Enumerating Functional Substructures of Genome-Scale Metabolic Networks: Stories, Precursors and Organisations

Paulo Vieira Milreu

N^o d'ordre: 267-2012 Année 2012

Thèse

Présentée

devant l'Université Claude Bernard - Lyon 1

pour l'obtention

du Diplôme de Doctorat

(arrêté du 7 août 2006)

et soutenue publiquement le
19 Décembre 2012

par

Paulo Vieira Milreu

Enumerating Functional Substructures of Genome-Scale Metabolic Networks: Stories, Precursors and Organisations

---

Directeur de thèse: Marie-France Sagot
Co-Directeur: Christian Gautier
Co-Encadrant: Vincent Lacroix

Jury:	Ludovic Cottret,	Examinateur
	Pierluigi Crescenzi,	Examinateur
	Joseph Heijnen	Rapporteur
	Steffen Klamt,	Rapporteur
	Alain Viari,	Président
	Peter Widmayer	Rapporteur

UNIVERSITÉ CLAUDE BERNARD-LYON 1

Président de l'Université	**M. François-Noël GILLY**
Vice-Président du Conseil d'Administration	M. le Professeur Hamda BEN HADID
Vice-Président du Conseil des Etudes et de la Vie Universitaire	M. le Professeur Philippe LALLE
Vice-Président du Conseil Scientifique	M. le Professeur Germain GILLET
Secrétaire Général	M. Alain HELLEU

SECTEUR SANTÉ

Composantes

Faculté de Médecin Lyon-Est - Claude Bernard	Directeur: M. le Professeur J. ETIENNE
Faculté de Médecine et de Maïeutique Lyon Sud - Charles Mérieux	Adm. prov.: M. le Professeur G. KIRKORIAN
UFR d'Ontologie	Directeur: D. BOURGEOIS
Institut des Sciences Pharmaceutiques et Biologiques	Directeur: Mme. la Professeure C. VINCIGUERRA
Institut Techniques de Réadaptation	Directeur: M. le Professeur Y. MATILLON
Département de Formation et Centre de Recherche en Biologie Humaine	Directeur: M. le Professeur P. FARGE

SECTEUR SCIENCES

Composantes

Faculté des Sciences et Technologies	Directeur: M. le Professeur F. De MARCHI
Département Biologie	Directeur: M. le Professeur F. FLEURY
Département Chimie Biochimie	Directeur: Mme. le Professeur H. PARROT
Département Génie Electrique et des Procédés	Directeur: M. N. SIAUVE
Département Informatique	Directeur: M. le Professeur S. AKKOUCHE
Département Mathématiques	Directeur: M. le Professeur A. GOLDMAN
Département Mécanique	Directeur: M. le Professeur H. BEN HADID
Département Physique	Directeur: M. le Professeur S. FLECK
Département Sciences de la Terre des Activités Physiques et Sportives	Directeur: M. le Professeur I. DANIEL
UFR Sciences et Techniques	Directeur: M. C. COLLIGNON
Observatoire de Lyon	Directeur: M. B. GUIDERDONI
Ecole Polytechnique Universitaire de Lyon 1	Directeur: M. P. FOURNIER
Ecole Supérieure de Chimie Physique Electronique	Directeur: M. G. PIGNAULT
Institut Universitaire de Technologie de Lyon 1	Directeur: M.R. BERNARD
Institut de Science Financière et d'Assurances	Directeur: Mme. la Professeure V. MAUME-DESCHAMPS

Acknowledgements

Merci beaucoup a tout le monde qui m'a aidé pendant les 5 ans de ma vie ou j'ai travaillé sur les sujets de cette thèse.

This is going to be a *sui generis* section of this thesis since I decided to write in three languages, including a general acknowledgement using my poor and under-developed french and some words in portuguese to my family.

Writing this thesis was not at all as I supposed it would be. It was much easier and fun. The reason is surely because I was surrounded by a team that conciliate highly competent and passionate professionals with gentle, humble and friendly people. Starting with the best directors I could think of: Marie-France, Vincent and Christian, each one of you contributed with your vision, orientation and patience for this to be the best work possible. Marie has become during these years much more than a director but a very special friend, a kind of friend that you want to spend Christmas together and with whom sharing a nice meal in a restaurant is always much more than eating and drinking well. Vincent (and Camille and Tao) has also become a friend. I am his first PhD student and I am really happy about it. Specially because he is an amazing and passionate professional who deeply understand both biology and computer science and discussing with him was always enlightening. Christian has arrived in the thesis more towards its end but I am also quite happy to have had the possibility to know more about biology, mathematics, science and laughing with you. This is, obviously, not an individual work and I would like to specially thank all my other co-authors, namely Alberto, Andrea, Cecilia, Etienne, Fabien, Fábio, Hubert Charles, Leen, Ludo, Matteo, Pilu and Vicho. You are the best and I cannot imagine this work without your presence, knowledge and support. Thank you also BleriBleri for sharing the same slide-template you used in your own PhD defense. I would also like to thank the reporters of my thesis: Joseph Heijnen, Steffen Klamt and Peter Widmayer and the president of my jury, Alain Viari.

Special thanks to all my very supportive friends that helped me directly or indirectly, far away in Brazil (ex: André e Alessandra, Bart e Paula, Be e família, Digão e Mel, Edson e Carmem, Joe e Elisa, Lurdinha, Mattheus, Paulo Vargas e Mariana, Robinson e Lu, Said e Odineia, Teddy, Teofilo e familia, Van) or physically closer in Europe (ex: Alexandre, Ana, Augusto e Lu, Blerina, Christian, Florence, Gustavo, Janice and Thomas, Lauranne, Leandro e Aline, Lilia, Mariana, Martin and Julie, PAF, Pat, Susan), and of course everybody else that were not explicitly mentioned but that were also part of this.

Para completar os agradecimentos não poderia faltar a minha família. Minha mamãozinha, Malu, pessoa que me deu a minha base moral e emocional e que é, portanto, a mais importante de todas. Agradeço também a minha irmã, Isabela, e aos meus sobrinhos queridos, Julia, João e Sofia. Agradeço a todos os demais familiares que me apoiaram com incentivos e muito afeto. Agradeço ao meu pai, que já se foi há alguns bons anos e, infelizmente, ainda tão jovem. Certamente, o senhor é parte integrante de mais essa etapa de minha vida.

Agradeço especialmente à Dri, por ter acreditado em mim e também por ter me feito

acreditar que sim, eu poderia. Sem você, nada disto teria sequer começado. Sou muito grato e feliz que você tenha dividido comigo cada momento desta trajetória que agora encerro, desde antes do início lá na 26 de agosto até o final, já aqui em Lyon. Este lugar é seu e estará para sempre gravado em mim.

Obrigado novamente a todos. Cette partie de l'histoire est fini...

Contents

Introduction

The work presented in this thesis may be seen as a collection of efforts to uncover structural properties in metabolic networks with the goal of arriving at a better understanding of their functional aspects. When I started writing this thesis, my idea was that I would have to put here all the things I had done for the past 6 years, from the point I got involved with bioinformatics and the minimal precursor set enumeration problem until now. But now that this document has a definitive shape, format and content, I can see that in fact this is more about what I have learned from the past 6 years, and this subtle difference made me much happier about it. It has surely been a multidisciplinary work, with big doses of Computational Theory and Algorithms, as much Biology as I could learn and also a lot of work on Software Engineering, which was one of the main areas I worked on before starting my master, back in 2006.

This work is mainly about metabolism and how to use computational tools, and specially graph algorithms to understand it better. Metabolism may be seen as a set of chemical reactions transforming sets of compounds (substrates) into other sets of compounds (products), while a metabolic network is an organized view of these reactions highlighting their possible interactions. Examples of interesting problems one may address on metabolism are: to detect what are the compounds needed for a given organism to produce key compounds such as amino-acids, to identify possible subnetworks that provide a biological hypothesis for a change in metabolite concentration that may be measured through metabolomics experiments, to identify possible metabolic exchanges between organisms living in community or in a symbiotic relationship, or finally, to identify what are the metabolic processes that take place in some special cell conditions such as growth, disease or some other stress situation.

All of these problems may be, at least partially, addressed through the enumeration of different metabolic structures, each one of them particularly designed with the goal of providing a better understanding of an underlying biological process. In this work, we present enumeration algorithms for minimal metabolic precursor sets, metabolic stories and chemical organisations, as well as some results on applying such enumeration procedures in order to (try to) answer some biological questions. Such applications confirm the fit of these structures as valid efforts for grasping the actual biological behaviour of the studied systems.

Although the biological questions that motivated each piece of work were diverse, the methodology for dealing and trying to answer them was quite similar and can be decomposed in the following three steps. Initially a formal definition capturing the biological features of interest has to be developed and accepted, specially by the biologists involved in the process, as a good candidate to represent the biological entity of interest. Once this modelling step is done, we can move to an algorithmic approach, first classifying the mathematical problem in terms of its time complexity, and then proposing algorithms for recovering solutions from the available data, even in the cases where the problem is computationally intractable. A biological validation of the solutions obtained closes this methodological cycle by providing a

new understanding of the model under study as well as proposing new directions and further improvements needed for an even better comprehension of the given system. During the time taken to achieve the results here presented, I have acquired some knowledge on Biology, more specifically on metabolism. This was possible mainly due to this methodological pipeline that my team follows since the modelling and the validation steps are inherently multi-disciplinary. I believe that for this reason already, this is a nice pipeline for dealing with biological questions because of its collaborative and complementary nature.

Another aspect of metabolism and its study using graph models is the different aspects one may concentrate on depending on the question one wants to answer. Of course, there are several pieces of information that are important to be taken into account from a biological point of view: biochemical reactions, metabolic pathways, reversibility of reactions, stoichiometric coefficients, thermodynamical constraints, enzymatic regulation, and so on. Currently, there is no model that takes all these informations into account, as far as our knowledge goes. Indeed, models vary from very simple representations concentrating only on the possible interactions between reactions to more complex linear programming models coupling stoichiometric and regulatory approaches in order to quantitatively predict the output of some objective function. This spectrum of models is explored in this thesis in the sense that each one of the three main problems addressed has different needs in terms of information and, hence, deals with different models of a metabolic network.

Metabolic Stories are maximal acyclic subgraphs with a constrained set of sources and targets (Acuña et al. (2012a)). The motivation behind this definition is going to be further detailed but, briefly, metabolic stories are our way of recovering chains of reactions that explain the flow of matter from a set of compounds S to another set T. Each story is an hypothesis to explain the concentration changes observed in a metabolomics experiment, or more generally to understand the interdependencies between a given set of vertices in a graph. The efforts to provide a formal definition required from us to separate biological issues from mathematical ones. Once the problem has been translated, it may be that it has been already considered in different domains and that there are ready-to-use algorithms that may be applied, or closely related algorithms that may be adapted. On the other hand, the formal definition may constitute a new computational problem, as seems to be the case for the stories problem. In this case, the algorithms we design to address the biological problem may be, in the future, useful for solving problems in completely different domains. This observation highlights another key feature of the previously mentioned pipeline, which is the need to move from the biological question to a general formally defined mathematical problem, which eases reusability of results.

Besides metabolic stories, this work also covers **minimal precursor sets** enumeration. The work on precursor sets was the one, as I previously mentioned, that introduced me to the field and I have never stopped working on it since then. During my master thesis, we published (Cottret et al. (2008)) our first version of the algorithm. This is an example in which the main result was the formal definition of a problem, since it was the first time that a definition of a minimal precursor set took cycles into account, which was a significant biological advance for the problem because cycles are very common in metabolic networks. In terms of algorithms, we presented a first version that in practice allowed us to work only on small networks, which was enough to provide us some biological insight on a relatively complex symbiotic system (Cottret et al. (2010a)). More recently, we developed new algorithms for exploring larger networks (Acuña et al. (2012b)).

Finally, we also worked on a mathematical object definition proposed in a different context, the **chemical organisations** (Dittrich and di Fenizio (2007)), and considered its use in the

context of metabolic network, with its specific properties such as mass and flux consistency. It is still the same pipeline but now we started directly from the second step and, to be honest, we did not reach the third yet. The work done for the moment answers our theoretical question of whether the enumeration algorithm could be made more efficient by considering such consistencies, and unfortunately the answer is negative. We however propose algorithms with improved expected performances over existing ones (Milreu et al. (2010)). The next and future step now is to consider practical algorithms that may run on our target networks, and to address the biological questions that possibly chemical organisations may help to answer.

This text is organized as follows. Chapter 1 presents the main biological concepts used throughout the work, namely metabolism, metabolic network reconstruction and metabolomics. Chapter 2 plays the same role for the main mathematical concepts, namely graph and hypergraph definitions and how to use them to model metabolic networks, the stoichiometric matrix, the concept of algorithmic enumeration problems and their time-complexity analysis. Chapter 3 presents the main contribution of this work, the enumeration of metabolic stories, detailing the whole process from the biological motivation, passing through the mathematical definition, the algorithmic and the complexity issues, to a validation of their biological value and, finally, to a different application whose result indicates that the method may also be adapted to answer a different biological question than the original one it was designed for. Chapter 4 presents a compilation of the work on the enumeration of minimal precursor sets, for which we have developed three different algorithms that will be detailed, as well as a biological validation and the time-complexity analysis of the problem. Chapter 5 presents our algorithmic approach for enumerating chemical organisations in metabolic networks under the assumption of mass and flux-consistency of such networks. Finally, a Conclusion and Perspectives chapter closes the discussion summarising the results obtained and highlighting interesting open problems. The software engineering aspects of the work, presenting diagrams and the general organisation of the code developed for addressing the metabolic precursors and stories problems, as well as information on how this library may be further extended to become a platform for metabolism-related software development is presented as an Appendix.

Chapter 1

Biological Concepts

Contents

The purpose of this chapter is to familiarise the reader with the basic biological concepts used throughout this thesis: metabolic networks and related notions such as metabolites and chemical reactions, as well as how to move from genomic information to obtain a representation of such a network.

1.1 Metabolic network

"Life is a chemical process involving thousands of different reactions occurring in an organized manner. These are called **metabolic reactions** and, collectively, **metabolism**" [Elliott and Elliott (2001)]. A chemical reaction may be described as a possible transformation of a set A of chemical compounds into a different set B. Ideally, both directions of such a transformation are possible, *i.e.*, one may observe $A \rightarrow B$ as well as $B \rightarrow A$. Such a reaction for which both directions are possible is called a **reversible** reaction and is denoted by $A \leftrightarrow B$. However, given thermodynamical constraints associated with the transformation in one of the directions of a reaction, it is possible to say that one of the two directions is so unlikely to happen that in practice the reaction is **irreversible**. Chemical compounds involved in reactions related to metabolism are called **metabolites** or, equivalently **compounds**. For a metabolic reaction $A \rightarrow B$, A is its **set of substrates or reactants** and B is its **set of products**. Metabolites may be imported inside the cell through so-called **transport reactions** and in this case they are called **nutrients**. Transport reactions are also responsible for exporting synthetized metabolites outside the cell.

A chemical equation may be used to describe the atoms being exchanged by the metabolites during a chemical reaction. For example, we describe the burning of methane as $CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O$. Notice that the substrates of the reaction are CH_4 and O_2 and the products are CO_2 and H_2O. The quantities of each of the substrates and products that are needed for the reaction to take place are known as its **stoichiometry** and the **stoichiometric coefficients** denote their proportions. In the given example, we may see that 1 molecule of CH_4 and 2 of O_2 react to produce 1 molecule of CO_2 and 2 of H_2O. Notice that both sides of a chemical reaction are expected to be mass balanced, *i.e.* the same atoms present in the substrates should be also present - even if rearranged - in the products.

Chemical processes in a cell usually are performed by a chain of chemical reactions such that, at each step a reaction consumes some compounds to produce some others that are going to be consumed by the next reaction and so on. This organized collection of reactions that performs some specific and useful task for a cell is known as a **metabolic pathway**. A metabolic pathway may still be seen as a transformation of a set of substrates into a set of products but now the task is performed by several successive metabolic reactions. Differently from reactions, metabolic pathways are in general irreversible. Indeed, after the collective effort of transporting the substrates into the cell and transforming them into an important product (amino-acids, cell walls, etc.), it would be a great loss if such a process was reverted by breaking again the product and reversing the whole pathway to expel the substrates out of the cell. For this kind of control, the irreversible reactions play an important role and, in general, the first and last steps of a metabolic pathway are performed by this kind of reactions. This does not mean that chemical processes inside the cell are irreversible but that they are done through different metabolic pathways depending on their direction, normally exchanging just the extremities of the pathway and keeping the reversible intermediate reactions. For example, "during violent exercise muscles convert glycogen to lactic acid. During rest, the lactic acid is converted to glycogen and the forward conversion switched off. To independently control the two directions (that is, to switch one on and the other off) there must be separated reactions to control; otherwise it would be possible only to switch both directions on and off together" [Elliott and Elliott (2001)].

Historically, metabolic pathways were inferred, validated experimentally and put into a catalog for further references. Even if the collection of all known metabolic pathways for a given cell provides a good view of its metabolic capacities, such a collection is far from representing all possible chemical transformations that may occur in a cell and, as we will further explore in Chapter 3 which presents our results on metabolic stories, even a metabolic pathway itself represents just one among several different possibilities for the transformation of a specific substrate set into a specific product set.

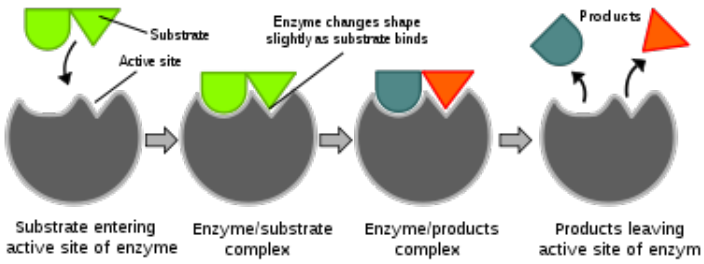
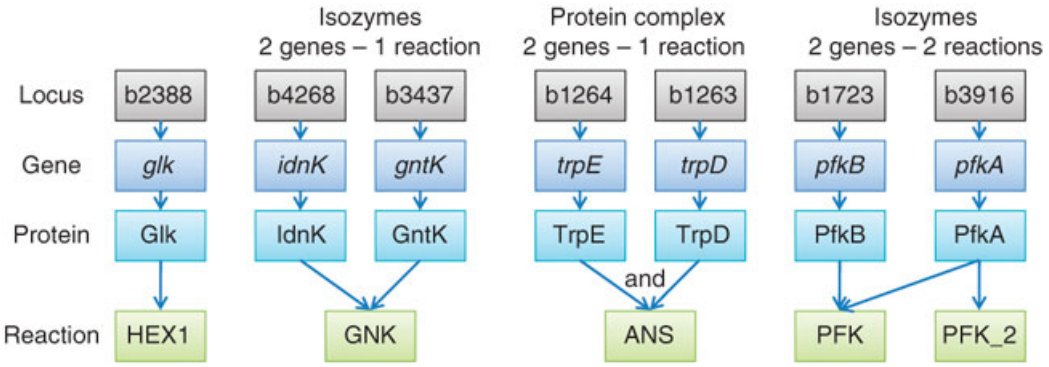


Figure 1.1: Illustration of the catalysis work performed by an enzyme. Source: Wikipedia

Reactions do not depend only on entropy, temperature, substrate concentrations, or energetic constraints to take place. Even if "thermodynamic considerations of sugar reacting with oxygen to form CO_2 and H_2O are highly favourable, sugar is stable in air. There is a barrier to the occurrence of chemical reactions, even if they involve large negative free energy changes" [Elliott and Elliott (2001)]. Usually, such constraints on chemical reactions indicate only if they may occur but the fact that they do occur depends on the presence of some catalytic agent such as **enzymes**. There may be spontaneous reactions, that happen without the need of any catalysis, but these are rare events. Metabolic reactions are, generally, catalysed by **enzymes**, *i.e.*, only in the presence of such special proteins do such reactions occur with the required speed. The role of the enzymes is exactly to accelerate thousands of times the

normal velocity in which chemical reactions occur. Figure 1.1 presents a schema on how the catalysis is done by the enzyme through the binding of the substrates to the active site of the enzyme, the formation of the enzyme-substrate intermediate, followed by the formation of the enzyme-products complexes, and finally, by the release of the products of the reaction.

Enzymes are biological catalysts and their synthesis by the cell is one of the main components for the cell to genetically control its metabolic activities. In general, each reaction is catalysed by a specific enzyme which is encoded by a specific gene but, usually, any other combination may occur in practice. There are three different possibilities for enzyme activity as exemplified in Figure 1.2. There may be multi-enzyme complexes that together may catalyse a single reaction like ANS. Conversely, there may also be multi-functional enzymes such as $PfkA$ that is able to catalyse two different reactions, PFK and PFK_2. Finally, there may also be the case that the same reaction may be catalysed by different enzymes and the reaction PFK is an example since it may be catalysed both by $PfkB$ or $PfkA$. Figure 1.2 thus illustrates the relation between gene content, enzymes and the reactions they catalyse.



Reaction abbreviation	Reaction name	E. C.number	GPR
HEX1	Hexokinase (D-glucose:ATP)	2.7.1.1	(b2388)
GNK	Gluconokinase	2.7.1.12	(b3437) or (b4268)
ANS	Anthranilate synthase	4.1.3.27	(b1264) and (b1263)
PFK	Phosphofructokinase	2.7.1.11	(b1723) or (b3916)
PFK_2	Phosphofructokinase (2)	2.7.1.11	(b3916)

Figure 1.2: Examples of Gene-Protein-Reaction (GPR) associations and their representation in Boolean format are shown for *Escherichia coli*. Source: Figure 5 in Thiele and Palsson (2010)

A **metabolic network** of a given organism may thus be seen as a collection of all possible reactions that may happen in a cell of this organism. The set of all metabolites in a metabolic network is called its **metabolome**.

1.2 Metabolic network reconstructions

The process of recovering the list of metabolic reactions that corresponds to the real metabolic network of an organism is known as **metabolic network reconstruction**. Basically, it consists in identifying the link between the gene content of an organism and its metabolic capabilites. Usually, this is done by identifying among the genes the ones that synthetize enzymes. Once a list of enzymes is available, it is possible to extend it to a list of the corresponding biochemical reactions catalysed by them. Figure 1.3 illustrates this

process. Another important point is that metabolism may be divided into small-molecule metabolism and large-molecule metabolism. The latter takes into account the biosynthesis of proteins and other large molecules. Most bioinformatics works are focused only on small-molecule metabolism and the metabolic network models used throughout this thesis have only metabolic reactions of this kind.

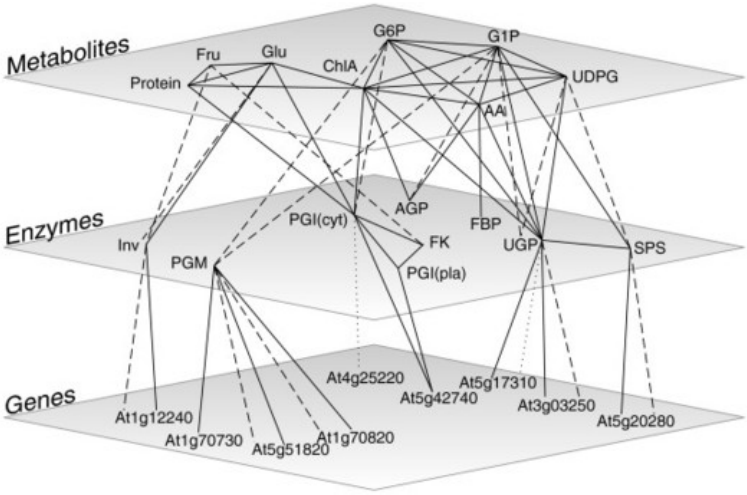


Figure 1.3: Metabolic network reconstruction showing how to move from the gene content to a set of enzymes and, finally, to the list of biochemical reactions corresponding to the metabolic network of an organism. Source: Wikipedia

The first challenge for reconstructing a metabolic network is to identify its genes and to assign a function to them, which is in itself a very interesting and well studied research topic in Bioinformatics (Reed et al. (2006); Lacroix et al. (2008); Thiele and Palsson (2010)). To apply comparative analysis in order to transfer knowledge that was acquired and curated on well studied organisms to less known and studied ones, phylogenetic information on the involved organisms is often required. Phylogenetics is the study of the history of evolutionarily related entities that may be, for instance, species or genes. Phylogenetic information is commonly expressed by means of a so-called phylogenetic tree in which the leaves correspond to current exemplars of the genes or species, and intermediate nodes to a prediction of their common ancestors until the root. Several variations of such a tree exist: the tree may be unrooted, or its edges (arcs) may be valued by the time estimated to have passed, etc. Details on phylogenetics are out of the scope of this work, since our goal here is just to illustrate how this can be used as a tool for metabolic network reconstruction. For this purpose only, we show a simple illustration of a phylogenetic tree of three species and three genes in Figure 1.4.

The main assumption for automatically assigning function to genes is that ortholog genes share a same function, *i.e.*, if in the past a speciation event happened, ortholog genes found in both species to be derived from the event probably have a same function. As an example, if gene A1 is known to catalyse reaction R1 in the mouse species and gene A1 has an ortholog gene in the rat species, it is fair to assume that this plays the same role also in rats. Identifying ortholog and paralog genes is done by comparing the DNA sequences that code for the gene. Paralog genes are less likely to keep a same function than ortholog ones since they occur after a duplication event. Indeed, it is less expected that a gene and its copy would be kept during evolution, performing exactly a same function. Of course, such assumptions depend strongly on the quality of the annotation of gene A1 in the mouse as well as on how phylogenetically

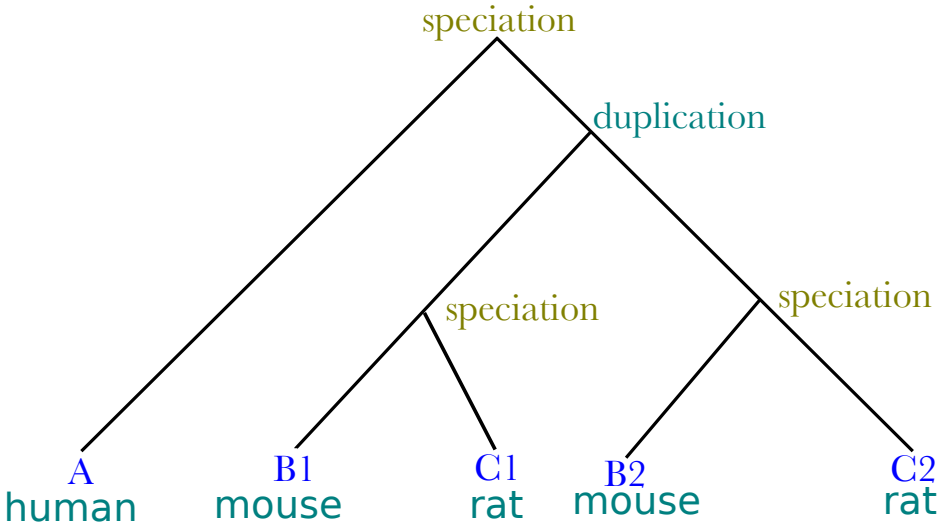


Figure 1.4: Phylogenetic tree involving three species (human, mouse and rat) and five genes (A,B1,B2,C1,C2). The genes B1 and C1 are ortholog genes, since they are derived from a speciation event. The genes B1 and B2, however, are paralog genes since they derived from a duplication event.

close mice and rats are. Once the function of the genes is known, it is a trivial task to select among them the catalytic related proteins, *i.e.*, the enzymes. A third step is then to identify, from this enzyme set, a list of biochemical reactions catalysed by them. To this purpose, there are several databases focused on enzyme data, such as BRENDA (Scheer et al. (2011)) and Uniprot (Consortium (2009)), from which the link between enzymes and chemical reactions may be recovered. Here, again, we assume that the link between enzymes and reactions is conserved through evolution. A final step is to identify which metabolic pathways may be present in the organism, based on the list of candidate reactions produced in the third step.

