Development of a computational platform for the visualization of metabolic models

Alberto Miguel Silva de Noronha
bertonoronha@gmail.com

Orientador:
Miguel Francisco de Almeida Pereira da Rocha

Julho, 2013

Acknowledgements

Foremost, I would like to express my gratitude to my advisor Prof. Miguel Rocha for the continuous support during these two years. His guidance helped me in all the time of research and writing of this thesis.

The work is partially funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT (Portuguese Foundation for Science and Technology) within projects ref. COMPETE FCOMP-01-0124-FEDER-015079 and PEst-OE/EEI/UI0752/2011.

To everyone at Bisbii, especially my lab mates, for transforming work into a pleasant experience and for all the fun we have had in the last years.

To everyone at Silico Life, especially Paulo Vilaça, for always having an open door to my doubts and for all his help and ideas that largely contributed to this work.

Finally, to my family, my brother and parents, and to Dirce. For everything they have done for me in my life, their caring and support, this work is as much as theirs as mine.

Abstract

The recent sequencing techniques and *omics* approaches are generating huge amounts of data that can provide ways to extract meaningful knowledge, by resorting to appropriate computational tools. One important technique resorts to the use of genome scale model reconstructions. These models are widely used in Metabolic Engineering, attempting to optimize an organism's functions, genetically modifying it to produce compounds of industrial interest.

Another area that became widely important within the fields of Systems Biology and Bioinformatics was network analysis and visualization. Networks can provide a way to better understand the relationships between biological entities, by allowing their visual representation. However, biological networks usually comprise a large number of entities and interactions, that cannot be easily interpreted by the human eye.

Integrating visualization and analysis is, therefore, a goal of high interest in several scientific areas, and this has been tackled by several visualization tools available. However, regarding the integration of metabolic engineering techniques with metabolic network visualization, there are still few examples of success. Usually, it is necessary to use more than one tool and the agility of the methods is limited.

In this work, a metabolic network visualization framework is presented, with the goal of being a tool that will help researchers in metabolic engineering projects. This framework is divided in two layers: the first deals with the importation and exportation of networks in different formats, while the other layer provides all the visualization and edition features.

A metabolic layout is based on the reactions contained in the metabolic model, and it can represent just a part of the metabolism of an organism. To have the possibility to use the same layout in different models, a strategy was defined to map the entities of the visualization with the entities of the model.

The layouts are displayed in a bipartite graph, with different node types and colors. It is possible to visualize additional information of the network by clicking the nodes. Some of the features include dragging, zooming and highlighting. On top of all this, it is also possible to apply filters and overlap information over these networks. The filters can change what is visible in the network, while the overlaps allow defining new labels, colors and shapes to the nodes, and new colors and thickness to the edges. Finally, the framework was also integrated within OptFlux, an open-source software to support metabolic engineering available at `www.optflux.org`, to provide a connection between visualization and metabolic simulation methods.

Resumo

As recentes técnicas de sequenciação e as abordagens "ómicas" estão a gerar enormes quantidades de dados que, através do uso de ferramentas computacionais adequadas, podem fornecer formas de extracção de conhecimento biológico significativo. Uma importante metodologia recorre à reconstrução de modelos metabólicos à escala genómica. Estes modelos são muito usados na Engenharia Metabólica, tentado-se optimizar o funcionamento do organismo, modificando-o geneticamente, de forma a maximizar a produção de compostos de interesse industrial.

Outra área de estudo que tem ganho bastante importância nos campos da Biologia de sistemas e Bioinformática é a análise e visualização de redes. As redes podem oferecer formas de melhor compreender as relações existentes entre entidades biológicas, fornecendo uma representação visual destes relacionamentos. No entanto, as redes biológicas, usualmente, são compostas por um elevado numero de entidades e relacionamentos, o que pode tornar difícil a sua interpretação a "olho nu".

A integração de visualização e análise sempre foi um objectivo de interesse em todas as áreas científicas, e respostas a este problema têm surgido sob a forma de diferentes ferramentas. No entanto, no que se refere à integração de técnicas de engenharia metabólica com visualização de redes metabólicas, existem ainda poucos exemplos com sucesso. Usualmente, é necessário o uso de diversas ferramentas e as funcionalidades e flexibilidade é ainda limitada.

Neste trabalho é apresentada uma plataforma para a visualização de redes metabólicas, com o objectivo de ser uma ferramenta que assista investigadores em projectos de engenharia metabólica. Esta plataforma está dividida em duas camadas: a primeira lida com a importação e exportação de redes em diferentes formatos, enquanto a outra camada oferece todas as funcionalidades de visualização e edição.

Um layout metabólico é baseado nas reacções contidas num modelo metabólico, e pode representar apenas uma parte do metabolismo do organismo. De forma a ser possível utilizar o mesmo layout em modelos diferentes, foi definida uma estratégia para mapear as entidades da visualização com as entidades do modelo.

Os layouts são representados sob a forma de um grafo bi-partido, com diferentes tipos de nodos e cores. É possível visualizar informação adicional sobre a rede clicando nos nodos. Algumas das funcionalidades incluem arrastar, focar e realçar partes da rede. Para além de tudo isto, é possível aplicar filtros e sobrepor informação sobre a rede. Os filtros permitem definir o que é visível na rede, enquanto a sobreposição permite definir novas etiquetas, formas e cores dos nodos e cores e espessura das conecções. Finalmente, a plataforma foi integrada no OptFlux, uma ferramenta de código aberto para engenharia metabólica que está disponível em `www.optflux.org`, de forma a estabelecer uma conexão entre a visualização de redes metabólicas e métodos de simulação do metabolismo.

Contents

List of Figures

List of Tables

Acronyms

AF Activity Flow

BFS Breadth-first search

CD-SBML Cell-Designer SBML

CLP Coin-or Linear Programming

CPLEX ILOG CPLEX Optimization Studio

EM Elementary Flux Mode

ER Entity Relationship

FBA Flux Balance Analysis

FDL Force Directed Layout

GLPK GNU Linear Programming Kit

GML Graph Markup Language

GRN Gene regulatory network

GSSMs Genome Scale Metabolic Models

GUI Graphical User Interface

KGML KEGG Markup Language

LP Linear Programming

ME Metabolic Engineering

MILP Mixed Integer Linear Programming

MOMA Minimization of Metabolic Adjustment

MVC	Model-view-controller
PD	Process Description
QP	Quadratic Programming
ROOM	Regulatory On-Off Minimization
SB	Systems Biology
SBGN	Systems Biology Graphical Notation
SBML	Systems Biology Markup Language
SPEA2	Strength Pareto Evolutionary Algorithm 2
XGMML	eXtensible Graph Markup and Modeling Language

Chapter 1

Introduction

1.1 Motivation

The surge of new experimental high-throughput techniques in the biomedical fields, including next generation sequencing and omics data, brought important advances and challenges. One of the fields that gained importance due to the generalization of these techniques was Metabolic Engineering, a field that involves designing genetic alterations in microorganisms to optimize the production of compounds of interest. The availability of genome scale data made possible the reconstruction of large scale metabolic models, integrating the knowledge obtained from classical biological techniques, with genome scale data.

Metabolic Engineering uses genome scale metabolic models (GSMMs) in an attempt to optimize an organism's functions, by means of predictions. In the last years, we have been experiencing a large increase in the sequencing of industrially viable micro-organisms that foster the interest on ME and GSMMs.

On the other hand, biological networks became more popular with the appearance of the recent sequencing techniques. They comprise a large number of entities and their interactions that cannot be easily interpreted by the human eye. One can have large networks simply by representing reactions and the metabolites they produce/consume, but the integration of information regarding protein coding, regulatory events, transcription factors or co-factors and the heterogeneity of such data takes the problem to even higher levels of complexity.

A lot of visualization tools arose to face this issue [2]. Most of them resort to the more intuitive way of representing such interactions, by mapping them as 2D graphs, where biological entities are represented as nodes and their interactions as edges. Several tools allow the representation of massive networks and also integrate diverse network analysis methods. The capability of allowing users to specify the representation of the nodes and edges also poses as an attractive property, as well as the features related to the integration with experimental data. The automatic layout capabilities are also very desirable assets, since most models in their raw forms do not comprise layout information.

In the majority of the visualization tools, biological entities/ interactions are represented as shapes/ lines, with different colors/formats standing for their classes. Although this seems a reasonable solution, the inherent complexity of the integrated information, the range of possible different interactions and the growing trend to more accurately represent them motivated the development of standard graphical notations. The most successful was the Systems Biology Graphical Notation (SBGN) [3], where networks are modelled in a state-transition way. One other successful standard format is the Systems Biology Markup Language (SBML), that aims at storing and exchanging biological models. Combined with the development of the SMBL Layout package, this makes up a very promising effort.

While in the pre-genomic era, the analysis and visualization of data patterns were approached as independent computational problems, presently it is desirable that these two levels are very well integrated in order to complement each other [4]. More particularly, this work aims at integrating two levels of analysis: network visualization and ME. Indeed, this work focuses on the development of a visualization framework for ME purposes. On the other hand, there will also be a focus on integrating the visualization framework within a ME platform, allowing the visualization of the models with phenotype simulation, overlapping results of those simulations in the metabolic network, providing a useful tool to assist researchers on ME projects.

1.2 Goals

The aim of this work will be to develop a computational platform for the visualization of metabolic models, integrated within the OptFlux workbench for ME, developed by the host group of this work.

This will encompass the following scientific/technological objectives:

- Review the main tools and standards within the field of metabolic model visualization;
- Implement a generic library for the visualization of metabolic models, well integrated within the OptFlux Metabolic Engineering workbench;
- Over the previous library, implement tools to allow importing/ exporting models and layouts in standard formats, including SBML, SBGN-ML, KEGG Markup Language (KGML) and eXtensible Graph Markup and Modeling Language (XGMML);
- Design and implement tools to allow the integration of the OptFlux visualization library with relevant tools in the field, such as Cytoscape and CellDesigner;
- Integrate the visualization of the models with phenotype simulation tools in OptFlux, overlapping results of simulations with models and exporting these results in standard formats;
- Develop methods to automatically generate layouts for specific pathways from a given metabolic model, and provide ability to edit those layouts;

- Apply the selected tools in selected case studies.

1.3 Structure of the document

This document is organized in the following way:

Chapter 2

Systems Biology and Metabolic Engineering

A brief introduction to the field of SB and ME. Some explanation on metabolic models and simulation methods are given. A survey of relevant ME tools is made.

Chapter 3

Biological Networks

Some basic definitions of networks and graphs are provided as well as some biological networks properties. Some of the main challenges of visualization of biological networks are discussed.

The chapter ends with a survey of some of the available network visualization tools.

Chapter 4

Results and Development

In this chapter the results of the work are presented. First a global view of the architecture of the visualization framework is made. Secondly, the features of the framework are described, as well as the features of the OptFlux's plugin developed. Finally, more technical details of the development of the framework and the plugin are also presented.

Chapter 5

Case Study

This chapter presents case studies of application of the developed framework.

Chapter 6

Conclusions and Future Work

In this chapter, some conclusions about the work are drawn. Some limitations of the framework are and future work possibilities are discussed.

Chapter 2

Systems Biology and Metabolic Engineering

2.1 Systems Biology

The increase in the amount of experimental data generated by high throughput sequencing and *omics* approaches has brought Bioinformatics to the spotlight in the field of the life sciences.

The need to intelligently store the huge amounts of data generated and provide ways to access knowledge provided by that data is perhaps the greatest challenge of bioinformatics today. In order to make sense of all these data, it is necessary to develop computational tools capable of analysing sheer volumes of data and extract meaningful knowledge interpretable by biologists. One of the major uses of such data is to provide knowledge enabling enhanced understanding of biological processes, and the complexity of their interactions. Due to this fact, systems biology arose recently as an important field in the biological sciences.

Systems biology (SB) aims at the system-level understanding of biological systems. While molecular biology currently focuses mainly on the identification of genes and functions, or the parts of the system, SB focuses on understanding how those parts interact, and is more concerned with behaviour. The identification of the components of a system provides limited knowledge, so it is essential that the study of these systems can also provide comprehension of what happens when certain stimuli or disruptions occur.

Systems biology combines efforts from several research areas, such as molecular biology, high-precision measurement, computer science and other engineering and biological fields. Genomics and molecular biology, computational analysis and simulations, analysis of dynamics and measurements are key areas in SB research [5].

The design of these systems should be performed to meet specific functional properties, and to achieve the complete understanding of a biological system. Indeed, the following properties should be met [4]:

- i **System Structure:** focuses on the identification of the interactions of the different compounds of the organism, such as regulation, interaction of proteins and metabolic pathways, as well as the physical structures of the system;
- ii **System Behaviour Analysis:** understanding the behaviour by analysing how the system reacts to certain stimuli or perturbations;
- iii **System Control:** determine methods that control the biological systems, allowing to obtain the knowledge on how to transform, for instance, a malfunctioning cell into a healthy one;
- iv **System Design:** in the best case scenario, develop methodologies that allow the design of biological systems with beneficial goals, such as, providing potential cures for diseases.

2.2 Metabolic Engineering and Metabolic Models

The reconstruction of genome scale models of cells is one of the major challenges of SB. The purpose of this effort is to integrate the knowledge obtained with the classic biological research methods, with genome scale data obtained, for instance, from modern sequencing techniques. Metabolic Engineering (ME) makes use of genome scale metabolic models (GSMMs) to better understand the organism's functions and predict alterations that can optimize the production of compounds of industrial interest. In the last years, we have been experiencing a large increase in the sequencing of industrially viable micro-organisms that foster the interest on ME and GSMMs.

GSMMs [6] are mathematical representations of the metabolic transformations of a cell. The main purpose of these models is to allow the simulation of the metabolism of the cell, taking into consideration specific environmental conditions, as well as to predict how the cell reacts to genetic modifications such as, for instance, gene deletions or under/over expression. These type of problems usually focus on two main goals, which are the maximization of the production of a given product, while keeping the cell viable(the usual strategy is maximizing the biomass production). These predictive models are also called *in silico* metabolic models.

There are two classes of models that are mostly used in ME:

- **Stoichiometric Models:** are derived from the metabolic network of an organism. The information about reaction stoichiometry is the starting point [7], and it is represented by a set of equations that describe the chemical transformations of the system [8].

  The metabolic network is composed by a set of compounds (nodes) and a set of reactions/fluxes (edges). The network is represented in a stoichiometric matrix of m rows (representing the metabolites), and n columns (representing the reactions).

- **Kinetic Models:** dynamically describe the mass balances for each metabolite of the network. The system is typically represented by a set of differential equations to simulate the behaviour of the organism. These models are less common due to the fact that they have a larger number of parameters and there is less data regarding the dynamic intra-cellular behaviour [7], creating greater difficulties in the reconstruction of such models.

The fact that stoichiometric models are more commonly used by the majority of metabolic engineering tools, will lead us, from now on, to focus solely on the analysis of these models.

2.2.1 Constraint based analysis

Stoichiometric models' starting point is the determination of the stoichiometric matrix that represents the translation of biological knowledge into mathematical terms. Then, it is possible to represent the mass balances of the metabolites with differential equations:

$$\frac{dc}{dt} = S.v - \mu.c \tag{2.1}$$

where c is the vector of the metabolites concentrations, v the flux vector and μ the growth rate. Each equation describes the evolution of the concentration of each metabolite over time.

The term $\mu.c$ is eliminated, due to the fact that in most scenarios this value (dilution) it is much smaller than the value of the fluxes [9]. A pseudo steady state assumption is normally taken [9] stating that the amount of intracellular compounds that is consumed is the same amount that is produced, which means that the mass balances can be described by a homogeneous system of linear equations:

$$S.v = 0 \tag{2.2}$$

Equation 2.2 defines all feasible flux distributions [10] instead of a particular solution. Despite being feasible that does not mean that those are biologically realistic solutions, which means that there is a necessity to further restrict the solution space. It is possible to restrict the model by limiting the value of the fluxes:

$$\alpha_i \leq v_i \leq \beta_i \tag{2.3}$$

where α_i is the minimum value, v_i the value of the flux, and β_i the maximum value for the flux i.

There are two main types of restrictions, adjustable and non-adjustable. Adjustable restrictions may change through evolution and can be specific for different organism cells [11], and are usually used to validate the cell under specific conditions. The non-adjustable restrictions are, for instance, restrictions imposed by thermodynamics or enzyme capacities.

The type of restrictions shown in equation 2.3 allows to set limits to the value of the fluxes, and it also provides ways to restrict the model by the reversibility of the reactions and to restrict fluxes to a constant value. A reaction is reversible if $\alpha_i \in \Re \vee \beta_i \in \Re$, and irreversible if either α_i or β_i are 0. Fixing the value of a flux is possible by giving the same value to α_i and β_i. Also, this can accommodate restrictions coming from other types of experimental data, such as metabolomics or fluxomics.

The process of stoichiometric modelling (Figure 2.1) of a cell is a process that starts by determining the stoichiometric matrix, which will give us the mass balances, by assuming a pseudo steady state of the system (Equation 2.2), and by following a constraint based approach, limiting the value of the fluxes through restrictions (Equation 2.3).

