications. The supported formats were already described in chapter 3.

The final sub-package is *auxiliarclasses*, which was created to contain classes that do not belong in any of the other packages, but are used in several data analysis methods or are used frequently both by InBiNA and TNA4OptFlux. Some of these classes include the *QRTable* and *QRReacTable* table formats, which are the standard output for many *BiologicalNetsCore*'s classes. Most of *auxiliarclasses*' methods, however, are used only internally with the notable exception of BFS, the class which implements the SBBFS algorithm described below.

The package *auxiliarclasses* also contains one sub-package called *filters* that contains generic network filtering classes. Bypass filters are defined in the *auxiliarclasses.filters.bypassfilters* package by three interfaces: *IBypassFilter*, which defines classes that execute the bypass operations, *IBypassTransformer*, which defines individual bypass operations, and *IBypassParameters*, which defines the input of the classes which both serve as inputs for implementations of *IBypassFilter* and execute the operations defined by the implementations of *IBypassTransformer*.

5.2.4 The package *jung*

While the package *core* defined the *BiologicalNetsCore* network structure and interfaces for the main operations, this package contains an implementation of the core package network interfaces based in the JUNG 2.0 library. Since this library contains extensive functionalities for network creation and analysis, for most part the implementations of the *jung* package serve as a layer of code between the Jung library and the applications that force the Jung networks to

follow the previously presented network structure thus assuring compatibility with *BiologicalNetsCore's* functionalities.

To facilitate the identification of the functions of each of its elements, the internal organization of the *jung* package was made deliberately similar to the core, with similarly named sub-packages:

- ***networks*** - probably the most straight forward of the sub-packages, it is here that *JungNetwork*, *JungEdge* and *JungNode*, implementations of *INetwork*, *IEdge* and *INode* respectively, are defined.
- ***metrics*** - this sub-package contains the Jung base implementations of *IRanker*, *IClusteringCoefficientsCalculator* and *IShortestPathCalculater*.
- ***fileformat*** - contains classes for the conversion of networks to and from two file formats, which were frequently during this work: SIF, a format used by tools such as Cytoscape, and MBNF, which was developed during this work.
- ***auxiliarclasses*** - contains classes which were used by both TNA4OptFlux and InBiNA but that do not belong to any other of the sub-packages; it also has a sub-package called *filters.bypassfilters* which contains implementations of core's bypass filtering interfaces specially created for JungNetwork objects. In order to give a wider range of possible bypass filters, two implementations of *IBypassFilter* and *IBypassTransformer* were written defining different algorithms for creating bypass operations and executing bypass operations. These algorithms are used to implement the different variations of bypass filters supported by InBiNA (chapter 3),

While this is the only implementation of this kind that was developed, the *jung* package represents only one possible implementation of the core package. Should in the future a network library more useful than JUNG 2.0 appear, it will be relatively easy to assure its compatibility with InBiNA or TNA4OptFlux by creating a similar package.

5.3 Application development methodology

In this section, the methodology used for the development of the final applications and their user interfaces is described.

5.3.1 Building user interfaces: AI Bench and MVC

Both applications developed in this work are built on top of *AIBench* [1], a software development framework that was born as a collaborative project between the host group and researchers from the University of Vigo in Spain. Building applications over *AIBench* brings important advantages to both the developers and the users, given its design principles and architecture. The applications incorporate three types of well defined objects, following the MVC (*model-view-controller*) design pattern.

The basic idea of the MVC paradigm is to split the application into three distinct mostly independent components, each of which can be replaced without affecting the others [2]:

1. *Model* - The combination of the data and the logic necessary to access and modify the data.
2. *View* - The methods used to present the data to the users.
3. *Controller* - The interfaces that receive input and pass the commands and information received to the model.

Normally, the initial development of a MVC project is time consuming since the three components have be developed in parallel and the correct communication between them as to be assured. However, *AIBench* acts as connector between components removing the burden of programming the communication, which not only greatly reduces the initial effort, but helps during the whole software development cycle.

In an *AIBench* based application, the user interaction is based on three main concepts:

- *Datatypes*: represent the distinct types of objects holding the relevant data to the applications (such as networks or analysis results). Each type can have multiple instances (objects) within the application. Each datatype can be a simple datatypes, a list containing a set of objects of another datatype or a complex datatypes that can contain any combination of other datatypes.