For some reference organisms such as *Escherichia coli* and *Saccharomyces cerevisae*, the list of chemical reactions inferred from their genomes is highly curated, *i.e.*, the presence in these organisms of a good percentage of those reactions was experimentally validated. For several other less studied organisms, this is not the case. Indeed, a metabolic network reconstruction for non-reference organisms may be automatically produced by several tools developed with this goal, for example the ones that integrate two of the most largely used platforms dedicated to metabolic information: Kegg (Kanehisa et al. (2008)) and BioCyc (Caspi et al. (2010)). As a matter of fact, both platforms store much more than simply metabolic information, providing also genomic and biochemical data and several tools for a large range of organisms. However, to obtain a better quality model, some further steps after the automatic reconstructions should be done, such as manual validation done by experts on the organism and/or an experimental validation of the proposed candidate reactions.

The quality of the first draft of a metabolic network reconstruction automatically obtained from one of these tools will depend basically on the taxonomy of the organism. This is due to the fact that the process of inferring functions for the genes is highly comparative. In the best case, a very close organism will be available and its gene content will be well annotated, providing good candidate enzymes for the metabolic network of the target organism. For the moment, this quality is the higher, the closer is the organism from the reference ones such as *E. coli* and *S. cerevisae*, for which the relation between gene and function is well validated. However, even in this ideal case, the resulting model will probably still contain errors, resulting

in metabolic networks with missing reactions, informally called *gaps*, reactions with wrong reversibility indications, etc. In order to improve the quality of the automatic reconstructions, several methods were developed for completing the missing information (Orth and Palsson (2010); Marashi and Bockmayr (2011); Satish Kumar et al. (2007a)).

Although the techniques of metabolic network reconstruction have still a lot of room for improvements, there exists well curated and experimentally validated reconstructions available, specially for model organisms such as *Escherichia coli* and *Saccharomyces cerevisiae*. Less studied organisms have become more and more focus of metabolic reconstruction projects and nowadays good quality data is also available for a wide spectrum of organisms. As the next section details, there are a variety of bioinformatics applications that were developed for the structural analysis of genome-wide metabolic networks. Even if the methods developed in this work may be applied on currently available data, they are expected to be even more useful when well curated metabolic network reconstructions become available.

Chapter 2

Mathematical Concepts

Contents

The purpose of this chapter is to familiarise the reader with the basic mathematical concepts used throughout this thesis, namely the definition of a graph, a directed graph and a hypergraph. Moreover, the chapter also gives an introduction to some classical algorithms in Graph Theory that are used repeatedly for the more complex methods of this work, specifically one of the main procedures for graph traversal: breadth-first search (BFS). A very brief introduction to algorithm analysis and the time-complexity model is also presented as well as its extension to enumeration problems. To this purpose, we propose to explore the feedback vertex set problem and its variation, the feedback arc set problem, that also form a basis for at least two of the three problems explored in this work (metabolic stories and chemical organisations) and that serve as a good illustration for algorithm time-complexity analysis, including the enumeration case. There is no intention to cover these subjects in detail. For more information on graphs and digraphs we refer to Bondy and Murty (1976) and Bang-Jensen and Gutin (2010), for more information on hypergraphs we refer to Berge (1976) and Ausiello et al. (2001), and for more information on algorithms on graphs and their complexity analysis we refer to Cormen et al. (2001), Garey and Johnson (1990) and Ausiello et al. (1999).

Finally, the metabolic network modelling aspects are also covered in this chapter. Indeed, we may use graphs, digraphs and hypergraphs to model the topology of such networks, each model having more or less details and, subsequently, being suitable for answering different kinds of biological questions. Graph models are static and qualitative models in contrast to dynamic and quantitative models of metabolic networks such as the description of a set of chemical reactions using differential equations. Such dynamic models are not considered in this work. An intermediate step moving from topology-based models to quantitative-based models are the so-called constraint-based models, that are also introduced in this chapter.

These models explore the stoichiometries and the irreversibility of the reactions as constraints on the possible fluxes of a network, the network is at the steady-state, *i.e.*, the concentrations of internal metabolites are balanced. In this work, such models are considered marginally, as a validation step in our chemical organisation enumeration algorithms and as perspectives for future work for the enumeration of both metabolic stories and minimal precursor sets. For a more detailed review on the differences between a quantitative and a qualitative modelling of metabolic networks, we refer to Stelling (2004).

2.1 Graphs and hypergraphs

A **directed graph**, or simply **digraph**, is a pair of disjoint sets (V, A). The set V represents the set of **vertices** or **nodes** while A, the set of **arcs** of the digraph, is a set of ordered pairs of vertices, *i.e.*, an arc is an ordered pair (u, v). An **undirected graph**, or simply **graph**, is a pair of disjoint sets (V, E), V is the set of vertices or nodes while E, the set of **edges** of the graph, is a set of unordered pairs of vertices, *i.e.*, an edge is an unordered pair (u, v). An arc or edge (u, u) is called a self-loop.

In a digraph, a vertex u is said to be **adjacent** to a vertex v if $(u, v) \in A$ and we denote it by $u \to v$. The set $N^-(u)$, called **in-neighbourhood** of u, is the set of vertices v such that the arc $v \to u$ exists and the set $N^+(u)$, called **out-neighbourhood** of u, is the set of vertices v such that the arc $u \to v$ exists. The **in-degree** of a vertex u, denoted by $d^-(u)$, is $|N^-(u)|$, while its **out-degree**, denoted by $d^+(u)$, is $|N^+(u)|$. For undirected graphs, the adjacency relation is symmetric, *i.e.*, if u and v are adjacent then both arcs $(u, v) = (v, u) \in E$ and we may define the neighbourhood $N(u)$ of a vertex u as the set of vertices v adjacent to u and the degree $d(u)$ of u as $|N(u)|$. A vertex is said to be **isolated** if it has an empty neighbourhood set or, equivalently, if its degree is zero. For the directed case, nodes with in-degree zero and out-degree positive are called **sources**, while nodes with in-degree positive and out-degree zero are called **sinks** or **targets**. In a digraph, a node is said to be isolated if both its in-degree and out-degree are zero. Figure 2.1 gives a visual representation of graphs and digraphs and examples of the above definitions.

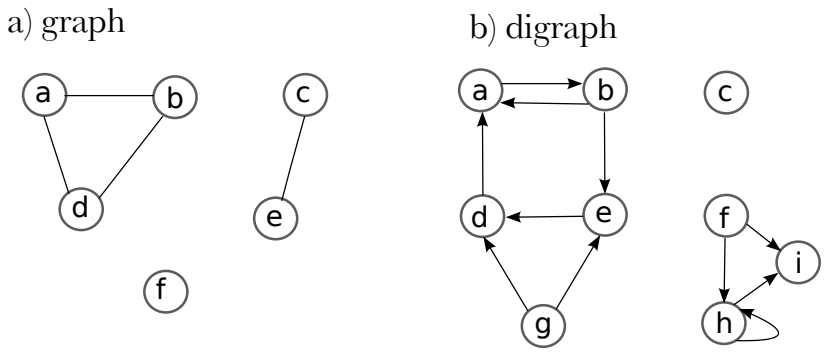


Figure 2.1: a) A graph with $V = \{a, b, c, d, e, f\}$ and $E = \{(a, b), (a, d), (b, d), (c, e)\}$. $N(a) = \{b, d\}$, $d(a) = 2$. Vertex f is isolated. b) A digraph with $V = \{a, b, c, d, e, f, g, h, i\}$ and $A = \{(a, b), (b, a), (b, e), (d, a), (e, d), (g, d), (g, e), (f, h), (f, i), (h, h), (h, i)\}$. $N^+(a) = \{b\}$ and $N^-(a) = \{b, d\}$, with $d^+(a) = 1$ and $d^-(a) = 2$. Arc (h, h) is a self-loop. Vertex c is isolated. Vertex g is a source and vertex i is a target.

A (elementary) **path** from a vertex u to a vertex v in a digraph $G(V, A)$ is defined as a sequence of vertices $\{p_0, p_1, \ldots, p_k\}$ such that $p_0 = u$, $p_k = v$ and $(p_i, p_{i+1}) \in A$, for $i = 0, \ldots, k-1$. The length of such a path is k. A path from u to v is denoted by $u \rightsquigarrow v$.

For example, in the digraph of Figure 2.1(b), there is a path $g \rightsquigarrow b = \{g, d, a, b\}$ of length 3. Notice that this is not the unique path between g and b, for instance you also have the path $g \rightsquigarrow b = \{g, e, d, a, b\}$ of length 4. If a path $p = u \rightsquigarrow v$ exists in a digraph G, we say that v is reachable from u in G. Again, the very same definition of a path may be extended for an undirected graph and the reachability relation is symmetric for this class of graphs. A **cycle** is defined as a path that starts and ends in the same vertex u, *i.e.*, a cycle is a path $u \rightsquigarrow u$. For example, in the graph of Figure 2.1, there is a cycle $\{a, b, d, a\}$ and in the digraph of the same figure, there is a cycle $\{a, b, e, d, a\}$ and a cycle $\{h, h\}$, which is a self-loop.

A **subgraph** $G'(V', A')$ of a digraph $G(V, A)$ is a digraph such that $V' \subseteq V$ and $A' \subseteq A$. For instance, the digraph $V = \{a, b, d\}$ and $A = \{(b, a), (d, a)\}$ is a subgraph of the digraph presented in Figure 2.1. Given a graph $G = (V, A)$ and a set of vertices $V' \subseteq V$, the subgraph $G'(V', A')$ induced by V', and denoted by $G[V']$, is such that $A' = \{(u, v) : (u, v) \in A \text{ and } u, v \in V'\}$. The same definitions may be directly extended for undirected graphs.

A graph is **connected** if, for any two vertices u and v, the path $u \rightsquigarrow v$ exists. In Figure 2.1, the graph G presented is not connected but the subgraphs $G[\{a, b, d\}]$, $G[\{c, e\}]$ and $G[\{f\}]$ are connected and are called the **connected components** of G. The connected components of a graph may be defined as the maximal connected induced subgraphs of G. A graph is said to be connected only if it has a unique connected component. A digraph is said to be **weakly connected** if, for any two vertices u and v, the path $u \rightsquigarrow v$ or the path $v \rightsquigarrow u$ exists, and it is said to be **strongly connected** if both paths exist. In Figure 2.1 the digraph G presented is neither weakly nor strongly connected but the induced subgraphs $G[\{a, b, d, e, g\}]$, $G[\{c\}]$ and $G[\{f, h\}]$ are weakly connected and are called the weakly connected components of G. The induced subgraphs $G[\{a, b, d, e\}]$, $G[\{g\}]$, $G[\{c\}]$, $G[\{f\}]$ and $G[\{h\}]$ are strongly connected and are called strongly connected components of G. Analogously to the undirected case, we may define the weakly (respectively, strongly) connected components of G as the maximal weakly (respectively, strongly) connected induced subgraphs of G.

A graph or a digraph that contains no cycle is said to be **acyclic**. A digraph with no cycles is known as a **directed acyclic graph** or simply **DAG**. A graph with no cycles is known as a **forest**. If the graph is acyclic and connected, then it is called a **tree**. In a forest, each connected component is a tree.

A digraph $G(V, A)$ is **bipartite** if its vertex set may be split in two disjoint sets such that any arc of the graph has one of its extremity in one set. Formally, $V = V_A \cup V_B$, such that $V_A \cap V_B = \emptyset$, and $\forall (u, v) \in A$, $u \in V_A \Rightarrow v \in V_B$. This definition may be directly extended for undirected graphs.

A **hypergraph** $H = (V, E)$ is a pair of sets V of vertices and E of unordered subsets of V. A directed hypergraph $H = (V, A)$ is a pair of sets V of vertices and A of pairs of subsets of V, *i.e.*, $a = (in(a), out(a)) \in A$ is an hyperarc of H and $in(a) \subseteq V$ and $out(a) \subseteq V$. These are the two classic definitions of undirected (Berge (1976)) and directed hypergraphs (Ausiello et al. (2001)) found in the literature. Figure 2.2 presents a graphical representation for the two proposed definition of undirected hypergraphs as well as for the directed version.

Graphs and hypergraphs, directed or undirected, may be extended with additional information associated both with their vertices or their arcs/edges/hyperarcs. A very common extension is to add quantitative information on the vertices using a mapping function $w : V \mapsto \mathbb{R}$ or on the arcs/edges/hyperarcs using a mapping function $w : A \mapsto \mathbb{R}$. The function w is said to assign **costs** or **weights** to the vertices, arcs, edges or hyperarcs. The value $w(v)$, $v \in V$, is the cost or weight of the vertex v. The value $w(a)$, $a \in A$, is the cost or weight of the arc a, and so on. A graph or hypergraph with an associated weight function is called a weighted graph or hypergraph.

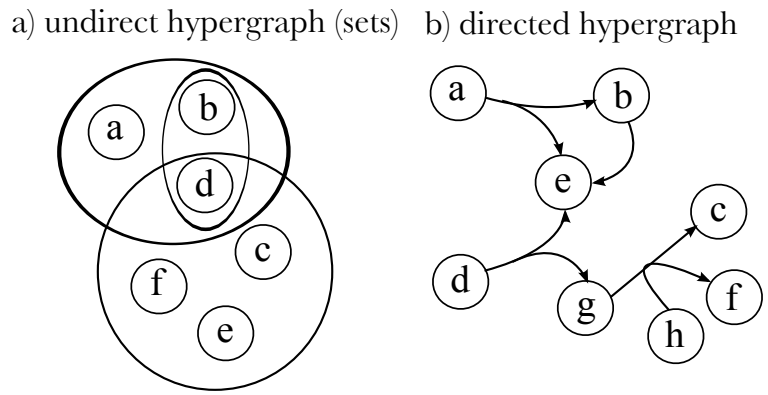


Figure 2.2: a) An hypergraph with $V = \{a, b, c, d, e, f\}$ and $E = \{(a, b, d), (b, d), (c, d, e, f)\}$. b) A directed hypergraph with $V = \{a, b, c, d, e, f, g, h\}$ and $A = \{(\{a\}, \{b, e\}), (\{b\}, \{e\}), (\{d\}, \{e, g\}), (\{g, h\}, \{c, f\})\}$.

2.2 Modelling metabolic networks

Metabolic networks may be seen as a collection of biochemical reactions that may occur in the metabolism of an organism. There are different ways to model metabolic networks, depending on the focus and the level of details desired. If one is mainly interested in the topological attributes of such a network, graph or hypergraph models are well suited.

A **compound graph** is a digraph modelling a metabolic network whose focus is on the compounds (also called metabolites). The vertex set of the digraph is the set of metabolites, and there exists an arc from a metabolite u to a metabolite v if there is a reaction in the network such that u is a substrate and v is a product of this reaction. A **reaction graph** is a digraph modelling a metabolic network whose focus is on the reactions. The vertex set of the digraph is the set of reactions, and there exists an arc from a reaction u to a reaction v if at least one of the products of reaction u is a substrate of reaction v. Figure 2.3(a-c) presents an example of how a list of reactions may be modelled into compound and reaction graphs. These representations of a metabolic network are very simple and significant information is lost (van Helden et al. (2002); Lacroix et al. (2008); Klamt et al. (2009)). For instance, if you look only at the compound graph in the figure, the information that you need both a and b in order to produce c and d is lost. Indeed, if we replace reaction $R1$ by two reactions $R4 : a \to c$ and $R5 : b \to d$, the compound graph representation will remain unchanged, but the biochemical transformations that we are modelling are not the same. In the same way, the reaction graph is supposed to capture the dependencies between reactions, but it is not possible to distinguish that the two arcs arriving in $R3$ from $R1$ and $R2$ mean that $R3$ depends on both reactions to happen for it to have all its substrates, while the two arcs arriving in $R2$ from $R1$ and $R3$ are two alternative ways for obtaining the only substrate $R2$ needs.

A bipartite representation of a metabolic network may overcome some of these problems by partitioning the set of nodes in two: metabolites and reactions. In this way, each substrate has an outgoing arc to the reaction that consumes it and each product has as incoming arc from the reaction that produces it, eliminating the ambiguity present in the previous models. However, the fact that **all** substrates of a reaction must be present for the reaction to take place and that, in such a case, **all** products will be synthesized is not explicit in the model, *i.e.*, it is still possible to find a path $a \to R1 \to c$, ignoring the information that the presence of b is mandatory for the reaction to take place. For this reason, an undirected or, preferentially, a

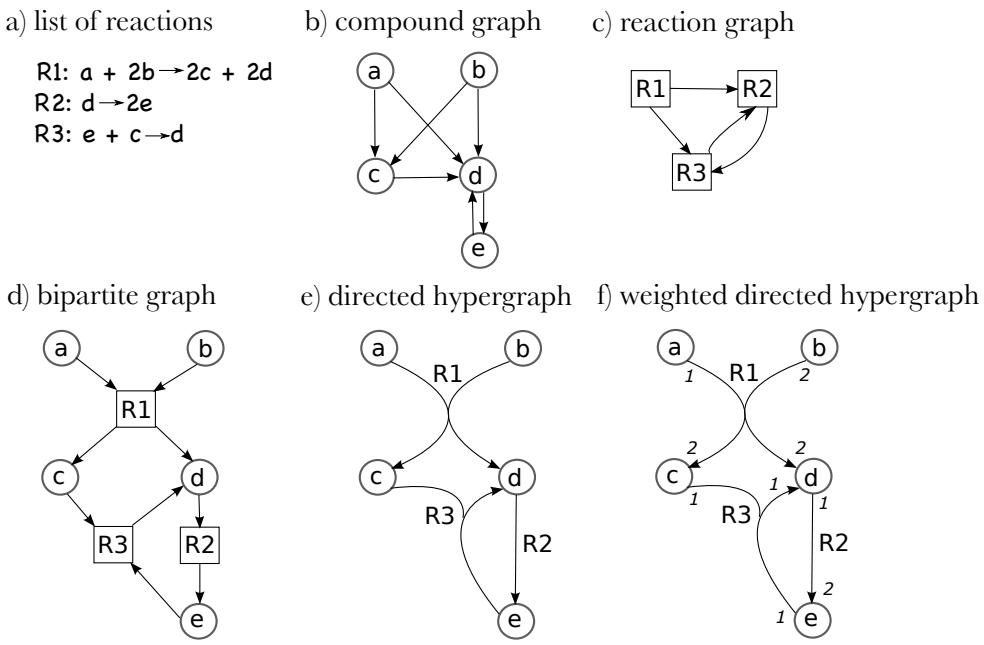


Figure 2.3: a) List of reaction corresponding to a metabolic network (b) Compound graph representation of the network (c) Reaction graph representation of the network. (d) Bipartite graph representation of the network (e) Directed hypergraph representation of the network (f) Directed hypergraph representation of the network with stoichiometric coefficients.

directed hypergraph is a more accurate representation of a set of reactions since the hyperarcs may be modelled to correspond exactly to the set of substrates and products of a reaction. Figure 2.3(d-e) presents the bipartite and directed hypergraph representation of the same metabolic network previously modelled as compound and reaction graphs. Notice that with the directed hypergraph representation, the imprecisions mentioned before are not present, which makes such models increasingly more used for modelling metabolic networks but also other sorts of biological networks, such as Protein-Protein Interaction networks (Klamt et al. (2009)).

A weighted version of a graph or hypergraph may include quantitive information on the model. For instance, one may want to consider the addition of the stoichiometric coefficients of the chemical reactions, detailing the proportions with which each metabolite is taken by the reaction. Figure 2.3(f) presents a weighted directed hypergraph with a weight assigned to each node in the hypearc, corresponding to the stoichiometric coefficients of the reaction modelled by the hypearc. However, this representation becomes cumbersome and a much more widely used approach is a so-called **stoichiometric matrix** representation of a metabolic network, which may be seen as the incidence matrix of the weighted directed hypergraph previously described. Each row of this matrix represents a metabolite of the metabolic network and each column represents a reaction. Considering a stoichiometric matrix S, the cell $S[i, j]$ denotes how many molecules of metabolite i is consumed or produced by reaction j. Negative values indicate consumption while positive values indicate production. If the metabolite is not involved in the reaction, the value of the corresponding cell is zero. A stoichiometric matrix corresponding to the metabolic network of Figure 2.3 is shown in Figure 2.4.

Notice that even if a stoichiometric matrix seems to capture as much information as a weighted directed hypergraph, this is not always true. A subtle case in which a directed

	R1	R2	R3
a	-1	0	0
b	-1	0	0
c	2	0	-1
d	2	-1	1
e	0	2	-1

Figure 2.4: Stoichiometric matrix corresponding to the list of reactions presented in Figure 2.3.

hypergraph is able to represent more information relates to the presence of autocatalytic metabolites, *i.e.*, metabolites that are involved in their own synthesis. This is an example in which hypergraphs may be more precise than the matrix, namely when a metabolite appears both as substrate and product, and its corresponding cell in the matrix contains the difference between production and consumption. Hypothetically, we could also model the enzymatic level with a directed hypergraph, by adding the enzyme catalysing a reaction in both the substrate and the product sets of the reaction, using stoichiometry one on both sides to model that the enzyme is not transformed by the reaction, *i.e.*, it remains unchanged after the reaction finishes. This is also not possible with a stoichiometric matrix, since a zero value will be assigned to the cell corresponding to the enzyme.

In constraint-based approaches, it is common to suppose that the system is at steady-state, which means that the concentration of the internal nodes (all metabolites except nutrients and substrates of large molecule metabolism) is balanced, *i.e.*, that the internal molecules that are being produced are being consumed at the same rate. A first constraint that comes from this assumption is to identify flux vectors $v \in \mathbb{R}^m$, with m the number of reactions of the network, satisfying $Sv = 0$, with S a stoichiometric matrix representing the metabolic network of a cell. Reversible reactions may have positive or negative fluxes, depending on which direction the transformation occurs but the irreversible reactions should always have a positive flux, which brings a second constraint to the model. The set of basis vectors that satisfy such inequations may be obtained by a convex analysis if we consider the inequalities $v[i] \geq 0$ for each irreversible reaction i of the network and the steady-state assumption $Sv = 0$. In such a case, the feasible flux space of a metabolic network may be described as a multidimensional cone, each axis corresponding to a set of flux through the reactions of the network. Two main methods have been proposed to describe such a cone, namely **elementary modes** (Schuster et al. (1999)) and **extreme pathways** (Schilling et al. (2000)). An elementary mode is a minimal (non-decomposable) set of reactions working at steady-state, and the set of all elementary modes describe all possible routes through a metabolic network. Extreme pathways may be seen as a minimal subset of elementary modes, *i.e.*, an extreme pathway may not be generated by a nonnegative linear combination of extreme pathways. A more detailed comparison between the two methods may be found in Papin et al. (2004). Elementary modes and extreme pathways are an inner description of the cone. More recently an outer description of such a cone, called **minimal metabolic behaviours** (Larhlimi and Bockmayr (2009)), has been proposed with the goal to provide a smaller description than the previous approaches.

Constraint-based approaches have also been extensively used in the context of flux balance analysis (FBA) (Varma and Palsson (1994a); Price et al. (2003); Kauffman et al. (2003)). In this context, in addition to the steady-state assumption and the constraints on the irreversibility of reactions, an optimal flux maximizing/minimizing some given objective function is sought. Typical flux balance analysis applications are to obtain *in-silico* predictive models for inferring growth or yield rates of some product of interest for several different organisms

(Varma and Palsson (1994b); Fong et al. (2005); Boyle and Morgan (2009)), to study the viability of mutant strains bio-engineered by gene knockouts or other genetic manipulations (Pharkya et al. (2004); Joyce and Palsson (2008); Suthers et al. (2009)), to check the flux consistency or curation of a metabolic network reconstruction (Satish Kumar et al. (2007b); Gevorgyan et al. (2008); Acuña et al. (2009)), and a wide variety of other possibilities. Notice that the choice of the objective function is fundamental for such approaches. For instance, a common objective function used for studying bacteria is to assume that in rich media they grow at maximum speed, which is modelled as an objective function in which the goal is to optimize the flux over a fake reaction, added to the model to emulate the production of biomass from compounds that should be essential for a bacterium to grow (Feist and Palsson (2010)).

2.3 Analysing structural properties of metabolic networks

The content of this work may be put in the general context of Structural Analysis of Metabolic Networks. The three subjects that form the kernel of this work, namely metabolic stories, metabolic precursor sets and chemical organisations, may be seen as subnetworks which satisfy some given topological and/or stoichiometric properties, and the proposed algorithms are therefore techniques for graph extraction focused on efficiently identifying these subnetworks. In this sense, the work here presented may be categorized as **global**, since genome-wide metabolic networks are explored and all possible subnetworks are taken into account, in contrast to a local analysis where the networks are only partially analysed. Another characteristic shared by the three methods is their **static, structural and mainly qualitative** nature, in opposition to the study of the dynamics of metabolic networks, or to quantitative analyses based on local or global measures of their graph representations.

There are several works that try to uncover biological knowledge based on analysing topological and/or quantitative properties of metabolic networks. A basic initial idea is to simply compute topological measures on the graph representation of metabolic networks. A natural question one may pose is how important is a node, *i.e.* if it plays a major role on the biological context under study, and centrality measures may thus be computed to try to give to each node some value of its importance, based on local or global properties. Local properties take into account information on the node and its surroundings but do not inspect the graph entirely, examples of local information is the degree of a node and the set of nodes directly connected to it. Global properties, on the other hand, take into account the whole graph in order to assign a value to a node, examples are the average distance of the node to the other nodes in the graph or how many shortest paths between two nodes passes through it. Some surveys on the use of graph measures for the analysis of metabolic networks and the issues and cautions needed for applying such techniques are Mason and Verwoerd (2007); Koschutzki and Schreiber (2008); Lacroix et al. (2008); Pavlopoulos et al. (2011); Klein et al. (2012b). Notice that these techniques may be applied not only to metabolic networks but also on other biological networks such as protein-protein interaction networks and regulatory networks.

In addition to the known issues related to measure-based approaches, we have recently argued (Klein et al. (2012b)) that care must be taken by methods relying on global measures or on identifying global subnetworks, such as our own. Our concern is about the difference between genome-wide metabolic network reconstruction and the subset of chemical reactions that are really active at any given moment in the cells. Metabolic network reconstructions provides information on all possible chemical reactions that may take place in the cell of an

organism and represents, thus, a **potential network**. However, only a subnetwork will be active at a given time interval, and we call each of these subnetworks, that are probably performing some task for the cell, a **realization** of a network. Local or, specially, global measures computed in the potential network may provide insights that are not true when one inspects all realizations of a same network, *i.e.*, it may be the case that even if the topological identified feature is potentially true, it is not true or it has a completely different interpretation for the real instances. For instance, identifying highly connected nodes, or hubs, on the potential network will provide some understanding on its essentiality or functional importance, while looking for nodes that are commonly hubs over a large set of realizations may have a completely different interpretation. In our case, applying metabolic stories enumeration on genome-wide metabolic network will provide all possible scenarios explaining transformations observed in a given metabolomics experiment. Ideally, we could apply the same enumeration algorithm on realizations of this network corresponding to the same conditions the cell was during the experiments, obtaining a smaller set of solutions and, probably, with a better quality. From an algorithmic point of view, however, this discussion has no impact since the method is exactly the same independent on the input being the whole network or some subnetwork.