Network pathway analysis is the attempt to understand and determine systematic properties that the metabolic network presents. It is possible to determine Elementary Modes [12] (EM) by analyzing the stoichiometric model constraints and the irreversibility of certain reactions. An EM is a flux vector that [12]:

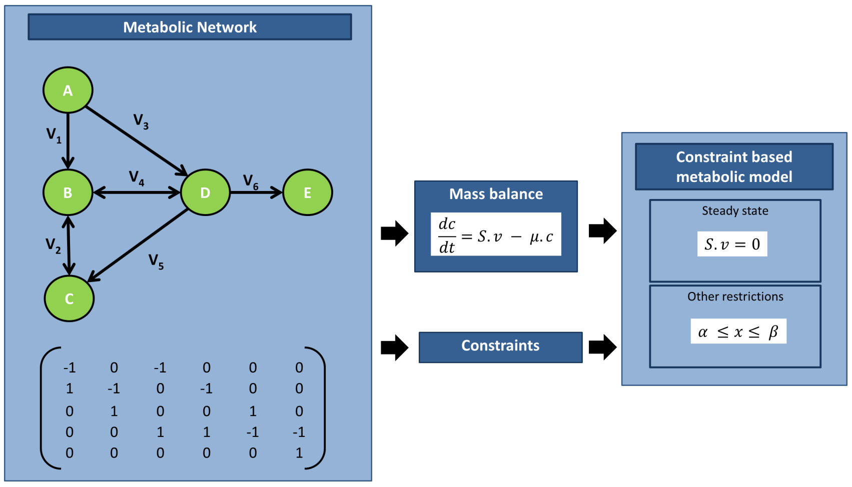

Metabolic Network
A
B
C
D
E
V_1
V_2
V_3
V_4
V_5
V_6
$\begin{pmatrix} -1 & 0 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$
Mass balance
$\frac{dc}{dt} = S.v - \mu.c$
Constraints
Constraint based metabolic model
Steady state
$S.v = 0$
Other restrictions
$\alpha \leq x \leq \beta$

Figure 2.1: Representation of the principles of stoichiometric modelling with a constraint based approach.

- satisfies the steady state;
- is thermodynamically feasible;
- there is no other non null vector that satisfies the previous constraints with a proper subset of its reactions.

EMs are the set of all routes through the network in steady state conditions that, from specific sets of substrates to products, cannot be decomposed to simpler routes [13], while maintaining steady-state, so they provide ways of analysing the set of pathways in the metabolic network.

As mentioned before, ME looks for ways to optimize the cell's production of desired compounds, and in order to do so, predictive analysis of the cells behaviour is an important step. When the point is reached when the stoichiometric model is fully built, there are methods that allow the calculation of flux distributions if a flux to maximize (or minimize) is chosen. Flux Balance Analysis (FBA) is the most common of these methods.

Phenotype simulation methods

Flux Balance Analysis [14] (FBA) uses linear programming (LP) to determine the steady-state flux distribution in a metabolic network by maximizing an objective function, such as the maximization of biomass or ATP production. Biomass production is in fact one of the most commonly used objective functions, based on the hypothesis that the metabolic objective of the cell is to maximize growth. While this may not seem correct, for instance, for human cells, it has been shown that for simpler uni-cellular organisms this principle may be applied [14]. To the existing model, additional restrictions are added usually to represent the model substrate uptakes. The result of a FBA simulation is an optimal steady state flux distribution (Figure 2.2).

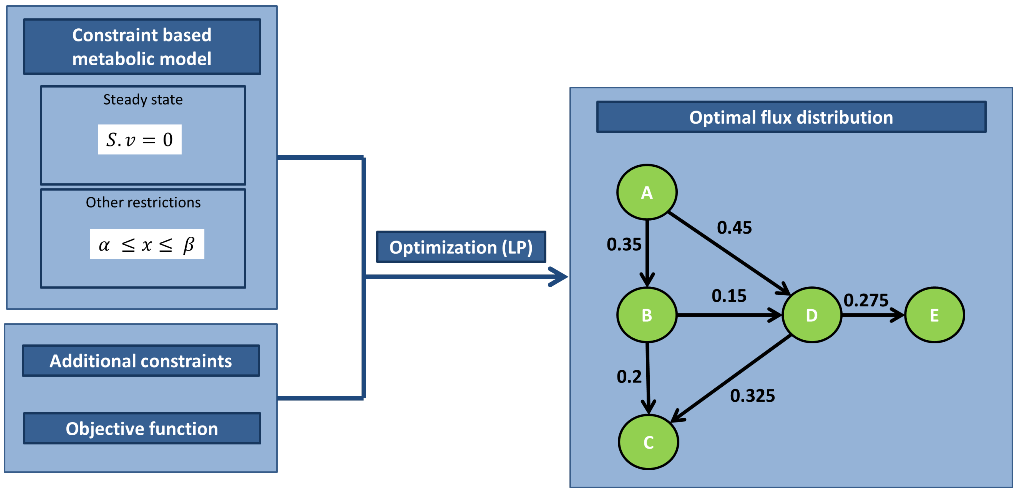


Figure 2.2: FBA major steps.

FBA assumes that the cell's optimal behaviour maximizes its growth, and can also be used to simulate mutant strains. The mutation may result from gene knockouts or changes in expression values. However, FBA's assumption of optimal growth may not be suitable for these cases, if we consider that the mutant cells were not exposed to long-term evolutionary pressure. There are other methods that can be better suited for mutant strain simulation.

The Minimization of Metabolic Adjustment [1] (MOMA) method assumes that the mutant cell tries to minimize the result of the perturbation caused by the mutation. This means that, if we consider the FBA solution space, the MOMA solution will be as close as possible from the optimal FBA solution. (Figure 2.3)

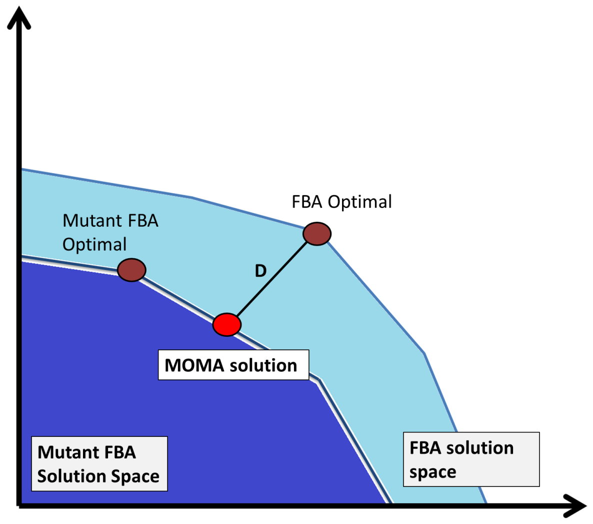


Figure 2.3: Relation between mutant and wild type FBA solution space. MOMA solution based on the distance. Adapted from [1].

Considering that the objective function is the minimization of the quadratic distance from both solutions, we are dealing with a quadratic programming (QP) problem.

Another common method for mutant simulation is the regulatory on/off minimization of metabolic fluxes [15] (ROOM). ROOM minimizes the total number of significant flux changes from the wild type flux distribution. This is made under the assumption that the cell minimizes the effort of adaptation to the new mutation, and it has been shown that there has been evolutionary pressure to minimize the cost of gene expression [15]. In terms of optimization methods, ROOM resorts to the use of a mixed integer linear programming (MILP) formulation.

In ROOM's predictions of growth rate, the flux distributions are very different from FBA's predictions, but the value of the growth rate is actually very close, while MOMA's are significantly lower. In fact, flux distributions from

ROOM are claimed to be more correlated with experimental data than the two other methods [15].

2.2.2 Existing software for Metabolic Engineering

In this section, some ME software tools are presented. The chosen tools focus on the capabilities to use metabolic models, perform steady state simulations and implement strain optimization methods.

OptFlux

OptFlux [16] `www.optflux.org` is a ME framework developed by the Bioinformatics and Systems Biology research group of the University of Minho. OptFlux aims at being the reference computational application in the field.

OptFlux has available a number of operations to visualize, import and export stoichiometric metabolic models, including reactions, metabolites, equations and, if available, gene-reaction associations. It also allows the use of stoichiometric metabolic models for phenotype simulation of both wild-type and mutant organisms, using several methods such as FBA, MOMA and ROOM. The strain optimization functionalities provide interfaces to identify sets of reaction deletions that maximize a given objective function related with a desired 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. OptFlux also provides a tool for EM calculation which provides a simple user interface that allows an intuitive filtering of the results that match given patterns.

OptFlux is open source, user friendly and compatible with standards, such as SBML [17], and utilizes free and commercial solvers such as Coin-or linear programming (CLP), the GNU Linear Programming Kit (GLPK) and IBM ILOG CPLEX Optimization Studio (CPLEX).

OptFlux is built in Java on top of AIBench (`http://www.aibench.org`), a software development framework that was born as a collaborative project between the host group and researchers from the University of Vigo in Spain. AIBench is a lightweight, non-intrusive, MVC-based Java application framework that eases the connection, execution and integration of operations with well defined input/output. This basic idea provides a powerful programming model to fast develop applications.

This tool was selected as the basis for the development of this project and, therefore, its implementation layers will be further analysed later in detail (Chapter 3).

COBRA

The COBRA Toolbox [18, 19] is a constraint-based reconstruction and analysis toolbox running in the Matlab environment. It allows for quantitative prediction of cellular behaviour using a constraint-based approach, and has methods to simulate, analyse and predict a variety of metabolic phenotypes using genome-scale models.

COBRA allows the use of stoichiometric metabolic models, that can be imported and exported in different formats, and a variety of phenotype simulation methods, including FBA, with several available objective functions (such as growth rate optimization, MOMA, flux variability analysis, among others). Solver support includes Gurobi, CPLEX and GLPK among others.

In version 2.0 several new features were added: ME Optimization algorithms (such as OptKnock [20] and OptGene [21]), model visualization with overlapping capabilities for flux distributions and metabolite concentrations, model reconstruction tools, etc.

COBRA is indeed a very powerful toolbox for ME and combined with the visualization functionalities added in version 2.0, it makes one of the most complete ME tools available. The fact that it is a Matlab toolbox means that it is not possible to use the toolbox without a paid license, and that is, probably, its main disadvantage.

FASIMU

FASIMU [22] is a command line oriented software for the computation of flux distributions. FASIMU aims at being a comprehensive, flexible and user-friendly computation environment for FBA.

Fasimu presents a really vast number of implemented algorithms, including biomass maximization, fitness maximization, MOMA and ROOM. For the solution of the optimization problem, and like the tools presented before, FASIMU uses different solvers, such as CPLEX, LINDO and GLPK.

For the visualization, FASIMU's team developed a plugin for BiNA (`http://www.bina.unipax.info/`), that allows the visualization of a computed flux distribution where the thickness and color of reaction arrows visualize the flux rate. FASIMU also prepares the input files needed by CellNetAnalyzer [23] and FluxViz [24], a plugin for Cytoscape [25].

FASIMU presents the most extensive list of FBA algorithms. The main disadvantages of FASIMU lie in the fact that it does not have an easy to use interface, and for the visualization a third party application is required.

Chapter 3

Biological Networks

Networks can be viewed as sets of entities related with each other, being used in several fields of science and engineering, which are rich in systems that can be represented by this approach. The difficulty of interpretation and analysis of large-scale databases makes networks useful tools for understanding the complexity of biological processes, being commonly used to represent biological entities and their interactions. This means that the analysis of networks is a rather important task, and to do so, a mathematical approach is necessary. Graph representation and graph theory assume a vital role in this approach.

3.1 Networks and graphs

3.1.1 Basic definitions

A graph can represent a network, and is composed by a set of objects that can be connected by links. The interconnected objects are usually called vertices, and the links are called edges. A graph can be represented as an ordered pair $G = (V, E)$ where V is the set of vertices and E the set of edges. An edge is represented by identifying the two vertices it connects, and a graphical representation of a graph can be easily achieved with points for vertices and lines or arrows between those points for the edges.

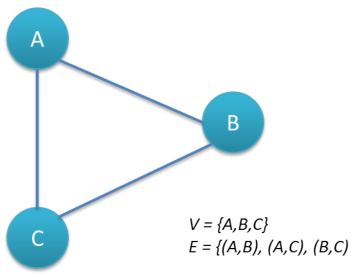


Figure 3.1: Representation of a graph with three vertices and three edges.

There are some basic definitions that must be taken into considerations when studying graphs:

- **Walk:** sequence of vertices and edges, where all vertices are connected to the next by the edge between them in the sequence (v1, e1, v2, e2, v3...);
- **Path:** walk where all the edges are unique;
- **Path length:** number of intermediate edges contained in the path;
- **Neighbour vertices:** vertices connected by an edge;
- **Vertex degree:** number of edges that connect to a vertex;
- **Strongly connected graph:** graph where, for all pair of vertices, there is an edge connecting them.

There are also some metrics associated with the analysis of the shortest paths of a graph:

- **Distance:** length of the shortest path between two vertices;
- **Characteristic path length:** mean of the distance of all pairs of vertices;

- **Efficiency:** inverse of the characteristic path length;
- **Diameter:** largest number of vertices which must be traversed to travel from one vertex to another without backtracking, detouring, or looping.

The attributes of the graph may affect the paths and the walks. It is extremely common that edges in a graph are not equivalent, and they can have a weight associated. This weight can represent cost, distance, time, etc.. Therefore, the sum of the weights of the path takes precedence on the length metric, and it can greatly change how the shortest paths are considered (for example, see Figure 3.2).

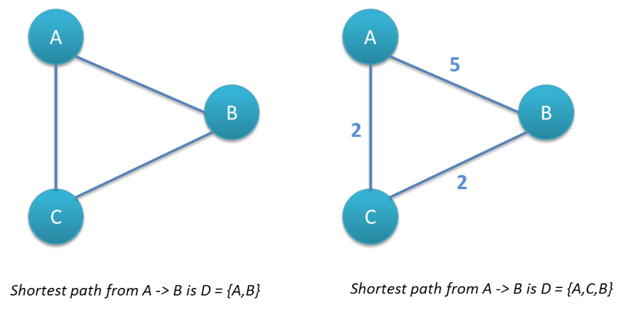

A
B
C
Shortest path from A -> B is D = {A,B}
A
5
2
B
2
C
Shortest path from A -> B is D = {A,C,B}

Figure 3.2: Changes in shortest path calculation on graphs with and without equivalent edges.

Calculating the shortest path can be very heavy computationally, and along the years some efficient algorithms arose, such as breadth-first search *(BFS)* [26] and the Dijkstra algorithms (we will not go into further detail about those in this work).

Another important aspect of a graph is its directionality. The directionality is associated with the edges of the graph, and it is classified as directed if the edges e = *(s,t)* take into consideration the order (holds only from s to t), and it is classified undirected if the order of the s and t are not taken into consideration. We can have three types of graph when considering directionality (see Figure 3.3): directed, undirected and mixed.

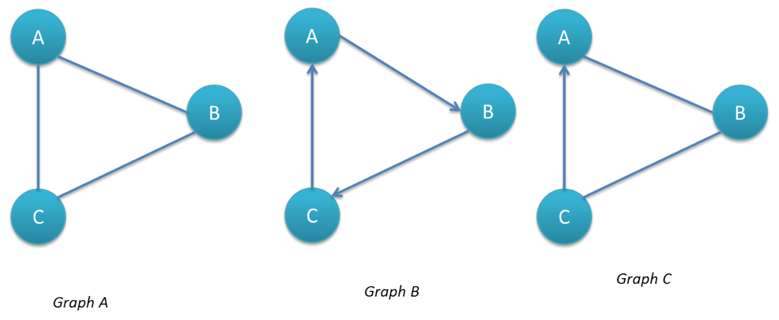


Figure 3.3: Different graph types, taking into account the directionality of the edges. Graph A is undirected, B directed and C mixed.

The directionality affects other metrics, such as the degree. When directionality is present (directed and mixed graphs), we can consider two types of degree of a vertex:

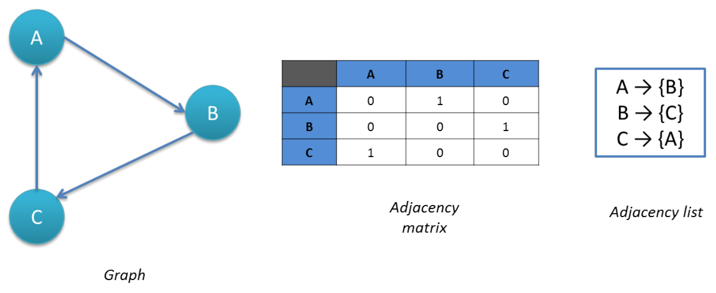
- **Indegree:** number of edges that end in the vertex;
- **Outdegree:** number of edges that start from the vertex.

The notions of walk and path must also take into consideration the directionality of the graph. If we have a directed edge $e = (v1,\ v2)$, walks can only consider this edge for paths that go from $v1$ to $v2$ and not otherwise.

3.1.2 Network representation

Besides the common graphical representation of a graph (vertices as dots and edges as lines or arrows), and the mathematical format - $G=(V,E)$ - there are other representations more suited for computational purposes and two of the most common representations of a graph are adjacency lists and adjacency matrices.

For adjacency matrices, the graph is stored in a matrix, where each row and column represents a combination of vertices. If the value of that cell is 0 (false), then there is not a connection between those two vertices, while if it is 1 (true) it means that those vertices are connected. If the connections between the vertices have an associated weight, it is very common to represent it in the connection cell. Adjacency lists, as the name indicates, are lists where for each vertex there is a vector of connections.



	A	B	C
A	0	1	0
B	0	0	1
C	1	0	0

Figure 3.4: Example of graphical representation, an adjacency matrix and and adjacency list, for the same graph.

3.1.3 Centrality and Clustering

Centrality

Determining which nodes are more important in a graph is one of the challenges of graph analysis. A centrality is a metric that allows the ranking process of the nodes by importance. The use of centrality measures is very common in social network studies, where persons or organizations are ranked by their position in the network, interpreted as their prominence in a social structure [27]. The importance of a vertex depends on the purpose and characteristics of the network. An important node in a social network can have different features when compared with an important node in a biological network. The metrics that are used to measure centrality must be chosen taking into account the specific characteristics of the problem. Centrality metrics have been successfully used for the study of multiple types of networks, such as citation networks, computer networks and metabolic networks [28]. Some of the most commonly used centrality measures are:

- **Degree centrality:** ranks vertices in descending order of their degree.