- *Views*: represent different ways to visualize the contents of data objects. Each datatype can have one or more methods to visualize its instances. If more than one view exists they are typically shown in different tabs.

- *Operations*: represent the software functionalities, i.e. available processing actions. Operations can be defined as transformations of a set of input datatypes into a set of output datatypes. When an operation is called, its interface is launched and the input data objects are selected. After being triggered, an operation typically changes or creates an instance of the output datatype.

Based on these concepts, a user-friendly Graphical User Interface (GUI) is developed for each specific application. Here, the strategy will be exemplified with the application InBiNA, presented in chapter 3. The layout of the main components of the application can be observed in the screenshots presented in Figure 5.2.

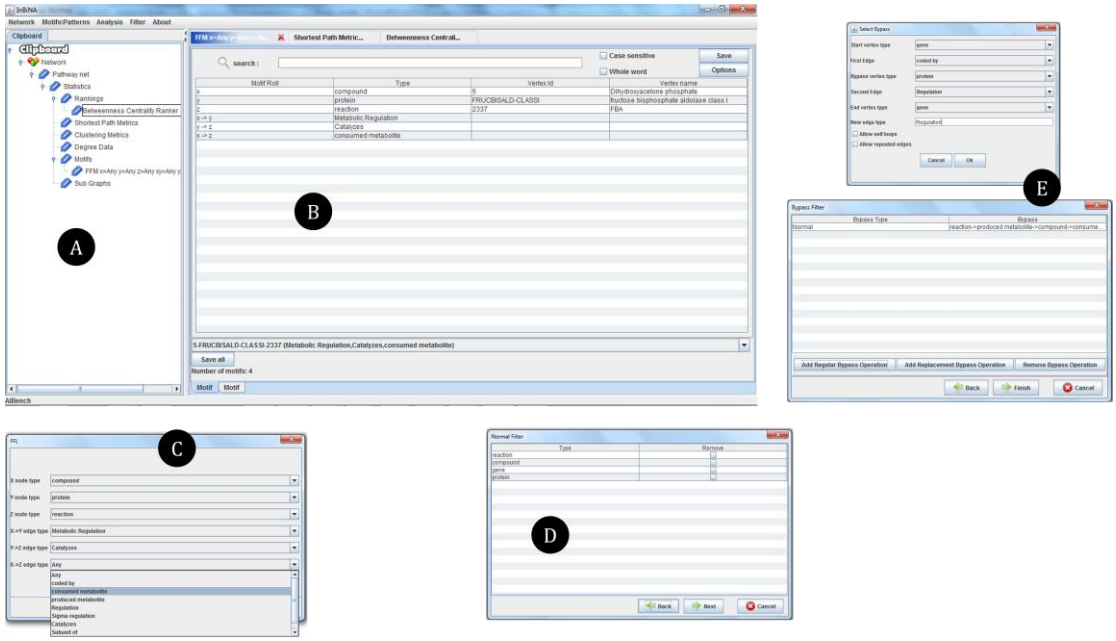


Figure 5.2 - Screenshots of InBiNA: a) Clipboard; b) One of InBiNA's table based views; c) the user interface for the operation allowing to detect instances of the FFL motif; d) wizard for simple filters; e) wizard for bypass filters.

The clipboard on the left (Figure 5.2a) keeps all data objects created within the application, in a logical hierarchy, grouped by their *datatypes*. The root of this tree is the *Network datatype* that keeps all objects related to a given network and the analysis performed with it. The components of a *Network datatype* are graphically shown in the form of explicit hierarchical containers, namely:

- *Statistics*: This complex *datatype* is directly below the *Network datatype*, grouping all *datatype* objects with information obtained from the analysis of a network.
- *Rankings*: A list *datatype* which contains all the results of ranking algorithms for a given network.
- *Shortest Path Metrics*: A *datatype* containing all the global shortest path metrics of a network. It also serves as an interface to the InBiNA functions which calculate vertex shortest path metrics and the shortest paths between pairs of vertices.
- *Clustering Coefficient*: A *datatype* containing a network's clustering coefficiente value and distribution.