Another topological analysis that may be done on metabolic and other biological networks is a modular decomposition of the network. The idea is to decompose a network in its functional modules that are commonly hierarchically grouped. Another possible benefit of a modular decomposition is that it allows the study of the modules independently and, since they are considerably smaller than the whole network, analytical methods that do not scale well for large networks may again be employed. For instance, by the means of the concept of reaction correlation coefficient, that tries to model the concept of an enzyme, hierarchical metabolic trees may be built where the leaves are individual reactions, the root of the tree is the whole network and intermediate nodes represent subsystems of reactions or metabolic modules (Poolman et al. (2007)). Different approaches for modular decomposition were proposed in the context of metabolic networks (Holme et al. (2003); Yoon et al. (2007); Verwoerd (2011)) and also for protein-protein interaction networks (Gagneur et al. (2004)).

Another structural approach is to compare the metabolic capabilities of two or more organisms, providing information on their similarities and their differences and, possibly some insight on the metabolic features that are conserved through evolution and may form a shared core of chemical reactions (or genes). A common application of such technique is to try to identify essential reactions (and, consequently, genes) that must form a core of the metabolic reactions that must be present in (almost) all compared organisms, usually focused on bacteria (Gabaldon et al. (2007); Barve et al. (2012); Klein et al. (2012a)).

This section has no hope of exhaustively covering all the various methods already employed for the structural analysis of metabolic networks. Subnetwork extraction techniques as well as methods for automatically proposing metabolic pathways are going to be covered in Chapter 3. Methods that explore the metabolic capabilities of an organism for nutrient-related analysis are covered in Chapter 4.

2.4 Algorithm complexity analysis

A **problem** is a general question to be answered and may be described by giving a general list of all its parameters and a statement of what properties the solution to the problem has to satisfy. An **instance** of a problem consists in specifying values to the parameters (Garey and Johnson (1990)). For example, the **shortest path problem** is the following: given a

graph G and two vertices u and v, what is the shortest path between u and v, *i.e.*, what is the minimum length of a path connecting these two nodes? An instance of this problem is, for example, the digraph presented in Figure 2.1 and the pair of vertices g and b, and the solution of the problem for this instance is 3.

An **algorithm** is a list of consecutive procedures or instructions to solve a problem. An algorithm is said to solve a problem if it can be applied to any instance of the problem and is guaranteed to finish and to produce a valid and correct solution for that instance (Garey and Johnson (1990)). Informally, an algorithm is usually seen as a recipe, a list of instructions that one should follow to have, at the end, the answer to the question the algorithm addresses. There are different ways of solving the same problem and, obviously, there are different algorithms that may solve the same problem. Normally, we are interested in the most efficient one but, in this case, we should be able to compare algorithms in terms of their efficiency. By efficiency, we mean the use of computational resources such as memory, CPU time, bandwidth, etc. Here we are mainly interested in the maximum computational time taken by the algorithm to solve the problem, *i.e.*, how much time will it need to solve the worst possible instance. In order to avoid differences due to computer hardware or software that would be extremely hard to take into account, the computational model commonly used in order to define the efficiency of an algorithm is based on a uniform cost per instruction, assuming all pieces of information stored in memory have almost the same size.

The theory that allows us to compare algorithms is based on **decision problems**, which are problems that may have only YES or NO as an answer. Even if this seems very restrictive, this is not the case since algorithmically solvable problems may be, in general, translated in terms of decision problems. For instance, problems known as **optimisation problems** are problems for which a solution is a "best" possible value, for example "find a shortest path between two nodes in a digraph". The decision version of such a problem is "is there a path between these two nodes of length at most k in a digraph".

Algorithmic analysis consists in specifying the execution time of an algorithm as a function of its input. A natural problem that arises in this case is that problems differ enormously on their inputs: these may be simply integers or strings, or yet more complex data structures. For this reason, we assume the inputs to be encoded in a string over some finite alphabet. We say that the encoded input has size n and, therefore, the complexity of an algorithm is a function $f(n)$. We say that a function $f(n)$ is $O(g(n))$ if there exist constants c, a and n_0 such that, for all $n \geq n_0$, $f(n) \leq cg(n) + a$. The constant a is independent of the size of the input and is usually related to the time spent on the preparation steps. If the number of steps performed by an algorithm in the worst-case is $O(g(n))$, we say that this algorithm has complexity $O(g(n))$, *i.e.*, for any instance with size n, the running time of such an algorithm is bounded from above by the value $g(n)$. This behaviour do not need to be valid for all input sizes, but there must be a point from which one function limits the other. A graphical illustration of such a comparison between two functions is given in Figure 2.5.

2.4.1 Algorithms on graphs

As an illustration of algorithms and their complexity analysis, let us provide an example. Consider the **breadth-first search (BFS)** algorithm, which is a classical algorithm in Graph Theory for traversing a graph: starting from a given source vertex s, at each step i of the algorithm, all nodes that have a distance i from u are visited. By the end of the execution, all reachable nodes from s have been visited and a so-called **BFS-tree** is obtained, as shown is Figure 2.6, that explores the induced subgraph $G[\{a, b, d, e, g\}]$ corresponding to the larger

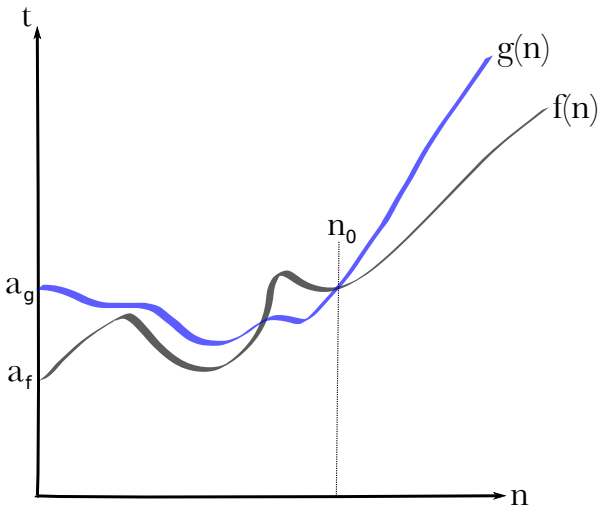


Figure 2.5: A function $f(n) \in O(g(n))$ as there exists $c > 0$ (*e.g.* $c = 1$) and n_0 such that $f(n) < cg(n)$ whenever $n > n_0$.

weakly connected component of the digraph presented in Figure 2.1(b).

Algorithm BFS(G, s)
Require: a digraph $G = (V, A)$ and a source vertex $s \in V(G)$.
  A BFS tree T rooted in s.
  $s.color$ = GREY
  $T.root \leftarrow s$
  for all $v \in V(G) - \{s\}$ **do**
    $v.color$ = WHITE
  put s in a queue Q
  while $Q \neq \emptyset$ **do**
    $u \leftarrow$ dequeue from Q
    for all $v : \{u, v\} \in A(G)$ **do**
      if $v.color$ = WHITE **then**
        $v.color$ = GREY
        hang v in T with u as its parent
        put v in Q
    $u.color$ = BLACK
Ensure: T

In order to analyse the BFS algorithm, we have to inspect how many computations it may take for it to solve the problem for any possible input. Classically, for graph algorithms, the size of the input is measured in terms of n, the number of vertices, or m, the number of arcs of the graph. For the given pseudo-code of the BFS algorithm, it is easy to verify that the algorithm performs some initialisation steps, including defining an initial white colour for all the nodes, signaling that they were not yet traversed by the algorithm. This consumes n operations. The core of the algorithm is performed by the main loop that analyses vertices in the queue. Notice that each vertex is added to the queue only if it has a white colour. Immediately after, its colour is changed to grey and is never changed again to white, which guarantees that each vertex is added just once to the queue. For each vertex, however, all its neighbourhood is inspected which means that in the end, all the arcs will be inspected once and then we can say that this main loop takes m operations. As a conclusion, we can say that the time complexity of BFS is $O(n + m)$.

Analysing a BFS tree, like the one in Figure 2.6, we may see that the BFS algorithm can be useful to solve the previously mentioned problem "given a graph G and two vertices u and

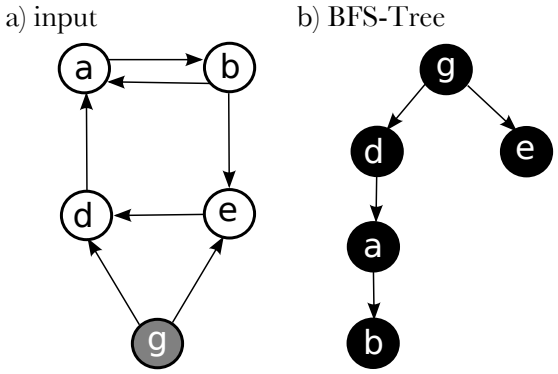


Figure 2.6: a) Input digraph G and the source vertex g, coloured in grey. b) BFS-Tree rooted in g.

v, find a shortest path between u and v". Indeed, by calling $BFS(G, u)$, we will obtain a BFS tree rooted in u, and we may easily search for v in this tree to compute the minimum distance between u and v. We say then that this problem is also $O(n + m)$, as there is an algorithm with this complexity to solve it. Every time the complexity of an algorithm is bounded by a polynomial of its input size, the problem is said to be **polynomial solvable** and to belong to a class P of problems.

2.4.2 Complexity analysis of algorithms

Finding the shortest path between two vertices of a graph may be done with an algorithm that makes a total number of steps that is a polynomial on the size of the input. These problems are said to be polynomial solvable. There are, however, problems for which no polynomial algorithm is known. Among those, there is a very interesting class of problems, known as the non-deterministic polynomial (or, *NP*) problems, for which a polynomial-time certificate algorithm exists, *i.e.*, if we provide an instance of the problem and a candidate solution, this certificate algorithm may answer, in polynomial-time, if the solution provided is a valid one or not. One of the most important theoretical open questions in Computer Science is to answer if P and NP are equal or not. The importance comes from the fact that there is a subset of the NP problems for which we may prove an equivalence that states that a polynomial-time algorithm solving any of them may also be adapted into a polynomial-time algorithm to solve any of the other problems in such a subset. This subset is known as the *NP-Complete* problems and is composed of several classical problems in Computer Science such as: the boolean satisfyability problem, the travelling salesman problem and the vertex cover problem (Karp (1972)). In order to prove a problem Y to be part of the *NP-Complete* class, one has to provide a polynomial-time certificate algorithm for Y as well as a polynomial-time algorithm that translates any instance of a *NP-Complete* problem X to an input of Y, with the sizes of the two instances polynomially comparable. To finish the proof, it is necessary to show that an algorithm solving Y will answer yes to some instance y of the problem Y if and only if an algorithm solving X also answers yes for the corresponding instance x of X. Basically, what these operations guarantee is a class of equivalent problems, in the sense that, once a polynomial algorithm is found for any of the problems in the class, automatically all other problems are also polynomially-solvable.

As an example of a *NP-Complete* problem, we present another problem in Graph Theory known as **feedback vertex set (FVS)**, that may be defined as follows: given a digraph $G(V, A)$, find a minimum size set $F \subseteq V$ such that $G[V - F]$ is acyclic. A variant of this

problem is the **feedback arc set (FAS)** problem, in which we want to find a minimum size set $F \subseteq A$ such that $G'(V, A - F)$ is acyclic. Informally, we want to find a minimum number of vertices or arcs that we need to remove from the graph to make it acyclic.

2.4.3 Enumeration problems

There exists another class of problems for which we are not interested in a single solution, as is the case for decision and optimisation problems, but in enumerating all of them. In this work, we mainly focused on this class of problems, since in biology it is common to work with incertitudes and errors on obtaining the input data, so that it is desirable to analyse sub-optimal solutions and, if possible, to really inspect all possible alternatives of explanation for some phenomena under study. In many cases, the decision to address the problem initially through an enumeration approach comes from the fact that no clear function to optimise is available.

The complexity analysis of enumeration algorithms, *i.e.*, algorithms devoted to find all possible solutions for a given problem, is different from the complexity analysis of decision or optimisation problems. The reason for this is clear, since the output of such an algorithm may be exponential in the input size, then just the time needed to write this output down will not be polynomial. Indeed, for this class of problems, we may have to consider not only the input size but also the output size. The classes of polynomial-time enumeration problems are **polynomial delay**, **incremental polynomial delay** and **polynomial total time** (Johnson et al. (1988)). More recently, a review on enumeration problems and their complexity specially focused on problems for which the order of the output is important has been published (Schmidt (2009)).

To explain the three classes of enumeration problems, we introduce three variables: $n(I)$ is the input size of the problem, $n(O)$ is the output size of the problem and $S(i)$ is a set containing all computed solutions until step i of the algorithm execution. We say that an enumeration algorithm has a *polynomial delay* complexity if at any step i of its execution, the time taken for the algorithm to compute the next solution or to state that there is no additional solution is bounded by a polynomial in $n(I)$. We say that an enumeration algorithm has an *incremental polynomial delay* complexity if at any step i of its execution, the time taken for the algorithm to compute the next solution or to state that there is no additional solution is bounded by a polynomial in $n(I) + |S(i)|$. We say that an enumeration algorithm has a *polynomial total time* complexity if it can be solved with time bounded by a polynomial in $n(I) + n(O)$.

As an illustration of such a process, we move back to the FVS and FAS problems, for which algorithms with *polynomial delay* complexity have been proposed (Schwikowski and Speckenmeyer (2002)). One of the enumeration problems addressed in that work (Schwikowski and Speckenmeyer (2002)) is the enumeration of all minimal feedback arc sets (MFAS) in a digraph. The algorithm proposed has a polynomial-delay, which means that the time interval needed to output each of the solutions is polynomial in the size of the input, in this case in the number of vertices of the graph. Their approach consists in two main ideas. The first, and most important one, is based on what they called a (v)-successor function, which is defined as a function $s(F, v) \rightarrow F''$ that has as input a MFAS F and a vertex v, and through a local modification in F, namely by replacing arcs from the in-neighbourhood of v by arcs in the out-neighbourhood of v, produces a new FAS F'. F' is surely a FAS because for any cycle passing through v that was covered in F by an arc (u, v), this cycle is also covered in F' by some arc (v, w). However, F' may be not minimal. In this case, it is easy to obtain a MFAS

$F'' \subseteq F'$ by just checking for each arc $a \in F'$ if $F - \{a\}$ is a FAS, keeping only the ones with a negative response to produce F''. Assuming this minimalization operation and an arbitrary and fixed order of the nodes, we may state that a unique MFAS is obtained when computing $s(F, v)$, for any MFAS F and vertex v. Based on this idea, the authors proposed a procedure that transforms any MFAS F into any target MFAS $F*$ by iteratively producing intermediate MFASs computed with the help of the defined successor function. The algorithm is illustrated in Figure 2.7 and works basically in the following way: initially a topological order T of the nodes of the graph is inferred based on the acyclic digraph obtained by removing the arcs in $F*$, and the algorithm starts with the MFAS $F_0 = F$. The algorithm iterates until a MFAS $F_k = F*$ is obtained. To move from a MFAS F_i to a MFAS F_{i+1}, a node v_i is selected that holds two properties: it is an endpoint of an arc contained in F_i but not in $F*$ and it is minimal with respect to T. A new MFAS F_{i+1} is computed as $s(F_i, v_i)$. Notice that v_i will always exist because $F_i \neq F*$, or else the algorithm would have stopped. An additional and important property of this procedure is that the number of iterations is bounded by $|V|$, since at each step a $v_{i+1} > v_i$ is chosen.

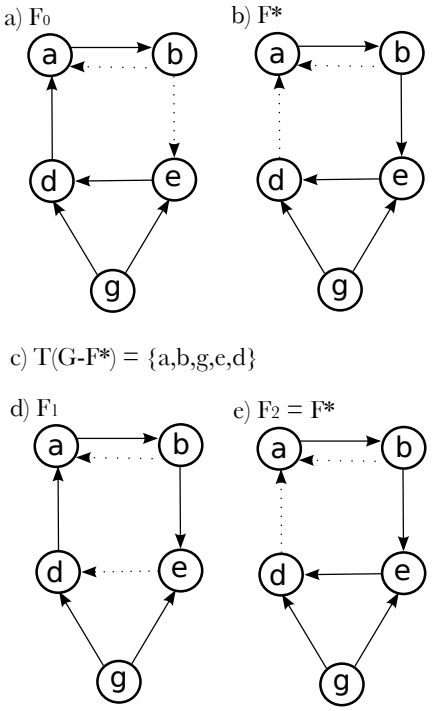


Figure 2.7: Example on how applying a carefully chosen successor function, one may transform any minimal feedback arc set F_0 into any target minimal feedback arc set $F*$ with a maximum number of steps given by $|V| - 1$. This is a fundamental result for the enumeration algorithm proposed in Schwikowski and Speckenmeyer (2002). a) Starting MFAS $F_0 = \{\{b, a\}, \{b, e\}\}$. b) Target MFAS $F* = \{\{b, a\}, \{d, a\}\}$. c) Topological order inferred from the DAG obtained by removing the arcs in $F*$. d) Computation of F_1: the only arc that is in F_0 and not in $F*$ is $\{b, e\}$ and therefore $v_1 = e$. Thus the arc $\{b, e\}$ is exchanged by the outgoing arcs from e, in this case only the arc $\{e, d\}$, covering exactly the same cycles than before. The feedback arc set obtained is minimalized and $F_1 = \{\{b, a\}, \{e, d\}\}$ is obtained. e) Computation of F_2: the only arc that is in F_1 and not in $F*$ is $\{e, d\}$ and therefore $v_2 = d$. Thus the arc $\{e, d\}$ is exchanged by the outgoing arcs from d, in this case only the arc $\{d, a\}$, covering exactly the same cycles than before. The feedback arc set obtained is minimalized and $F_2 = \{\{b, a\}, \{d, a\}\}$ is obtained. As $F_2 = F*$, the algorithm finishes.

The second idea, that is in fact the enumeration algorithm, is to build a graph ϕ where

$V(\phi)$ is the set of all MFASs of a graph G and ϕ has an arc between two vertices $F, F' \in V(\phi)$ if $s(F, v) = F'$ for some vertex v. A nice property of ϕ is that it is strongly connected given the previous result on the transformation of any MFAS into another by applying the successor function. The enumeration algorithm consists in dynamically building ϕ starting from any initial MFAS and traversing (finding) its nodes by the use of the successor function for all possible $v's$. In order to guarantee that only nodes that were not yet expanded are considered, a dictionary to keep track of already traversed nodes is used. This approach achieves the goal of producing a new MFAS with a delay of $O(|V||E|(|V| + |E|))$ per minimal solution. In this case, the proposed algorithm may be said to be efficient, since a polynomial-delay method is the best one may expect for an enumeration problem.

Chapter 3

Metabolic Stories

Contents

3.1 Introduction

The theoretical results presented in this chapter are strongly based on our paper Acuña et al. (2012a) while the biological application will soon be submitted to a Computational Biology journal. A classical goal of metabolic studies is to try to understand which are the metabolic processes involved in the adaptation to an environmental change. Recently, metabolomic techniques gained the spotlight by providing a way to monitor metabolism by measuring the concentration of metabolites in different conditions or at different time points. A typical result from such an experiment is a list of metabolites whose concentrations significantly changed when the cell was exposed to some stress. How to interpret this list became then a new research topic, consisting in identifying the metabolic processes that link the metabolites of interest, possibly explaining the observed variations in their concentrations.

The simplest idea one may think of is to simply highlight the set of metabolites identified in the experiment, let us call them **interesting compounds**, and then to visually analyse their interconnections. For genome scale networks, the metabolism of a whole organism is considered, which may thus be very large (Thiele and Palsson (2010)), while a metabolic perturbation caused by some stress condition may impact only a small portion of this complex

network. However, even if in some cases it might be possible to keep track of some monitored metabolites, the internal mechanisms that lead to the observed variation are not easy to identify. As an example of how a visual analysis of such processes may be difficult, consider Figure 3.1 that shows a representation of the metabolic network of *Saccharomyces cerevisiae*, which contains more than a thousand reactions and metabolites, and some highlighted nodes that correspond to the metabolites identified in a metabolomics experiment in which the yeast cell was exposed to cadmium. Nodes in red are the ones corresponding to the metabolites whose concentration decreased after exposition to cadmium while the green nodes correspond to an increase of concentration.

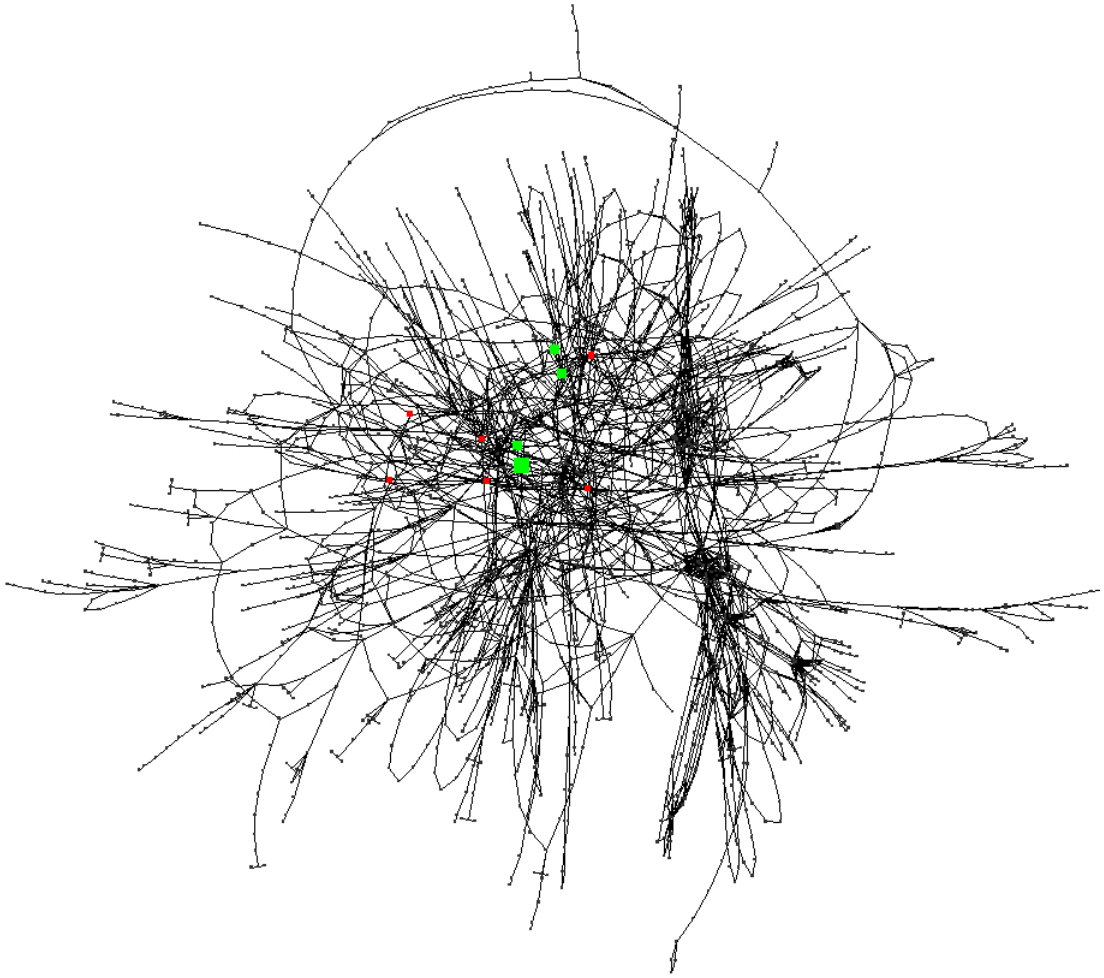


Figure 3.1: Graph representation of the metabolic network of Yeast (Model IMM904) with 6 nodes highlighted in red (methionine, serine (mitochondrial and cytoplasmatic), glycine (mitochondrial and cytoplasmatic) and homoserine) and 4 in green (cystathionine, gamma-glutamyl-cysteine, reduced glutathione, cysteinylglycine) that correspond to metabolites whose concentration decreased (red) or increased (green) during a metabolomics experiment.

Recently, automatic methods have been proposed to deal with this kind of data (Dittrich et al. (2008); Antonov et al. (2009); Faust et al. (2010); Leader et al. (2011)). A natural idea is to try to link all interesting compounds through chains of reactions. In the context of Computer Science, a Steiner tree is a minimum cost tree connecting all nodes in a predefined subset called *terminals*. It was therefore natural to explore the computation of Steiner trees

to connect all interesting compounds (Scott et al. (2005); Dittrich et al. (2008)). However, any pair of metabolites may be connected through several alternative paths within a network, which makes the extraction of subgraphs more relevant than the extraction of trees. Another point is that no criterion of minimality could be easily employed in order to correctly predict which is the real underlying process observed in the experiment.

A simplification of the subgraph extraction process is to concentrate on a pair of interesting compounds, searching for subgraphs corresponding to source-to-sink paths between two interesting compounds (van Helden et al. (2002); Croes et al. (2006); Faust et al. (2010)). A step further is the approach based on unifying the shortest paths, limited by some maximum length k, between each pair of metabolites (Antonov et al. (2009)), which can lead to still large networks (if k is too big) or to disconnected ones (if k is too small). In more detail, the method of Antonov et al. (2009) for dealing with the list of interesting compounds consists in building graphs denoted by D_i containing the interesting compounds and the paths between them with length up to i. Thus, D_1 corresponds to a graph containing the interesting compounds and the arcs directly connecting them (if they are present in the original network), while D_2 is D_1 with the addition of any path between two interesting compounds passing through one intermediate node, and so on. For each D_i graph, a p-value P_i is computed as the probability of inferring models of the same or larger sizes, using the same graph as input and a randomly generated list of compounds. The output of the method is therefore a collection of graphs connecting as many interesting compounds as possible through shortest paths. There is a possible filter for outputting only the ones found to be statistically significant. However, the solution may be disconnected, *i.e.*, in the reported models it may happen that not all the interesting compounds are present in the same connected component. Despite this, it may also be that the final model remains large and hard to analyse.

The previously mentioned methods are based only on the topology of the network but one could consider different approaches based on flux distributions over the set of reactions, such as elementary modes (Schuster and Hilgetag (1994); Schuster et al. (1999)) that are minimal subnetworks working at steady-state. There are, however, issues that need to be taken into account if one decides to use constraint-based techniques such as elementary modes enumeration to explain metabolomics experimental data. Flux-based models need stoichiometric values as well as a definition of the boundaries of the system under analysis, which are not always simple to identify, particularly in the case of a metabolomics experiment in which the list of interesting compounds does not directly define the boundaries of the system. Indeed, the very same metabolite may play different roles in different metabolic processes, being source in one, intermediate in a second and target in a third one; for instance *pyruvate* is the input of gluconeogenesis while it is the output in glycolysis. The inability, and the unwillingness to tell, a priori, the role of the interesting compounds in the metabolic stories to be found is a key factor of our approach. In fact, we are interested in metabolic stories that not only connect all the interesting compounds but also establish their individual role in each story.

Our approach for connecting all interesting compounds is also a subgraph extraction technique. We want to find maximal directed acyclic subgraphs whose set of sources and targets are interesting compounds. We call such subgraphs **metabolic stories**. However, we do not simply want to find one single story but to enumerate all of them. This allows us to avoid any *a priori* optimisation criterion, which makes the method general enough to be applied in different contexts. For the specific case of metabolomics data analysis, we propose a score function that assigns value to the stories based on how the concentration of the interesting compounds is observed to vary in the experiment. This procedure allows a very good filter of the solutions, selecting stories that best fit the experimental data.