This metric is limited by the fact that it only takes into consideration the immediate neighbours of the vertex;

- **Closeness Centrality Measures:** defined in terms of geodesic distance between nodes in a graph. The importance of a node resides in how close and how quickly it can communicate with the other nodes in the network. Some closeness measures were developed in the context of resource allocation and were used in metabolic and protein interaction networks [29];

- **Betweenness Centrality Measures:** concept of betweenness centrality was introduced as a means of quantifying an individual's influence in a social network [30]. An important node will be present on a high proportion of shortest paths between other nodes in the network [29];

- **Eigenvector Centrality Measures:** also introduced in the analysis of social networks, the main idea is that a node is important if it is connected with important nodes.

Clusters

Cluster analysis or clustering is the task of grouping by similarity a set of objects. These groups of objects are called clusters. Networks often present strongly connected clusters, and in the case of biological networks, these clusters can have a biological significance. Within a cluster, two vertices that are both neighbours of the same third vertex have a heightened probability of also being neighbours of one another. The metric that quantifies this phenomenon is called the clustering coefficient [31].

3.2 Biological Networks

3.2.1 Types of biological networks

A large variety of biological systems can be represented as networks. In this section, a brief introduction to some of the most common types of biological networks is made. Some of those are given as follows:

- **Metabolic networks:** Focus on cell metabolism, and are composed of two main entities: compounds and reactions. The chemical compounds, or metabolites, can be converted by the cell into cellular building blocks, or decomposed to generate energy or other compounds.
- **Regulatory networks:** In a cell, the metabolism must be able to adapt to the changes of environmental conditions. The expression of genes must be regulated for the genes to be expressed at the correct time and in the proper amounts to ensure the functional integrity of the cell. Gene regulatory networks (GRNs), involving interactions between genes and their regulators, have been mapped onto graphical diagrams and networks that are used to analyse the regulatory relationships and understand global principles of gene regulation [32].
- **Signalling networks** The ability of biological cells to respond to signals from the external environment can be seen as a fundamental characteristic of life. In fact, cells have evolved highly elaborated and complex networks of signalling pathways [33]. Studying this type of networks will provide the understanding of how the cell coordinates and integrates the responses to external and internal signals.

These typically focus on specific systems, so the integration of these systems and understanding how they connect and interact with each other can give a larger comprehension of how organisms function as a whole. Biological network analysis can be a very important tool in this effort.

3.2.2 Metabolic Networks

Metabolism can intuitively be perceived as a network of chemical transformations. Metabolic pathways are series of chemical reactions that share several compounds and combining those pathways into the same network can provide a global overview of the metabolism of an organism.

Taking into account the goals of this work, metabolic networks are the most relevant type of networks, and thus will be analysed here in more detail.

The genome sequencing projects and the modelling of micro-organisms, discussed in Chapter 2, provide the scientific community with resources to develop this type of networks.

Of all the possible representations of metabolic networks, three must be referred:

- **Reaction-Compound networks:** In this type of networks, there are two types of vertices, the reactions and the metabolites, while the edges represent the directionality of the reactions. This is the most complex representation, but it is the most intuitive, and the one that provides the largest amount of information. The graph is, by definition, bipartite.

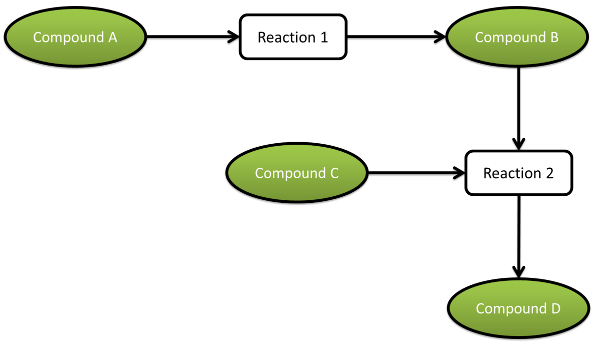


Figure 3.5: Reaction-Compound metabolic network.

- **Reaction-Reaction networks:** In these networks, the vertices represent the reactions, while the edges point from reactions that produce given compounds to reactions that use those compounds as substrates. This approach has advantages when studying relations between reactions.

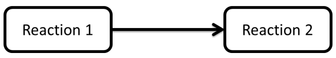


Figure 3.6: Reaction-Reaction metabolic network.

- **Compound-Compound networks:** This approach is similar to the reaction-reaction networks, but instead vertices represent compounds, and edges represent reactions. This brings advantages for studying the relationships between the compounds. A disadvantage is that several edges may represent the same reactions, which may lead to difficulties in analysing the network.

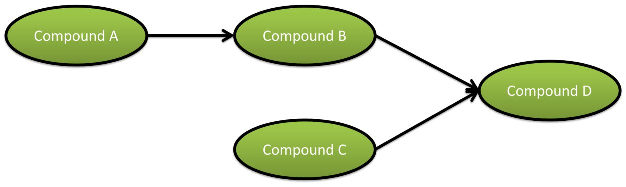


Figure 3.7: Compound-Compound metabolic network.

All these approaches use directed graphs, and for reversible reactions it would be necessary to duplicate all edges with the opposite directions. In the case of this work, that focuses on visualization of the network and not on the analysis, an approach using a mixed graph was chosen with the Reaction-Compound topology, that will be detailed later.

3.3 Visualization of biological networks

3.3.1 Important features in network visualization

Biological networks may represent processes from simple metabolic, signalling or regulatory pathways to broader ways of cellular organization. While it was common to manually curate maps and iteratively improve as new information became available, the appearance of high-throughput genetic techniques provided genome-scale analysis capabilities, which brought to the light the importance of software tools capable of automatically generating the visualization of these large networks.

Being able to capture a cell state in a visual form can provide insight to the biology of an organism, and network analysis has shown that a multitude of organisms share relevant properties [2]. Probably, the most important finding on biological network studies, was the fact that most cellular networks follow (or approximate) a scale-free topology [34]. Scale-free networks are characterized by having a few highly connected nodes, called 'hubs', while the others are connected by a few neighbours. Biologically this can explain the fact that biological systems are resistant to most attacks and very vulnerable to few specific ones[35]. Scale-free networks are, therefore, resistant to random node removal, but vulnerable to the removal of a few 'hubs' that will disrupt all network connectivity.

Visualization of these large networks is, therefore, problematic. While scalability of the networks can be successful addressed by generic visualization packages and graph algorithms, usually the generic layouts available produce unsatisfying results. This happens mostly due to the fact that these layout algorithms do not take into consideration biological knowledge, such as cell localization or molecular functions.

Another problem comes with the filtering of the networks. It is necessary to have an easy way to query the network and visually filter it to specific sets of nodes of interest, allowing an improvement of the analysis.

Finally, another common problem comes with the fact that there is an increasing number of the types of interaction between cellular entities, which brings problems to network modelling, and as a consequence to the visualization process.

In order to address all these problems a visualization tool should offer some basic features: layouts, graphical notation, integration of analysis and user interface [2].

Layouts

The first thing that comes to mind, when dealing with biological networks, is the ability to automatically draw the network, or build the network layout. There are several known layout categories:

- **Circular:** simple circular layout, where nodes are placed in a circumference form and links are straight lines between them;
- **Force-directed** [36]**:** Force-directed algorithms are iterative processes that attempt to place nodes in equillibrium. This is achieved by assigning forces to edges that pull linked nodes together and pull unlinked nodes apart. These algorithms are commonly used because they produce relatively good results, and are simple to adapt for specific purposes, such as using subcellular localization to better position the nodes.

  The main disadvantage also comes from the iterative nature of the algorithms, that may be time consuming reaching the equilibrium state, which can be solved (from the user's point of view) by animating the layout building.
- **Hierarchical** [37]: aims at highlighting the main direction of a directed graph. The nodes are placed in arranged layers such that the majority of the edges show the same overall orientation. The order of the nodes is reached to reduce the number of edges crossing;
- **Simulated annealing:** named from annealing in metallurgy, this method follows a notion of slow cooling, or in this case a slow decrease in the probability of accepting worse solutions as it explores the solution space. For layouts, the solution space is composed of network layouts that are selected according to an associated energy so that low energy states correspond to potential solutions. Accepting worse solutions

allows for a more extensive search for the optimal solution. The main disadvantage is the slowness of the method which limits the size of the network to be visualized.

- **Hierarchical clusters:** it has already been shown that the hierarchical clustering can produce simplified layouts of biological networks [38]. This is particularly true for protein-protein networks as the existence of general protein clusters attest [39, 40].

Graphical Notation

As mentioned before, graphical notation is a very important aspect of network visualization. In the case of biological networks, these rely on the classical graph representation, where the nodes identify the entities (sometimes distinguishable by shape) and edges relations between the edges. Colors can also be used to represent some network properties, for instance, subcellular localization or gene expression.

In terms of visualization, there is not a well defined standard, or at least the attempts of standardizing biological networks representation have not been very successful. This has been solved by the development of informal standards, usually imitating approaches followed by more popular tools.

There are, however several attempts at developing these standards that are promising: the Systems Biology Graphical Notation (SBGN) [3], for instance, is a good example, which uses state transition diagrams to model biological processes, and it is used in CellDesigner [41, 42]. The SBML Layout package is also worth mentioning, being a package that gives the possibility to add SBML annotations with layout information, which would allow tools to exchange SBML files without loosing information on layout alterations.

Integration with analysis

Ii is very common that tools specialize only on visualization or analysis of data. The ideal scenario is that these two features are incorporated in one, and developers should always aim at this scenario.

Sometimes this can be solved by recurring to an integration with another software, which highlights the importance of plugin-based software, that can be the solution for the integration of visualization and analysis.

Another case worth mentioning is the incorporation of external data sources, for both analysis and visualization purposes, that are common in several tools. There are a lot of public repositories with data that can be useful, bringing once more to light the problem of the lack of standards. The translation of data from various sources with different formats is a non-trivial task. There are, however, some biological network tools that that offer partial solutions to this problem.

User interface

A visualization tool should support a graphical user interface where it is possible for the user to interact with the network, as easily as possible. Network edition, such as creation and removal of nodes, and access to information, for instance, by clicking on the nodes, are desirable features for a better user experience.

However, in the case of biological networks, what happens, in most cases, is that the features provided by the tool are not sufficient for the specific tasks the users want to perform. This is partially solved by the fact that several visualization tools can be expanded by the development of plugins. This allows advanced users to develop features they desire, but the development of these plugins requires time and expertise.

3.3.2 Software tools for biological network analysis and visualization

In this section, some of the available software for biological network visualization and analysis are presented, and some of their main capabilities are highlighted.

Cytoscape

Cytoscape is a popular bioinformatics package, that became a standard tool for integrated analysis and visualization of biological networks [25]. It supports the visualization of molecular interaction networks, biological pathways and the integration of experimental data using attributes which map nodes or edges to specific data values, such as gene expression levels or protein functions [43].

Cytoscape is a flexible visualization and editing tool that allows the user to create and edit networks using a user-friendly graphical user interface (GUI). It has several functionalities, such as "visual styles" that allow users to customize the appearance of the network, and associate several visual styles for the same network. It also supports several layout algorithms, but they tend not to perform well with large networks. On top of all this, Cytoscape is a plugin based framework, and there is a vast number of plugins available, that expand the functionalities provided by the core packages. There are several plugins for network analysis, that use some of the metrics mentioned before, such as CentiScaPe [44] for centrality metrics, NetworkAnalyzer [45] that computes a large number of network topological parameters, ClusterViz and MCODE [46] for clustering, ShortestPath for shortest path calculations, among others.

One plugin that must be mentioned, since it is closely related to the scope of this work, is FluxViz [24], a plugin for the visualization of flux distributions in networks. Primarily developed for FASIMU [22], a software for flux-balance computation, it uses the generated result files as input for visualization. The fact that the visualization depends only on the flux distribution and network structure, makes FluxViz independent from the simulation tool, making the importation of the flux distributions also possible for other formats (CSV and Cytoscape attributes).

FluxViz features include:

- importation of flux distributions (FASIMU val files, CSV and Cytoscape attributes). Networks can be imported using Cytoscape capabilities;
- all layout algorithms available in Cytoscape;
- multiple flux distributions in one session;
- filtering of the view based on flux values or node attributes and generation of subnetworks;
- flexible mappings of flux information to the visual node and edge attributes;
- exportation of the network view for multiple formats (PDF, SVG, EPS, JPEG, PNG, BMP).

Another important plugin that is closely related to this work is the CySBML [47], a plugin designed to work with Systems Biology Markup Language (SBML) with the following main features:

- importation of SBML files;
- support for SBML layout package;
- support for qualitative model packages;
- network navigation based on SBML structure;
- SBML validation.

Cytoscape is a powerful and easy to use visualization platform, but the visualization of flux distributions requires the usage of an additional tool. Besides that, flux distributions are numeric values, which brings some limitations when distinguishing, for instance, a reaction that has a flux of 0 from reactions that was knocked-out as an effect of a gene deletion.

CellDesigner

CellDesigner [41, 42] is a user-friendly editing and visualization tool for biochemical networks. CellDesigner supports Systems Biology Markup Language (SBML) [17], an XML format that tries to provide the systems biology community with a standard for model and network representation, and uses a graphical notation and listing of symbols based on the proposal by Kitano[48].

The tools provided by CellDesigner allow the user to build readable networks, with several shapes for edges and vertices, and multiple layout algorithms are available. It is also possible to extract networks from a series of databases.

Users can also use their own SBML model created by CellDesigner for simulations on all SMBL compliant applications.

CellDesigner is a powerful visualization tool, whose main disadvantage lies in the fact that it lacks network analysis methods, and only supports network models in SBML.

Vanted

VANTED [49] (Visualization and Analysis of Networks containing Experimental Data) is an application for the visualization and analysis of networks with related experimental data. VANTED has a user friendly interface; networks can be manually created or imported from a series of formats such as Graph Markup Language (GML) and SBML, as well as direct importation from available databases, such as KEGG; it also supports several layout algorithms.

VANTED's main advantage is the fact that it handles experimental data quite nicely. The data can be associated with the network and if the necessary elements are not all present they are automatically added into the network which means that it is possible to create networks from scratch just with the experimental data. It is then possible to use VANTED's statistical methods to analyze the resulting network.

VANTED's functionalities can be expanded by the available add-ons that can be obtained from the application website and there are two in particular that are closely related to this work:

- **FluxMap:** allows the visualization of measured or simulated fluxes in the network by changing the appearance (thickness) of the edges. Available visualization options and interaction possibilities enable comparison of complex experimental setups in an interactive way.
- **FBASimVis:** add-on for constraint-based analysis of metabolic models, with a special focus on the dynamics and visual exploration of metabolic flux data resulting from model analysis. It supports wild type and knock-out Flux Balance Analysis simulations with the models represented by the networks. The graphical representation of the network is changed to represent the fluxes and critical reactions.

Patika

Patika [50] is an integrated software environment designed to provide the scientific community with a solution for modelling and analysing cellular processes. Its main goal is to provide ways of modelling and storing a vast amount of cellular pathway data in a centralized database.

Patika uses a representation of the pathways very similar to the chemical equations in metabolic networks. The network represents each state and transition in specific nodes. States can be molecules or complexes of physical phenomena, while transitions can be group additions and removals, complex formations, transportations as well as other cellular events. Patika is a client-server based application that requires an internet connection to function. The server manages the database, queries, submissions and users, while the client provides the user an easy to use interface for querying, retrieving and manipulating pathways from the database.

Pathways are created on the fly and their information is obtained from the database. Then, the user can manipulate that data and even add more information. This new pathway can be stored locally or be submitted to the server, and after an evaluation it can be added to the Patika database.

The client provides several edition tools, such as zooming, scrolling, selection, dragging, event-handling and persistent storage, as well as automatic layouts functions.

One of the particular aspects of Patika is that it allows a series of queries to the database, not only to search particular pathways but also to find relationships not noticed before. It is possible to query certain states or transitions, search for paths between states or transitions, shortest paths between states, among others.

Later, PATIKAweb [51] was also introduced and it provides a Web interface for retrieving and analysing biological pathways in the PATIKA database. The server contains data integrated from several public databases, and this provides a Web-based service with a user friendly interface without requiring any registration or installation, and usable with any internet browser.

Chapter 4

Results and Development

In this chapter, the development of the visualization framework and its functionalities will be presented. Implementation details and technical aspects will also be covered. Firstly, the goals and architecture adopted will be detailed, followed by the description of the features of the visualization framework and the integration with OptFlux. Finally, technical aspects of the development will be detailed, from the core libraries to more specific technical characteristics.

4.1 Development methodology and overall architecture

The main goal of this work is to create a visualization tool that will allow researchers to perform visualization tasks in the context of ME projects. The main focus of the visualization is, therefore, the metabolism. Metabolism can be represented as a series of transformations of chemical compounds/ metabolites, and that makes it relatively easy to represent as a graph.

4.1.1 Layout components and model mapping

There are two main entities that will have to be addressed by the visualization: the reactions and the compounds/metabolites. A reaction is a chemical transformation that uses a set of metabolites as reactants and produces another set of metabolites as products, that can be used as reactants for other reactions.