- *Degree Data*: This *datatype* contains the degree values (including the in and outdegree) of all the vertices of a network, as well as the table and charts with the degree distribution.
- *Motifs*: A list *datatype* which stores all the patterns found in the associated network.

When an object in the clipboard is double-clicked, the views corresponding to its *datatype* will be launched on the right side of the working area (if more than one view is available, those are accessible in different tabs). An example of a view is shown in Figures 5.2b.

All the available operations are easily accessible, either through the menu in the top or by right clicking the item in the clipboard area, an action that displays all operations that work over that type of argument. Snapshots of some operations for finding instances of the FFL motif, applying a normal filter and applying a bypass filter are shown in Figure 5.2c, d e, respectively.

All operations are, at the maximum possible level, default-oriented, thus hiding behind scenes their complexity (e.g. definition of non-obvious parameters). Nevertheless, they allow more advanced users to fine-tune the parameters available to a given operation.

5.3.2 Main datatypes and operations for the InBiNA application

Table 5.1 shows the main datatypes developed in InBiNA and some of the operations that allow creating these datatypes. This list is by no means exhaustive being complemented by the information in InBiNA's web site.

Table 5.1: Main datatypes developed within InBiNA and the operations that allow to create them.

Datatype	Operation(s) which create it
Network	Open New Network From MBNF New Network From SIF New Network From SBML
Statistics	-
Rankings	-
Betweenness Centrality Ranker	Betweenness Centrality Ranker
HITS Ranker	Hubs-and-authorities Ranker
Closeness Centrality Ranker	Closeness Centrality Ranker
Shortest Path Metrics	Get Shortest Path Metrics
Clustering Metrics	Clustering Coefficient
Degree Data	Get Degree Data
Motifs	-
Motif	FFL Reversed FFL SIM EFL DEFL Bowtie DOR Costume pattern from file
Sub Graphs	Find sub-graphs

5.2.3 Implementation of the TNA4OptFlux plug-in

The TNA4OptFlux is, as previously mentioned, a plug-in for the OptFlux platform, previously described in section 4.1.3. The whole OptFlux platform, and related plug-ins, has been developed over *AIBench* according to the principles described in section 5.3.1.

In OptFlux, the main datatype is the *Project datatype* that keeps all objects related to a given metabolic model and the analysis performed with it. The principal components of a project are:

- The metabolic model, including the sets of metabolites (internal and external), the set of reactions with their flux bounds and stoichiometry, the steady state equations and the encoding genes and the gene-reaction association rules;
- Sets of simulation and optimization results;
- Other optional objects including: a model graph for visualization, environmental conditions, lists of essential genes/ reactions, among others.

The main operations available in the core OptFlux software are related with model creation and importation/ exportation, model simplification, phenotype simulation (see section 4.1.1) and strain optimization (see section 4.1.2).

TNA4OptFlux adds a number of datatypes to the core OptFlux, namely:

- The *Network analyzer datatype,* which is associated with a *Project datatype* and is used to store all the other *datatype* related with TNA4OptFlux.
- The *network datatype,* which contains is used to store individual networks and any *datatype* with data related to their analysis.
- Several other *datatypes* related with network analysis, mostly similar to the *datatypes* of InBiNA presented in the previous section.

Once again, this information is not exhaustive, being complemented by the information available in the documentation from the OptFlux's web site, regarding this specific plug-in.

5.3 Selected algorithms

In this work, a number of features were implemented based on pre-existing software and therefore it would be fastidious to enumerate the algorithms of all analysis and metrics. However, some novel algorithms have been developed and some of those are worth a more detailed analysis, being presented below.

5.3.1 Shortest path calculation

In section 2.1.2. the classical algorithms for shortest path calculation were described, namely the breadth-first search (BFS) and the Dijkstra algorithm. When the graph has unweighted edges the BFS typically is the best alternative. However, a problem arises when representing reversible reactions in metabolic networks.

In the *BiologicalNetsCore* network structure, reversible reactions are typically represented using parallel directed edges pointing in opposite directions, which led to a problem in finding paths in networks with reversible reactions. This network organization opens the possibility of including in a given path a reaction in which a pair of metabolites participate as substrates or products as a step between them. Naturally, these paths are meaningless from a biological point of view.