This chapter is organised in the following way. Section 3.2 presents the motivation behind our proposed modelling of metabolic stories while Section 3.3 formally defines the model and the algorithmic problem. Section 3.4 presents some operations that allow a huge simplification of the input graph without losing solutions as well as a polynomial time algorithm for finding one story, and also a proof that the problem of finding stories with a specific set of sources and targets is NP-complete. Subsections 3.4.3 and 3.4.4 propose two different approaches to enumerate stories: the first one makes use of a minimal feedback-arc-set enumerator but can only be applied to a specific class of graphs while the second is an extension of our algorithm to find one story based on an initial permutation of the nodes, and can be used for any graph. Section 3.5 provides complexity results for an alternative definition of stories. Finally, Section 3.6 presents some biological applications of our method and Section 3.7 points out perspectives and open problems.

3.2 Modelling metabolic stories

Informally, we call a set of metabolic reactions linking all the metabolites of interest a *metabolic story*. For instance, one metabolomics study compared a yeast cell under two conditions, with and without exposition to cadmium (Madalinski et al. (2008)). The metabolic network reconstruction of Yeast has about 1300 metabolites and the experiment identified a list of 24 metabolites whose concentrations changed. Figure 3.1 presents a graph representation of the Yeast metabolic network. The green and red nodes correspond to a subset of the metabolites identified in the experiment, specifically the ones whose concentration changed the most. Green nodes are the metabolites whose concentration increased while the red ones are those whose concentration decreased. The concentration of the other metabolites did not change significantly. Figure 3.2 shows a metabolic story that corresponds to the current knowledge on the metabolic processes under study (Madalinski et al. (2008)). It was found to be part of the set of solutions computed by our method to enumerate metabolic stories that will be detailed in this chapter. The chain of reactions present in this story goes along with the conclusions found in Madalinski et al. (2008), since the order in which the metabolites are transformed - *i.e.*, the flow of matter - is the same. There are several other possible scenarios close to the one of this story that could also be considered and might provide new insights on the underlying metabolic process. Finding and enumerating all these alternatives is what we call the “metabolic stories problem".

First, let us introduce the main features we believe a metabolic story should have. A metabolic story should capture the relationship between all the nodes of interest. Each individual story should explain how some metabolites are derived from others through a chain of reactions. In other words, our definition of a metabolic story should allow us to define a flow of matter from a set of source metabolites to a set of target metabolites. The candidates to be the endpoints (sources or targets) of a story should belong to the set of interesting compounds. Finally, we want each story to explain as many alternative pathways connecting the interesting compounds as possible. In order to ensure these features, we introduce the following constraints: only red and green nodes are allowed to be sources and targets in a story, even if they can appear as intermediate nodes in some stories. In order to guarantee that stories will have at least one source and one target, we introduce the acyclicity constraint. These two combined constraints lead us to search for directed acyclic subgraphs (DAGs) with sources and targets contained in the given subset of interesting compounds. We also want such a DAG to be maximal, in the sense that alternative pathways between these nodes should be included, provided that their addition does not create cycles. Finally, in a first step, we

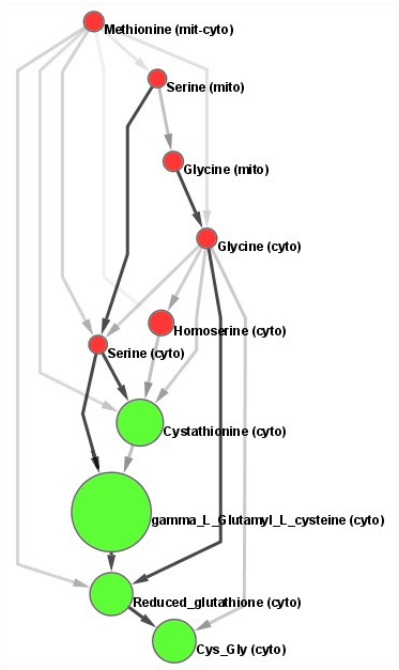


Figure 3.2: One possible metabolic story connecting all metabolites of interest. Each arc in this graph corresponds to a reaction (black arcs) or to a chain of reactions (grey arcs) in the original metabolic network. The colour of the nodes corresponds to decreased (red) or increased (green) concentration while their size corresponds to their concentration change in the metabolomic experiment.

prefer to model red and green nodes as a single set, called **black nodes**. There are two main reasons supporting this decision. The first one is to keep our definition more general, since it may be that more than two conditions are being evaluated in a metabolomics experiment and in such case, the roles of the interesting compounds are not clearly defined. The second reason is that this more general definition allows us to explore all possible scenarios including the ones in which red nodes are targets or green nodes are sources, and these may correspond to interesting biological cases that would not be considered otherwise.

More formally, we introduce a constrained version of the problem of enumerating all maximal directed acyclic subgraphs of a graph G (Schwikowski and Speckenmeyer (2002)). In our version, only a given subset $\mathbb{B}$ of the nodes are allowed to be sources or targets of the DAGs to be enumerated. The subset $\mathbb{B}$ corresponds to compounds that have been experimentally identified. The aim is then to extract all the interaction dependencies among the compounds in $\mathbb{B}$ which do not create cycles, but at the same time involve as many compounds and reactions as possible. These may require intermediate steps that concern compounds not in $\mathbb{B}$, but the initial and final steps must involve only compounds in $\mathbb{B}$. A solution, that is a possible scenario of metabolic dependencies, is called a *(metabolic) story*. The problem is then to "tell" all possible stories given as input a graph G and a subset $\mathbb{B}$ of the nodes of G.

The idea of connecting a set of nodes in a graph suggests that our problem could be related to a Steiner problem, whose applications have been already widely explored in biology (Betzler (2005)) for different problems. However, a major difference is that Steiner problems look for minimal structures while we are searching for a maximal one. On the other hand, enumerating maximal DAGs in a graph is equivalent to enumerating feedback arc sets (FASs), which is also a widely studied problem. However, as we will later show, the constraint on the

sets of sources and targets is enough to change the nature of the problem. More precisely, as previously presented in more detail in Chapter 2, a feedback arc set is a minimal set of arcs that break all the cycles, *i.e.* it is the complement of a DAG. In this sense, enumerating stories is a generalisation of enumerating FASs, since the complement of a story is a minimal set of arcs that breaks all the cycles and also avoids sources or targets that are not in $\mathbb{B}$. We call such minimal sets of arcs *story arc sets* (SASs). Hence every SAS is a FAS. We show that indeed not every FAS is a SAS, and give evidence that telling stories is possibly harder than enumerating feedback arc sets.

3.3 Definitions

Let $G = (\mathbb{B} \cup \mathbb{W}, R)$ be a directed graph such that $\mathbb{B} \cap \mathbb{W} = \emptyset$. We write $V = \mathbb{B} \cup \mathbb{W}$. Nodes in $\mathbb{B}$ are said to be *black* while those in $\mathbb{W}$ are said to be *white*. Let $d^+(u)$ and $d^-(u)$ denote, respectively, the in-degree and the out-degree of a node u. Node u is called a *source* if $d^+(u) = 0$ and $d^-(u) > 0$ and a *target* if $d^-(u) = 0$ and $d^+(u) > 0$.

A **pitch** of G is an acyclic subgraph $G' = (\mathbb{B} \cup \mathbb{W}', R')$ of G with $\mathbb{W}' \subseteq \mathbb{W}$ and $R' \subseteq R$ and, for each node $w \in \mathbb{W}'$, $d^+(w) > 0$ and $d^-(w) > 0$. A trivial pitch is $G' = (\mathbb{B} \cup \emptyset, \emptyset)$: the subgraph containing all the black nodes and no arc. We define a **story** as a maximal pitch. We denote by $\Sigma(G)$ the set of stories of G.

Problem ENUM-STORIES(G): Given $G = (\mathbb{B} \cup \mathbb{W}, R)$ enumerate $\Sigma(G)$.

For independent reading, we define a **feedback arc set** (FAS) of a directed graph $G = (V, E)$, which is a subset F of E such that $G_{F^-} = (V, E \setminus F)$ is acyclic. A FAS is said to be *minimal* if there exists no $f \in F$ such that $F \setminus \{f\}$ is a FAS. We notice that, if $V = \mathbb{B} \cup \mathbb{W}$, the complement of a FAS is not always a story since G_{F^-} may contain white sources or targets. Indeed, the FAS enumeration problem is a particular instance of our problem in which every node is black, *i.e.*, $\mathbb{W} = \emptyset$. We define a **story arc set** (SAS) as a FAS S with the extra property that no white node in G_{S^-} is a source or a target. A SAS is said to be *minimal* if there exists no subset S' of S such that $S \setminus S'$ is a SAS. This implies that if S is minimal, then for every $s \in S$, the graph $G_{S^-,s^+} = (\mathbb{B} \cup \mathbb{W}, (E \setminus S) \cup \{s\})$ either contains a cycle or contains a white source or target. If S is a minimal SAS, then G_{S^-} is a story. A SAS is also a FAS. However, the example in Figure 3.3 shows that, as expected, not every minimal FAS is a minimal SAS and, more surprisingly, that not every minimal SAS is a minimal FAS.

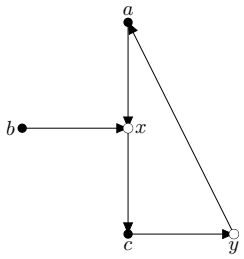


Figure 3.3: In this case, $\mathbb{B} = \{a, b, c\}$ and $\mathbb{W} = \{x, y\}$. There are 4 possible minimal FASs: $\{(a, x)\}$, $\{(x, c)\}$, $\{(c, y)\}$, and $\{(y, a)\}$. Only one of these minimal FASs (that is, the first one) is also a minimal SAS. For example, the second one is not a SAS since $G_{\{(x,c)\}^-}$ contains a white target (that is, node x). On the other hand, another minimal SAS is $\{(c, y), (y, a)\}$, which is not a minimal FAS (even though it is a FAS).

In order to enumerate all stories, one might consider using the polynomial-time delay enumeration algorithm for minimal FAS proposed in Schwikowski and Speckenmeyer (2002): however, as in the example in Figure 3.3 this FAS enumerator would not solve our problem since some minimal SAS may not be detected.

We shall see in a next section that this is not the case when we restrict ourselves to some particular class of graphs.

3.4 Algorithms and complexity for finding and enumerating stories

In this section, we present five operations for preprocessing the graph in order to reduce it while focusing only on the interactions between the black nodes. The graph obtained after this preprocessing, that we call **compressed network**, captures in a very compact way the relationships between these nodes. The first preprocessing consists in simply computing all-pairs lightest paths between all black nodes and to compute a new graph with their union. This first step is specially important when the method is applied in metabolic networks represented as compound graphs as detailed further and could be avoided if one wants to really examine all possible paths between black nodes.

The other four operations derive immediately from our formal definition of a metabolic story, taking advantage of the topological properties that may be inferred allowing simplifications on the input graph without losing solutions. This preprocessing step is a first practical result since it allows a reduction of the size of the input graph to less than 20% of its initial size. These much smaller graphs, focused on the interactions between the black nodes only, are already valuable objects that may be visually analysed by biologists.

Opposite to the metabolic story that is an acyclic object, the compressed network contains usually plenty of cycles. Naturally, while visually inspecting such a structure we concentrate ourselves on how some subset of the nodes follow a chain of reactions to be transformed into another subset of products. A particular case of this problem that is worth investigating both from an algorithmic and also from a biological point of view is: given a fixed set of sources and targets, can we find a story with these sets? In terms of algorithm complexity, the answer to this question is no and we provide a NP-completeness proof for the problem of finding a single story with a fixed set of sources and targets.

Another way to explore the alternative transformation scenarios hidden in a compressed network is exactly to enumerate the metabolic stories contained on such a graph. We show then two different algorithms to perform this task. The first one is to use a minimal feedback arc set enumeration algorithm (Schwikowski and Speckenmeyer (2002)) but this approach is restricted to instances where no so-called **bad nodes** are present. The second one is our algorithm that explores the relationship between total orders of the nodes and a story.

3.4.1 Preprocessing the graph

First, we show how a graph may be simplified without essentially changing the set of its stories. The simplifications allow, from a theoretical point of view, shorter proofs of our results and, from a biological point of view, the simplified graphs obtained by these preprocessing steps turn out to be interesting since they correspond to a more compact representation of graphs that is equivalent in terms of story sets.

For the applications of the stories enumeration method described in this work, the input graph is a compound graph representation of a metabolic network. In this case, not all paths

between black nodes are biologically meaningful. Metabolic reactions are in many cases of the form $m_1 + s_1 \rightarrow m_2 + s_2$, where m_1 and m_2 are the main compounds and s_1 and s_2 are so-called side compounds. For example, a typical pair of side compounds is ATP and ADP, that appear in many reactions with the role of providing energy for the reaction to take place. The problem, however, is that a path from m_1 to ATP will be present in the compound graph even if there was no direct exchange of atoms between these two compounds. Consider a second reaction transforming m_3 into m_4 that also uses these side compounds. Figure 3.4 shows a shortcut connecting m_1 to m_4 passing through ATP. Notice that we could also connect m_3 to m_2 simply passing through ADP. Considering that a large portion of the reactions are reversible, it is clear that almost any pair of nodes may be connected with a few steps. One way to avoid these unrealistic paths is to compute a graph corresponding to the union of all lightest paths between all pairs of black nodes and to eliminate the remaining of the network, given some definition of arc weight.

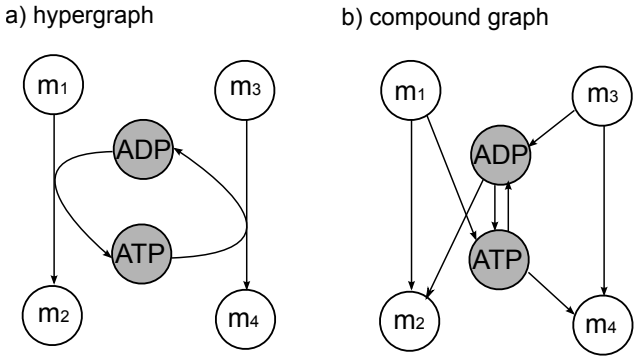


Figure 3.4: An example of shortcuts in a metabolic network caused by the presence of hubs like ATP and ADP. Notice how in the compound graph representation there is a path from m_1 to m_4 passing through ATP that does not exist in the hypergraph representation.

For the computed paths to be more biologically meaningful than simply the shortest ones in terms of reactions, we may adopt as arc weight the out-degree of the target vertex. In this way, the weight of a path is the sum of the weights of the arcs in the path. Naturally, ATP and ADP have high degrees in the compound graph since they are involved in a large number of reactions as side compounds and, therefore, shortcuts passing through them tend to be avoided with the selected weight policy. An even more sophisticated approach to compute meaningful paths is to consider the exchange of carbon atoms between the metabolites and to trace a route between the source and the target of interest in such a way that in every step there is a carbon atom passing from one metabolite to the other (Boyer and Viari (2003); Arita (2004); Blum and Kohlbacher (2008)). However, this approach needs more detailed data and algorithms, such as a graph representation of the chemicals of the whole network, that were not available at the time our experiments were performed. We intend, though, to use this approach in the near future in order to check how it improves our results, specially for the application on automatically recovering metabolic pathways that will be further described later. In the remaining of this chapter, we consider that the input graph is a collection of lightest paths between all-pairs of black nodes.

We now define the following four simplification operations:

- A **white source and target removal** consists in removing iteratively a white node from the graph that is either a source or a target. Clearly such nodes cannot appear in any story. Let $\mathtt{de}(G)$ be the graph resulting from such removals.

- A **self-loop removal** consists in removing all arcs of the form (u, u). Since stories are acyclic, such arcs do not appear in any story. Let $\texttt{sl}(G)$ be the graph resulting from such removals.
- A **forward bottleneck removal** consists in removing a white node v whose out-degree is equal to 1, and directly connecting any predecessor of v to the unique successor of v (without creating multi-arcs). Let $\texttt{fb}(G, v)$ be the resulting graph.
- A **backward bottleneck removal** consists in removing a white node whose in-degree is equal to 1, and directly connecting the unique predecessor of v to the successors of v (without creating multi-arcs). Let $\texttt{bb}(G, v)$ be the resulting graph.

We prove that the last two operations leave the set of stories essentially unaltered. First an observation:

Observation 1. *Let v, p, and s be three nodes such that $(p, v), (v, s), (p, s) \in E$ and v is a (white) bottleneck. Then, for any story S, $(p, v), (v, s) \in S$ if and only if $(p, s) \in S$.*

Proof. The lemma follows from two observations. First, (p, v) and (v, s) belong to cycle C of G if and only if the arc (p, s) belongs the cycle C' of G, which contains, next to (p, s), all the arcs of C except (p, v) and (v, s). Second, (p, s) and the pair $(p, v), (v, s)$ will create the same white sources or targets, if any. □

Given three nodes $v, p, s \in V$ with $(p, v), (v, s) \in E$ and $(p, s) \notin E$, let $\texttt{ab}(G, v, p, s)$ denote the graph obtained by adding to G the arc (p, s).

Lemma 1. *Let $v \in \mathbb{W}$ be a forward bottleneck and let $p, s \in V$ be such that $(p, v), (v, s) \in E$ and $(p, s) \notin E$. Then there exists a bijection from $\Sigma(G)$ to $\Sigma(\texttt{ab}(G, v, p, s))$.*

Proof. For any story $S \in \Sigma(G)$, we define $f(S) = S \cup \{(p, s)\}$ if $(p, v) \in S$ (and hence, $(v, s) \in S$ since v is a forward bottleneck), otherwise $f(S) = S$. To prove that $f(S) \in \Sigma(\texttt{ab}(G, v, p, s))$, we use Observation 1 to show that $f(S)$ is acyclic if and only if S is acyclic. We now show that $f(S)$ is maximal. Indeed, if $(p, s) \in f(S)$, then no set of arcs could be added to $f(S)$ since otherwise it could also be added to S. Otherwise, if (p, s) could be added to $f(S)$, then, from Observation 1 also (p, v) and (v, s) could be added to $f(S)$ and, hence, these two arcs could be added to S.

Let us now prove that, if S_1 and S_2 are two stories such that $S_1 \neq S_2$, then $f(S_1) \neq f(S_2)$. If $(p, v) \notin S_1 \cup S_2$, then $f(S_1) = S_1 \neq S_2 = f(S_2)$. Otherwise, if $(p, v) \in S_1 \cap S_2$, then $f(S_1) = S_1 \cup \{(p, s)\} \neq S_2 \cup \{(p, s)\} = f(S_2)$. Finally, if $(p, v) \in S_1 \setminus S_2$ (the other case can be dealt with similarly), then $(p, s) \in f(S_1)$ while $(p, s) \notin f(S_2)$ and, hence, $f(S_1) \neq f(S_2)$.

It remains to show that, for any $S' \in \Sigma(\texttt{ab}(G, v, p, s))$, there exists a $S \in \Sigma(G)$ such that $f(S) = S'$. Define $S = S' \setminus \{(p, s)\}$. Since S' is acyclic, so is S. If $(p, s) \notin S'$, then $S = S'$ and $S \in \Sigma(G)$, since the only difference between G and $\texttt{ab}(G, v, p, s)$ is the arc (p, s). Otherwise, from Observation 1, it follows that $(p, v), (v, s) \in S'$ and, hence, $(p, v), (v, s) \in S$: the maximality of S then follows from the maximality of S', since any set of arcs that could be added to S could also be added to S'. □

By this lemma we may assume that, for any forward bottleneck $v \in \mathbb{W}$ whose unique successor is s, and for any predecessor p of v, the graph contains the arc (p, s). To complete the forward bottleneck removal operation, we then need to delete the vertex v without changing the stories set of the graph. Consider now the following operation: given a graph G with a forward bottleneck v, $\texttt{dp}(G, v)$ denote the graph obtained by deleting from G the vertex v and all incident arcs.

Lemma 2. *Let $v \in \mathbb{W}$ be a forward bottleneck and s its unique successor. Suppose that for any predecessor p of v, the graph contains the arc (p, s). Then there is a bijection from $\Sigma(G)$ to $\Sigma(\mathtt{dp}(G, v))$.*

Proof. For any $S \in \Sigma(G)$, we define $f(S) = S \setminus \{v\}$, that is the subgraph obtained by removing v and all incident arcs from S if $v \in S$. Since S is acyclic, so is $f(S)$. Moreover, from Observation 1, it follows that if $(p, v), (v, s) \in S$, then $(p, s) \in S$ and, hence, $(p, s) \in f(S)$. The maximality of $f(S)$ then follows from the maximality of S, since any set of arcs that could be added to $f(S)$ could also be added to S.

Let us now prove that, if S_1 and S_2 are two stories such that $S_1 \neq S_2$, then $f(S_1) \neq f(S_2)$. If $(p, s) \notin S_1 \cup S_2$, then $(p, v), (v, s) \notin S_1 \cup S_2$ and $f(S_1) = S_1 \neq S_2 = f(S_2)$. Otherwise, if $(p, s) \in S_1 \cap S_2$, then $(p, v), (v, s) \in S_1 \cap S_2$ and $f(S_1) = S_1 \setminus \{(p, v), (v, s)\} \neq S_2 \setminus \{(p, v), (v, s)\} = f(S_2)$. Finally, if $(p, s) \in S_1 \setminus S_2$ (the other case can be dealt with similarly), then $(p, s) \in f(S_1)$ while $(p, s) \notin f(S_2)$ and, hence, $f(S_1) \neq f(S_2)$.

Finally, let S' be a story of $\mathtt{dp}(G, v)$. Then S obtained by adding to S' the path $(p, v), (v, s)$ for every predecessor p of v such that $(p, s) \in S'$ is clearly a story and $f(S) = S'$. □

Using the two previous lemmas, we obtain a justification for the third simplification operation.

Theorem 1. *For any forward bottleneck $v \in \mathbb{W}$, $\Sigma(G) = \Sigma(\mathtt{fb}(G, v))$.*

Analogously, we can justify the fourth operation.

Theorem 2. *For any backward bottleneck $v \in \mathbb{W}$, $\Sigma(G) = \Sigma(\mathtt{bb}(G, v))$.*

For any graph G, let $\mathtt{fb}(G)$ (respectively $\mathtt{bb}(G)$) denote the graph obtained by applying as many times as possible the forward (respectively backward) bottleneck removal operation. Notice that, even if G does not contain self-loops, it might happen that $\mathtt{fb}(G)$ (respectively $\mathtt{bb}(G)$) contains self-loops created by one bottleneck removal. Remember also that $\mathtt{sl}(G)$ denotes the graph obtained by the removal of all self-loops from G and $\mathtt{de}(G)$ denotes the graph obtained by the iterative removal of all white sources and targets from G. Our simplification procedure can now be described as follows.

(1) Let $G_0 = \mathtt{sl}(\mathtt{de}(G))$ and let $i = 0$.

(2) Let $G_{i+1} = \mathtt{sl}(\mathtt{bb}(\mathtt{sl}(\mathtt{fb}(G_i))))$.

(3) If $G_{i+1} = G_i$ then return G_i, otherwise let $i = i + 1$ and go to Step 2.

As a consequence of the previous results, we have that if H is the graph returned by this procedure, then there is a bijection between $\Sigma(G)$ and $\Sigma(H)$, and we may enumerate $\Sigma(H)$ instead. Hence from now on, we assume that any $v \in \mathbb{W}$ has $d^+(v) > 1$ and $d^-(v) > 1$. Notice that this avoids graphs like the one shown in Figure 3.3. Indeed, in this case, the two arcs (c, y) and (y, a) would disappear and the arc (c, a) would be inserted. Furthermore, also x will disappear and we get arcs (b, c) and (a, c). Observe also that this simplification procedure does not guarantee that a minimal FAS enumerator would produce all possible minimal SAS.

We applied the preprocessing steps described in this section on a collection of 107 metabolic networks obtained from MetExplore (Cottret et al. (2010a)). We randomly chose sets of black nodes with sizes varying from 5% to 15% of the total number of nodes of the graph. For each pair of metabolic network and set of randomly chosen black nodes we then computed subgraphs corresponding to the lightest paths between all pairs of black nodes. These subgraphs

vary on number of vertices from 42% to 98% with respect to the number of vertices in the original input graph and from 46% to 68% with respect to the number of arcs. In average, extracting all lightest-paths between the black nodes gives a graph with 68% of the nodes and 69% of the arcs of the original input graph. Over this new collection of graphs we applied then the four simplification operations: white source and target removal, self-loops elimination and forward and backward bottlenecks removals. The compression ratio on the number of nodes goes from 65% to 98% with an average reduction of 83%, while the compression ratio on the number of arcs goes from 56% to 99% with an average reduction of 77%, with respect to the original input graph. Overall it is 60% of reduction due to the lightest paths and an additional 20% because of the graph simplifications. This more compact representation of the interactions between black nodes greatly facilitates the visualisation and analysis of the input data.

3.4.2 Finding single stories

We move now to the problem of finding some story. We show that this can be done in polynomial time. Our algorithm basically starts with a pitch and grows it into a story by adding paths between black nodes while avoiding cycles. We can start with a trivial pitch such as the subgraph containing all the black nodes and no arcs. We present the algorithm COMPLETE_PITCH for completing a pitch into a story. Figure 3.5 presents an example for which the algorithm is applied to a graph containing five black nodes and one white node, completing a starting pitch that contains only the black nodes and no arcs. Assuming lexical ordering of the nodes, the algorithm traverses each of them trying to find paths from this initial node to other nodes belonging to the pitch. Once such paths are found they are examined to verify if their addition creates cycles and, in the negative case, the paths are added to the pitch. A partial order of the nodes is inferred from the pitch and it is used in order to decide whether the addition of a new path $a \rightsquigarrow b$ will generate a cycle. This verification is simply done by verifying on the partial order if a and b are incomparable or if a comes before b and both cases are safe. After any modification of the pitch the partial order of the nodes have to be updated. The algorithm finishes when all nodes are traversed and no new path may be added, *i.e.*, the pitch is now a story.

Algorithm COMPLETE_PITCH(G, P)
Require: a graph $G = (\mathbb{B} \cup \mathbb{W}, R))$ with $\mathbb{B} \cap \mathbb{W} = \emptyset$ and an initial pitch P;
Ensure: A story completing P

```
i ← 1
π ← any topological order of P
while i ≤ |V(P)|) do
    u ← i-th element according to π with u ∈ V(P)
    Apply BFS(u, G \ E(P)) until reach a node v ∈ V(P)
    if π(u) < π(v) ∨ (u and v are incomparable) then
        include the path u ⇝ v in P and update π
        i ← 1
    else if no such node v exists then
        i ← i + 1
return P
```

Theorem 3. *A story can be determined in polynomial time.*

Proof. The algorithm COMPLETE_PITCH determines a story by completing a starting pitch P. It chooses a topological order π of the nodes consistent with the pitch. Starting in u, which can be any of the first nodes in this order that has not been scanned yet, a breadth-first search

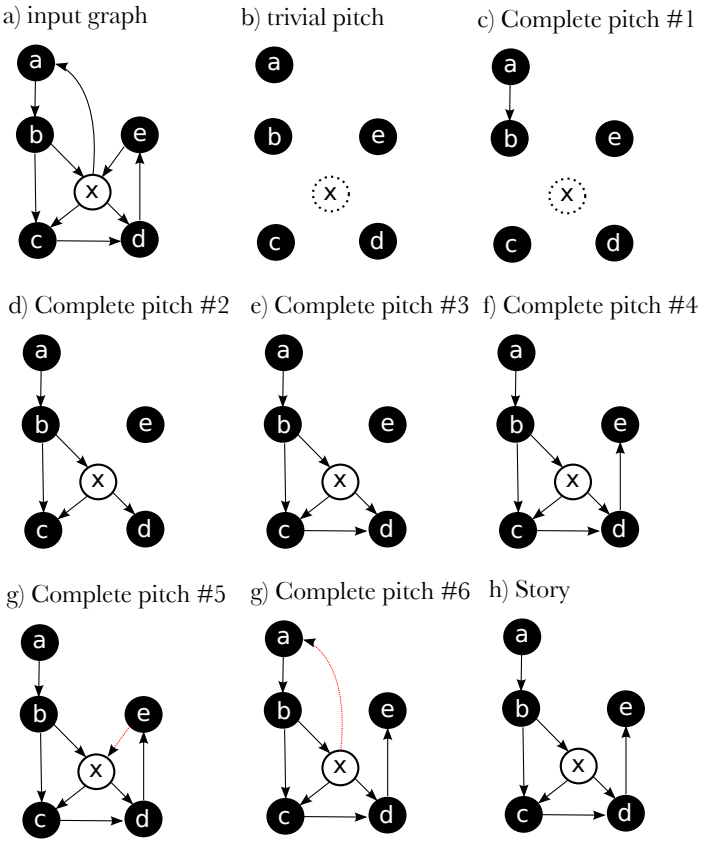


Figure 3.5: a) The input graph with set of black nodes $B = \{a, b, c, d, e\}$ and white nodes $W = \{x\}$. b) The starting pitch, which is simply a graph $V = B$ and no arcs. c) The path $a \rightsquigarrow b$ is added to the pitch. d) Three paths starting from b are added to the pitch: $b \rightsquigarrow c$, $b \rightsquigarrow x \rightsquigarrow c$ and $b \rightsquigarrow x \rightsquigarrow d$. Notice that as x did not belong to the pitch at this point, the algorithm goes further and stops only when nodes in the pitch are found. e) The path $c \rightsquigarrow d$ is added to the pitch. f) The path $d \rightsquigarrow e$ is added to the pitch. g) The path $e \rightsquigarrow x$ is evaluated but may not be added since e comes after x in the current partial order inferred from the pitch, and therefore such an addition creates at least one cycle, for instance $e \to x \to d \to e$. h) The path $x \rightsquigarrow a$ is evaluated but may not be added since x comes after a in the current partial order inferred from the pitch, and therefore such an addition creates at least one cycle, for instance $x \to a \to b \to x$. i) There is no more nodes to traverse, the final object is a maximal pitch, *i.e.*, a story.