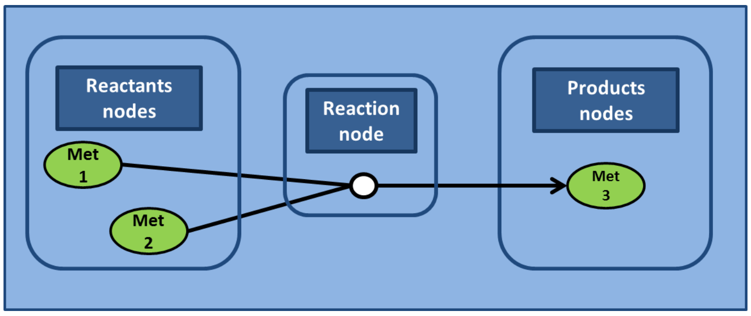


Figure 4.1: Reaction representation in a reaction-compound network.

As seen in Chapter 2, there are three main types of metabolic networks, but for visualization purposes it seems that the most descriptive, and at the same time, most visually attractive is the reaction-compound network. For these reasons, this representation was used, where a reaction is composed by two sets of nodes - one for the reactants and the other for the products - connected by a reaction node, as depicted in Figure 4.1.

A metabolic layout is based on the reactions contained in the metabolic model, and since one of the goals of this work is to provide a link between the visualization and the metabolic model, a strategy must be defined to map the entities of the visualization with the entities of the model. On top of this, it is also desired that a layout can represent just a part of the metabolism of an organism, and the possibility to use the same layout on different models (e. g. for different strains of different model versions).

Another aspect of metabolic networks is the fact that, to make them visually more understandable, some nodes may be replicated. It is very common to replicate metabolites such as cofactors, or highly connected hubs that do not have a high interest (e.g. water), to make the network more readable. These metabolites are typically referred as currency metabolites. It is desired that these nodes are differentiated from the other metabolites but it also means that the metabolic identifiers from the metabolic model may be present in several nodes of the same layout.

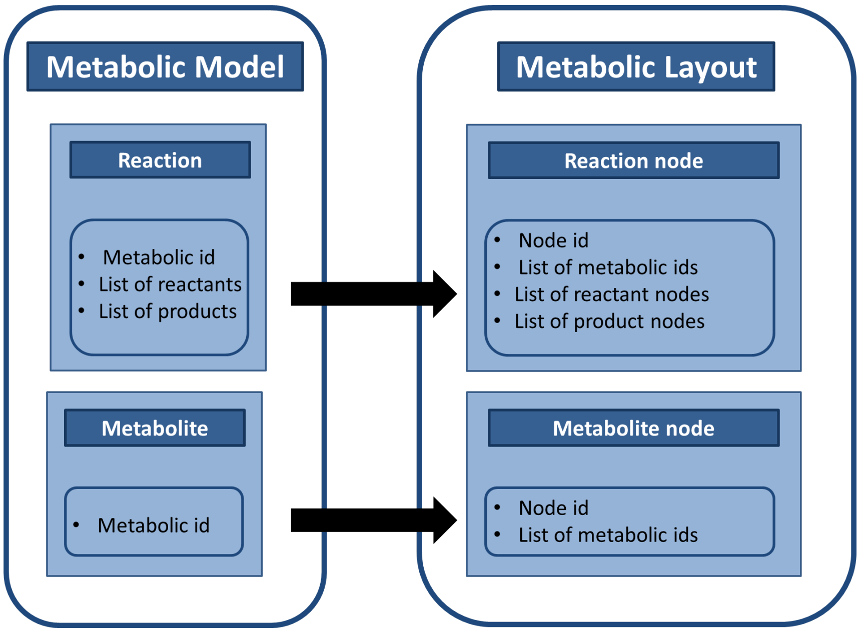


Figure 4.2: Mapping of metabolic reactions and metabolites with the nodes of the layout.

To comply with the previously mentioned features, each reaction node and metabolite node will have a list of metabolic identifiers that will provide the link between the metabolic model and the layout (Figure 4.2).

If we take into consideration the definition of a reaction node, that is composed by the metabolic identifiers and by the reactant and product metabolite nodes that represent the substrate and products of the metabolic reaction, it is possible to define a layout with a list of these reactions. The visual representation will have three types of nodes, the reaction nodes, the metabolite nodes and the currency nodes - used to represent the previously mentioned cofactors or hubs with little interest. On top of this, the user who develops layouts may want to associate some kind of information to a reaction, which will be possible by adding another type of node: the information node.

Thus, each reaction in the layout is composed by the following set of attributes:

- *Unique identifier*: uniquely identifies the reaction node;
- *Label*: text that will be shown in the reaction node;
- *Metabolic identifiers*: identification of the reactions that the node represents from the model;
- *Coordinates (x,y)*: position of the node in the layout;
- *List of reactants*: nodes that represent the metabolites consumed by the reaction;
- *List of product*: nodes that represent the metabolites produced by the reaction;
- *Information nodes*: information nodes that provide additional information about the reaction.

Each reaction has two sets of nodes: one that represents the metabolites consumed and another that represents the metabolites produced. Additionally, there is a third set of nodes, that are the information nodes, that give additional information about the reaction. Each of these nodes are defined by the following information:

- *Unique identifier*: uniquely identifies the node;
- *Label*: text that will be shown in the node;
- *Metabolic identifiers*: identification of the metabolites the node represents from the model;
- *Node type*: indicates whether it is a normal or a currency metabolite or an information node;
- *Coordinates (x,y)*: position of the node in the layout.

4.1.2 Overall architecture

With the definition of the layout and the composition of each of its elements, an overall architecture of the visualization framework was designed.

The two main tasks of the software to be developed are to be able to build these layouts from external sources (being able to export them as well), and visually representing them. This leads to the idea of a 2-layer architecture (Figure 4.3), where one layer has the capabilities to read and write metabolic layouts, while the other layer, the visualization layer, handles the visualization and edition of the metabolic layout. The main features of each are given below:

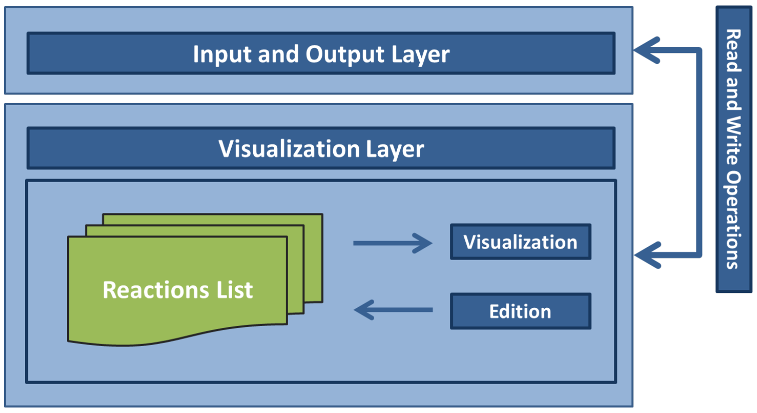


Figure 4.3: 2-layer architecture of the metabolic visualizer.

- **Visualization layer:** this layer should provide all the functionalities related with the visualization and edition of a layout. Some of the features should include automatic layout, creation and edition of layouts, visual filters and actions to change the aspect of the network (colors, shapes, etc.).

- **Input and output layer:** There are several tools that provide network creation and exportation capabilities for a multitude of file formats. The Input and Output Layer has the objective of providing the capability to read a given network in a specific file format, and building the metabolic layout, usable by the visualization layer. At the same time, it shall also provide the possibility to export those layouts into some of those formats.

4.2 Main features of the visualization framework

4.2.1 Visualization layer

As stated previously, the visualization layer provides all the functionalities related with visualization and edition of the metabolic layout. In Figure 4.4, it is possible to see how a metabolic layout is shown in the visualizer. In this figure, it is possible to see how the four node types are represented. The green ellipses are the regular metabolites, while the smaller yellow ellipses are the currency metabolites. The blue rectangles are the information nodes, while the small gray squares are the reaction nodes (the edges have the same color as the reactions to give the visual notion of a single element).

The visualization tool graphical user interface (GUI) is composed of two major elements, as seen in Figure 4.5: the network view, where it is possible to edit the network and click/drag the nodes, and the side panel where filters, overlaps and node information are available. This way, it is possible, for the user to easily interact with the network, using all the features this interface offers.

Highlight, node information, zooming and dragging

The visualizer provides a highlighting functionality. If an user hovers the cursor over a node it will highlight it. If the node is a metabolite, and it is represented in other nodes of the network, these will be highlighted as well. If the node is a reaction, it will highlight all the elements of that reaction.

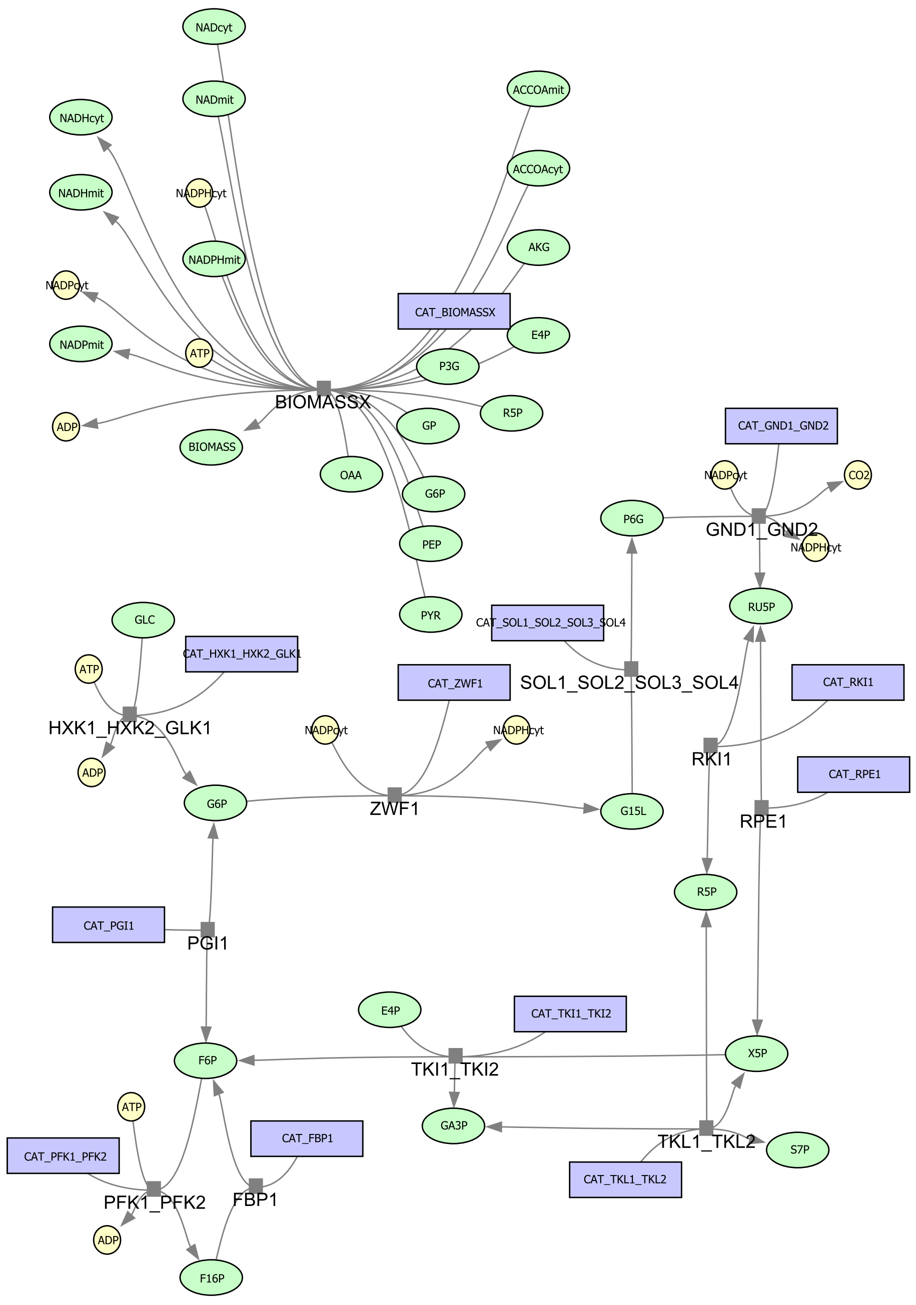


Figure 4.4: Metabolic layout in the visualization tool: an example.

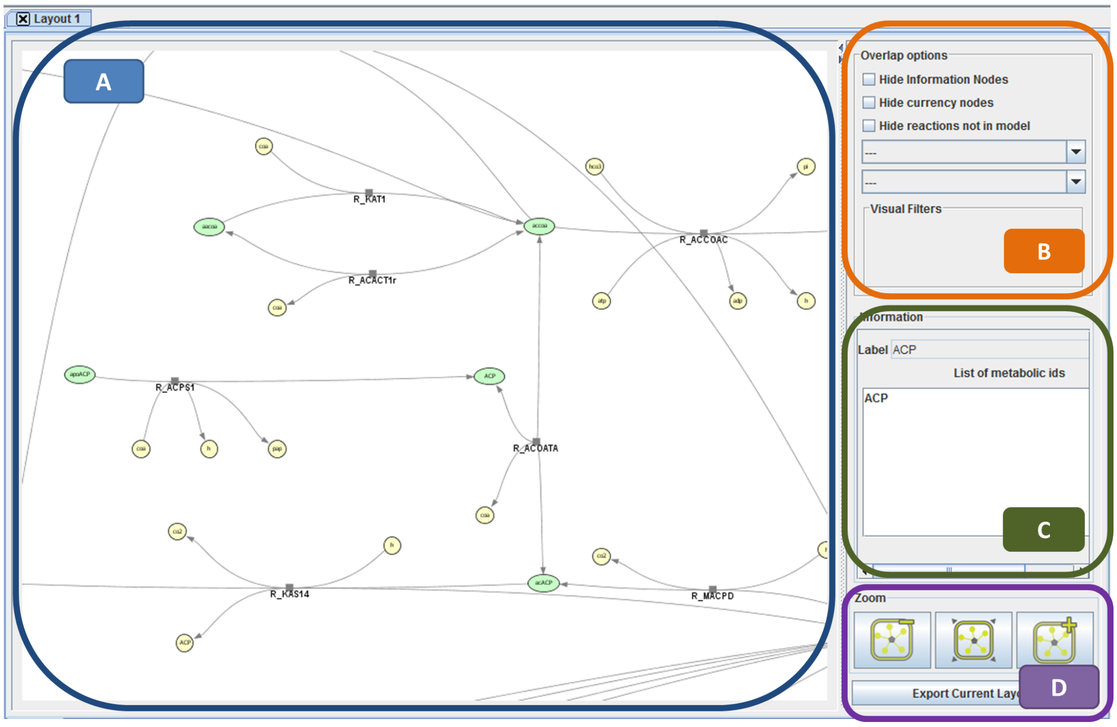


Figure 4.5: Visualization tool GUI: A- metabolic network visualization panel; B - filters and overlaps panel; C - node information panel; D - Zoom panel and export button.

By clicking a node, the node information panel (Figure 4.5 C) will display information on that node. It is possible for advanced users to implement an information panel and add it to the visualizer, to visualize information of specific interest. This will be explained in more detail later.

Zooming and dragging allows the user to navigate the layout. Zooming can be made by using the mouse wheel or using the zoom buttons in the side panel (Figure 4.5 D), while dragging can be made by pressing the right mouse button and moving the screen. It is also possible to drag nodes or sets of nodes. To drag a set of nodes, it is necessary to select several nodes, which can be done by pressing the control key while selecting nodes, or selecting an area of the network pressing the left mouse button.

Layout Algorithms

One of the characteristics desired for the visualizer is the capability to load layouts from different sources. Some layouts may not specify the coordinates of the nodes, at least for some of the nodes. For instance, if the original network is not a bi-partite graph, it is necessary to have a layout algorithm that will take this into account.

The layout used by the visualizer, by default, is the Force Directed Layout (FDL). Force directed based algorithms determine the nodes positions by assigning forces among the nodes and edges based on physical laws of attraction and repulsion.

Prefuse (www.prefuse.org) libraries already provide a FDL layout, which was altered to support fixed nodes. These nodes already have a specified position, and will be fixed in that position, influencing the position of the variable nodes, which will be placed by the FDL algorithm.

Edition tools

Another crucial aspect of the visualizer is the ability to edit the metabolic layout. This feature combined with the import and export capabilities of the visualizer provides the users with the means to create and edit their layouts, and export them for later use.

By right clicking on the nodes of the layout, the possible edition options are shown to the user (Figure 4.6). The possible actions are:

- **Fix/Unfix nodes:** this option allows the user to fix a node to the specific position it is in, or drag it to a desired position; unfix a node will remove the position information of the node, making it susceptible to the FDL forces, adjusting its position automatically according to its surroundings. It is also possible to unfix and fix nodes by type, allowing to fix/unfix all reaction or metabolite nodes at the same time.

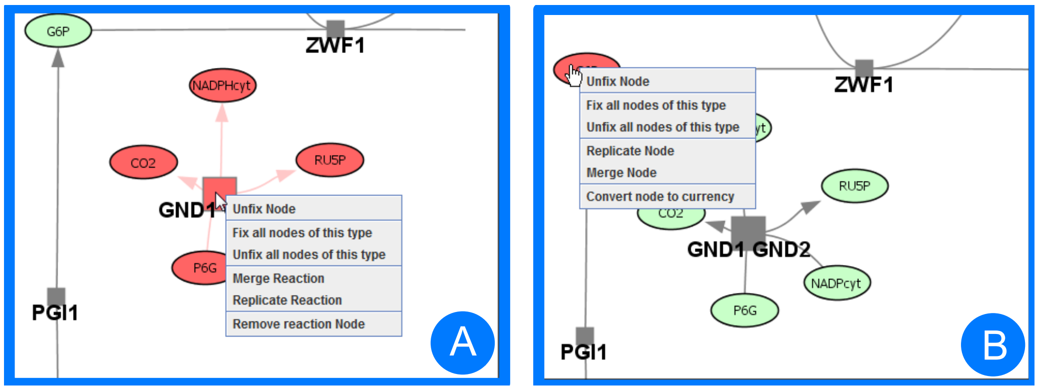


Figure 4.6: Right click menu of reaction nodes (A) and metabolite nodes (B).