To address this issue a novel algorithm was developed, called set-based breadth-first search (SBBFS). This is a variation of the original BFS, by adding the notion of edge sets, i.e. edges have an associated attribute called the set that is kept in the edge metadata. The process of path finding must obey the condition that if a path enters a vertex through an edge of a set, it must exit that node through an edge belonging to the same set or an edge with no set. Figure 5.3 illustrates the approach.

With the SBBFS algorithm, if a metabolic network is correctly built, it is possible to identify all valid shortest paths from a selected vertex to all the vertices it is

connected to. It should be noted that since SBBFS must store the paths and edge information, it is more memory expensive than the normal BFS. The detailed algorithm is given as Algorithm 5.1 below.

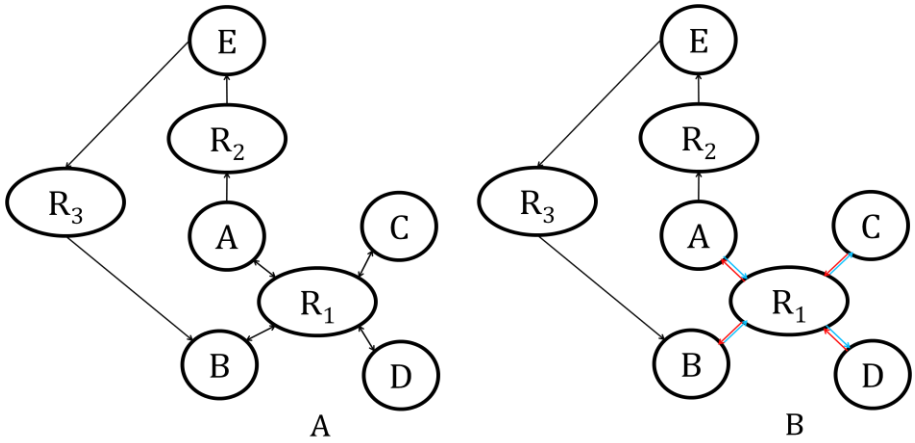


Figure 5.3 - Illustration of the SBBFS approach for shortest path calculation in metabolic networks (a) Using BFS, the shortest path between A and B is {A, R1, B}, a non valid path, since A and B are in the same side of reaction R1, so they will always be consumed or produced together; (b) Using SBBFS, the shortest path between A and B will be {A, R2, E, R3, B}, a longer but biologically meaningful path.

Algorithm 5.1 – Set Based Breadth first search

```
SBBFS(G, s)

Create a queue Q, initially with s
Get list lv of all vertices in G
Create integer array distances[lv.length]
Create vertex matrix connectedBy[lv.length][]
Create string matrix connectedBySet[lv.length][]
z=0
For all vertices v2 in lv
       if v2 is s then distances[z] = 0
       else distances[z] = -1
       z = z + 1
change = true
dis = 0
current = {v}
While change is true
       Dis = dis + 1
       change = false
       new_current = {}
       for all vertices v2 in current
              Get index of v2 i2 in lv
              enteredBySet = connectedBySet[i2]
              For all edges ed that start in v2
```

```
                    if ed has not associated set or if ed set in
                    enteredBySet or if enteredBySet.length = 0
                           valid = true
                    if ed set in enteredBySet or if ed has no set and
                    enteredBySet.length=0
                           KeyComplet = true
                    else
                           KeyComplet = false
                    if valid
                           get vertex v3 to which ed points
                           Get index of v3 i3
                           if distances[i3] is -1
                                  distances[i3]=dis
                                  connectedBy [i3] add v2
                                  if KeyComplet is false and ed has set
                                         connectedBySet[i3]  = ed's set
                                  new_current add v3
                                  change = true
                           else if distances[i3] is dis
                                  connectedBy [i3] add v2
                                  if KeyComplet is false and ed has set
                                         connectedBySet[i3]  ed's set
      current = new_current
```

5.3.2 Pattern finding

A number of algorithms have been implemented to address the location of all instances of a defined pattern within networks. These include a general purpose algorithm for the detection of user defined patterns, as well as specific patterns found of general interest. The detailed algorithms for the specific motifs described in section 2.4.2 are given in supplementary material included in the web site http://darwin.di.uminho.pt/phdthesis-jpp.