(BFS) is performed using only arcs not in $E(P)$. Any branch of the BFS tree is pruned as soon as it hits a vertex $v \in V(P)$. If v has $\pi(u) < \pi(v)$ or u and v are incomparable, then the path $u \rightsquigarrow v$ is added to P and the topological order is updated. This addition creates no cycle since there was no path $v \rightsquigarrow u$ in P due to the fact that $\pi(u) < \pi(v)$ or u and v were incomparable, which can be checked in polynomial time. Moreover, since P contained no white source nor target before the addition of the path, then it does not contain any after adding the path because u and v, which are the only candidates to become source or target, were already present in P. Hence, the addition of $u \rightsquigarrow v$ to P creates a new pitch.

This procedure is repeated until no new path starting from u can be found. At this point, we continue with the next node in the updated order π. Every time a new path is found, π is updated and the procedure is started from the minimum node according to the new order. Since at each updating of the topological order, we add at least one arc, the algorithm terminates in polynomial time. The final pitch produced by this procedure is maximal and, therefore, a story. □

We proceed by showing that the problem becomes NP-complete if we wish to identify a specific single story, *i.e.*, one having a particular set of sources and/or targets.

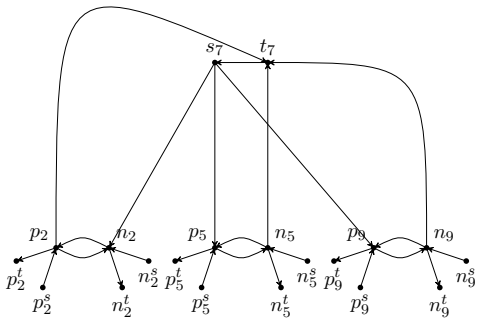


Figure 3.6: The subgraph corresponding to the clause $C_7 = \neg x_2 \vee x_5 \vee x_9$

Theorem 4. *Deciding whether there exists a story with a given set of sources and targets is NP-complete.*

Proof. In order to prove this theorem, we show how the `3-SAT` problem (Garey and Johnson (1990)) is reducible to the problem of deciding whether, given a directed graph $G = (V, E)$ and two subsets S and T of V, G contains a maximal DAG with its set of sources equal to S, and its set of targets equal to T. If this is true for maximal DAGs, it is also true for stories since any story is also a maximal DAG.

Consider a 3-CNF Boolean formula φ with clauses C_i, $i = 1, \ldots, m$, over a set Boolean variables x_j, $j = 1, \ldots, n$. We define a directed graph G as follows (see also Figure 3.6).

- For each variable x_j, we create a set of six nodes, $p_j, p_j^s, p_j^t, n_j, n_j^s, n_j^t$, and for each clause C_i, two nodes s_i and t_i. We define the set $S = \{p_j^s, n_j^s \mid j = 1, \ldots, n\} \cup \{s_i \mid i = 1, \ldots, m\}$ and the set $T = \{p_j^t, n_j^t \mid j = 1, \ldots, n\} \cup \{t_i \mid i = 1, \ldots, m\}$.
- The set of arcs of G includes the six arcs

  $$(p_j^s, p_j), (p_j, p_j^t), (p_j, n_j), (n_j, p_j), (n_j^s, n_j), (n_j, n_j^t)$$

  related to each variable x_j and the arc (t_i, s_i) for each clause C_i.
- For each clause $C_i = l_i^1 \vee l_i^2 \vee l_i^3$, we introduce for each literal two arcs: if $l_i^h = x_j$ then we create the arcs (s_i, p_j) and (n_j, t_i), and if $l_i^h = \neg x_j$ the arcs (s_i, n_j) and (p_j, t_i), $h = 1, 2, 3$.

We prove that φ is satisfiable if and only if G includes a maximal DAG whose sets of sources and targets are, respectively, S and T.

Suppose φ is satisfiable and let τ be a satisfying truth-assignment. In the FAS F we include the arc (n_j, p_j) if $\tau(x_j) =$ `true` and the arc (p_j, n_j) if $\tau(x_j) =$ `false`. Moreover, for each clause C_i, we include in F the arc (t_i, s_i) (see Figure 3.7). Clearly, the resulting subgraph G_{F^-} is a DAG whose set of sources (respectively, targets) is equal to S (respectively, T). Moreover, G_{F^-} is maximal since removing any arc from F would create either a two-node variable cycle or, for some clause C_i, at least one six-node cycle corresponding to a true literal in C_i.

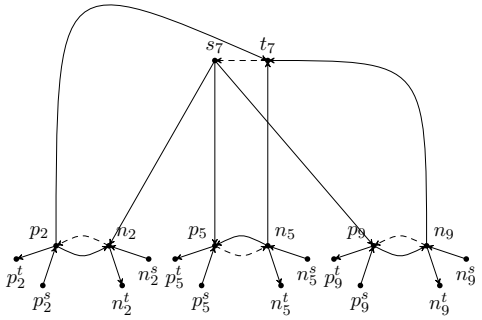


Figure 3.7: The directed acyclic subgraph corresponding to the truth assignment $\tau(x_2) = \texttt{true}$, $\tau(x_5) = \texttt{false}$, and $\tau(x_9) = \texttt{true}$ that satisfies the clause $C_7 = \neg x_2 \vee x_5 \vee x_9$: the dashed arcs are in the FAS

Now suppose that G' is a maximal DAG with sources S and targets T. Clearly, for each clause C_i, the arc (t_i, s_i) is not in G'. Maximality of G' implies that for each variable x_j, exactly one of (p_j, n_j) and (n_j, p_j) is in G'. All other arcs are included in G'. Let τ be a truth-assignment defined as follows: for each variable x_j, $\tau(x_j) = \texttt{true}$ if and only if (p_j, n_j) is in G'. We prove that this assignment satisfies φ. Suppose, to the contrary, that there exists an unsatisfied clause C_i. Wlog we may assume that $C_i = x_1 \vee x_2 \vee x_3$ (see Figure 3.8). Then the three cycles containing the arc (t_i, s_i) are broken both by this arc and by the three arcs (n_j, p_j), $j = 1, 2, 3$ not in G'. Hence, G' is not maximal since the arc (t_i, s_i) can be added to G' without creating any new cycle. This contradicts the hypothesis on G'. □

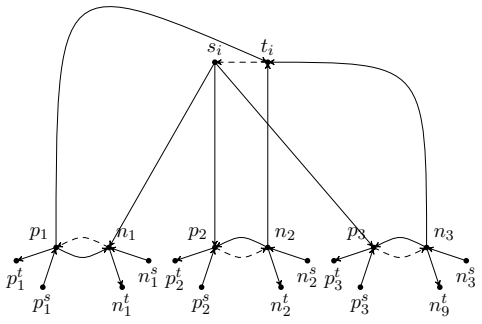


Figure 3.8: A directed acyclic subgraph (the dashed arcs are in the FAS) corresponding to the truth assignment $\tau(x_2) = \texttt{true}$, $\tau(x_5) = \texttt{false}$, and $\tau(x_9) = \texttt{false}$ that does not satisfy the clause $C_7 = \neg x_2 \vee x_5 \vee x_9$: the DAG is not maximal since the arc (t_7, s_7) can be taken out from the FAS.

It is easy to modify the previous reduction in order to prove that the same result holds even if we specify only the set of sources *or* only the set of targets.

3.4.3 Enumerating stories by enumerating FASs

We already noticed that there exist graphs for which the set $\mathcal{S}(G)$ of minimal SASs and the set $\mathcal{F}(G)$ of minimal FASs are not comparable in terms of the inclusion relation. In this section, we show that, for some particular cases, $\mathcal{S}(G)$ is contained in $\mathcal{F}(G)$.

A white node $v \in \mathbb{W}$ is called **bad** if, for any predecessor p of v and for any successor s of v, there exists a cycle containing the arcs (p, v) and (v, s) (see Figure 3.9).

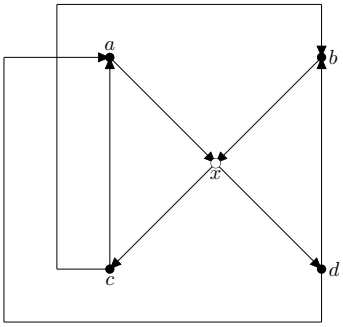


Figure 3.9: Example of a bad node. The minimal SAS $\{(a, x), (b, x), (x, c), (x, d)\}$ is not a minimal FAS.

Proposition 1. *If G does not include any bad node, then any minimal* SAS *is a minimal* FAS.

Proof. By absurdum, assume that S is a minimal SAS which is not a minimal FAS. Then, there exists an arc $e = (u, v) \in S$ such that $S \cup \{e\}$ is a FAS but not a SAS, that is, in G_{S^-,e^+} either u is a white source or v is a white target. We can restrict ourselves to consider the latter case, since the former one can be dealt with in a similar way. According to the results of the previous section, we can assume that the in-degree and the out-degree of v are both greater than 1. Since in G_{S^-,e^+} v is a white target, we have that all arcs incident to v are in S. Moreover, since v is not bad, there exists a predecessor p and a successor s such that (p, v) and (v, s) are not included in the same cycle. Let $S_0 = S - \{(p, v), (v, s)\}$. In $G_{S_0^-}$ the node v is neither a source nor a target. Moreover, $G_{S_0^-}$ is acyclic. Indeed, any cycle including (p, v) (respectively, (v, s)) is hit by an arc outgoing from (respectively, incoming to) v different from (v, s) (respectively, (p, v)). However, it might be that either p is now a source or s is a now a target. Let us analyse the first case (since the the other case can be analysed in a similar way). According to the results of the previous section, we can assume that there exists a path $p_k, p_{k-1}, \ldots, p_1, p$ with $k \geq 1$ such that p_k is black and p_i is white, for any i with $1 \leq i < k$. Let j be the minimum i with $1 \leq i < k$ such that p_i is not isolated in $G_{S_0^-}$: if no such j exists, then we define $j = k$. We show that $S' = S_0 - \{(p_j, p_{j-1}), \ldots, (p_1, p)\}$ is a SAS, thus proving that S is not minimal. Clearly, no white node in $G_{S'^-}$ is a source or a target. Moreover, if (p_h, p_{h-1}) belongs to a cycle, then either another arc of this cycle incident to p_{h-1} belongs to S' or (p_{h-1}, p_{h-2}) belongs to the same cycle. Inductively, we have that either this cycle is hit by an arc in S' or it includes (p, v): in this latter case, the cycle is hit by another arc in S' outgoing from v. □

Proposition 2. *Any $v \in \mathbb{W}$, which is not bad, belongs to every story.*

Proof. Consider a pitch P not containing v. As v is not bad, it has a predecessor p and a successor s such that there exists no cycle containing the arcs (p, v) and (v, s). By simplification rule 2, there exists a path $p_k, p_{k-1}, \ldots, p_1 = p$ with $k \geq 1$ such that $p_k \in \mathbb{B}$ and $p_i \in \mathbb{W}$, for any i with $i < k$. Let j be the minimum $i < k$ such that $p_i \in P$: if no such j exists, then we define $j = k$. Similarly a path $s = s_1, \ldots, s_{\ell-1}, s_\ell$ ending in a black node exists, and let $s_{j'}$ be the first node on that path belonging to P, or $s_{j'} = s_\ell$ if no such node exists.

Then $P' = P \cup \{(p_j, p_{j-1}), \ldots, (p, v), (v, s), \ldots, (s_{j'-1}, s_{j'})\}$ has no white source nor target as p_j and $s_{j'}$ are not white sources or targets in P. Moreover, P' is acyclic as P is acyclic and any cycle containing the additional path would contradict the fact that v is not a bad node. Thus any pitch not containing v is not maximal, hence not a story. □

Corollary 1. *If G does not include any bad node, then any minimal* SAS *is a minimal* FAS.

Proof. By absurdum, assume that A is a minimal SAS which is not a minimal FAS. Then, there exists an arc $e = (u, v) \in A$ such that $A \setminus \{e\}$ is a FAS but not a SAS. This implies that in $G_{(A\setminus e)^-}$, either u is a white target or v is a white source. We restrict ourselves to consider the latter case, since the former one can be dealt with similarly. Since v is a white source in $G_{(A\setminus e)^-}$, and it is not in G_{A^-}, all arcs incident to v are in A. In other words, the story corresponding to A does not contain v, which contradicts Proposition 2. □

The previous proposition and its corollary state that, in a graph with no bad nodes, each story corresponds to a minimal FAS. This suggests that for such graphs, we could enumerate all stories by enumerating all the minimal FASs and by checking for each of them whether the resulting graph is a story (which can be done by checking that no white node is source or target). Unfortunately, there are graphs with no bad nodes in which the number of minimal FASs is exponentially larger than the number of minimal SASs. An example is given in Figure 3.10.

Figure 3.10: Graph with no bad node and in which the number of minimal FASs is 2^n and the number of minimal SASs is 2.

3.4.4 Enumerating stories by enumerating permutations

We previously presented a method for enumerating all stories in the case of graphs with no bad nodes. Unfortunately, many graphs arising from the biological application briefly described in the introduction of this chapter contain a huge number of bad nodes. The rule, indeed, is that bad nodes are very common instead of being rare. For instance, any white node which is present in a strongly connected component of the input graph containing at least five nodes is a bad node. The simplification rules previously described not only reduce the size of the input graph but also make it much more dense than the input graph and in the transformed graphs all remaining white nodes are usually bad nodes.

We thus need a method for enumerating stories which is able to deal with these cases.

Remember how we can find a single story as explained in the proof of Theorem 3. Consider the following two simple operations, CLEAN and CONSISTENT_ARCS. For any graph $G(\mathbb{B} \cup \mathbb{W}, E)$, and for any total order π of the nodes:

$G'(\mathbb{B} \cup \mathbb{W}, E') \equiv$ CONSISTENT_ARCS(G, π): for each arc $(u, v) \in E$, $(u, v) \in E'$ if $\pi(u) < \pi(v)$;

$G'(\mathbb{B} \cup \mathbb{W}', E') \equiv$ CLEAN(G): recursively remove white nodes that are sources, targets or isolated in G.

We can thus define the composed operation

$$\text{PITCH}(G, \pi) = \text{CLEAN}(\text{CONSISTENT\_ARCS}(G, \pi)).$$

PITCH produces a pitch since the resulting graph G' contains only arcs that respect the order π and therefore is acyclic. Moreover, due to the cleaning step, G' is guaranteed to have neither white sources nor white targets.

Theorem 5. *For any story S, there exists a permutation π such that* $\text{PITCH}(G, \pi) = S$.

Proof. It is enough to show that, for any story S of $G = (\mathbb{B} \cup \mathbb{W}, E)$ and for any topological order π of $V(S)$, $\text{PITCH}(G, \pi) = S$. Because of the maximality of a story, it suffices to show that $S \subseteq \text{PITCH}(G, \pi)$. Given an arc (u, v) of S, we have $\pi(u) < \pi(v)$. Therefore (u, v) is in $\text{CONSISTENT\_ARCS}(G, \pi)$. Since (u, v) is an arc of S, there exists a path p in S between two black nodes containing u and v. Then p is also in $\text{CONSISTENT\_ARCS}(G, \pi)$, and thus u and v are both black or, if one or both of them is white, then they are neither source nor target in $\text{CONSISTENT\_ARCS}(G, \pi)$. Since $\text{CLEAN}(\text{CONSISTENT\_ARCS}(G, \pi))$ removes neither black nor white nodes that are neither source nor target, we conclude that (u, v) is also in $\text{CLEAN}(\text{CONSISTENT\_ARCS}(G, \pi)) = \text{PITCH}(G, \pi)$. □

This theorem together with Theorem 3 suggest an approach to enumerate stories which simply consists in generating all permutations π of the nodes of G and computing $P = \text{PITCH}(G, \pi)$: if P is not a story, then we use COMPLETE_PITCH to grow it into a story, like illustrated in Figure 3.11.

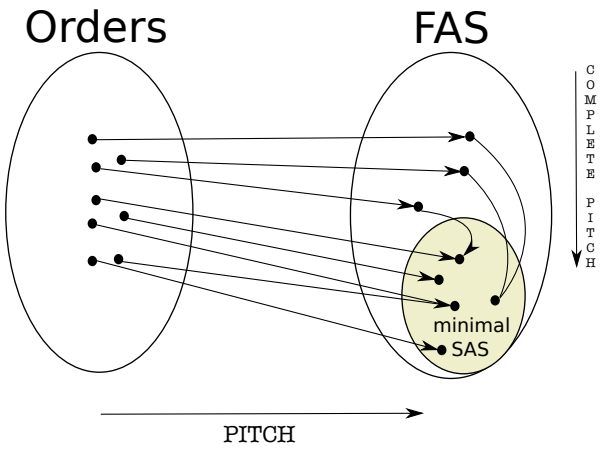


Figure 3.11: Illustration of the enumeration algorithm based on sampling the space of orders of the nodes and computing a pitch through the procedures CLEAN and CONSISTENT_ARCS. This procedure is guaranteed to produce all storiesaccordingly with Theorem 5. For practical purposes only, in the cases where the pitch do not correspond to a story we maximalize it into one for the algorithm to produce a story for all orders explored, even if the story produced may be not unique.

3.5 Alternative definition of a story

It is clear that, according to our definition of a story, no white node can be either source or target in the original graph, since otherwise such a white node would not belong to any story. This implies that the original graph can be seen as the union of a finite set $\mathcal{P}$ of directed

paths between black nodes: in particular, if $\mathcal{P}$ includes all paths between every pair of black nodes, then it is easy to verify that a story is a maximal subset $\mathcal{S}$ of $\mathcal{P}$ such that the graph defined as the union of the paths in $\mathcal{S}$ is acyclic and there exists no path p in $\mathcal{P} - \mathcal{S}$ that can be added to $\mathcal{S}$ while preserving acyclicity. Let us call this alternative definition of story a *path-story*. A minimal number of paths to be removed from $\mathcal{P}$ such that the union of the remaining paths is a path-story is called a *feedback path set*.

A natural question is whether the problem changes when a set $\mathcal{P}$ is given as input, and the graph $G_{\mathcal{P}}$ is defined by the union of the paths of $\mathcal{P}$, where the endpoints of the paths in $\mathcal{P}$ form the set of black nodes of $G_{\mathcal{P}}$. Clearly, since $\mathcal{P}$ may not contain all paths between every pair of black nodes in $G_{\mathcal{P}}$, the set of path-stories of $G_{\mathcal{P}}$ is different from the set of stories of $G_{\mathcal{P}}$ (see for an example Figure 3.12). We will prove that enumerating path-stories is at least as hard as enumerating hitting sets, which is a well-known enumeration problem (for a survey, we refer to Eiter et al. (2008)) with its computational complexity still open, after more than 28 years.

Moreover, another important observation is that even if the original problem is a special case of the path-story problem, we cannot simply state that this new definition is a generalisation of the previous one. The case in which the two problems are equivalent is when $\mathcal{S}$ contains all possible paths between the black nodes. However, transforming the input of the original problem into the equivalent case of the second demands the computation of all paths between the black nodes. The sizes of the two inputs are therefore not polynomially bounded. For this reason the path-stories problem is not a generalisation of the stories problem.

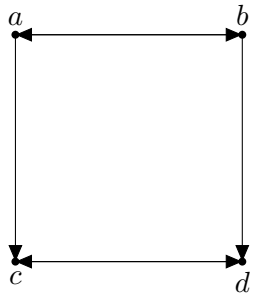


Figure 3.12: Graph obtained by two paths (a, b, d, c) and (b, a, c, d). According to the alternative definition, this graph clearly contains only two stories, which correspond to the two paths. According to the original definition, instead, the graph contains the following four minimal SAS: $\{(a, b), (c, d)\}$, $\{(a, b), (d, c)\}$, $\{(b, a), (c, d)\}$, and $\{(b, a), (d, c)\}$. Note that these four minimal SAS originated four stories which are all different from the two stories obtained according to the second definition.

Theorem 6. *Enumerating path-stories is at least as hard as enumerating minimal hitting sets.*

Proof. Hitting Set. Let $\mathcal{C}$ be a collection of subsets of a domain set X. $H \subset X$ is a *hitting set* of $\mathcal{C}$ if for any $C \in \mathcal{C}$, $H \cap C \neq \emptyset$.

We reduce $\mathcal{C}$ to a collection $\mathcal{P}$ of paths, such that there is a bijective correspondence between (minimal) hitting sets of $\mathcal{C}$ and (minimal) feedback path sets of $\mathcal{P}$ and, hence, between hitting sets of $\mathcal{C}$ and path-stories of $\mathcal{P}$.

We order all sets of $\mathcal{C}$ and all elements of X. Within any set of $\mathcal{C}$ the elements are ordered. For each element in each set, we create a vertex of the graph $G_{\mathcal{P}}$. For each set $C_i \in \mathcal{C}$

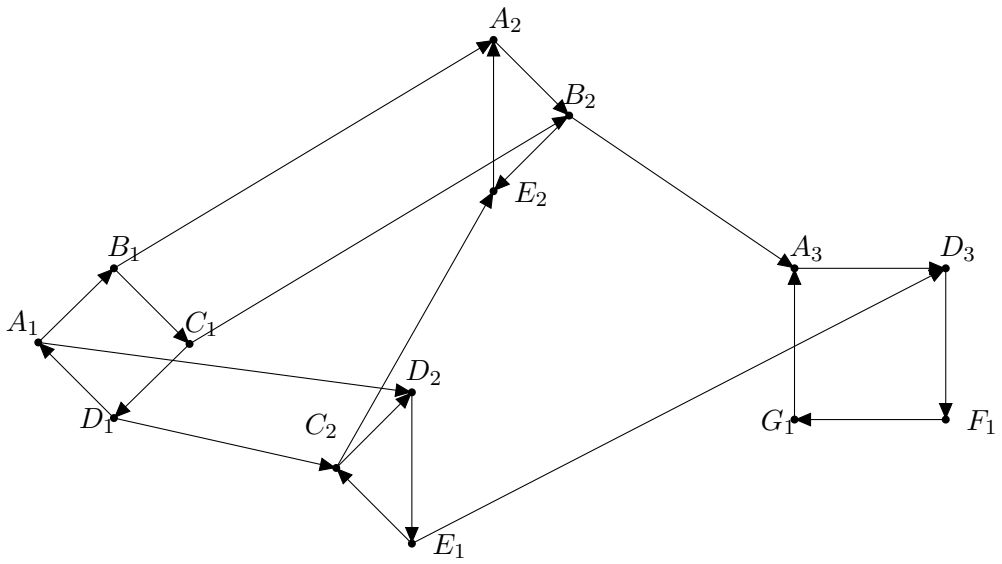


Figure 3.13: An example of reduction: $C_1 = \{A, B, C, D\}$, $C_2 = \{C, D, E\}$, $C_3 = \{A, B, E\}$, and $C_4 = \{A, D, F, G\}$.

with $C_i = \{x_{i_1}, \ldots, x_{i_{k_i}}\}$, create a cycle by introducing the arcs $(x_{i_\ell}, x_{i_{\ell+1}})$, $\ell = 1, \ldots, k_i$ and $(x_{i_{k_i}}, x_{i_1})$. We call this cycle also C_i. Moreover, suppose that $x_{i_\ell} = x_j$ is the h-th occurrence of x_j and x_{r_t} the next occurrence, then we introduce an arc $(x_{i_{\ell+1}}, x_{r_t})$, *i.e.*, there is a path of two arcs between any two consecutive occurrences of the same element. Let us call the latter set of arcs the *element-arcs* and the set of arcs on the cycles the *set-arcs*. Notice that the element arcs are not in any cycle.

Now for each element x_j, we define a path $P_j \in \mathcal{P}$, by starting in the vertex of the first occurrence of x_j, and every time selecting the two arcs connecting it to the next occurrence vertex, until we arrive at the last occurrence vertex.

The graph induced by $\mathcal{P}$ contains all the edges just introduced. In particular it contains all the cycles corresponding to the sets in $\mathcal{C}$. An example of the reduction is shown in Figure 3.13.

It is easy to see that a path P_j cuts cycle C_i if and only if x_j hits the set C_i. Hence there is a one-to-one correspondence between a minimal path set of $\mathcal{P}$ and a minimal hitting set of $\mathcal{C}$. This proves the theorem. □

For the practical part of the work, we have mainly focused our attention on the first definition of stories, since we cannot define *a priori* which paths should be examined in a real application, moving back to the case in which all possible paths between black nodes must be considered. In such a case, there is an equivalence between stories and path-stories.

3.6 Biological application

In this section, we present some illustrations on how to explore metabolic stories enumeration for studying metabolic networks. The software developed to compute metabolic stories, called **Gobbolino**, has as input a metabolic network and a list of black nodes and is presented in more detail in the Appendix. Currently, the accepted format of the metabolic network is the SBML format which describes in an XML document all the metabolites and chemical reactions of a given organism (Hucka et al. (2004)). This input data is then trans-

formed into the internally defined format of a metabolic network with black nodes. This internal format is called the NEL format, for node-edge-list, and is a simple text file listing in the following order: the black nodes, the complete list of nodes and the list of arcs. There are two consecutive simplifications that are applied to the NEL input file. First we compute the lightest path network which will contain a union of all lightest paths between all pairs of black nodes, according to the rationale explained in Section 3.4. After that, a compressed network is computed by applying the four simplification operations described previously: forward and backward bottleneck removal, self-loop removal and white source/target removal. Finally, the metabolic stories are enumerated on this simplified network. To analyse the results, it may be useful to group the best classified stories, if a meaningful classification function may be applied, into a single object, that we call an **anthology**, highlighting the frequency with which each of the arcs are present inside the selection. An anthology may be useful to show which sets of arcs form the backbone of the grouped stories and which arcs correspond to less used alternatives. This schema is shown in Figure 6.5.