- **Merge and Replicate metabolite/currency nodes:** as stated above, the same metabolite can be represented several times in a layout by a multitude of nodes. To allow the user to choose if he wants a single node or several nodes representing a metabolite, it is possible to replicate the node (if several reactions are connected to it) or merge it to have several reactions connected to that specific node (as seen in Figure 4.7).

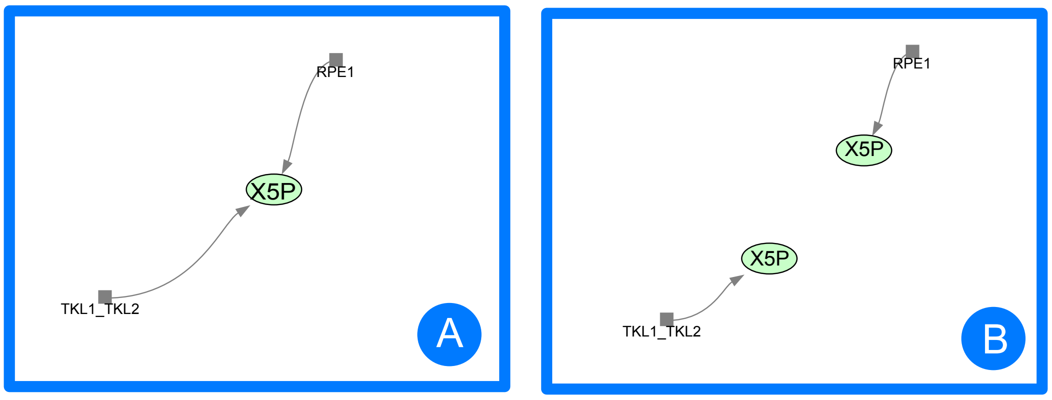


Figure 4.7: Metabolite node merged (A) and replicated (B). It is possible to go from one state to the other by merging or replicating the node.

- **Change metabolite node type:** it is possible to change the type of a simple metabolite to a currency metabolite and vice versa. When the type is changed, if there are other nodes representing the same metabolic compound, they are also changed. This can be useful if, for instance, the user wants to hide the currency nodes (a filter defined by default in the visualization interface).

- **Merge and Replicate reaction nodes:** it is possible to replicate a reaction node that has more than one metabolic identifier. This will result in two, or more, reactions connected to the same metabolite nodes. Merging two reactions is only possible if the reactions to merge are exactly the same (as seen in Figure 4.8), i.e. they are connected to exactly the same nodes (if different nodes represent the same metabolic compound they can be merged before merging the reaction);

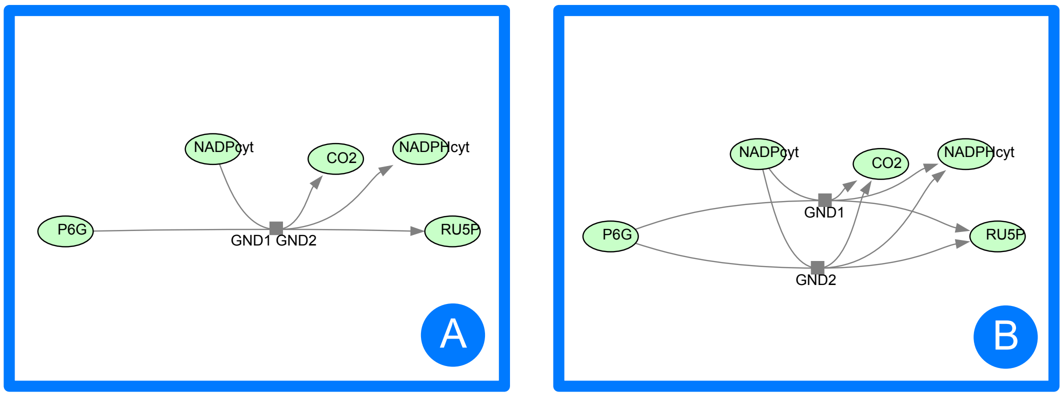


Figure 4.8: Reaction node merged (A) and replicated (B). It is possible to go from one state to the other by merging or replicating the reaction node.

- **Delete reaction node:** it is possible to delete a reaction node, and when this is done all edges and nodes that are only connected to that reaction node are also deleted.

Filters and Overlaps

On top of the edition capabilities, the visualizer was also developed to support "visual overlaps", that are the functionalities that allow the application of filters to the network, changing the color, shape and labels of nodes as well as the thickness of the edges. By applying overlaps to the network it is possible to manipulate its aspect.

The visualizer supports visual filters in two distinct ways:

- By type: it is possible to hide nodes by type, for instance hide all currency metabolites, or information nodes.
- By reaction: a list of reactions to hide; this will also hide metabolites that are connected only to these reactions.

Overlapping supports changing the direction of the edges, thickness and colors, while for nodes it is possible to change the color and shape (Figure 4.9).

These functionalities allow, for instance, the visual representation of steady-state simulations that will be better described in the OptFlux interaction, in a later section. All the overlaps are shown in the Overlap panel ordered by category. For instance, a simulation result is displayed in the category "Simulations".

4.2.2 Input and output layer

The features of this layer are the support for the different input/output formats it provides. The supported file formats are:

- **CellDesigner SBML (CD-SBML):** CD-SBML is a graphical notation system proposed by Kitano [52], where layouts are stored using a specific extension of the Systems Biology Markup Language (SBML).

  The structure of a CD-SBML is a lot similar to the structure of the metabolic layout, it also has the notion of metabolites and reactions which facilitates the conversion.

- **eXtensible Graph Markup and Modeling Language (XGMML):** This format is based on the Graph Modelling Language (GML), being

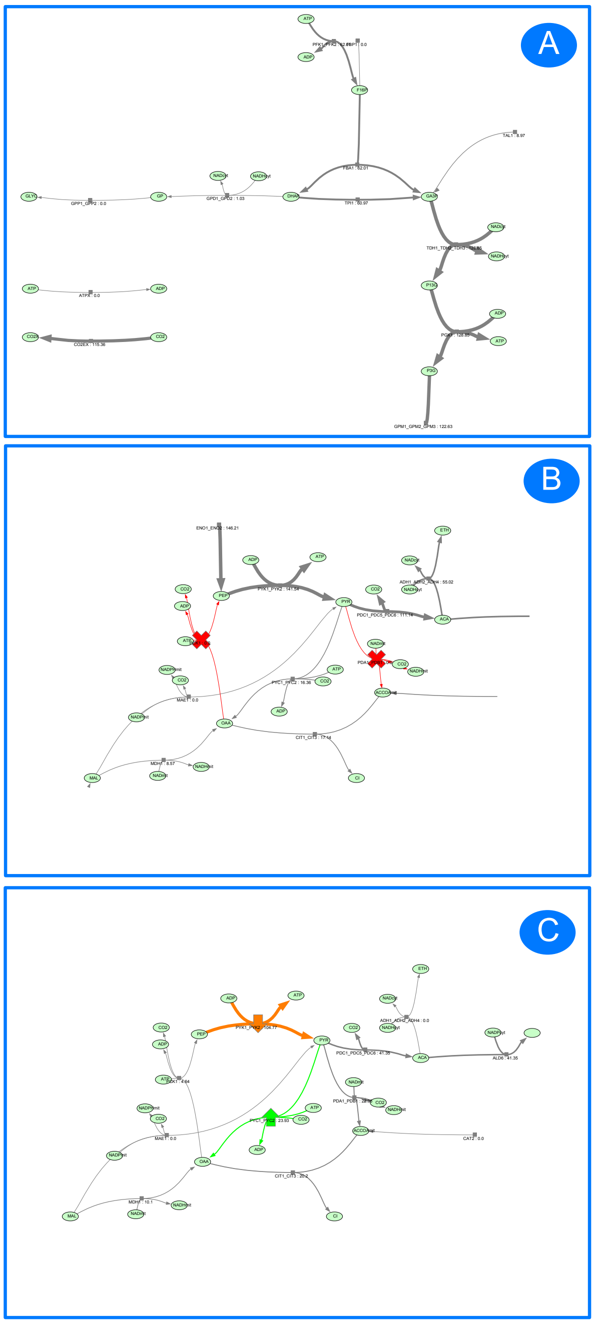


Figure 4.9: Some of the possible changes to the networks using overlaps and filters. A has thickness and label changes; B has nodes with cross shapes and red edges; C has nodes with arrow shapes and edges with different colors.

used for graph description using XML tags to describe nodes and edges of a graph. It is supported, for instance, by Cytoscape.

To create a metabolic layout from an XGMML file it is necessary to specify some fields for the nodes: the metabolic identifier, node type (metabolite, currency, information, reaction) and, in the case of reaction nodes, the reversibility of that reaction.

The exportation of a layout in this format also adds Cytoscape fields, so it is possible to export these generated networks to Cytoscape with a specific visual aspect.

- **KEGG Markup Language (KGML):** KGML is an exchange format created for the Kyoto encyclopaedia of genes and genomes (KEGG)[53] maps automatic drawing. KEGG pathways' conversion is relatively easy because they contain the notion of reaction, but other types of problems arose with this format. The fact that this format exclusively uses KEGG-ids while genome scale metabolic models do not use these identifiers as primary source of identification of their compounds and reactions, brings the need to provide a map of identifiers to make possible the connection between the model and the layout.

  Another problem comes with the fact that it is very usual that a KEGG reaction node represents several reactions (and the same with compounds), which may require some work over the layout before it is ready for use.

- **Systems Biology Graphical Notation (SBGN)**[3]**:** SBGN is a visual language developed by biochemists, modellers and computer scientists that aims at being the standard for visual representation of biological processes.

  SBGN defines three visual languages: Process Description (PD), Entity Relationship (ER) and Activity Flow (AF). For the purpose of this work, which focuses on metabolism, support was only developed for PD language files. The PD language was based on Kitano's proposal used in CellDesigner's graphical representation, using bi-partite graph representations of metabolic processes, which is the same approach used for our metabolic layouts defined previously. It is also possible to export simple PD language files for metabolic layouts.

- **COBRA Layouts:** maps developed for COBRA Toolbox. There are, at the moment,several maps on this format for many of the models hosted in the BiGG knowledgebase (http://bigg.ucsd.edu). These maps can be used on different models that have similar pathways, with a correct mapping of the identifiers between the layout and the BiGG model. The files are text tab-delimited files divided in four sections, and since they were created to map a metabolic model for a ME tool, the conversion to the metabolic layout presented in this work was relatively straight forward. These files contain the notion of molecules (metabolites/currency), reaction nodes (transformations) and reactions (reaction nodes).

Besides the support for all the formats mentioned, there is another feature that is included in this layer, but it is slightly different from the ones mentioned before, the pathway generation layout.

It is possible to generate a layout by using a list of reactions from a genome scale metabolic model. This can be done following two strategies: choosing a list of reactions, or in the case that the model has pathway information, building layouts with the reactions from those pathways. It is also possible to use another metabolic layout as a base, and choose other reactions or pathways to add to that layout. This functionality, along with the edition capabilities of the visualization layer, provides the means to create and edit layouts.

4.2.3 Integration with OptFlux

As mentioned before, OptFlux is a ME tool developed by the University of Minho's Bioinformatics and Systems Biology research group. OptFlux is open source, easy to use and compatible with standards.

Since OptFlux's main goal is to provide a ME tool that is easy to use, simple and intuitive, the user interaction was based in three main concepts:

- **Datatypes:** represent the data of the application. These datatypes can be simple or combinations of multiple datatypes. Users can manage objects, instances of the different datatypes, through the use of an hierarchical clipboard.
- **Views:** represent the visualization of the datatypes. Each datatype can have one or more views, represented in different tabs.
- **Operations:** available actions that allow the creation of instances of the datatypes. Operations are transformations that have a set of arguments as input, and a set of output objects. Operation's user interface can be automatically generated by AIBench or defined by the developer.

Based on these concepts, a user-friendly Graphical User Interface (GUI) was developed using the original layout of the components that can be observed in the screenshots presented in Figure 4.4.

OptFlux is also plugin based, which allows new features and services to be easily added. This facilitates reusing and integrating new functionalities.

With all these issues considered, it was the chosen tool to integrate the visualizer packages described before, integrating the phenotype simulation capabilities with the visualization of metabolic models. This led to the development of a visualization plugin for OptFlux, named *MetabolicVisualizer4OptFlux3*.

A visualization plugin for OptFlux

This plugin is included in the core set of plugins for OptFlux 3, being readily available when the application is downloaded and installed. To obtain the software and further information, check the website `www.optflux.org`. A tutorials of this plugin operations' is also available on-line at the OptFlux Wiki (`http://darwin.di.uminho.pt/optfluxwiki/`).

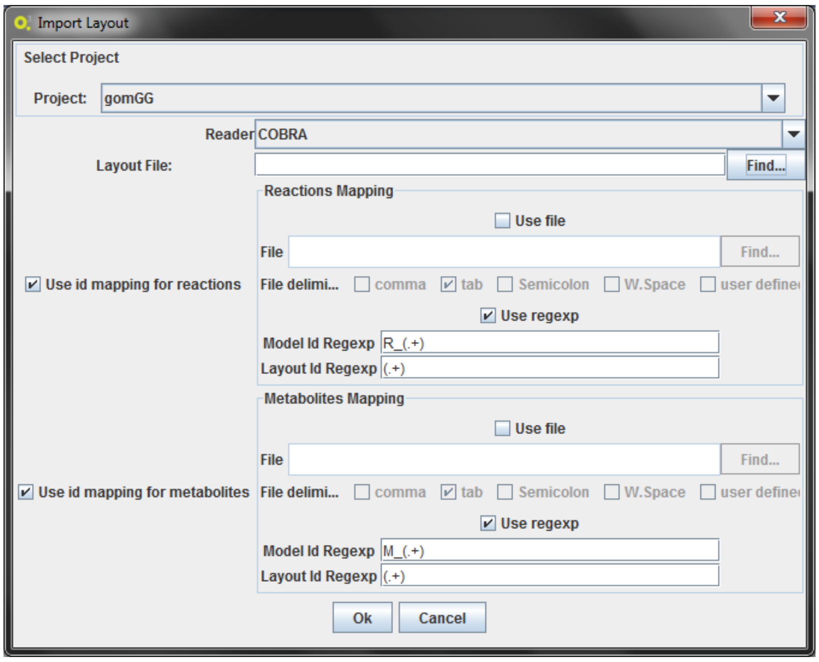
The visualization plugin has three operations that allow importing layouts, listed in Table 4.1. The Import Layout operation, is the operation that allows the importation of layouts from XGMML, SBGN, CellDesigner or COBRA files. One thing that must be kept in focus is the fact that this plugin has the goal to make the connection between the genome scale metabolic models loaded in Optflux, and layouts from the visualization framework. The identifiers of the metabolites and reactions on the metabolic model are unique by nature, but as shown before, a metabolite can be represented by more than one node, and the node can have more than one metabolic identifier associated with it.

This operation offers ways for the user to map the identifiers of the metabolic model with the metabolic identifiers of the reactions and metabolite nodes of the layout. In this operation there are two different methods of mapping available: loading a simple two column file with the mapping of the identifiers (from metabolic model identifier to layout identifier), or by applying regular expressions to the identifiers in the model and/or the layout. The regular expressions must only contain one group (as seen in Figure 4.10), and if the groups from both identifiers match, the layout metabolic identifier will be replaced with the identifier from the model.

Another available operation allows the importation of KGML layouts. These layouts can be automatically downloaded from the KEGG site, removing that workload from the user. The mapping of the identifiers in this case can also be made following two distinct methodologies. In the first, the imported KGML layout will have the KEGG identifiers as metabolic identifiers in the layout, so it is necessary to map these identifiers with others from the metabolic model. Sometimes, metabolic models already have KEGG identifiers, and in OptFlux those identifiers are stored in the "additional information" of the

Operation	Descritption
Import Layout	Importation of layout from a XGMML, SBGN, CellDesigner or COBRA layout file. This operation also has the necessary tools to map the identifiers from the model to the layout.
Import KGML Layout	Importation of a KGML layout, that can be automatically downloaded or loaded from a file on the local system.
Create Pathway Layout	Generate a pathway layout from a list of reactions, with the possibility to use another layout as base.

Table 4.1: MetabolicVisualizer4Optflux3 operations.


Import Layout

Select Project

Project: gomGG

Reader COBRA

Layout File:

Reactions Mapping

[✓ Use id mapping for reactions]

Use file

File

File delimi... comma [✓ tab] Semicolon W.Space user define

[✓ Use regexp]

Model Id Regexp R_(.+)

Layout Id Regexp (.+)

Metabolites Mapping

[✓ Use id mapping for metabolites]

Use file

File

File delimi... comma [✓ tab] Semicolon W.Space user define

[✓ Use regexp]

Model Id Regexp M_(.+)

Layout Id Regexp (.+)

Figure 4.10: Import layout operation GUI.

model. When loading the layout, it is possible to specify this case, and the mapping will be done automatically. The other way is, like in the default import operation, by loading a two-column file with the mapping.