The general-purpose algorithm will be examined here in more detail. Writing an algorithm capable of detecting an user defined motif was a complex task, which involved developing a computer data format that could be used for storing any possible network patterns and an algorithm for searching networks for those patterns.

The first step was to develop the data format for storing patterns that permitted them to be compared against possible occurrences in a network. Naturally this format would have to be capable of representing all possible patterns but had to be as simple as possible. This data structure is composed by an array **V** which identifies the number and type of the vertices in the pattern and by two double

arrays, **L** and **T**, which define the edges. Each of the arrays corresponds to one of the vertices of **V**, in the case of **L** these identify the corresponding vertex neighbors, while **T** identifies the types of the edges which start in the vertex (Figure 5.4).

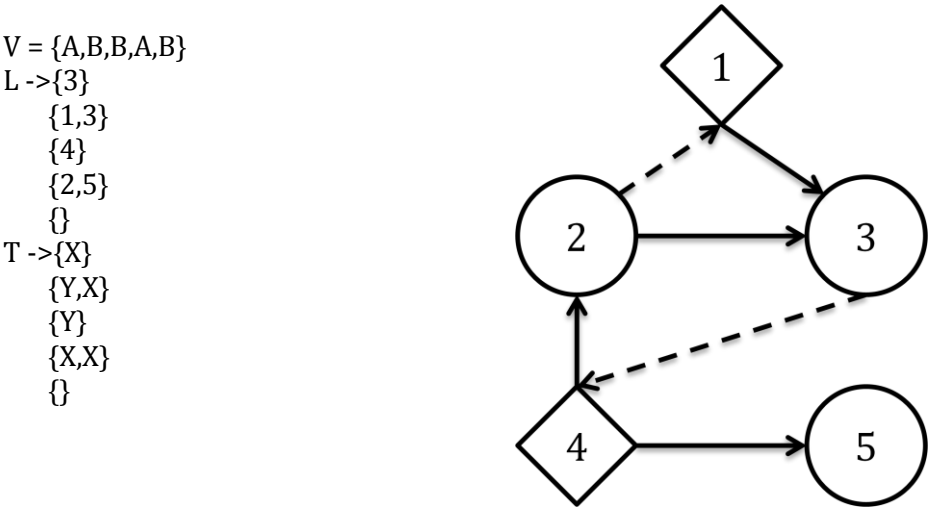


Figure 5.4 -Example of the data structure created for storing network patterns: two types of vertices, A (the lozenge) and B (the circles), and two types of edges, X (the normal lines) and Y (the dotted lines), were used.

The pattern location algorithm takes all vertices in the network of the type corresponding to the first element of **V** and then searches for pattern matches from them using an BFS based method. The main difference between an normal BFS method and this algorithm is that when searching for pattern occurrences it looks both at vertices connected by edges that start and end in the current vertex. Thanks to this modification, this algorithm can search for pattern occurrences by starting in any point of the pattern where in an normal BFS method the pattern would have to be arranged as a tree and the first vertex would necessarily have to be the root.

Another crucial step in this algorithm is that when descending for the next tree level the already found vertices are checked to see if they have all the required

interconnections, a process which involves looking at the already found vertices. This step is necessary because in a pattern, unlike trees, a vertex can be connected to any other vertices.

The full algorithm is given in supplementary material included in the web site http://darwin.di.uminho.pt/phdthesis-jpp.

5.3.3 Bypass filters

A final set of novel algorithms developed in this work is related with bypass filters, as described in section 3.1.3.. Bypass filters were created to have a way of changing the format of biological networks, without having to manually rewrite all the edges. For example, with a bypass filter a metabolic network in which the representation of both the reactions and the metabolites is made through vertices could be converted in one where the vertices represent only reactions without losing the connection between the reactions. Regular and sequential bypass filters were implemented and the detailed algorithms are also given in supplementary material included in the web site http://darwin.di.uminho.pt/phdthesis-jpp.

5.4 Other implementation details

5.4.1 Used libraries

In this work, apart from the mentioned JUNG and AI Bench, a number of third party Java libraries were used from which the following can be highlighted:

- JFreeChart [3] is a free Java library for drawing plots and charts, which is used to draw degree distribution charts.