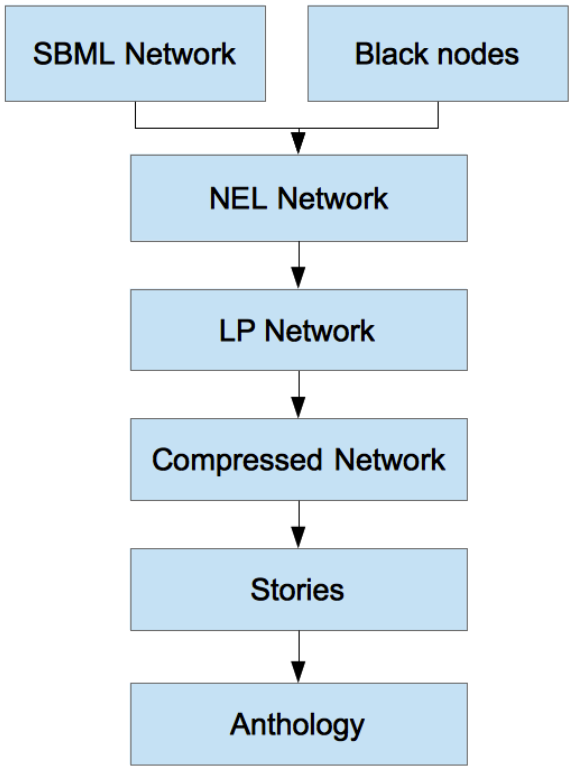


Figure 3.14: Successive steps needed to compute stories and the anthology using the software Gobbolino.

One first application is the original motivation for developing the method which is the study of metabolomics experiments. We focus here on a real study performed with the aim of better understanding which metabolic pathway yeast employs for the detoxification of cadmium, which is a heavy metal. To this purpose, we first produce a compressed network that provides a very compact representation of the interactions between the nodes identified in the experiment. In a second step, we compute metabolic stories, classify them according to how they fit to the experimental data and group the best ones in an anthology.

A second application is to use the metabolic stories enumeration to automatically identify metabolic pathways (Section 1.1 provides a brief introduction to metabolic network and

metabolic pathways). To this purpose, we selected as black nodes the sources and targets of 69 annotated yeast metabolic pathways downloaded from MetaCyc (Caspi et al. (2010)). Using a different classification function that prefers stories that respect a given order of the metabolites, *i.e.*, that preserves the orientation of the metabolic pathway we are supposed to recover, we compare the obtained best stories with the original metabolic pathway.

3.6.1 Enumerating stories for interpreting metabolomics experiments

This is the original motivation for the development of the method. A widely studied metabolic pathway in *Saccharomyces cerevisiae* is the one responsible for glutathione biosynthesis, since it is related to the detoxification process of the cell when exposed to high concentrations of the heavy metal cadmium (Fauchon et al. (2002); Lafaye et al. (2005); Madalinski et al. (2008)). Previous studies demonstrated that the presence of such a metal in the environment induces a huge impact in terms of gene expression and metabolism, showing that there is a strong response both at the metabolomic and proteomic levels. Basically, glutathione needs to be produced because it is a thiol-metabolite linked to the detoxification of cadmium through a process called chelation (Li et al. (1997)). Plants are the natural biotope of *S. cerevisiae* and it is known that they are able to tolerate cadmium and other metals up to 1% of their dry weight, which is believed to provide defense against herbivores and pathogenic microorganisms (Fauchon et al. (2002)). This exposition to cadmium in natural conditions provides a reason for yeast cells to keep a detoxification pathway. However, biosynthesis of glutathione requires high quantities of sulfur. In order to save sulfur, there is a replacement of sulfur-dependent enzymes related to other metabolic processes by isozymes that are sulfur free, *i.e.*, other enzymes that have the same function but a different chemical composition. More specifically, protein PDC1P is replaced by its isozyme PDC6P in order to save sulfured amino acids and two other isozymes are employed in order to save both energy and sulfur demanded in the glutathione pathway (Fauchon et al. (2002)). Another adjustment is that sulfur metabolism is normally directed to produce methionine and cysteine but under the presence of cadmium is redirected to glutathione synthesis. All these adjustments affect a big portion of the metabolic network and collectively constitute the mechanisms used by the cell to survive under this specific stress condition. A schema of the known glutathione biosynthesis metabolic pathway is presented in Figure 3.15.

More recently, a metabolomics experiment was performed to measure the metabolites whose concentration significantly change in yeast cells under cadmium exposition (Madalinski et al. (2008)). The experiment was performed using the strain *s288c* of *Saccharomyces cerevisiae*, whose metabolic reconstruction is available in MetExplore (Cottret et al. (2010a)). This metabolic network contains 600 metabolites and 949 arcs. The authors identified a list of 24 metabolites whose concentration significantly changed, shown in Table 3.1.

We decided to perform two experiments in order to use our method to explore the effect on *Saccharomyces cerevisiae* cells of exposition to the nonessential heavy metal cadmium (Cd^{2+}). We first enumerated metabolic stories using a set of black nodes restricted to the measured metabolites that are known to participate to the biosynthesis of glutathione. The idea is to check whether our method is able to recover one or more stories that correspond to the known metabolic pathway. In a second step, we enumerated metabolic stories using the entire list of 24 interesting compounds identified in the metabolomics experiments. The idea is that this will allow a comparative approach that may give insights on possible alternative pathways for glutathione synthesis or enable to better understand side effects of cadmium exposition.

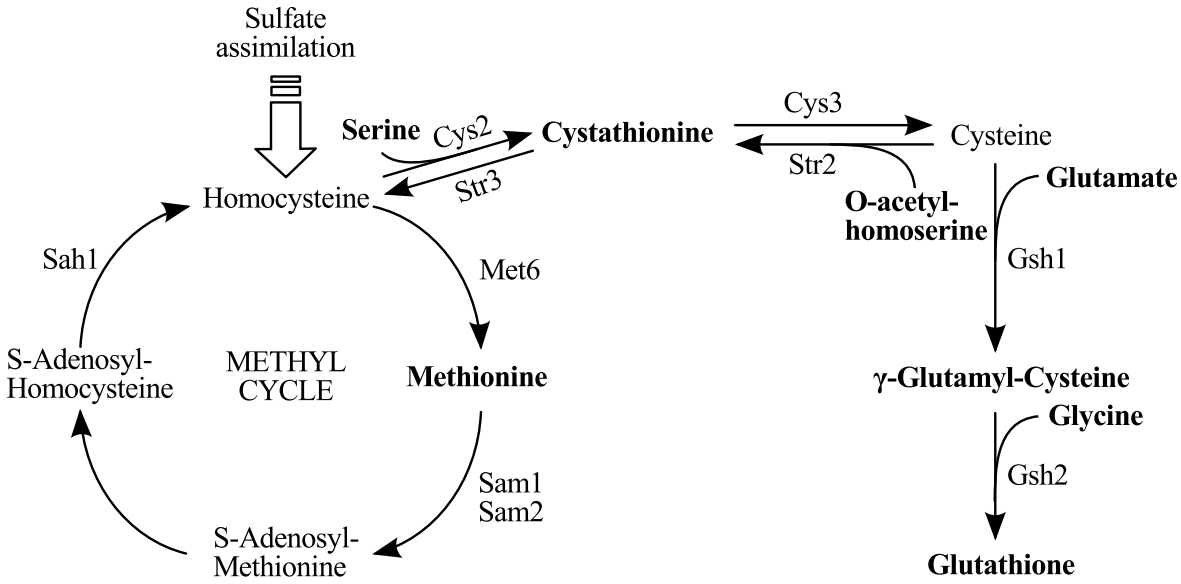


Figure 3.15: Glutathione biosynthetic pathway. Source: Adapted from Figure 1 in Lafaye et al. (2005)

First experiment: black nodes from the biosynthetic pathway of glutathione

We first consider the previously mentioned metabolic pathway directly involved in cadmium detoxification, namely the glutathione biosynthetic pathway, in order to enumerate stories and check whether we are able to recover a story that fits current knowledge on the biological process. We thus selected as black nodes for this first experiment only the metabolites that were measured in the experiment (Madalinski et al. (2008)) and also that are known to participate in the glutathione biosynthetic pathway (Fauchon et al. (2002)). These compounds are the 8 ones presented in Table 3.1 with the third column marked as "yes": glutathione, O-acetylhomoserine, methionine, glutamate, glutamylcysteine, serine, glycine and cystathionine.

Working with a small set of black nodes produces a small compressed network and, consequently, not so many metabolic stories. The compressed network obtained for the reduced set of black nodes contains 10 nodes and 25 arcs, *i.e.*, represents more than 98% of compression in terms of nodes and more than 97% in terms of arcs, with respect to the original input size of *S. cerevisiae* metabolic network. The resulting compressed network is shown in Figure 3.16. For this experiment, we set the two stop conditions for GOBBOLINO: producing half a million stories or not producing a new story after one minute of computation. The algorithm reached the second stop condition and the computation stopped after one minute in which no new story was found. In total, 222 metabolic stories were identified. It is expected that the total number of stories is not much bigger than that since this stop condition indicates that the space of orderings of the nodes is well explored.

From a formal point of view, there is no qualitative difference between any two stories. In this sense, whether a given interesting compound is a source, an intermediate node or a target in a story is indifferent for the enumeration process since all possible scenarios satisfying the three properties given by the definition, namely maximality of paths, acyclicity and source/target constraint, have to be computed. This is not true from a biological point of view, since we may use a priori knowledge on the chemicals identified as interesting compounds, or on the experiment under study in order to distinguish the stories. In fact, in the

Table 3.1: **List of interesting compounds for the *Saccharomyces cerevisiae* cell exposed to cadmium**

Metabolite ID	intensity ratio	Present in the pathway
arginine	1.9	no
reduced glutathione	33.9	**yes**
O-acetylhomoserine (*)	0.5	**yes**
2-aminoadipate (*)	0.5	no
niacinamide (*)	4.8	no
pyridine-3-aldoxime (*)	4.8	no
pyrroline-hydroxy-carboxylate	0.7	no
methionine	0.3	**yes**
citrulline (*)	0.7	no
threonine	0.6	no
homoserine	0.6	no
glutamine	0.7	no
glutamate	0.8	**yes**
glutamylcysteine	192.2	**yes**
5-methylthioadenosine	11.0	no
serine	0.2	**yes**
glycine (*)	0.3	**yes**
cystathionine	50.5	**yes**
lysine	0.7	no
cysteinylglycine (*)	35.9	no
leucine/isoleucine	1.2	no
tyrosine	2.9	no
histidine	1.2	no
alanine	0.8	no

List of 24 metabolites from the yeast metabolic network whose concentration significantly varied under cadmium exposed. The intensity ratio column presents the ratio from the stress condition to the control. The 3rd column indicates whether the compound is present in the glutathione biosynthetic pathway (Fig 3.15) or not. Metabolites identified with an (*) after their names needed further validation to be confirmed in the list.

specific case of yeast cells exposed to cadmium, we may define a score function using the data from the observed concentration changes in the experiment in order to identify the stories that best fit the data. This score function is then computed for each story, assigning to each a value that reflects its agreement with the global flow of mass observed.

Using additional information such as the concentration of the interesting compound in the metabolomics experiment allows us to differentiate between interesting compounds that have their concentration increased or decreased during cadmium exposure, *i.e.*, to explore also the data in the second column of Table 3.1 in order to identify the stories where the interesting compounds whose concentration increased (ratio greater than 1), such as reduced glutathione, cysteinylglycine, cystathionine and glutamylcysteine, are preferentially produced and, on the other hand, stories where the interesting compounds whose concentration decreased (ratio smaller than 1) such as methionine, homoserine, serine and glycine, are preferentially consumed. To differentiate these two sets of black nodes, let us distinguish again as **green nodes** the group of metabolites whose concentration increased and **red nodes** the group of metabo-

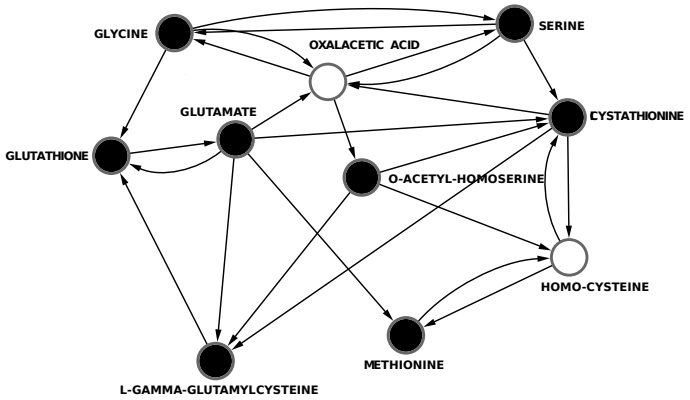


Figure 3.16: The compressed network computed considering as black nodes the 8 compounds of Table 3.1 marked as present in the glutathione biosynthetic pathway.

lites whose concentration decreased. The other nodes in the network will continue to be called white nodes.

Table 3.2: **Weights for the different kinds of interactions between nodes in a story**

Outgoing arcs			
	To Red	**To Green**	**To White**
From Red	0.0	1.0	1.0
From Green	−1.0	0.0	−1.0
From White	−1.0	1.0	0.0

Table exhibiting the weights for interactions between green, red and white nodes used for computing the score of a story in the context of a metabolomics experiment.

The idea of the score function is to capture the relationship between green, red and white nodes inside a story. It is computed in the following way: $s(S) = \sum_{v \in V(S)} ni(v) \times af(v)$, where the score $s(S)$ of a story S is the sum of the *normalized intensity* ni times the *adjacency factor* af of each node v in the story. The normalized intensity $ni(v)$ of a node v is its intensity (*i.e.*, the relative variation of its concentration) divided by the maximum intensity observed in the experiment (if v is a green node) or the minimum intensity observed in the experiment divided by the intensity of the node (if v is a red node). The adjacency factor $af(v)$ of a node v is the sum of the values presented in Table 3.2 corresponding to the adjacency of the node, capturing all possible interactions between nodes and weighting them with values representing the normal expected behaviour of the system, *i.e.*, positive weights for arcs producing green nodes or consuming red nodes, and corresponding negative weights for the opposite case, *i.e.*, arcs representing consumption of green nodes or production of red nodes. The remaining arcs have a weight of zero. White nodes are also assumed to have a zero intensity, since they were not measured in the metabolomics experiment. Figure 3.17 presents a small story containing 4 black nodes and 1 white node and its associated score.

Using this score function to assign values to the 222 stories computed in our experiment we found that only one story appeared at the top of the ranking with a score of 4.067. This highest score story is shown in Figure 3.18, and it is interesting to compare it with the original metabolic pathway shown in Figure 3.15, since it corresponds quite well, as expected, to the

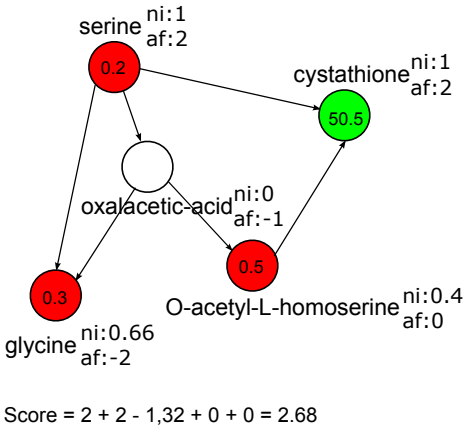


Figure 3.17: Considering the story with 5 nodes presented we may compute its score as 2.68. The minimum concentration observed in the story for the red nodes is 0.2 and the maximum concentration observed for a green node is 50.5. Therefore, we may compute the following node intensities: $ni(serine) = 0.2/0.2 = 1$, $ni(cystathione) = 50.5/50.5 = 1$, $ni(glycine) = 0.2/0.3 = 0.66$ and $ni(O - acetyl - L - homoserine) = 0.2/0.5 = 0.4$. The adjacency factors are $af(serine) = 2$, counting the two arcs producing a green and a white node, $af(cystathione) = 2$, counting the two arcs producing a green node, $af(glycine) = -2$ since it is a red node being produced by two different arcs and $af(O - acetyl - L - homoserine) = 0$ since it is a red node that is being produced (-1 factor) but is also being used to produce a green node (+1 factor). White nodes have no intensity, since they were not measured in the experiment and therefore they are assumed to have zero intensity and they do not contribute to the global score of the story.

flow of matter there observed. The story agrees with the biological conclusion of the paper with the metabolomics experiment from where the data was taken (Madalinski et al. (2008)), *i.e.*, to the metabolic flux corresponding to the detoxification of cadmium by glutathione, explaining that the levels of metabolites involved in the glutathione biosynthesis pathway (homocysteine, cystathionine, Glu-Cys and glutathione itself) were increased following cadmium exposure.

Second experiment: black nodes from the metabolomics experiment

For the second experiment, all 24 black nodes were considered. One of them, pyrroline-hydroxy-carboxylate, was eliminated when computing the lightest paths between all pairs of black nodes since it was part of a small disconnected component of the original input graph, most probably due to missing information in the metabolic network reconstruction as the metabolite was present in the metabolome of the strain. The computed compressed network contains 34 nodes and 76 arcs, *i.e.*, a compression of 94% in terms of nodes and 92% in terms of arcs. The resulting compressed network is shown in Figure 3.19.

For this experiment, we set the same two stop conditions for Gobbolino: producing half a million stories or not producing a new story after one minute of computation. The algorithm stopped after reaching the first condition, which strongly indicates that the space of solutions was under-explored and that the total number of stories must be much greater than that. However, for the purpose of our analysis, we decided to proceed with the stories produced so far. As an example of the result, we present in Figure 3.20 the first two metabolic stories out of the 500000 that we enumerated.

Using our score function for assigning values to the computed stories, we can restrict our analysis to the ones with the highest values, allowing us to focus on the stories that best fit the experimental data. For instance, only 7 out of the half a million stories computed have

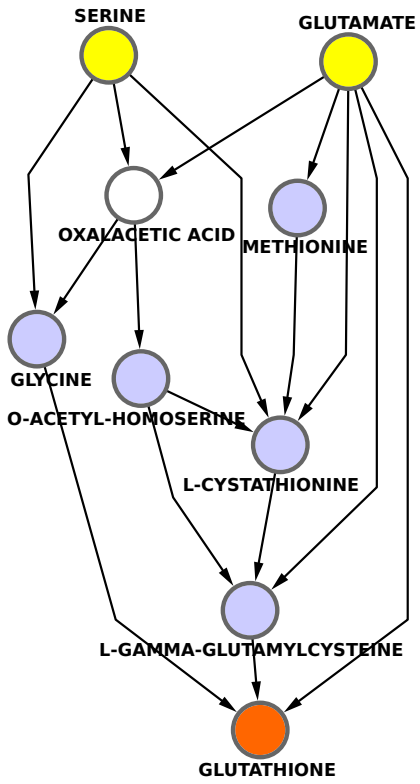


Figure 3.18: The best story generated for our experiment taking into account only the intersection between the metabolites measured in the metabolomics experiment and the ones known to be present in the glutathione biosynthesis. The set of black nodes is coloured in the following way: yellow nodes denote sources of a story, orange nodes are targets and blue nodes are intermediate. The white nodes are white.

the highest computed score of 19.556. Figure 3.21 presents an histogram of the distribution of the scores over the whole set of computed stories. Gobbolino may display each story individually for analysis, allowing one to investigate in detail the differences between each of the 7 stories with the highest score, or any of the sub-optimal ones. Again, an alternative way to perform the analysis is to compute an anthology, which is a collection of stories. One may generate the anthology of the 7 highest score stories (shown in Figure 3.22) or include sub-optimal solutions, such as all stories whose score is at most 0.5 points below the optimum. As an anthology is a collection of two or more stories, it will contain cycles and the roles of the nodes as source or target may not be so clear anymore. In order to help the analysis, we draw an anthology giving a size to the nodes that corresponds to their concentration change, red ones being those whose concentration decreased and green ones those whose concentration increased. The thickness of the arcs corresponds to the frequency with which each arc appears in the stories grouped in the anthology.

Comparing the two experiments

With the goal of contrasting the smaller and more local sets of stories against the bigger and more global anthology produced for the second experiment, we computed an anthology for the results of our first experiment, grouping the 6 stories that were close to the best one, *i.e.*, whose difference in the score was not bigger than an arbitrarily chosen 0.5 points range.

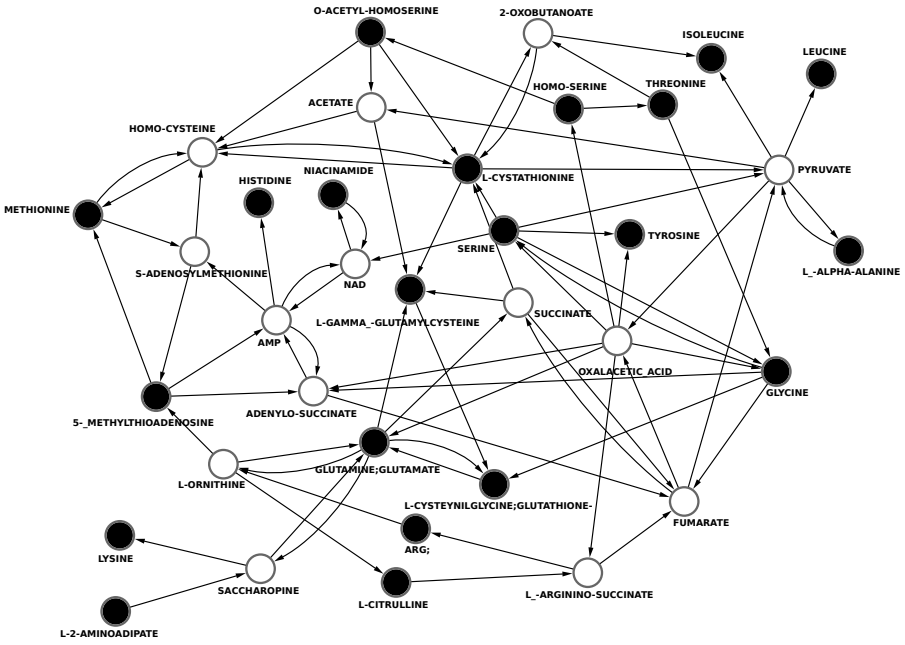


Figure 3.19: The compressed network computed for the whole list of interesting compounds of Table 3.1 and the metabolic network of the yeast strain s288c.

This anthology is shown in Figure 3.23. The backbone still corresponds quite well to the known pathway but now some variations may also be observed. This is a strong point of our method since it allows exploring alternative but close scenarios through the analysis of these and other high-scoring stories, which might provide new insights on the underlying processes that took place under the given conditions.

For ease of analysis, we produced a new drawing of the anthology considering the whole list of black nodes, highlighting the 8 nodes known to be part of the glutathione biosynthetic pathway, as shown in Figure 3.24. We may therefore try to analyse what portion of the anthology may be explained by the sulfur redirections that are known to be part of the response of yeast cells to cadmium (Fauchon et al. (2002); Lafaye et al. (2005); Madalinski et al. (2008)); and which portion may be seen as an indirect consequences of such response. Among the 27 metabolites present in the anthology, 8 have sulfur in their chemical structure: S-adenosyl-L-methionine, L-gamma-glutamylcysteine, S-methyl-adenosine, O-acetyl-L-homoserine, cysteinylglycine, glutathione, cystathionine and L-methionine. Among these sulfured metabolites, the only one which is not involved in the glutathione biosynthesis is S-methyl-adenosine, and is instead involved in the methionine salvage pathway. Notice also that serine, which is one of the precursors in the glutathione biosynthesis, is involved in many other processes, specially involving amino acids such as alanine, leucine, isoleucine, tyrosine, glutamate, glutamine and arginine, but also in the synthesis of a vitamin, niacinamide. Interestingly, the compounds involved in the methyl cycle, which is part of the glutathione biosynthesis pathway (see Figure 3.15), were not recovered in the highest score story found in our first experiment. The reason is that the lightest path found between methionine and cystathione in that experiment passed through the reversible reaction homocysteine S-methyltransferase. In the second experiment, when the whole list of interesting compounds was considered, the com-

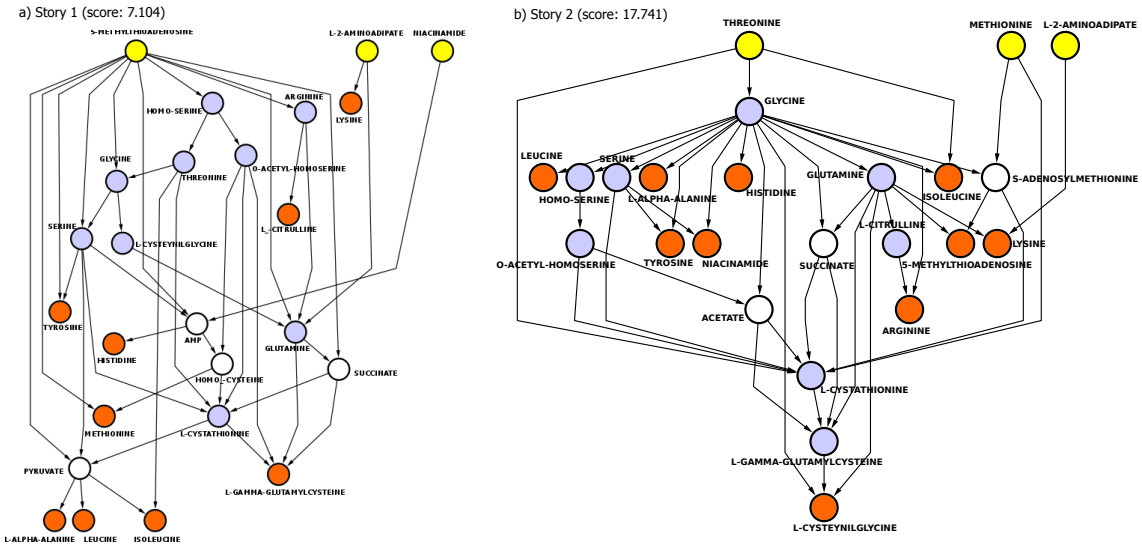


Figure 3.20: The first two random stories generated for our experiment taking into account the whole list of black nodes. The set of black nodes is coloured in the following way: yellow nodes denote sources of a story, orange nodes are targets and blue nodes are intermediate. The white nodes are white.

pounds in the methyl cycle are recovered, with the white node S-adenosylmethionine present in the anthology and the metabolites S-adenosyl-homocysteine and homocysteine compressed into the arc from S-adenosylmethionine to cystathionine. The role, if any, of the other white nodes recovered in this anthology such as adenylo-succinate, succinate, acetate and adenosine monophosphate (AMP), need more detailed analysis to check if their presence may be related to the response of the yeast cells to cadmium or if they are side effects related to some other metabolic processes.