The third operation allows the creation of layouts from reactions of a metabolic model. In this case the conversion of identifiers is not necessary, due to the fact that the identifiers come directly from the reactions of the model. The generation of this type of layouts can be made by selecting a pathway from the model, in the case that the model has that information, or by selecting a list of reactions manually. It is also possible to select an existing layout of the same project as a base for the new layout. This will allow to create new layouts or add new reactions to existing layouts (Figure 4.11).

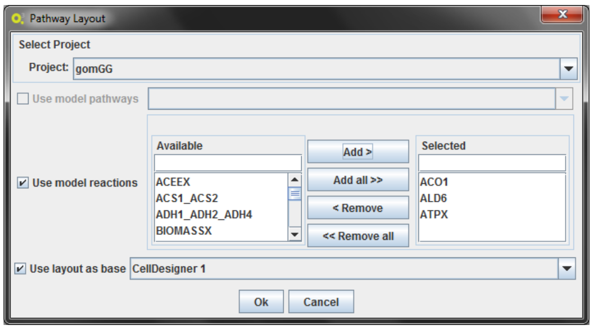


Figure 4.11: OptFlux plugin pathway layout creation operation interface. It is possible to use pathways (if the model has that information) or manually select a list of reactions. It is also possible to use an already existent layout as a base.

The most desired functionality of the connection between a metabolic engineering tool such as Optflux, and a visualization tool, is the ability to visualize phenotype simulations in the network, and hopefully, be able to use the visualization to better understand the organism, and improve it.

In order to visualize alterations in the network, the visualizer provides the filters and overlap functionalities. To allow this operation, there must be a conversion from a steady state simulation result from Optflux, to an overlap object and pass it to the visualization.

In OptFlux, simulation results have two major elements of interest for the visualization: the flux distributions and the genetic conditions. The flux distributions, as the name indicates, represent the flux values of the reactions. To represent the flux distribution in the network, first a conversion of identifiers is needed. The identifiers of the reaction nodes are specific of the visualization tool, but as mentioned before, they can have multiple metabolic identifiers associated. It can happen that two or more fluxes are present in the same reaction node, and the methodology chosen was to sum all those values. In the end, all these flux values, now mapped by reaction node identifier, are normalized to a numeric range of 1 to 10, which will be the thickness of the edges. Additionally, the labels of the reaction nodes are also changed, adding the value of the flux in front of the reaction name (Figure 4.12).

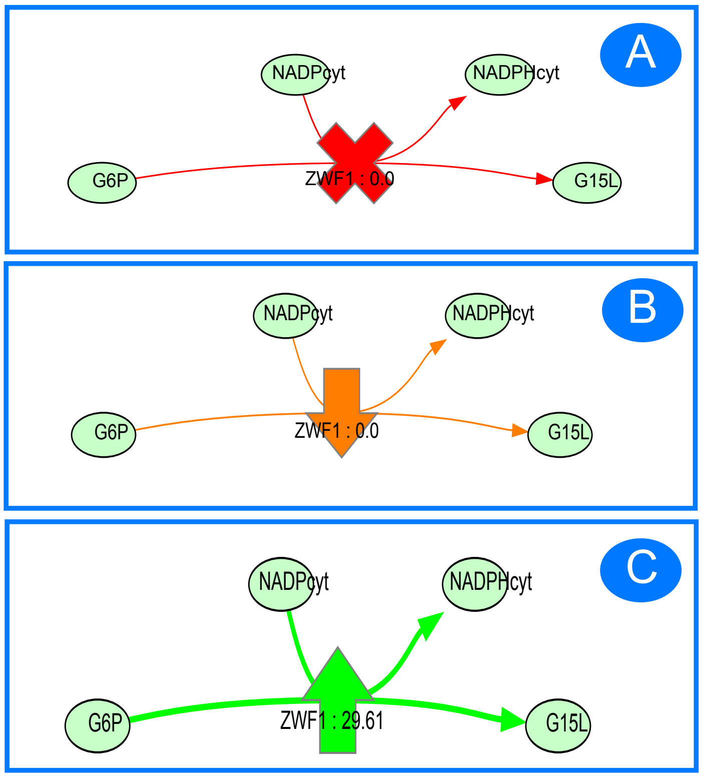


Figure 4.12: Phenotype simulation genetic conditions visual conversion: A is a knocked-out reaction; B is an under-expressed reaction; C is an over-expressed reaction.

The genetic conditions of a simulation are all genetic alterations made to the organism for that specific simulation. It contains all knock-outs, and under/over expressed reactions. For the visual representation some node shapes and colors were adopted to represent these reactions. As seen in Figure 4.12, a knocked-out reaction will have the shape of a red cross, and the reaction edges will also have a red color. An over-expressed reaction will have the shape of an arrow pointing up and with a green color for the node and the edges. Finally, an under-expressed reaction will have the shape of an arrow pointing down and an orange color to the node and edges.

Elementary Flux Modes Plugin integration

In Optflux, there is a plugin that gives the tool the capability to determine the set of Elementary Flux Modes (EFMs) of a model. This plugin also provides an interface that allows filtering the results, including the selection of EFMs based on presence/absence of external metabolites or sorting by yield. It is possible to select sets of EFMs browsing these results, and it is possible to visualize the EFMs in a column-wise table. From this point, it is possible to obtain the flux values for each EFM.

The visualization plugin converts these flux distributions to an overlap, in a similar way to the one used for the simulation results. Considering that in this case, we are dealing with simple flux distributions, only the thickness and labels of the edges are passed to the overlap, and a visual filter can be applied that hides the zero value fluxes, allowing a visualization of the EFM in the visualization tool. The overlaps obtained from EFMs are stored in the category "Flux Distributions" (Figure 4.13).

On top of this, if the model is loaded from a Cell Designer file it can be exported to Cell Designer and for each reaction in the EFM, the line in the Cell Designer layout is represented with a thickness that is proportional to the value of the flux.

4.3 Implementation details

The strategy adopted in the development of the visualization framework had the goal of creating a tool that can be used independently, but at the same

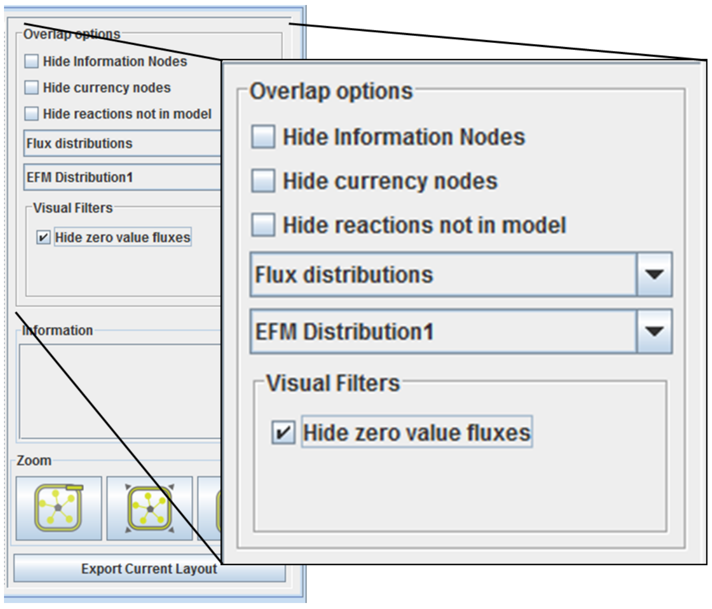


Figure 4.13: EFM flux distribution overlap in the visualization plugin.

time built in a way such that the integration with an ME tool (OptFlux in this case) was facilitated.

This brings to light the importance of the MVC (model-view-controller) design pattern. MVC's principle is to split applications into three independent components, making the replacement of any of these components an easier task [54]:

- **Model:** the data and logic of the application.
- **View:** representation of the data to the outside world.
- **Controller:** means of communication between the outside world and the model. Receives the input and passes information to the model, that may result in changing the data of the application.

Building an application following the MVC design pattern may turn out to be very time consuming. This may happen because all the three components have to be developed separately and at the same time guarantee that the communication between them is fully functional. But, at the same time, it grants the applications independence between the different components, so it is easier to update or add new functionalities.

The importance of MVC in this work was highly relevant, not only in the development of the framework, but also in the integration with OptFlux, that also follows this design principle.

4.3.1 Core Libraries

In Figure 4.14, it is possible to see how the visualization framework packages are organized. As mentioned before, there are two logic layers that deal with the reading and writing of layouts, as well as visually representing them. A third layer is related to the GUI and user interaction.

The metabolic layout is defined by the interface `ILayout`, specifying that classes implementing it should include the methods `getReactions()` and `getNodes()` (metabolites, currency and information). All implementations must, therefore, have a built layout with all the nodes and reactions. The class `LayoutContainer` is such an implementation, and as the name indicates it is a container for a layout. The `CellDesignerContainer` is an extension of the `LayoutContainer` keeping the structure of the CD-SBML for exportation purposes. In a later phase of development, other methods were added to the `ILayout` interface that are related with the edition of the layout. The metabolites and reactions nodes are also defined by two interfaces: `INodeLay` and `IReactionLay`, that define the rules that were presented in the previous section.

The class `LayoutVisualizer` is the responsible for the generation of the visualization of the `ILayout`. The visualization is provided by the *Prefuse* packages as it will be explained in detail in the next section. This class builds a table containing all the necessary information on the graph, and an object of the class `Visualization` is built from this table.

All overlaps and filters are processed over this table, which will result in a

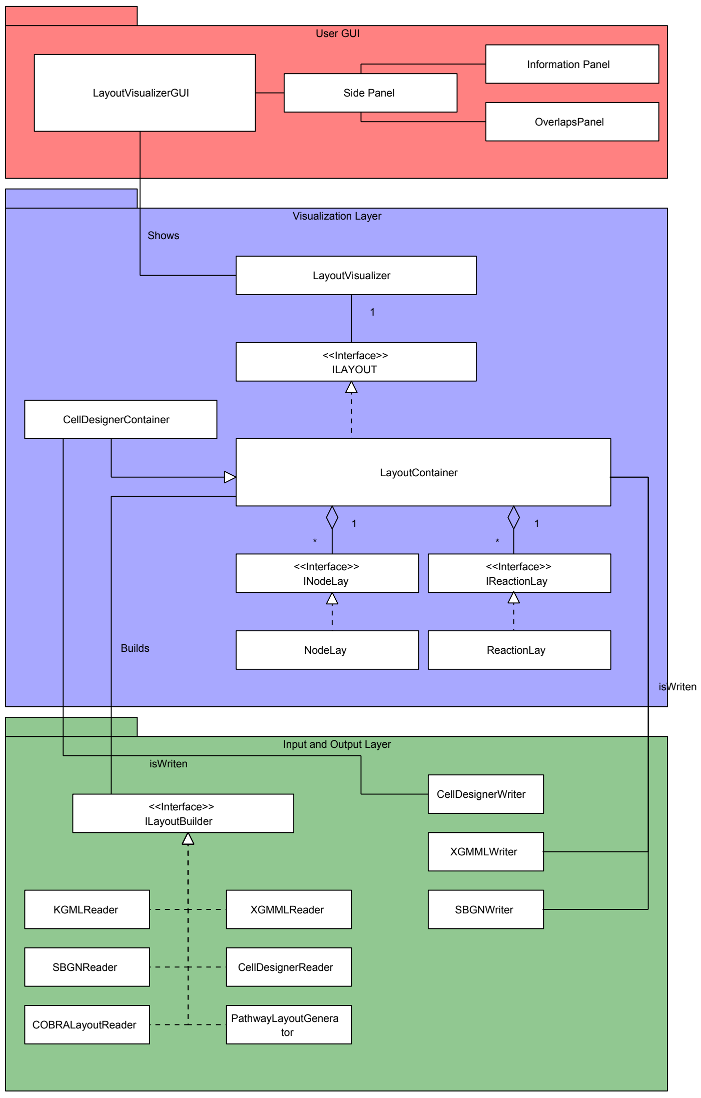


Figure 4.14: Visualization framework core packages class diagram (using UML).

change in the visualization. This table is managed by the `LayoutVisualizer` and can be accessed by the interface in several ways. The interface is the bridge between the user and the `LayoutVisualizer` class. The central element of the interface is the `LayoutVisualizerGUI` class, that provides the basic interface of the visualizer, with additional information and actions provided by the other elements, the `InformationPanel` and `OverlapPanel`.

The `InformationPanel` is an abstract class that extends a `JPanel` and implements the methods `showNodeInformation()` and `showReactionInformation()`. This allows for advanced users to develop their own information panels, with node and reaction information for specific needs.

Importation and exportation of `ILayouts` are made by the respective readers and writers. All the readers implement the interface `ILayoutBuilder`, that only has one method, `buildLayout()`. This was made to facilitate adding support for new formats. `CellDesignerContainer` is exported in a different way, due to the fact that the CellDesigner SBML must be preserved. To read and write the CellDesigner SBML file the JSBML was used, a Java library for reading, writing, and manipulating SBML files and data streams [55]. It is also possible to export the layouts as images to PDF and SVG formats directly from the visualizer. This was accomplished using the libraries Batik Java SVG Toolkit (`http://xmlgraphics.apache.org/batik/`) and VectorGraphics2D (`http://trac.erichseifert.de/vectorgraphics2d/`).

4.3.2 Implementing the visualization layer with prefuse

The Prefuse software framework (`www.prefuse.org`) was created for the development of interactive visualization applications, using the Java programming language. It simplifies the process of visually representing and manipulating data. With Prefuse, it is possible to map data using their spatial position, size, shape, color, etc, while being able to directly manipulate the visualized data.

The design of the Prefuse toolkit is based upon the information visualization reference model[56], depicted in Figure 4.15.

In the Figure 4.15 the first step of the reference model is not mentioned (the Source Data), because in this work it was not used. The reading and writing of data is handled by the Input and Output layer described previously.

For this work, the first step that used Prefuse is the construction of the data tables. The Prefuse packages provide a `Graph` data structure for represent-

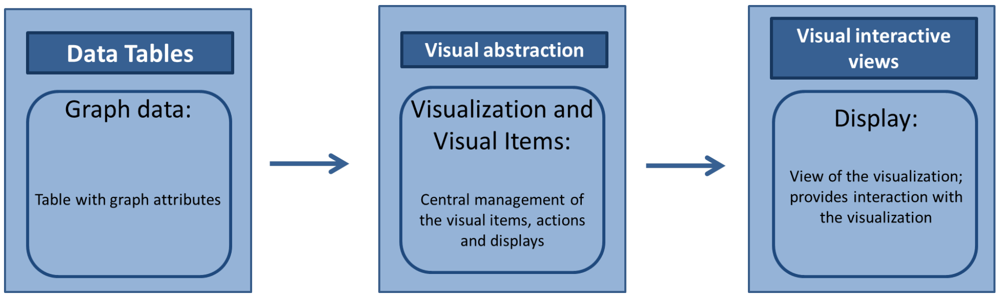

Data Tables
Graph data:
Table with graph attributes
Visual abstraction
Visualization and Visual Items:
Central management of the visual items, actions and displays
Visual interactive views
Display:
View of the visualization; provides interaction with the visualization

Figure 4.15: Prefuse toolkit overview.

ing data, providing the data tables of the reference model. Table rows are represented by the `Tuple` class, while the `Node` and `Edge` classes represent the entities of the graph. The Graph class is implemented using a `Table` instance to store the node and edge data.

After building the data tables, and following the reference model, it is necessary to create a visual abstraction, that will contain all the necessary information needed to draw the graph. This visual abstraction of a data set can be created by adding the data to the Prefuse `Visualization` class. This class is a data structure that has the original data provided by the tables, but also visualization-specific fields such as coordinates, color, size and font values. For every node and edge of the table an object is created of the class `VisualItem`, providing access to the original data of the table, as well as the newly-added visual fields.

The last step is the actual rendering of the defined structures through the views that provide the means of interaction with the user.

4.3.3 Implementing the OptFlux plugin

To understand the development of the OptFlux plugin it is first necessary to understand how Optflux is organized. Optflux is built on top of AIBench (`http://www.aibench.org/`), a software development framework developed by researchers from the University of Vigo in Spain.

AIBench follows the previously described MVC design pattern, and comes

Datatypes and Views		
Datatypes	LayoutBox	Datatype that stores the layout.
	CellDesignerLayoutBox	Datatype that stores a CellDesigner layout.
Views	PathwayView	View that displays the layout in Optflux.

Table 4.2: MetabolicVisualizer4Optflux3 datatypes and views.

as an answer to the effort that takes developing applications following MVC. Indeed, it provides the abstraction that was also described earlier, defining the connection between the components of the MVC. Thus, AIBench provides an easier way to develop user-friendly applications, and Optflux is such an example.

The three operations defined for the plugin were already described previously, since they represent the features that the plugin offers. Their results are stored in the datatypes developed for the plugin: the `LayoutBox` that, as the name indicates, is a datatype that has a `LayoutContainer`, and the `CellDesignerLayoutBox` that works similarly to the `LayoutBox` but for `CellDesignerContainer`. This was implemented in this way due to fact that OptFlux needs to serialize its datatypes to keep the workspace functional. Since the `CellDesignerContainer` is different in structure from the other container (it keeps the structure of the CellDesigner-SBML file), a differentiation is necessary in the serialization processes provided by the two datatypes. This allows both datatypes to be displayed in the same view, the `PathwayLayoutView`. The summary of the datatypes and views is shown in Table 4.2.

Another crucial point in the development of the plugin, was the mapping of the metabolic model's entities and the ones in the layout.

The basis of the implementation of the operations in OptFlux, is the `Metabolic` library. Its packages implement all the ME methods and algorithms used in

OptFlux. It contains the implementation of methods for phenotype simulation such as FBA, pFBA, MOMA and ROOM, as well as strain optimization methods together with many other features for ME. It also contains all structures used to represent metabolic models, and reading/writing files in different formats.