- Prefuse [4] is a toolkit that provides multiple methods for data modeling, visualization, and interaction. In TNA4OptFlux it is used to draw the

partial graph representation. Prefuse was used instead of JUNG's native graphical capabilities because it was determined to possess superior graph drawing methods.

References

[1] F. F.-Riverola, D. Glez-Peña, M. Reboiro-Jato, P. Maia, M. Rocha, F. Díaz, "AIBench: a rapid application development framework for translational research in biomedicine.," *Computer methods and programs in biomedicine*, vol. 98, no. 2, pp. 191-203, 2001.

[2] S. Stanchfield, "Applying MVC in VisualAge for Java." [Online]. Available: http://javadude.com/articles/vaddmvc1/mvc1.htm.

[3] "JFreeChart's website." [Online]. Available: http://www.jfree.org/jfreechart/.

[4] "Prefuse's webpage." [Online]. Available: http://prefuse.org/.

Chapter 6

Conclusions and future work

6.1 Summary and main contributions

This work started even before the PhD described in this thesis as a simple project to design an application to help researchers of the host group to create biological networks from data extracted from different sources. While the application was eventually created, the final result was a relatively poor tool. Indeed, the study of large-scale biological networks proved to be more difficult than originally anticipated.

When this PhD started, the ideas behind that original application were revisited and expanded and the work began. The development of an application for the analysis of large-scale biological networks started, this time with a clearer view of the challenge ahead. The first piece of software developed was the Integrated Biological Network Analyser (InBiNA) (chapter 3), a stand alone application based in the AIBench framework [1]. InBiNA possesses a series of functionalities, which allow users to analyse and manipulate multiple types of biological networks. Besides topological analysis, InBiNA is also capable of detecting network patterns, a capability included so that the application would also be useful for motif/ pattern finding [2][3].

The main contributions of InBiNA lie in its capabilities for easily creating and analysing large-scale integrated cellular networks, with distinct types of biological entities (e.g. genes, proteins, compounds) and interactions. This allows to represent the different cellular sub-systems (metabolic, regulatory, signalling) and the interconnections among those. The ability to perform topological analyses over these integrated networks and to search for different types of

patterns revealed itself as a valuable asset for researchers in the Systems Biology arena.

During the development of InBiNA it became evident that OptFlux [4], an application for Metabolic Engineering created by the host group, could be expanded by the addition of network analysis capabilities. A plug-in called Topological Network Analysis for OptFlux (TNA4OptFlux) (chapter 4) was then developed. Initially, TNA4OptFlux was essentially a "light" version of InBiNA for OptFlux. TNA4OptFlux is capable of obtaining networks directly from the metabolic models used by OptFlux and analyse them; additionally, it also possesses a functionally, called simulation filtering, which allows it to filter networks based in simulation results.

However, the development of TNA4OptFlux diverged from InBiNA's when the focus of this work shifted to network comparison. This concept had been initially introduced in both pieces of software as a way for the user to compare the properties of networks diverged from a common origin through filtering processes. The combination of network comparison with TNA4OptFlux's simulation filtering led to the development of the variation network methodology, which offered a new way of analysing OptFlux's simulation results. After their potential was identified, variation networks quickly became the major focus of work in TNA4OptFlux. The final result was a plug-in which combines model-based phenotype simulation methods and network-based topological analysis methods.

The main contributions of the TNA4OptFlux plug-in are related to its usefulness in helping Metabolic Engineering researchers to interpret results from phenotype simulation/ strain optimization algorithms. The methods for network comparison, when applied to the analysis of mutant strains versus reference wild-type strains helped to highlight important features in the understanding of the strategies followed by mutants that are able to accomplish an industrial goal (such as the overproduction of a given compound). When combined with other

features in OptFlux, these tools have shown to be a valuable asset for researchers in Biotechnology.

The capabilities of both InBiNA and TNA4OptFlux were verified through case studies that show their usefulness. These case studies make use of the large body of knowledge in *Escherichia coli* available in the host group. This fostered the collaboration of the author with other members of the group, helping to improve the tools and inspiring new functionalities. No software tool can evolve without users. Also, the results obtained in those collaborations have helped to improve the modelling efforts on specific tasks within the group.