3.6.2 Enumerating stories for recovering metabolic pathways

Although established for dealing with metabolomics experiments, our definition of metabolic stories in terms of a subgraph extraction problem is general and may thus prove useful in different contexts than the original one. In the previous application on interpreting metabolomics experimental data, we started by showing how we could recover a story that corresponded well to the known metabolic pathway of glutathione biosynthesis. We now propose to apply the enumeration of metabolic stories outside of the context of metabolomics and try to generalise the result obtained before, *i.e.*, apply the method for metabolic pathway automatic discovering. A metabolic pathway is a chain of reactions that explains how a set of source metabolites is transformed into a set of target metabolites. The idea behind this experiment is that among the stories enumerated, one of them will correspond to the traditionally defined metabolic pathway, and the others will correspond to possible alternatives. To validate this idea, we set up an experiment in which we used a collection of known metabolic pathways of *Saccharomyces cerevisiae*, but in this case the *IMM904* strain which has been extensively curated recently and is also available in MetExplore (Cottret et al. (2010a)). A collection of 69 metabolic pathways annotated in *S. cerevisiae* were used as our benchmark. This data was taken from MetaCyc (Caspi et al. (2010)), version 15. For this second experiment, the set of black nodes was selected specifically for each of the metabolic pathways by manually identifying through visual inspection of the pathways the main source and target metabolites of each

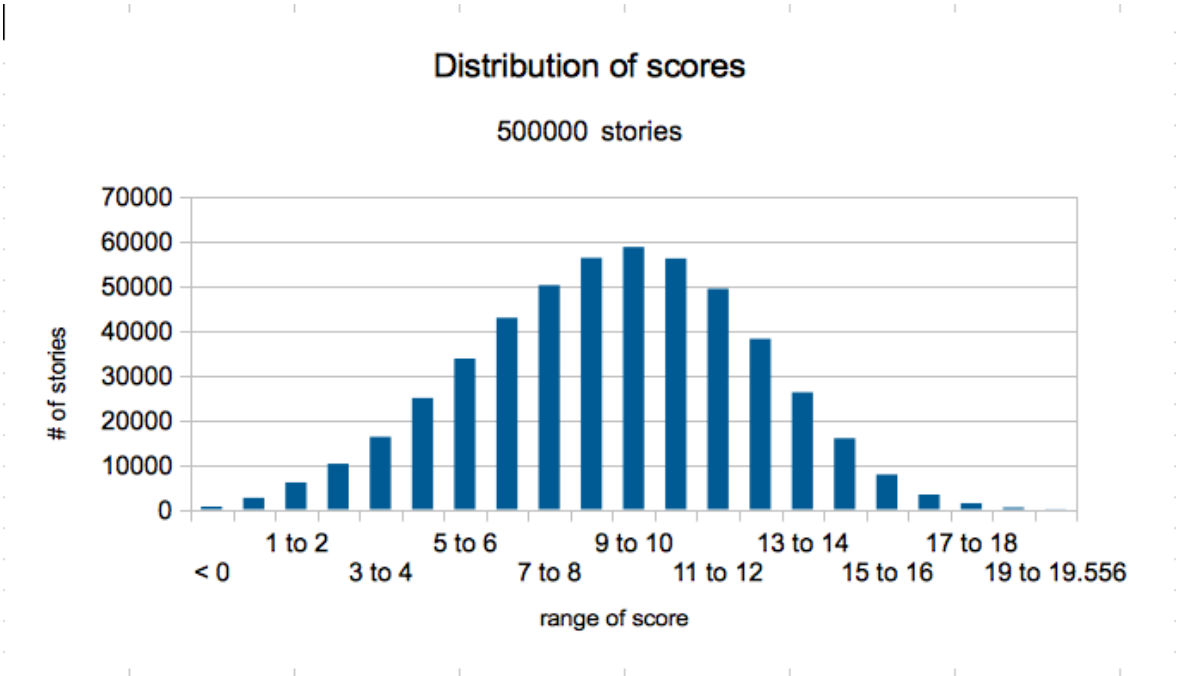


Figure 3.21: Distribution of the scores over the 500000 computed stories.

pathway, *i.e.*, the metabolites that are involved in the first and last steps of the metabolic pathway. Side compounds and compounds that are not involved in carbon metabolism were also filtered out (such as water, proton, ATP, ADP, etc.). Table 3.3, put at the end of the section, presents the list of metabolic pathways used in the experiment as well as the black nodes chosen for each of them.

The experiment consisted in enumerating metabolic stories for each metabolic pathway and their corresponding sets of black nodes. After this, we wanted to compare the stories found with the original pathways, measuring the specificity and sensitivity of our method to recover in the set of computed metabolic stories one that corresponded to the known chain of reactions of the pathway while, in parallel, providing alternative metabolic pathways for the same set of black nodes.

Since this experiment differs from the previous one also in the sense that, in this case it did not originate from a metabolomics experiment, we have to design a different score function that does not use data on metabolites concentration. For the metabolic pathway retrieval experiment, we defined a score function based on a partial order of the nodes, which prefers stories where some ordering of the metabolites is preserved, namely the metabolites involved in the starting (respectively, ending) point of the original metabolic pathway should preferably be sources (resp., targets) of the story. This score function needs as input a set of ordered pairs of metabolites indicating their preferable order and assigns a value to the final story according to how many of these orderings are respected. For example, Figure 3.25 presents one of the 69 metabolic pathways used in our experiment, namely the superpathway of glucose fermentation. The nodes selected to be the black nodes in this example are glucose, ethanol and acetate. The secondary input file with partial order information will have two entries: (glucose,ethanol) and (glucose,acetate). This indicates that stories in which glucose comes before ethanol and acetate should have higher scores than the ones in which these orderings are not preserved. Notice that this function may be useful in different contexts for which no

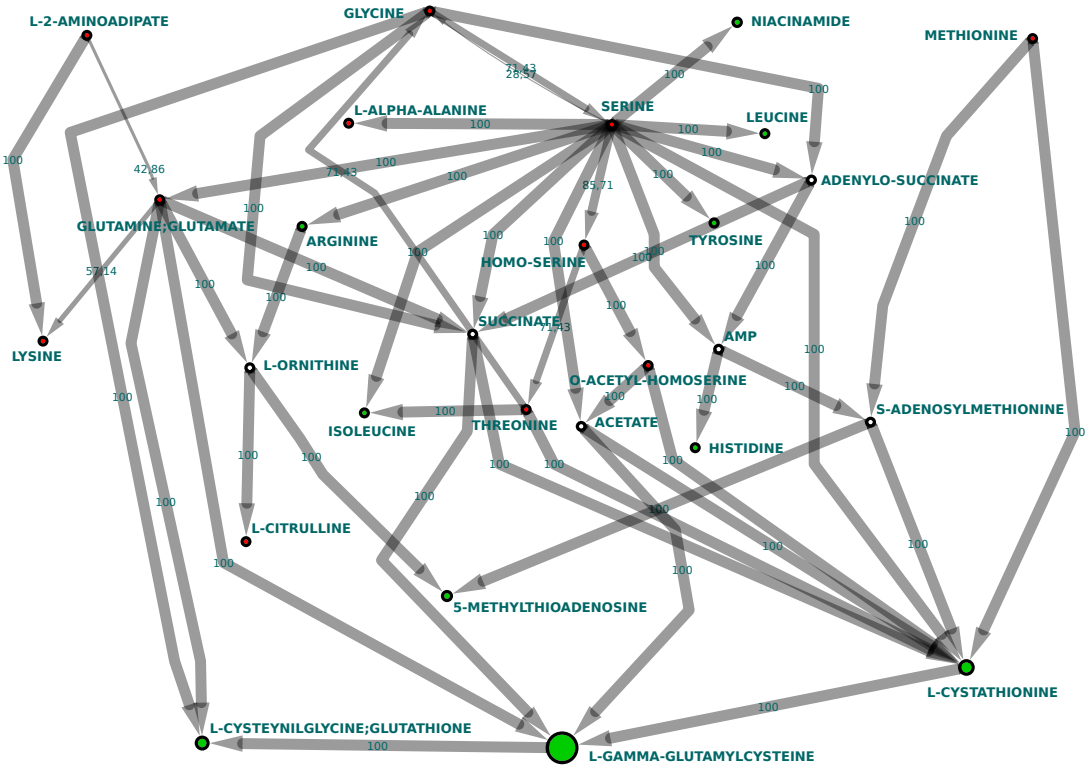


Figure 3.22: Anthology corresponding to the 7 highest scored stories computed for the experiment on the yeast s288c exposed to cadmium. Red nodes correspond to interesting compounds whose concentration decreased and green nodes correspond to those whose concentration increased in the metabolomics experiment. The thickness of the arc represent the frequency of the arc among the stories in the anthology.

concentration or other quantitative information is available but there exists some knowledge on the metabolic process or on the chemical transformations under study.

The experiment consisted in two steps. The first one was to enumerate stories for the 69 metabolic pathways and their sets of black nodes. If the number of black nodes was below 8, we can guarantee that we found all stories. Above, we used the previously described randomized approach with the same stop conditions: half a million stories computed or one minute of computation with no new story produced. The second step is to compare the set of solutions with the original metabolic pathways and to measure the specificity and sensitivity of the method to recover them. Notice that the stories are computed on a compound graph representation of the metabolic network, where there is no direct correspondence between reactions and arcs/nodes.

In order to be able to compare our stories with the reference metabolic pathways, we used the compound graph representation of the reference metabolic pathway considering only the lightest paths connecting its black nodes, *i.e.*, exactly the same preprocessing done for computing stories. The comparison is then based on the sets of arcs of both graphs by following this simple rule: given an arc $u \to v$ in a story if the exact same arc is present in the pathway then we have a true positive prediction (TP), otherwise we have a false positive prediction (FP). On the other hand, if there is an arc $u \to v$ in the reference pathway that is not present in the story we have a false negative prediction (FN). To compute the quality of the method for pathway recovery purposes, we use the same accuracy metric as in a similar metabolic pathway recovery method (Faust et al. (2010)), that is $\sqrt{Sn * PPV}$, where Sn is

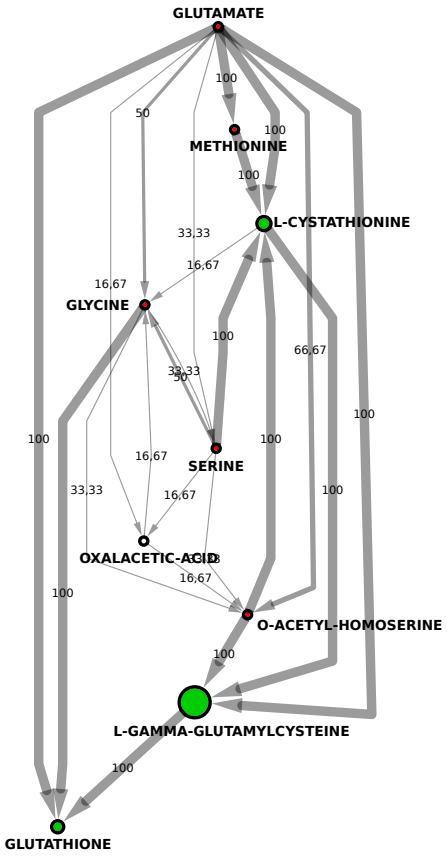


Figure 3.23: Anthology corresponding to the 6 highest scored stories computed for the experiment on the yeast s288c exposed to cadmium using only nodes known to be involved in the glutathione biosynthesis as black nodes. Red nodes correspond to interesting compounds whose concentration decreased and green nodes correspond to those whose concentration increased in the metabolomics experiment. The thickness of the arc represent the frequency of the arc among the stories in the anthology.

the sensitivity measured as the ratio of correctly inferred nodes versus all reference nodes ($TP/(FP + FN)$) and PPV, or Positive Predictive Value, is the ratio of correctly inferred nodes versus all inferred nodes ($TP/(TP + FP)$). Table 3.3 presents the complete list of metabolic pathways, chosen black nodes and the accuracy measures.

On average, 62.72% of the reference metabolic pathways could be recovered using the stories enumerator, which is lower but similar to the results presented in Faust et al. (2010), where they compared four different methods specifically designed for metabolic pathway retrieval in directed graphs. However, it is interesting to notice that our method outperformed in accuracy some of the methods shown in Faust et al. (2010). This is unexpected since the methods they examined are specifically taylored to pathway recovery while ours has a different and more general purpose, which is to extract all interactions between a set of interesting compounds using an enumeration approach. However, there are two main differences that make the comparison of our experiments with the ones performed in Faust et al. (2010) hard. The first one is their use of a reaction graph instead of the compound graph. The second is the fact that our algorithm runs once for each pathway and set of black nodes and

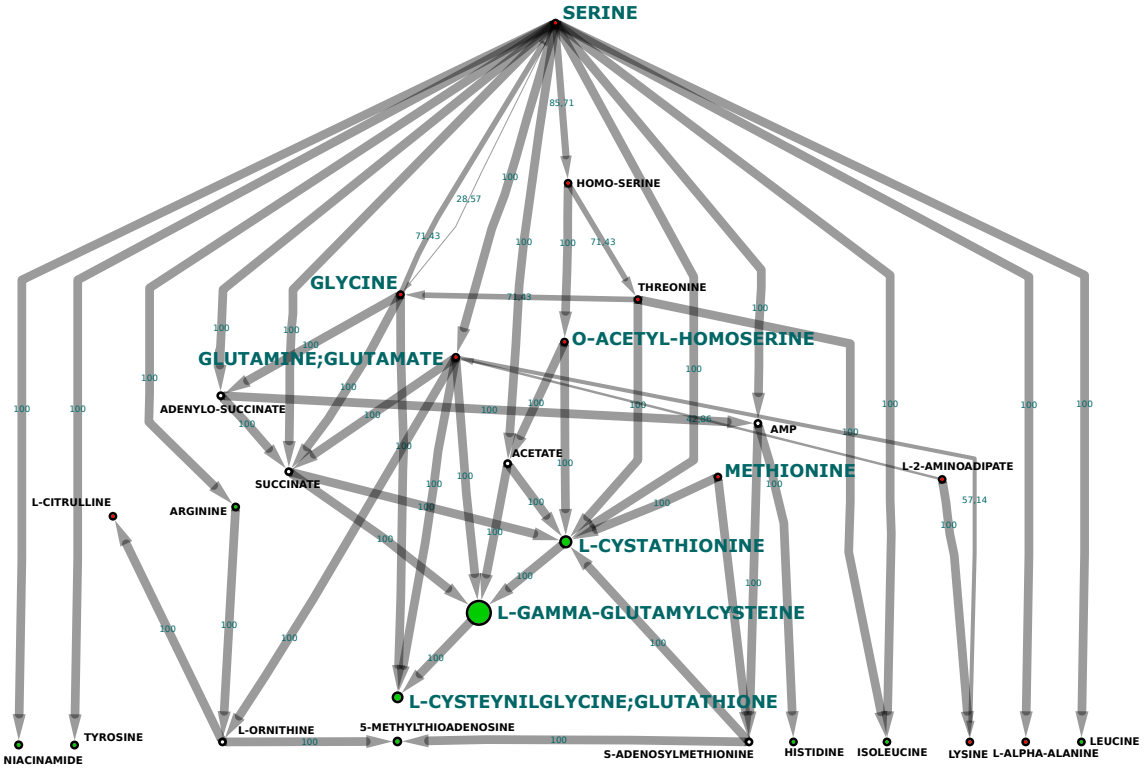


Figure 3.24: Same anthology presented in Figure 3.22 but with highlighting the nodes present in the anthology shown in Figure 3.23.

inspects the accuracy obtained by each of the stories found, selecting the best one, while in their experiments the authors report the average accuracy obtained after running the examined algorithms for some iterations. In more detail, in Faust et al. (2010) each experiment starts with an initial set of nodes, that is increased at each iteration with a random selected node among the internal nodes of the metabolic pathway. The experiment finishes when all reactions of the pathway are found. What is reported is then the mean accuracy over all iterations. Ideally, we should then compare our results with the first iteration of each of their methods, which was not done for this thesis but is planned for a near future. The practical effect is that this increasing set of black nodes makes their accuracy higher than ours since at each run the algorithm uses more information on the reference metabolic pathways in order to recover them.

The best accuracy obtained in the 110 experiments for 7 different methods and 3 different weight policies performed in Faust et al. (2010) was of 76,81% by an hybrid approach combining both Takahashi-Matsuyama (Takahashi and Matsuyama (1980)) and random k-Walks (Dupont et al. (2006)) algorithms. The first algorithm, Takahashi-Matsuayma, starts with a network containing one randomly selected interesting node. It proceeds by iteratively merging to the current network a lightest-path between any of the remaining nodes and any of the nodes already in the network. It finishes when all interesting nodes are connected to the network. The second algorithm, random k-Walks, consists in building a subgraph that connects the interesting nodes through a set of the most relevant arcs. The relevance of an arc is measured as the expected number of times it is visited along random walks connecting the interesting nodes. A subgraph is obtained by applying a threshold on the relevance, which in the experiments performed in Faust et al. (2010) was automatically set as the value such that

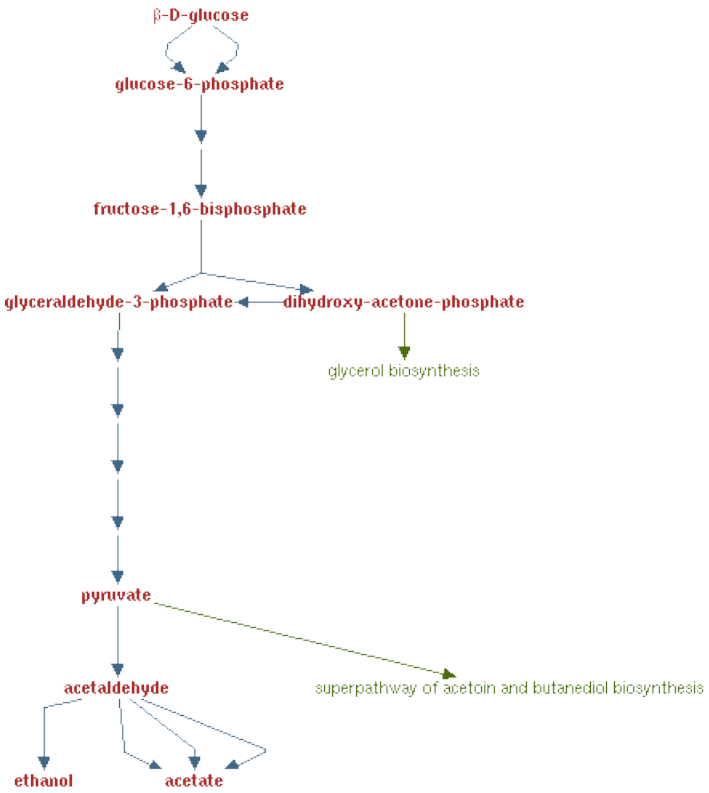


Figure 3.25: Superpathway of glucose fermentation in *Saccharomyces cerevisiae*. Source: Metacyc.

the graph induced by the selected arcs is weakly connected. In the combined approach, first a subnetwork was computed using the random k-Walks algorithm and then this network was used as an input for the Takahashi-Matsuyama algorithm. The fact that the difference between this hybrid algorithm and ours is around 10% and considering that we did not perform the previously mentioned procedure of increasing the set of interesting compounds during several iterations allow us to believe that our method is comparable to algorithms specifically designed for pathway recovery and that it may be worth developing and adapting it in this direction. Moreover, our method was designed to be more sensitive than specific, since stories are maximal subgraphs and this could be a relevant difference in favour of our method in the context of metabolic pathway recovery, since we may be able not only to recover the known metabolic pathway but also to propose alternative pathways involving the same interesting compounds.

Interestingly, in 12 cases (chorismate biosynthesis, lysine degradation, non-oxidative branch of the pentose phosphate pathway, pyridoxal 5'-phosphate salvage pathway, ubiquinone biosynthesis from 4-hydroxybenzoate, phosphatidylinositol phosphate biosynthesis, superpathway of methionine salvage pathway, methionine salvage pathway, de novo biosynthesis of pyrimidine ribonucleotide, tryptophan biosynthesis, tryptophan degradation via kynurenine and galactose degradation), a story corresponded exactly (100% accuracy) to the reference metabolic pathway and, further, the accuracy was greater or equal to 80% in 25 cases (36%). On the other hand, for 7 cases (allantoin degradation, lipid-linked oligosaccharide biosynthesis, glycolysis, lysine biosynthesis, de novo NAD biosynthesis, superpathway of TCA cycle and glyoxylate cycle and TCA cycle aerobic respiration), the correspondence between the stories and the pathway was 0% and the accuracy was smaller or equal to 20% in 8 cases (12%). The

average accuracy was 62,72%.

Significantly, all the cases for which there is no correspondence between the stories and the reference pathway occurred when the computed lightest-path graph connecting the black nodes, which is the first step of our method, did not contain the reference pathway. This means that our method had no chance to find the correct metabolic pathway since it tried to find it in a network where this pathway was not present. This observation highlights the fact that computing the lightest path between metabolites using the degree of the nodes as weights may avoid passing through pool metabolites but is not guaranteed to cover all biologically valid transformations since some known metabolic pathways do not correspond to lightest paths between their terminals. For instance, the lightest path from allantoin to carbon-dioxide has only one step through urate and does not correspond at all to the reference allantoin degradation pathway. Another example concerns glycolysis, where the computed lightest path between glucose and pyruvate is through proton which is a lighter path than the chain passing through glc-6p, fructose-6p, fructose 1,6 diphosphate, gap, dpg, g3p, 2pg and phospho-enol-pyruvate. Finally, the lightest path between pyruvate and malate is a direct connection between them through the reversible reaction oxaloacetate decarboxylating (NADP+). The fact that this was the selected path between these two nodes explains the 0% accuracy obtained for the superpathway of TCA cycle and glyoxylate cycle and the TCA cycle aerobic respiration pathways, since both of them have exactly pyruvate and malate as black nodes.

The issue of selecting wrong candidates between pairs of compounds has a direct impact in the cases where there are linear pathways and, thus, only two black nodes to connect because the method may simply ignore from the start the pathway that it should recover. Notice, for instance, that all the 0% cases have two black nodes. However, this issue also affects the results for more complicated topologies. For instance in the de novo NAD biosynthesis, the lightest path between TRP and NAD is wrongly inferred, passing through indole-pyruvate and indole acetaldehyde instead of the known reference reactions. As a result we have a 0% accuracy for this pathway, but for the superpathway of NAD biosynthesis (accuracy of 29,8%) the same pair of metabolites is present and the same error appears, even if the subpathway between nicotinade riboside and NAD is entirely recovered. This specific pathway also indicates that the maximality feature of the stories definition makes the number of false positive predictions very high, in this case presenting alternative connections between nicotinamide ribose and nicotinate riboside and TRP, which are not part of the reference pathway. This may be not good for the accuracy measure but is a nice way of proposing composed scenarios or superpathways in which different alternative transformations are analysed together, *i.e.*, for the chosen metrics our method is not so good in specificity since stories are maximal but we have a good sensitivity and we are able to include alternative pathways.

The observation that the main factor inducing a decrease in accuracy of our method is the absence of the reference pathway in the lightest-path network strongly indicates that our method would benefit from replacing this approach by some atom mapping routing approach (Boyer and Viari (2003); Arita (2004); Blum and Kohlbacher (2008)), since this will guarantee that chains of reactions will correspond to exchange of carbon atoms at each step and, therefore, are more likely to correspond to the metabolic pathways even in the cases in which they do not correspond to any shortest/lightest path.

Table 3.3: **List of metabolic pathways from *Saccharomyces cerevisiae* used in the pathway retrieval experiment.**

	Pathway	**nodes**	**arcs**	**Black nodes**	TP	FP	FN	**Acc.**
1	ALL-CHORISMATE superpathway of chorismate	36	43	GTP ERYTHROSE-4P THF TYR PHE TRP 7-8-DIHYDROPTEROATE 5-METHYL-THF	37	44	6	62,7%
2	ALLANTOINDEG allantoin degradation	5	4	ALLANTOIN CARBON-DIOXIDE	0	4	4	0%
3	ARGDEG-YEAST arginine degradation (aerobic)	3	2	ARG GLT	2	1	0	81,6%
4	ARO chorismate biosynthesis	3	2	ERYTHROSE-4P CHORISMATE	2	0	0	100%
5	BRANCHED-CHAIN-AA-SYN superpathway of leucine, isoleucine and valine biosynthesis	17	24	PYRUVATE VAL LEU ILE	21	7	3	81%
6	COMPLETE-ARO superpathway of phenylalanine, tyrosine and tryptophan biosynthesis	19	21	ERYTHROSE-4P TYR PHE TRP	18	16	3	67,4%
7	DENOVOPURINE3 de novo biosynthesis of purine nucleotides	18	18	PRPP ATP dGTP dATP	3	8	15	21,3%
8	ERGOSTEROL-SYN superpathway of ergosterol biosynthesis	18	19	ACETOACETYL-COA ACETYL-COA CO-A ERGOSTEROL	12	13	7	55,1%
9	FASYN-ELONG2 fatty acid elongation	8	9	ACYL-ACP OH-ACYL-ACP PALMITOYL-ACP PALMITYL-COA ACP	7	2	2	77,8%

continued on next page

Table 3.3: continuation

	Pathway	**nodes**	**arcs**	**Black nodes**	TP	FP	FN	**Acc.**
10	FASYN-INITIAL fatty acid biosynthesis, initial steps	11	11	BCCP-BIOTIN ACETYL-COA |R-3-Hydroxypalmitoyl-ACPs| BUTRYL-ACP	8	4	3	69,6%
11	FOLSYN folate biosynthesis	14	13	CHORISMATE GTP 5-METHYL-THF	11	4	2	78,8%
12	GLUCFERMEN superpathway of glucose fermentation	6	5	GLC ETOH ACET	1	15	4	11,2%
13	GLUCONEO gluconeogenesis	11	11	MAL GLC-6-P DIHYDROXY-ACETONE-PHOSPHATE	4	7	7	36,4%
14	GLUCOSE-MANNOSYL-CHITO-DOLICHOL lipid-linked oligosaccharide biosynthesis	15	14	DOLICHOLP CPD3O-410	0	2	14	0%
15	GLYCOLYSIS glycolysis	4	3	GLC-6-P PYRUVATE	0	8	3	0%
16	GLYOXYLATE-BYPASS glyoxylate cycle	7	7	MAL SUC GLYOX	7	12	0	60,7%
17	HEXPPSYN hexaprenyl diphosphate biosynthesis	7	6	DI-CH3-ALLYL-PPI DELTA3-ISOPENTENYL-PP HEXAPRENYL-DIPHOSPHATE	6	1	0	92,6%
18	HISTSYN histidine biosynthesis	13	12	ATP HIS AICAR PRPP	12	3	0	89,4%
19	ILEUSYN isoleucine biosynthesis	6	5	THR ILE	5	1	0	91,3%
20	IPPSYN mevalonate pathway	8	7	CO-A ACETYL-COA DI-CH3-ALLYL-PPI	7	4	0	79,8%
21	LYSDEGII lysine degradation	7	6	LYS GLUTARATE	6	0	0	100%
22	LYSINE-AMINOAD	9	8	LYS	0	14	0	0%

continued on next page

Table 3.3: continuation

	Pathway	**nodes**	**arcs**	**Black nodes**	TP	FP	FN	**Acc.**
	lysine biosynthesis			2-KETOGLUTARATE				
23	NADSYN de novo NAD biosynthesis	10	9	TRP NAD	0	3	9	0%
24	NONOXIPENT non-oxidative branch of the pentose phosphate pathway	3	2	GAP RIBULOSE-5P	2	0	0	100%
25	P4 superpathway of threonine and methionine biosynthesis	10	9	HS MET THR L-ASPARTATE	8	6	1	71,3%
26	PANTOSYN2 pantothenate and coenzyme A biosynthesis	12	11	CO-A SPERMINE 2-KETO-ISOVALERATE	8	7	3	62,3%
27	PENTOSE-P pentose phosphate pathway	7	7	GAP GLC-6-P FRUCTOSE-6P	1	1	6	26,7%
28	PHOS superpathway of phosphatidic acid and phospholipid biosynthesis	21	22	SER GLYCEROL CHOLINE CDPDIACYLGLYCEROL CARDIOLIPIN PHOSPHATIDYLCHOLINE-CMPD DIHYDROXY-ACETONE-PHOSPHATE 1-ACYL-DIHYDROXYACETONE-PHOSPHATE DIACYLGLYCEROL	20	21	2	66,6%
29	PHOSLIPSYN2 phospholipid biosynthesis	12	11	SER GLYCEROL CDPDIACYLGLYCEROL CARDIOLIPIN GLYCEROL-3P PHOSPHATIDYLCHOLINE-CMPD	11	17	0	62,7%