OptFlux's datatype for simulation results is the `SteadyStateSimulation-ResultBox`, that contains a class from the Metabolic packages named `Steady-StateSimulationResult`, and from this class it is possible to access all the necessary information of a specific simulation.

In a previous section, it was mentioned that converting a simulation result to an overlap required two key elements: the flux values and the genetic conditions that describe the genetic alterations made to the organism. The flux distribution is represented by the class `FluxValueMap`, that as the name indicates, contains the simulation flux values.

The `GeneticConditions` class represent all genetic alterations made to the organism for that specific simulation. It is possible to access the reactions affected by these changes as a list of pairs of values. Each pair has the identifier of the reaction and an associated value. If this value is 0, than that reaction is knocked-out, if it is between 0 and 1, than the reaction is under-expressed and if the value is higher than 1 then that reaction is over-expressed.

The conversion of overlaps for EFMs distributions is simpler. The EFM distributions implement an interface called `IFluxValueContainer` that has only one method that returns a `FluxValueMap`. The conversion is similar to the one of the fluxes of a simulation result, explained above.

Chapter 5

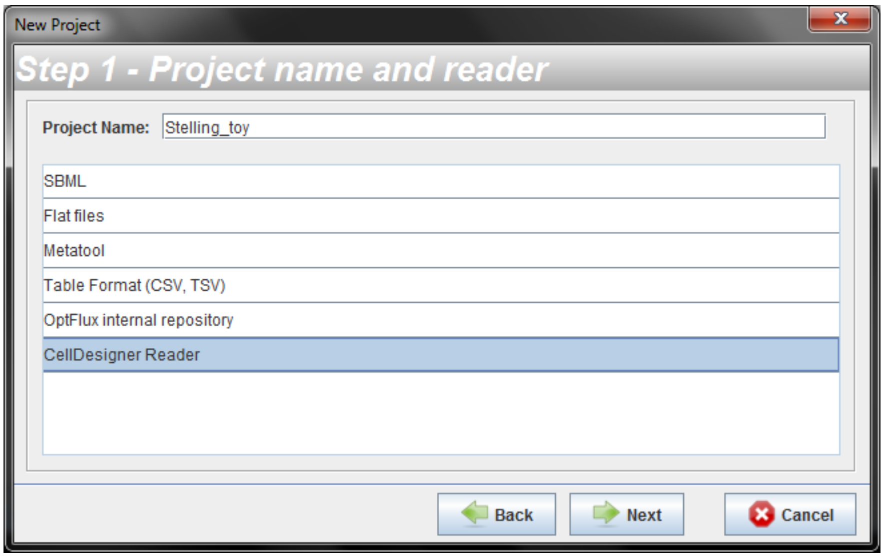
Case Study

5.1 Case study I

For the first use case a simple model was chosen to show some of the features from the visualization framework integrated with OptFlux. The chosen model was an example from Z.Szallasi et al book [57]. The model comes with a CellDesigner SBML, so it is possible to load it using the reader from the wizard in OptFlux (Figure 5.1).

The model is composed of 10 reactions and 10 metabolites. In order to simulate it, it is necessary to change the bounds of the drains, so substrates can enter the cell. This is done by defining environmental conditions in OptFlux. In this way, the reactions R1 and R2 were open, and it is possible to perform simulations with this model.

In Figure 5.2, it is possible to see several of the possible instances of the model layout available from the operations of the the visualization plugin. Figure 5.2 A shows the model layout, B shows the model overlapped with a wild-type simulation, with the environmental conditions described previously. It is possible to see that the sum of the quantities of substrate that enter the model are the same that are produced. C shows the model overlapped with a simulation with a knockout to the reaction R2, and this means that half the


New Project

Step 1 - Project name and reader

Project Name: Stelling_toy

SBML
Flat files
Metatool
Table Format (CSV, TSV)
OptFlux internal repository
CellDesigner Reader

Figure 5.1: OptFlux visualization plugin reader, in the OptFlux new project wizard readers list.

quantity of substrates will enter the network, resulting in half the products. It is also possible to see that the R2 reaction changes shape according to the genetic conditions of the simulation. Finally Figure 5.2 D shows an EFM flux distribution, calculated with the EFM plugin described previously.

5.2 Case study II

Succinic acid is a member of the C4-dicarboxylic acid family and plays a role in the citric acid cycle, an energy-yielding process. It is used as precursor to a wide range of products, from pharmaceuticals, to polyesters, and even on food and beverage industry as an acidity regulator[58].

Succinic acid has been produced by chemical processes, but due to pollution problems there has been an effort to use microbial fermentation processes with anaerobic bacteria [59]. Optimizing micro-organisms to over-produce succinic acid, is one goal of interest for ME researchers.

For this case study, the core *E. coli* metabolic model was used. This model is

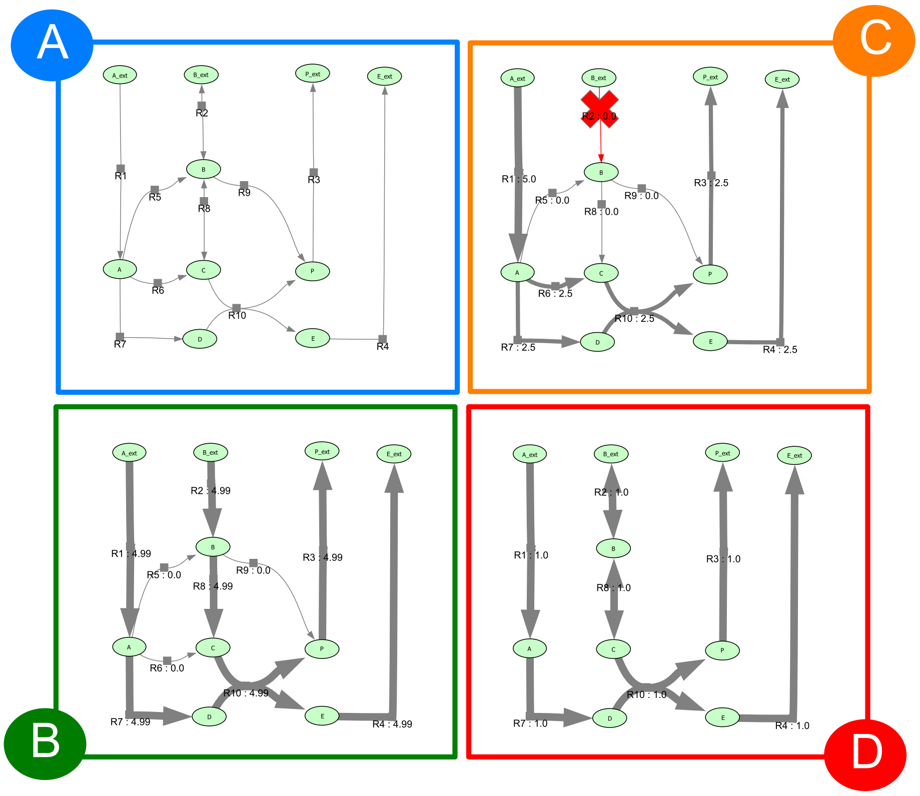
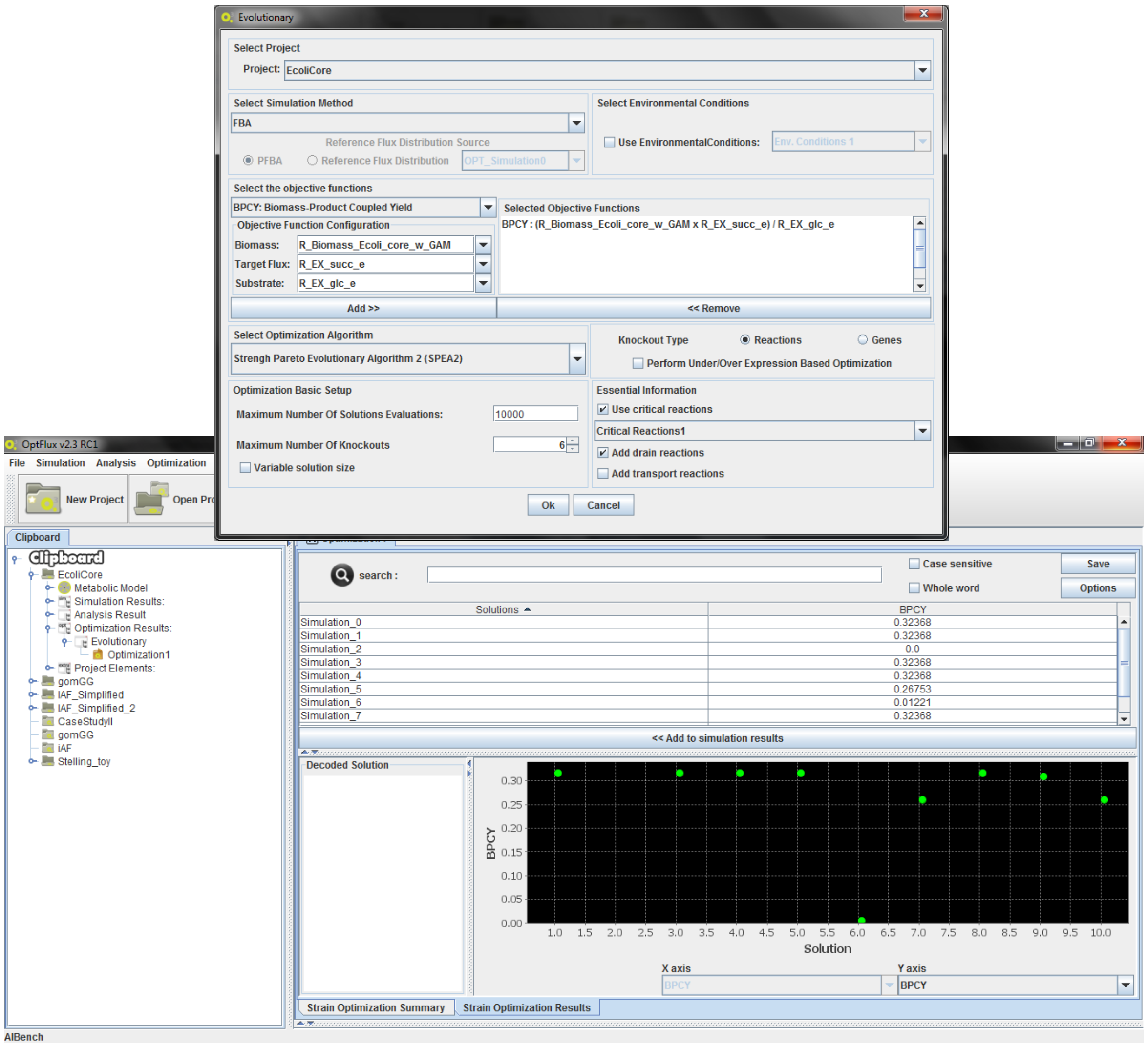


Figure 5.2: Examples of operations with the visualization plugin: A is the layout view; B is a wild-type phenotype simulation; C - is a simulation with a reaction knock-out; D is a EFM with the zero value fluxes filter.

a subset of the genome-scale metabolic reconstruction iAF1260 and it is used in the EcoSal chapter of the book by Orth et al [60]. The model is available for download directly from OptFlux's internal repository. This model has 95 reactions (20 external and 75 internal), 72 metabolites (20 external, 52 internal) and 137 genes with 95 gene rules.

For visualization purposes, a COBRA layout was loaded for this particular model, available at BiGG Database (`http://bigg.ucsd.edu/`). The first problem appeared with the identifiers of the model. With the application of regular expressions using the OptFlux plugin interface the majority of the identifiers were mapped. Still, some of the identifiers were not mapped. To solve this, the layout was exported to XGMML, and those identifiers were corrected manually. With this new layout, a second version was available and able to be fully mapped to the *E. Coli* Core Model.

The next step was to run a knock-out optimization, using the optimization capabilities that OptFlux offers. The optimization was set up with the production of succinate as a target, with glucose as substrate. The optimization algorithm used was the Strength Pareto Evolutionary Algorithm 2 [61] (SPEA2), with 10000 evaluations. A set of solutions were obtained (Figure 5.3), being possible to add the simulation results of these solutions to the OptFlux clipboard.


Evolutionary

Select Project
Project: EcoliCore

Select Simulation Method: FBA
Reference Flux Distribution Source: [✓ PFBA] Reference Flux Distribution OPT_Simulation0

Select Environmental Conditions
☐ Use EnvironmentalConditions: Env. Conditions 1

Select the objective functions: BPCY: Biomass-Product Coupled Yield
Objective Function Configuration
Biomass: R_Biomass_Ecoli_core_w_GAM
Target Flux: R_EX_succ_e
Substrate: R_EX_glc_e

Selected Objective Functions
BPCY : (R_Biomass_Ecoli_core_w_GAM x R_EX_succ_e) / R_EX_glc_e

Select Optimization Algorithm: Strengh Pareto Evolutionary Algorithm 2 (SPEA2)
Knockout Type: [✓ Reactions] Genes
☐ Perform Under/Over Expression Based Optimization

Optimization Basic Setup
Maximum Number Of Solutions Evaluations: 10000
Maximum Number Of Knockouts: 6
☐ Variable solution size

Essential Information
☑ Use critical reactions
Critical Reactions1
☑ Add drain reactions
☐ Add transport reactions

Solutions	BPCY
Simulation_0	0.32368
Simulation_1	0.32368
Simulation_2	0.0
Simulation_3	0.32368
Simulation_4	0.32368
Simulation_5	0.26753
Simulation_6	0.01221
Simulation_7	0.32368

X axis: BPCY
Y axis: BPCY

Figure 5.3: OptFlux optimization setup and results.

All solutions, except Simulation 2 show results for succinate production. From these there are five that have better results than the others: Simulation 0, 1, 3, 4 and 7. Since the goal of the optimization was to maximize the succinate production the focus on the visualization analysis shall be on these 5 solutions (Table 5.1).

Solution	BPCY	Succinate
Simulation 0	0.3237	5.6347
Simulation 1	0.3237	5.6347
Simulation 2	0.0	—
Simulation 3	0.3237	5.6347
Simulation 4	0.3237	5.6347
Simulation 5	0.2675	4.0795
Simulation 6	0.0122	0.67815
Simulation 7	0.3237	5.6347
Simulation 8	0.3161	5.3528
Simulation 9	0.2675	4.0795

Table 5.1: Succinate optimization results.

All these five solutions have the same values for BPCY, biomass and succinate production. By checking the knock-outs for each solution we can see that they are very similar (Table 5.2), and this can mean that they can be instances of the same solution, i.e. the same overall flux distribution. This can happen because when we face chains of linear reactions, knocking-out any of the reactions from the chain leads to the same result, in terms of fluxes, although the solution for the optimization algorithm (knock-out set) is different. This means that if we look at the small variations of these solutions, we can begin to understand how they improve the production of succinate. To address this task the visualization plugin was used, allowing to visualize the different solutions overlapping the original layout of the model.

Figure 5.4 represents the *E. coli* core layout loaded in the visualization framework. Over it is possible to overlap the simulations one by one to see the differences between them.

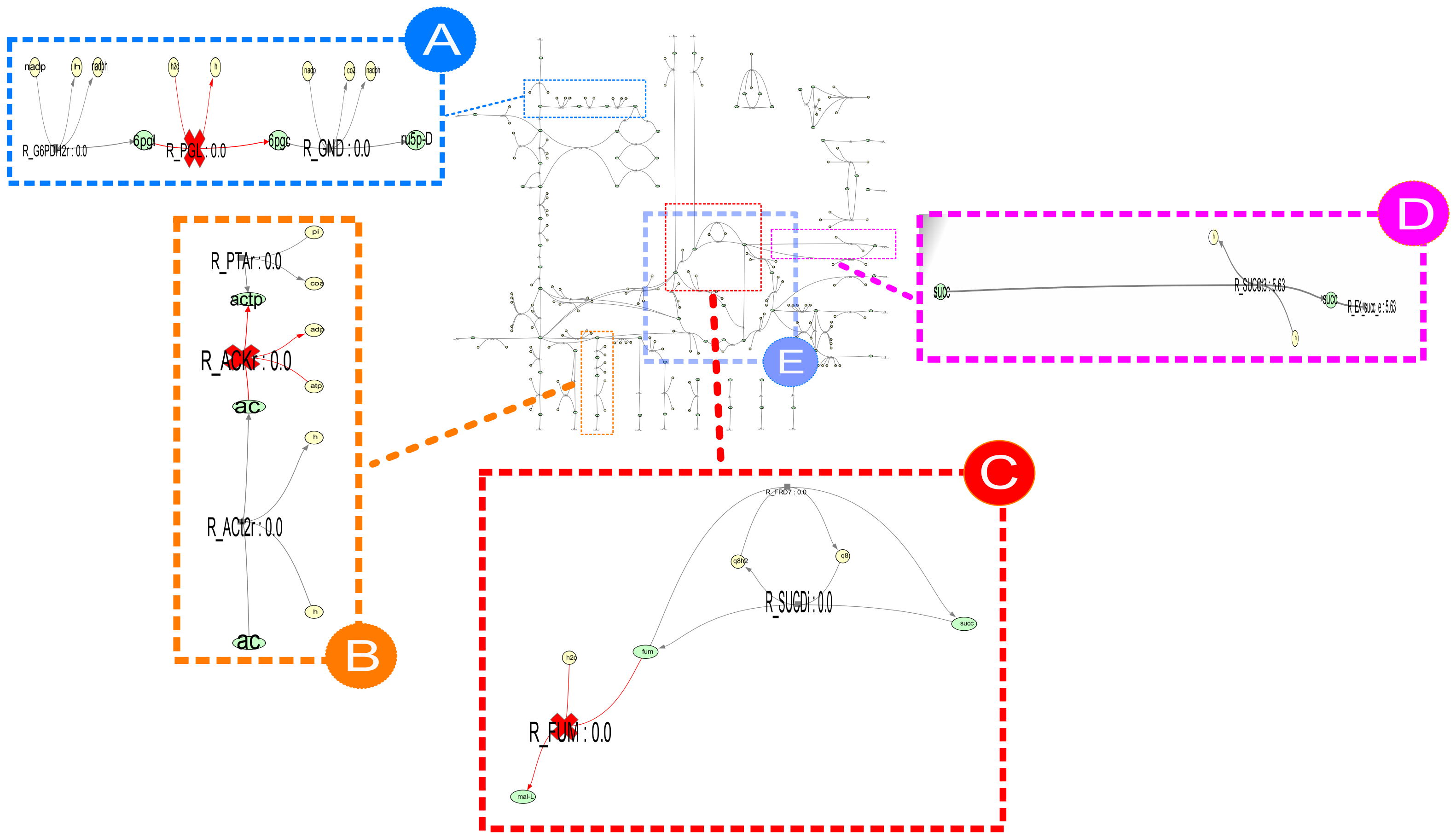


Figure 5.4: *E. Coli* core layout in the visualization. All the six simulation knock-outs are present in the image. Knock-outs to force flux to go to the TCA-cycle (E) are in A and C; B are knock-outs to prevent succinate consumption; D is the production of succinate.