At the core of InBiNA and TNA4OptFlux is *BiologicalNetCore* (chapter 5), a library created to implement all the networks creation and analysis functions developed during this PhD. The reasons for separating the functions from the main application were twofold. First, the use of an independent library facilitated the transfer of functionalities between InBiNA and TNA4OptFlux. Second, since, like all software developed during this PhD, *BiologicalNetCore* is open source and other parties interested in large-scale network analysis can easily access the code contained in it without having to look at the source code of either InBiNA or TNA4OptFlux.

6.2 Limitations

During the software development process, the focus was more in the network analysis rather than manipulation. In fact, even the network filtering capabilities were initially added to facilitate the network analysis through the removal of currency metabolites [5]. As a consequence of this focus in analysis, neither InBiNA nor TNA4OptFlux possess options for modifying the networks through the addition of new elements. This is usually not a problem for TNA4OptFlux, but it can be troublesome when working with InBiNA.

Another limitation of both applications is the way they store and expose metadata. In the model used, the metadata for both vertices and edges is

organized by element type. Consequently, all network elements of the same type must have the same metadata fields associated, even if they lack the respective data. This facilitates the exposition of the data in a table format. When dealing with networks in which the metadata distribution is not homogenous, however, this organization can result in tables with too many fields, most of which are empty. Adding new metadata to networks can also be problematic, because of the number of network elements that have to be modified.

Finally, when using filters to remove currency metabolites from metabolic networks, there is a problematic situation that can occur. In most reactions the removal of currency metabolites is essential for the application of many network analysis methods. There are, however, a few reactions in which the currency metabolites are essential to understand their place in the networks. These are reactions in which the currency metabolites do not act as currency metabolites and are in fact essential elements of the reactions. Ideally, the filters should be capable of identifying these exceptional situations and maintain the currency metabolites connections to the correct reactions. Unfortunately, that is not possible and so, in this case, users must rely on their knowledge of the metabolic network.

6.3 Future work

As mentioned in the former section, InBiNA and TNA4OptFlux could be improved by adding new network editing capabilities and modifying the metadata storage model. These alterations would require modifications in not only InBiNA and TNA4OptFlux code, but also in the *BiologicalNetCore* library.

This library could be expanded with the addition of new network topology metrics. This would benefit both InBiNA and TNA4OptFlux since it would be relatively simple to add the new metrics to either application.

Besides the addition of new analysis metrics, improvements to InBiNA would probably involve increasing the possible data sources it can use to create

networks. This would mean adding new supported file formats and possibly adding to the application the capability of connecting with existing data repositories. Another possible upgrade for InBiNA is the addition of some graphical capabilities. Naturally, this would not mean the visualization of complete large-scale networks, but something like TNA4OptFlux, which shows only small selected parts of the network.

Any refinements for TNA4OptFlux will probably be related with the variation network functionalities, with a special focus on defining new variation identification methods. However, the development of new methods to combine model-based phenotype simulation methods and network-based topological analysis methods is another possible avenue of improvement.

References

[1] F. F.-Riverola, D. Glez-Peña, M. Reboiro-Jato, P. Maia, M. Rocha, F. Díaz, "AIBench: a rapid application development framework for translational research in biomedicine.," *Computer methods and programs in biomedicine*, vol. 98, no. 2, pp. 191-203, 2001.

[2] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks.," *Science (New York, N.Y.)*, vol. 298, no. 5594, pp. 824-7, Oct. 2002.

[3] A.-L. Barabási and Z. N. Oltvai, "Network biology: understanding the cell's functional organization.," *Nature reviews. Genetics*, vol. 5, no. 2, pp. 101-13, Feb. 2004.

[4] I. Rocha et al., "OptFlux: an open-source software platform for in silico metabolic engineering.," *BMC systems biology*, vol. 4, p. 45, Jan. 2010.

[5] P. Gerlee, L. Lizana, and K. Sneppen, "Pathway identification by network pruning in the metabolic network of Escherichia coli.," *Bioinformatics (Oxford, England)*, vol. 25, no. 24, pp. 3282-8, Dec. 2009.