Solution	Knock-outs
Simulation 0	R-PGL, R-SUCDi, R-ACKr
Simulation 1	R-GND, R-SUCDi, R-ACKr
Simulation 3	R-G6PDH2r, R-SUCDi, R-ACKr
Simulation 4	R-PGL, R-FUM, R-ACKr
Simulation 7	R-G6PDH2R, R-FUM, R-ACKr

Table 5.2: Knock-out sets obtained in the best solutions from the optimization.

In all the solutions, a knock out appears in one of the reactions R-G6PDH2r, R-PGL or R-GND (Fig. 5.4 A) and on the reaction R-ACKr. This seems to happen to direct more flux to the TCA-cycle (Figure 5.4 E). Another thing in common in all 6 solutions is a knock-out in the TCA-cycle in the R-FUM or R-SUCDi reactions (Fig. 5.4 B). This happens because the reaction R-SUCDi consumes succinate, and by knocking-out one of these reactions there is a greater accumulation of succinate in the network, directed later to the succinate drain (Figure 5.4 D).

Analysing the network and overlapping the solutions facilitates the understanding of the basic principles behind the solutions obtained in the strain optimization. The genetic modifications are made to maximize the influx on the TCA-Cycle, and removing the reactions that lead to succinate consumption results in an increase of accumulation of this compound.

Chapter 6

Conclusions and Future Work

6.1 Discussion

In the last years, the problem of visualizing biological networks has been tackled by several available tools. More particularly, metabolic networks have been addressed by some of these tools, and many network analysis tools are used to study the structure and properties of these networks. However, integration with ME tools and/or phenotype simulation methods is not very common. Usually, phenotype simulations results or flux distributions can be visualized by using a third-party software, even though there are good examples, like COBRA, that already integrate these two features.

In this work, a metabolic network visualization framework was presented. This framework has the ability to load networks from a variety of formats and display those networks using a layout. It provides features of creation and edition of these layouts, as well as exportation capabilities. On top of this it is possible to overlap the network with visual changes, a functionality that allows, for instance, to visualize fluxes in phenotype simulations.

The framework was integrated with OptFlux, by the development of a plugin. This allows ME researchers to use the visualization directly from OptFlux, and use a series of operations that will allow loading and exporting layouts with a user-friendly interface.

This framework presents itself as an useful tool that can help researchers involved in ME projects to have a way of integrating the visualization of the metabolic networks they are studying easily. The ability to dynamically visualize phenotype simulations is also an important asset.

This framework, and the combination of visualization with ME simulation and optimization processes, will help researchers to achieve knowledge about the structure and functioning of organisms of interest that was not available before.

The visualization framework and the plugin were developed in JAVA and are a part of the development effort of the Bioinformatics and Sistems Biology research group of the University of Minho, integrating researchers from the Computer Science and Technology Center (CCTC), the University of Minho research unit in Informatics, and the Center of Biological Engineering (CEB).

The work developed in this thesis was partially published in the 7th International Conference on Pratical Applications of Computational Biology and Bioinformatics (PACBB) [62], while an extended article is under preparation for a journal.

6.2 Future Work

The goals proposed in this work were generally accomplished, but there are some points that can be improved in the future. Some are given next:

- to develop ways of comparing several simulations in the same overlap, instead of only being able to visualize one simulation result at a time. This would drastically improve the analysis capabilities of the framework;
- find an improved way to map the layouts with the metabolic model, or at least, develop a more effective and easier way for the user to make

that mapping;

- add different types of reactions, for instance, support for simplified reactions, where it would be possible to simplify a layout by merging equivalent reactions and returning to the initial mode if wanted;
- adding support for regulation: there are currently efforts on integrating regulatory models with metabolic models, and visualization support for those would also be a good asset.

Bibliography

[1] Segre D, Vitkup D, and Church GM. *Analysis of optimality in natural and perturbed metabolic networks.* Proceedings of the National Academy of Sciences, 99(23), 15112–15117, National Acad Sciences, 2002.

[2] Suderman M and Hallett M. *Tools for visually exploring biological networks.* Bioinformatics, 23(20), 2651–2659, Oxford Univ Press, 2007.

[3] Le Novere N, Hucka M, Mi H, Moodie S, Schreiber F, Sorokin A, Demir E, Wegner K, Aladjem MI, Wimalaratne SM, *et al. The systems biology graphical notation.* Nature biotechnology, 27(8), 735–741, Nature Publishing Group, 2009.

[4] Azuaje F and Dopazo J. *Data analysis and visualization in genomics and proteomics.* John Wiley & Sons, 2005.

[5] Kitano H *et al. Foundations of systems biology.* MIT press Cambridge, MA, 2001.

[6] Patil KR, Åkesson M, and Nielsen J. *Use of genome-scale microbial models for metabolic engineering.* Current opinion in biotechnology, 15(1), 64–69, Elsevier, 2004.

[7] Llaneras F and Picó J. *Stoichiometric modelling of cell metabolism.* Journal of Bioscience and Bioengineering, 105(1), 1–11, Elsevier, 2008.

[8] Ostergaard S, Olsson L, and Nielsen J. *Metabolic engineering of Saccharomyces cerevisiae.* Microbiology and Molecular Biology Reviews, 64(1), 34–50, Am Soc Microbiol, 2000.

[9] Stephanopoulos G, Aristidou AA, and Nielsen J. *Metabolic engineering: principles and methodologies.* Academic Press, 1998.

[10] Wiback SJ, Famili I, Greenberg HJ, and Palsson BØ. *Monte Carlo sampling can be used to determine the size and shape of the steady-state flux space.* Journal of theoretical biology, 228(4), 437–447, Elsevier, 2004.

[11] Llaneras F and Picó J. *An interval approach for dealing with flux distributions and elementary modes activity patterns.* Journal of theoretical biology, 246(2), 290–308, Elsevier, 2007.

[12] Gagneur J and Klamt S. *Computation of elementary modes: a unifying framework and the new binary approach.* BMC bioinformatics, 5(1), 175, BioMed Central Ltd, 2004.

[13] Papin JA, Stelling J, Price ND, Klamt S, Schuster S, and Palsson BO. *Comparison of network-based pathway analysis methods.* Trends in biotechnology, 22(8), 400–405, Elsevier, 2004.

[14] Kauffman KJ, Prakash P, and Edwards JS. *Advances in flux balance analysis.* Current opinion in biotechnology, 14(5), 491–496, Elsevier, 2003.

[15] Shlomi T, Berkman O, and Ruppin E. *Regulatory on/off minimization of metabolic flux changes after genetic perturbations.* Proceedings of the National Academy of Sciences of the United States of America, 102(21), 7695–7700, National Acad Sciences, 2005.

[16] Rocha I, Maia P, Evangelista P, Vilaça P, Soares S, Pinto J, Nielsen J, Patil K, Ferreira E, and Rocha M. *OptFlux: an open-source software platform for in silico metabolic engineering.* BMC systems biology, 4(1), 45, BioMed Central Ltd, 2010.

[17] Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, *et al.* *The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models.* Bioinformatics, 19(4), 524–531, Oxford Univ Press, 2003.

[18] Becker SA, Feist AM, Mo ML, Hannum G, Palsson BØ, and Herrgard MJ. *Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox.* Nature protocols, 2(3), 727–738, Nature Publishing Group, 2007.

[19] Schellenberger J, Que R, Fleming RM, Thiele I, Orth JD, Feist AM, Zielinski DC, Bordbar A, Lewis NE, Rahmanian S, *et al.* *Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2. 0.* Nature protocols, 6(9), 1290–1307, Nature Publishing Group, 2011.

[20] Burgard AP, Pharkya P, and Maranas CD. *Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization.* Biotechnology and bioengineering, 84(6), 647–657, Wiley Online Library, 2003.

[21] Patil KR, Rocha I, Förster J, and Nielsen J. *Evolutionary programming as a platform for in silico metabolic engineering.* BMC bioinformatics, 6(1), 308, BioMed Central Ltd, 2005.

[22] Hoppe A, Hoffmann S, Gerasch A, Gille C, and Holzhütter HG. *FASIMU: flexible software for flux-balance computation series in large metabolic networks.* BMC bioinformatics, 12(1), 28, BioMed Central Ltd, 2011.

[23] Klamt S, Saez-Rodriguez J, and Gilles ED. *Structural and functional analysis of cellular networks with CellNetAnalyzer.* BMC systems biology, 1(1), 2, BioMed Central Ltd, 2007.

[24] König M and Holtzhünter HG. *Fluxviz: cytoscape plug-in for visualization of flux distributions in networks.* In *Genome Informatics 2010: The 10th Annual International Workshop on Bioinformatics and Systems Biology (IBSB 2010): Kyoto University, Japan, 26-28 July 2010*, 24, page 96. World Scientific, 2010.

[25] Cline MS, Smoot M, Cerami E, Kuchinsky A, Landys N, Workman C, Christmas R, Avila-Campilo I, Creech M, Gross B, *et al.* *Integration of biological networks and gene expression data using Cytoscape.* Nature protocols, 2(10), 2366–2382, Nature Publishing Group, 2007.

[26] Junker BH and Schreiber F. *Analysis of biological networks*, volume 2. John Wiley & Sons, 2008.

[27] Brandes U. *A faster algorithm for betweenness centrality*.* Journal of Mathematical Sociology, 25(2), 163–177, Taylor & Francis, 2001.

[28] Newman ME. *A measure of betweenness centrality based on random walks.* Social networks, 27(1), 39–54, Elsevier, 2005.

[29] Mason O and Verwoerd M. *Graph theory and networks in biology.* Systems Biology, IET, 1(2), 89–119, IET, 2007.

[30] Freeman LC. *A set of measures of centrality based on betweenness.* Sociometry, pages 35–41, JSTOR, 1977.

[31] Girvan M and Newman ME. *Community structure in social and biological networks.* Proceedings of the National Academy of Sciences, 99(12), 7821–7826, National Acad Sciences, 2002.

[32] MacNeil LT and Walhout AJ. *Gene regulatory networks and the role of robustness and stochasticity in the control of gene expression.* Genome research, 21(5), 645–657, Cold Spring Harbor Lab, 2011.

[33] Sauro HM and Kholodenko BN. *Quantitative analysis of signaling networks*, volume 86. Elsevier, 2004.

[34] Barabási AL and Oltvai ZN. *Network biology: understanding the cell's functional organization.* Nature Reviews Genetics, 5(2), 101–113, Nature Publishing Group, 2004.

[35] Albert R, Jeong H, and Barabási AL. *Error and attack tolerance of complex networks.* Nature, 406(6794), 378–382, Nature Publishing Group, 2000.

[36] Fruchterman TM and Reingold EM. *Graph drawing by force-directed placement.* Software: Practice and experience, 21(11), 1129–1164, Wiley Online Library, 1991.

[37] Sugiyama K, Tagawa S, and Toda M. *Methods for visual understanding of hierarchical system structures.* Systems, Man and Cybernetics, IEEE Transactions on, 11(2), 109–125, IEEE, 1981.

[38] Schwikowski B, Uetz P, and Fields S. *A network of protein–protein interactions in yeast.* Nature biotechnology, 18(12), 1257–1261, Nature Publishing Group, 2000.

[39] Murzin AG, Brenner SE, Hubbard T, and Chothia C. *SCOP: a structural classification of proteins database for the investigation of sequences and structures.* Journal of molecular biology, 247(4), 536–540, Elsevier, 1995.

[40] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, *et al.* *Gene Ontology:*

tool for the unification of biology. Nature genetics, 25(1), 25–29, Nature Publishing Group, 2000.

[41] Funahashi A, Morohashi M, Kitano H, and Tanimura N. *CellDesigner: a process diagram editor for gene-regulatory and biochemical networks.* Biosilico, 1(5), 159–162, Elsevier, 2003.

[42] Funahashi A, Matsuoka Y, Jouraku A, Morohashi M, Kikuchi N, and Kitano H. *CellDesigner 3.5: a versatile modeling tool for biochemical networks.* Proceedings of the IEEE, 96(8), 1254–1265, IEEE, 2008.

[43] Smoot ME, Ono K, Ruscheinski J, Wang PL, and Ideker T. *Cytoscape 2.8: new features for data integration and network visualization.* Bioinformatics, 27(3), 431–432, Oxford Univ Press, 2011.

[44] Scardoni G, Petterlini M, and Laudanna C. *Analyzing biological network parameters with CentiScaPe.* Bioinformatics, 25(21), 2857–2859, Oxford Univ Press, 2009.

[45] Assenov Y, Ramírez F, Schelhorn SE, Lengauer T, and Albrecht M. *Computing topological parameters of biological networks.* Bioinformatics, 24(2), 282–284, Oxford Univ Press, 2008.

[46] Bader GD and Hogue CW. *An automated method for finding molecular complexes in large protein interaction networks.* BMC bioinformatics, 4(1), 2, BioMed Central Ltd, 2003.

[47] König M, Dräger A, and Holzhütter HG. *CySBML: a Cytoscape plugin for SBML.* Bioinformatics, 28(18), 2402–2403, Oxford Univ Press, 2012.

[48] Kitano H. *A graphical notation for biochemical networks.* Biosilico, 1(5), 169–176, Elsevier, 2003.

[49] Junker BH, Klukas C, and Schreiber F. *VANTED: a system for advanced data analysis and visualization in the context of biological networks.* BMC bioinformatics, 7(1), 109, BioMed Central Ltd, 2006.

[50] Demir E, Babur O, Dogrusoz U, Gursoy A, Nisanci G, Cetin-Atalay R, and Ozturk M. *PATIKA: an integrated visual environment for collaborative construction and analysis of cellular pathways.* Bioinformatics, 18(7), 996–1003, Oxford Univ Press, 2002.

[51] Dogrusoz U, Erson E, Giral E, Demir E, Babur O, Cetintas A, and Colak R. *PATIKAweb: a Web interface for analyzing biological pathways*

through advanced querying and visualization. Bioinformatics, 22(3), 374–375, Oxford Univ Press, 2006.

[52] Kitano H, Funahashi A, Matsuoka Y, and Oda K. *Using process diagrams for the graphical representation of biological networks.* Nature biotechnology, 23(8), 961–966, Nature Publishing Group, 2005.

[53] Kanehisa M, Goto S, Sato Y, Furumichi M, and Tanabe M. *KEGG for integration and interpretation of large-scale molecular data sets.* Nucleic acids research, 40(1), 109–114, Oxford Univ Press, 2012.

[54] Applying MVC in VisualAge for Java.
`http://javadude.com/articles/vaddmvc1/mvc1.htm`.

[55] Dräger A, Rodriguez N, Dumousseau M, Dörr A, Wrzodek C, Le Novère N, Zell A, and Hucka M. *JSBML: a flexible Java library for working with SBML.* Bioinformatics, 27(15), 2167–2168, Oxford Univ Press, 2011.

[56] Chi EHH. *A framework for information visualization spreadsheets.* Ph.D. thesis, Citeseer - University of Minnesota, 1999.

[57] Szallasi Z, Stelling J, and Periwal V. *System modelling in cellular biology.* MIT Press, Cambridge, MA, 2006.

[58] Zeikus J, Jain M, and Elankovan P. *Biotechnology of succinic acid production and markets for derived industrial products.* Applied Microbiology and Biotechnology, 51(5), 545–552, Springer, 1999.

[59] Lee PC, Lee WG, Lee SY, Chang HN, and Chang YK. *Fermentative production of succinic acid from glucose and corn steep liquor byAnaerobiospirillum succiniciproducens.* Biotechnology and bioprocess engineering, 5(5), 379–381, Springer, 2000.

[60] Reconstruction and Use of Microbial Metabolic Networks: the Core Escherichia coli Metabolic Model as an Educational Guide.
`http://ecosal.org/`.

[61] Zitzler E, Laumanns M, Thiele L, Zitzler E, Zitzler E, Thiele L, and Thiele L. *SPEA2: Improving the strength Pareto evolutionary algorithm.* pages 1–21, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001.

[62] Noronha A, Vilaça P, and Rocha M. *Network Visualization Tools to Enhance Metabolic Engineering Platforms.* In *7th International Conference*

on Practical Applications of Computational Biology & Bioinformatics, pages 137–144. Springer, 2013.