continued on next page

Table 3.3: continuation

	Pathway	**nodes**	**arcs**	**Black nodes**	TP	FP	FN	**Acc.**
30	PLPSAL pyridoxal 5'-phosphate salvage pathway	6	5	PYRIDOXAL PYRIDOXINE PYRIDOXAMINE PYRIDOXAL-PHOSPHATE	5	0	0	100%
31	PRPP superpathway of histidine, purine and pyrimidine biosynthesis	31	37	GLN CTP HIS RIBOSE-5P DGTP ATP DATP	25	17	12	63,4%
32	PWY-2201 folate transformations	6	9	THF 5-FORMYL-THF 5-METHYL-THF METHYLENE-THF	7	0	2	88,2%
33	PWY-821 superpathway of sulfur amino acid biosynthesis	15	17	CYS SULFATE L-ASPARTATE S-ADENOSYLMETHIONINE	9	24	8	38%
34	PWY3O-1 salvage pathways of purines and their nucleosides	10	8	GMP IMP GUANINE XANTHOSINE-5-PHOSPHATE INOSINE ADENINE ADENOSINE	5	6	3	53,3%
35	PWY3O-1109 p-hydroxybenzoate biosynthesis	8	7	TYR CHORISMATE |4-hydroxybenzoate|	7	3	0	83,7%
36	PWY3O-19 ubiquinone biosynthesis from 4-hydroxybenzoate	10	9	UBIQUINONE-6 |4-hydroxybenzoate|	9	0	0	100%
37	PWY3O-2 superpathway of phospholipid biosynthesis	20	19	SER CHOLINE GLYCEROL-3P	19	34	0	59,9%

continued on next page

Table 3.3: continuation

	Pathway	**nodes**	**arcs**	**Black nodes**	TP	FP	FN	**Acc.**
				CARDIOLIPIN GLYCEROL MYO-INOSITOL CPD-482 CDPDIACYLGLYCEROL ETHANOL-AMINE PHOSPHATIDYLCHOLINE-CMPD				
38	PWY3O-20 folate polyglutamylation	6	6	5-METHYL-THF 10-FORMYL-THF 7-8-DIHYDROPTEROATE	4	4	2	57,7%
39	PWY3O-2220 salvage pathways of adenine, hypoxanthine and their nucleosides	6	5	IMP INOSINE ADENINE ADENOSINE	2	4	3	36,5%
40	PWY3O-242 phosphatidylinositol phosphate biosynthesis	5	8	CPD-482 PHOSPHATIDYL-MYO-INOSITOL-45-BISPHOSPHA 1-PHOSPHATIDYL-1D-MYO-INOSITOL-35-BISPH	8	0	0	100%
41	PWY3O-261 superpathway of serine and glycine biosynthesis	7	6	G3P THR GLYOX SER	6	11	0	59,4%
42	PWY3O-285 superpathway of purine biosynthesis and salvage pathways	16	20	ATP DGTP PRPP GUANINE DATP ADENINE ADENOSINE	11	31	9	38%
43	PWY3O-351 superpathway of methionine salvage pathway	3	2	SPERMINE SPERMIDINE PUTRESCINE	2	0	0	100%
44	PWY3O-402 inositol phosphate biosynthesis	11	12	CPD3O-768 PHOSPHATIDYL-MYO-INOSITOL-45-BISPHOSPHA	12	1	0	96,1%

continued on next page

Table 3.3: continuation

	Pathway	**nodes**	**arcs**	**Black nodes**	TP	FP	FN	**Acc.**
				DIPHOSPHO-1D-MYO-INOSITOL-TETRAKISPHOSPH				
45	PWY3O-4107 NAD salvage pathway	5	5	NAD DEAMIDO-NAD	1	0	4	44,7%
46	PWY3O-4158 superpathway of NAD biosynthesis	13	12	NAD TRP CPD-8259 NICOTINAMIDE-RIBOSE	4	11	8	29,8%
47	PWY3O-45 folate biosynthesis II	12	11	GTP THF CHORISMATE	11	3	0	88,6%
48	PWY3O-64 methionine salvage pathway	7	6	MET 5-METHYLTHIOADENOSINE	6	0	0	100%
49	PWY3O-69 superpathway of heme and siroheme biosynthesis	12	11	GLY SIROHEME PROTOHEME	11	2	0	92%
50	PWY3O-697 folate interconversions	4	4	5-METHYL-THF THF 10-FORMYL-THF	1	1	3	35,4%
51	PWY3O-7 superpathway of threonine biosynthesis	8	7	PYRUVATE THR	3	2	4	50,7%
52	PWY3O-862 superpathway of ubiquinone biosynthesis	24	23	TYR CHORISMATE UBIQUINONE-6 DI-CH3-ALLYL-PPI DELTA3-ISOPENTENYL-PP	23	10	0	83,5%
53	PWY3O-94 superpathway of TCA cycle and glyoxylate cycle	7	6	PYRUVATE MAL	0	1	6	0%
54	PWY3O-954 superpathway of methionine biosynthesis	8	7	HS MET L-ASPARTATE	6	3	1	75,6%
55	PWY3O-981	3	2	PYRUVATE	2	1	0	81,6%

continued on next page

Table 3.3: continuation

	Pathway	**nodes**	**arcs**	**Black nodes**	TP	FP	FN	**Acc.**
	superpathway of acetoin and butanediol biosynthesis			BUTANEDIOL				
56	PYRIMID-RNTSYN de novo biosynthesis of pyrimidine ribonucleotides	2	1	GLN CTP	1	0	0	100%
57	SPHINGOLIPID-SYN sphingolipid metabolism	12	11	MIP2C SER PALMITALDEHYDE PHOSPHORYL-ETHANOLAMINE	11	4	0	85,6%
58	TCA-EUK TCA cycle, aerobic respiration	5	4	PYRUVATE MAL	0	1	4	0%
59	THREOCAT2 threonine degradation	15	17	GLY THR 1-AMINO-PROPAN-2-OL METHYL-GLYOXAL HOMO-SER CYSTATHIONINE O-SUCCINYL-L-HOMOSERINE PROPIONATE	12	16	5	55%
60	TRIGLSYN triglyceride biosynthesis	6	5	ACYL-ACP GLYCEROL-3P TRIACYLGLYCEROL	3	7	2	42,4%
61	TRPSYN tryptophan biosynthesis	6	5	CHORISMATE TRP	5	0	0	100%
62	TRYPTOPHAN-DEGRADATION-1 tryptophan degradation via kynurenine	8	7	NICOTINATE-NUCLEOTIDE TRP	7	0	0	100%
63	YEAST-ARG-SYN arginine biosynthesis	5	4	GLT ARG	2	3	2	44,7%
64	YEAST-DE-NOVO-PYRMID-DNT de novo biosynthesis of pyrimidine deoxyribonucleotides	11	10	CDP UDP DCMP DUMP TTP	10	8	0	74,5%
65	YEAST-FAO fatty acid oxidation pathway	6	5	ACYL-COA |Fatty-Acids|	3	2	2	60%

continued on next page

Table 3.3: continuation

	Pathway	**nodes**	**arcs**	**Black nodes**	TP	FP	FN	**Acc.**
				CIS-DELTA3-ENOYL-COA				
66	YEAST-GALACT-METAB galactose degradation	5	4	GLC-6-P GALACTOSE	4	0	0	100%
67	YEAST-RIBOSYN riboflavin, FMN and FAD biosynthesis	10	9	GTP FAD RIBULOSE-5P	5	7	4	48,1%
68	YEAST-RNT-SALV salvage pathways of pyrimidine ribonucleotides	6	5	UMP CTP CYTIDINE	5	4	0	74,5%
69	YEAST-SALV-PYRMID-DNTP salvage pathways of pyrimidine deoxyribonucleotides	8	7	TMP DUMP DEOXYCYTIDINE CYTOSINE DCMP	4	10	3	40,4%
	Average	10	9,8		7,1	6,61	2,74	62,72%

3.7 Open problems and perspectives

From a practical point of view, for some graphs, the number of stories found is extremely large and therefore the analysis of the results is compromised without the use of some classification approach like the score functions we used in both of our biological applications.

One possible way to explore the spectrum of solutions is to group stories based on their sets of sources and targets. Unfortunately, in our experiments, when using this approach almost all possible combinations of such sets were highly represented and the number of stories in each group was large and close to a uniform distribution.

Instead of using approaches that intend to classify or to group the numerous solutions, an alternative way to proceed is to add more constraints to the model to be able to filter *a priori* the set of solutions. This observation led us to consider the problem from a modelling point of view. For instance, the acyclicity constraint could be relaxed allowing cycles between white nodes. An even less strict approach would be to allow “intermediate" cycles even if intermediate black nodes participate in such cycles, but with some reasonable rule to define the frontier between the top/bottom level of a story in which black nodes should be sources and targets, and the middle part of a story where cycles are allowed.

Another alternative is to consider integrated models, adding to the metabolic network other sources of information such as regulation, or taking the stoichiometry of the reactions into account. Stoichiometry should be explored in a different way than simply filtering out stories that are not stoichiometrically valid, because stories are in general stoichiometrically

valid if the input metabolic network have mass-consistent reactions, which is expected to be the case. Indeed, as stories are acyclic, we may assume that they are consistent in the sense that it is possible to find a flux over the reaction set that reaches the steady-state, assuming that the inputs and outputs do not need to be balanced. A potentially interesting idea to follow would be to work with an alternative definition of a story exploring the stoichiometric coefficients in order to allow cycles that keep the reaction set balanced. However, considering stoichiometry in such a way cannot be handled with a compound graph representation of the metabolic network and, therefore, we should move to a bipartite graph or, ideally, to a hypergraph representation of the network.

We could consider the definition of a *hyperstory* which is a set of hyperarcs such that the induced hypergraph is maximal (with respect to hyperarcs), acyclic and contains no white source-set or white target-set. This would be the equivalent in terms of hypergraphs of our current definition. Basically, as an hyperarc a in a directed hypergraph is divided in two sets $in(a)$ and $out(a)$, a white source-set is a set $in(a)$ for some hyperarc that contains only white nodes and equivalently a white target-set is a set $out(a)$ for some hyperarc a that contains only white nodes. This alternative definition of a story in terms of a set of hyperarcs may also be a future direction of the current work.

Finally, a current challenge in metabolomics is to correctly predict which are the metabolites corresponding to the peaks in the spectrum and whether the changes in concentration are actually significant, which suggests that the model should also account for noisy data corresponding to an incertitude on the red/green/white labels assigned to the nodes. The presence in the input of such “grey" nodes that could play the role of either a white or a black one is another interesting variation of the original problem. Considering even more details on the metabolomics experiments, it could also be interesting to integrate in the modelling experiments in which we have a time series, *i.e.*, a collection of linked measurements performed in different moments. Finally, it could also be taken into account the fact that some metabolites are not possible to be measured experimentally, which is a yet different kind of “grey" node.

From a theoretical point of view, the complexity of the stories enumeration problem is unknown. The enumeration algorithm we proposed, even if it works well in practice, gives no guarantee on the delay between the output of two consecutive solutions. Notice that any change in the definition might imply a revision of the formal results presented here.

Chapter 4

Precursor Sets

Contents

4.1 Introduction

The results presented in this chapter are strongly based on three papers we published on the enumeration of minimal precursor sets and its applications (Cottret et al. (2008, 2010a); Acuña et al. (2012b)). In Cottret et al. (2008), we formalised the concept of a *minimal precursor set* which corresponds to a set of metabolites that an organism may obtain from its environment and that enables it to produce a set of metabolic targets of interest. In this model, we proposed the first definition of precursor set that takes into account the fact that metabolic networks contain many cycles and that the solutions are expected to pass through them. In both of our first two papers (Cottret et al. (2008, 2010a)) on this subject, we applied our method to enumerate all minimal precursor sets for a given set of targets in order to validate the usefulness of the algorithm for a metabolic analysis, including the study of a relatively complex symbiotic system (Cottret et al. (2010a)). In this case, the environment is represented by an insect, *Homalodisca coagulata*, which hosts within its cells two bacteria, respectively *Baumannia cicadellinicola* and *Sulcia muelleri*. The identification of the precursor sets for the sets of metabolites each bacterium gives to the symbiotic system (host and co-resident endocytobiont) enabled to refine the analysis that had been done previously (McCutcheon and Moran, 2007) of the complementarity between the metabolisms of the two bacteria and their host. It also suggested that both *B. cicadellinicola* and *S. muelleri* might be completely independent of the metabolites output by the co-resident endocytobiont to produce the carbon backbone of the metabolites provided to the symbiotic system. More recently, two new algorithms were proposed that are faster and have an improved memory usage than the original version of the algorithm, allowing it to be applied to genome-wide metabolic network reconstruction (Acuña et al. (2012b)).

A current limitation of our definition of precursor sets is the fact we do not consider the stoichiometry of reactions. However, the values that are currently obtained for the stoichiometric coefficients may often be not accurate. For instance, 51% of the reactions in KEGG were considered to be unbalanced in 2004 (Feist et al., 2009) and solving these cases may become challenging for more complex reactions (Thiele and Palsson, 2010). Our definition proposes a model based only on the topology of a metabolic network, that is, that considers the set of substrates and products of each reaction without considering the amounts of each molecule that are involved in such a reaction. The collection of precursor sets thus defined should therefore be considered as potential/candidate solutions which could be confirmed or discarded *a posteriori* by other sources of information.

While some results were achieved with the first algorithm we proposed (Cottret et al. (2008, 2010a)), the method suffered of a memory consumption problem due to the necessity to construct and store in memory a data structure – called the *replacement tree* - which could be exponentially larger than the input metabolic network. Moreover, the enumeration procedure explored such a tree and it was, therefore, not the most efficient way to enumerate all minimal precursor sets. For small networks (less than 250 nodes), the previous method runs in an acceptable time, but for bigger networks, it usually runs out of memory. This limitation restricted the analysis basically to organisms with a very small metabolic network, which makes it difficult to use it out of the context of endosymbiosis, for which usually the organisms have undergone a large genomic reduction due to its stable and rich environment in the host cell, which implies smaller and specialised metabolic networks. More recently, we presented new algorithms for enumerating all minimal precursor sets that address both memory requirements and time efficiency (Acuña et al. (2012b)), enabling the method to be applied also to large metabolic networks such as the human one.

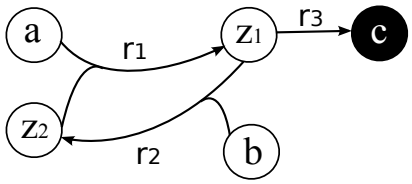


Figure 4.1: A metabolic network containing 3 reactions and 5 metabolites. Considering c as a target metabolite, previous methods could not identify a precursor set for c. Our formal definition of precursor sets that explicitly deals with cycles allows us to identify the set $\{a, b\}$ as a candidate solution, provided that an initial amount of z_1 or z_2 are available. The algorithm must also guarantee that any internal metabolite used in the synthesis of the target may be regenerated.

The two main contributions of our methods are the formal definition of a minimal precursor set and the fact that this formalism takes cycles into account. Before our paper (Cottret et al. (2008)), previous methods related to the identification of precursor sets lacked in formalism (Romero and Karp (2001)) or were not able to deal with cycles in the network (Handorf et al. (2008)). Considering only the topology of a hypergraph, hypercycles like the one shown in Figure 4.1 may be identified in real metabolic networks and, previously to our first method, the target c had no precursor set, since it needs z_1 that could not be produced due to the mutual dependance with z_2 in the cycle formed by reactions r_1 and r_2. However, assuming that either z_1 or z_2 are present in the cell, then the use of the precursors a and b may produce c and also reproduce z_1 or z_2, since they are involved in their own synthesis. Considering this kind of autocatalytic compounds in the definition of precursor sets allowed us to explore solutions that were not taken into account before.

With the objective of detecting inconsistencies in EcoCyc (the database inside BioCyc that is dedicated to the bacterium *Escherichia coli*), Romero and Karp (Romero and Karp (2001)) used a whole-network approach to find precursors. They defined and used an iterative process, called **forward propagation**, that will be presented in more detail later. Basically, it consists in starting with a set of available metabolites and increasing it, iteratively, by adding the products of any reaction whose substrates are available (*i.e.*, are present inside the set). Considering topological sources as potential precursor sets and computing the forward propagation on this set produces a sub-network, that was later called the **scope** (Handorf et al. (2005)) of the initial set. Inspection of the essential metabolites that were not produced (not present in the scope) allowed them to identify missing nutrients (possibly missing transport reactions) or reactions whose reversibility was not correctly identified in the annotation process, as well as other kinds of errors in the metabolic network reconstruction.

More recently, a method to identify minimal precursor sets required by an organism to produce all metabolites contained in a target set was proposed but it also did not consider cycles in its procedure (Handorf et al. (2008)). The algorithm consisted in starting with an ordered list of all metabolites that are not in the target set. If the scope of this list produces the targets, the list corresponds to a precursor set of the target set. The algorithm then proceeds by minimalising the list, iteratively checking whether the removal of a metabolite from the list makes one of the metabolites in the target set not be produced anymore. If this is the case, the metabolite is put back into the list, otherwise the metabolite is identified as not necessary for target production and is definitively removed from the list. The solution obtained may change depending on the order in which the metabolites are considered. Computing all the solutions would require to compute the scope of all possible orderings of the metabolites, which is not computationally realistic. In order to get an approximation of the solution space, several random orderings of the metabolites are computed. Even though the method was applied to more than 400 organisms in order to predict their nutrient requirements, the solutions cannot be considered minimal precursor sets since cyclic solutions were not taken into account.

This led to the idea of introducing the biologically well-founded concept of "self-regenerating" metabolites, that will be internally supplied by the cell itself. Those are metabolites one cannot consider as available in infinite supply, as is the case for precursors, which are provided by the environment. Such metabolites therefore need to be continuously regenerated, but they have the ability to participate, in one or more steps, to their own regeneration, and to the subsequent generation of other metabolites. Self-regenerating metabolites will be part of at least one cycle. They represent the metabolites that one may consider as continuously available in the network even though they are not in infinite supply as the precursors are assumed to be. In Romero and Karp (2001), the authors very informally define what they called bootstrapping compounds that may be related to our self-regenerating metabolites.

In this chapter, we will cover the evolution of the three proposed algorithms, highlighting their differences and key features, as well as a time comparison analysis between them after running the different algorithms for several combinations of metabolic networks and target sets. We will also provide full proofs of the complexity results of the problem and present some biological applications of the method. We also refer to the appendix for details on the implementation of these algorithms and how to use the developed methods.

4.2 Definitions

A metabolic network is modelled as a *directed hypergraph* $G = (M, R)$ with M the set of *vertices* corresponding to metabolites (also called compounds) and R the set of *hyperarcs*

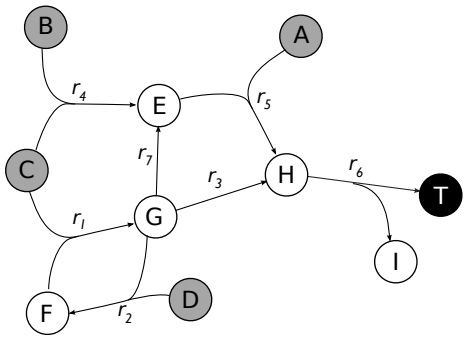


Figure 4.2: A metabolic network. Nodes represent metabolites and hyperarcs represent reactions. Grey nodes are sources while the black node is the target.

corresponding to reactions. A directed hyperarc of a reaction $r \in R$ is an ordered pair of metabolite sets $r = \big(Subs(r), Prod(r)\big)$ where $Subs(r)$ is the set of substrates of r and $Prod(r)$ is the set of products of r. Reactions are supposed to be irreversible: each originally reversible reaction is replaced by two irreversible reactions of opposite direction.

We consider also a set of *sources* $\mathcal{S} \subseteq M$ representing the metabolites that are *potentially* available in infinite external supply. Sources used as substrates of reactions produce other metabolites, thereby increasing the set of available ones. In addition, the set $T \subseteq M$ denotes the *target set*, that is a set of metabolites for which we are interested in inspecting the alternative ways the cell has to produce them. Given a source set $\mathcal{S}$ and a target set T of metabolites, the aim is to find subsets of $\mathcal{S}$ which are able to produce all metabolites of T. We need now to formally define the meaning of: *being able to produce the target*. When stoichiometric information is missing or not (fully) reliable, two definitions have been proposed to model this concept that can give different solutions to a particular instance.

Before comparing the two approaches, we introduce some notation. Let C be a set of metabolites of M. We define $\mathcal{R}_C$ as the set of reactions that can be fired when the metabolites in C are present. In other words, $\mathcal{R}_C = \{r \in R \mid Subs(r) \subseteq C\}$. For a given set of reactions $H \subseteq R$, we define the sets $Subs(H) = \cup_{r \in H} Subs(r)$ and $Prod(H) = \cup_{r \in H} Prod(r)$.

Sequential production of the target

The *forward propagation* of C, denoted by $Fwd(C)$, is the set of metabolites successively produced from C using the reactions of the network. Formally, $Fwd(C)$ is the result of the recursion $C_{i+1} = C \cup Prod(\mathcal{R}_{C_i})$ starting from $C_0 = C$ and until a fixed point is reached. For instance, in the network of Figure 4.2, if $C_0 = \{a, b, c\}$ then $C_1 = \{a, b, c, e\}$, $C_2 = \{a, b, c, e, h\}$, and so on until the fixed point $\{a, b, c, e, h, i, t\}$ is reached. Thus, $Fwd(\{a, b, c\}) = \{a, b, c, e, h, i, t\}$. A pseudocode for the forward propagation algorithm is presented below.

Romero and Karp considered a subset X of the sources $\mathcal{S}$ as a precursor set of a target set T, when $T \subseteq Fwd(X)$ (Romero and Karp, 2001). For instance, the set of sources $X = \{a, b, c\}$ is a precursor set of the target set $T = \{t\}$ since $Fwd(\{a, b, c\})$ contains t. This *iterative* way to calculate what is available from X may however not be enough to model some real cases. Indeed, the network could have cycles whose metabolites need to be consumed and produced *all at the same time*. For instance, in Figure 4.2 reactions r_1 and r_2 form a cycle that consumes metabolites c and d to produce f and g. However $Fwd(\{c, d\}) = \{c, d\}$, that is, it contains neither f nor g.

Algorithm *Fwd*(C)
Require: Implicitly, a metabolic network represented as a hypergraph $G = (M, R)$ and a set $C \subseteq M$ of available metabolites;
Ensure: a set X of metabolites that can be reached from C.

```
X ← C
newMetabolite ← TRUE
while newMetabolite = TRUE do
    for all r ∈ R_X, N ← ⋃prod(r)
    if N ⊆ X then
        newMetabolite ← FALSE
    else
        X ← X ∪ N
return X
```

Considering cycles in the target production

In Cottret et al. (2008), we proposed a model which defines a precursor set using a different approach. Instead of starting the propagation from a subset of the sources, the use of other metabolites (called *internal supply*) is allowed, provided such metabolites are produced by some reaction in a future step of the forward propagation (in addition to the production of the target). Formally, we define $Fwd_Z(C)$, the forward propagation of C with Z (as internal supply), as the result of the recursion $C_{i+1} = C \cup Prod(\mathcal{R}_{C_i \cup Z})$ starting from $C_0 = C$ and until a fixed point is reached. A subset X of the sources is a precursor set of T if T and Z are both included in $Fwd_Z(X)$. For instance, in the network of Figure 4.2, $Fwd_{\{f\}}(\{c, d\}) = \{c, d, f, g, e, h, i, t\}$. Thus, it produces t but also *re-produces* f to keep the cycle working.

We define a set of sources $X \subseteq \mathcal{S}$ to be a **precursor set of** $T \subseteq M$ if there exists a set $Z \subseteq M$ such that $T \cup Z \subseteq Fwd_Z(X)$. In this case, we say that Z is an *internal supply* of the precursor set X.

Of course, the internal supply may be not unique for a given precursor set. In Figure 4.2, both $Z = \{f\}$ and $Z = \{g\}$ are internal supplies for the precursor set $X = \{c, d\}$. Observe also that any set of metabolites which is a precursor set by the Romero and Karp definition will continue to be a precursor set for this definition just considering $Z = \emptyset$.

Precursor cut set

Suppose now that the target is a set of metabolites whose production we want to *avoid*. In this case, we can define the notion of a **precursor cut set** or simply *cut set*, that is, a subset X of sources such that, if they are not present, then the target cannot be produced by any combination of the remaining sources. This concept has a biological application, for instance, in the case where we want a bacterium to avoid producing some given metabolite while providing it with a maximal set of resources that enables it to continue doing its other specific tasks. Notice that in practice, a precursor cut set corresponds to restrictions imposed directly to the environment (for instance, a controlled environment where some nutrients are missing) or to block the transport reactions associated with the precursors in the cut set, in order to avoid the cell to retrieve it from the environment. As an example, in Figure 4.2, the set $\{a, d\}$ is a cut set of $\{t\}$. Formally, we define a set of sources $X \subseteq \mathcal{S}$ to be a **precursor cut set** of $T \subseteq M$ if and only if the set $\mathcal{S} \setminus X$ is **not** a precursor set of T.

If the target contains more than one metabolite, a cut will avoid the production of the *whole* target set but could still produce *some* of their elements (a strict subset of T). If we want to block *each* element of the target, we can slightly modify the network in the following way: given $T = \{t_1, \ldots, t_\ell\}$, we can define a new target metabolite t_{target} and reactions

$r_{t_1}, \ldots, r_{t_\ell}$ with $Subs(r_{t_i}) = \{t_i\}$ and $Prod(r_{t_i}) = \{t_{target}\}$. Clearly, a cut set of the new target set $T' = \{t_{target}\}$ must block the production of each metabolite in T.

Factories of metabolites

We give now a simpler and more natural way to grasp the characterisation of a precursor set X of T by considering, instead of metabolites, a set of reactions $F \subseteq R$ that *connects* X with T. Clearly the reactions must verify these two properties:

1. (*Feasibility of reactions*) Each substrate of the reactions in F is contained in X or is produced by some reaction in F;
2. (*Production of target*) Each metabolite in the target set T (which is not in X) is produced by some reaction in F.

These two conditions can be summarised in one: $T \cup Subs(F) \subseteq X \cup Prod(F)$. In this case, we say that F is a **factory from X to T**.

Theorem 7. *A set of sources $X \subseteq \mathcal{S}$ is a* **precursor set of** *$T \subseteq M$ if and only if there exists a factory from X to T.*

Proof. If X is a precursor set of T, there exists $Z \subseteq M$ such that $T \cup Z \subseteq Fwd_Z(X) = X \cup Prod(\mathcal{R}_{Fwd_Z(X)})$. The set of reactions $F = \mathcal{R}_{Fwd_Z(X)}$ is such that $T \subseteq X \cup Prod(F)$ and $Subs(F) \subseteq Fwd_Z(X) \subseteq X \cup Prod(F)$. Therefore, F is a factory from X to T. Inversely, let F be a set of reactions such that $T \cup Subs(F) \subseteq X \cup Prod(F)$. Defining $Z = Subs(F)$, we have $Fwd_Z(X) = X \cup Prod(\mathcal{R}_{Fwd_Z(X) \cup Subs(F)}))$ which clearly contains $X \cup Prod(F)$. Therefore, $T \cup Z \subseteq Fwd_Z(X)$. □

In the example of Figure 4.2, the set $\{c, d\}$ is a precursor set of $\{t\}$, since the set of reactions $F = \{r_1, r_2, r_3, r_6\}$ is such that $\{t\} \cup \{c, d, f, g, h\} \subseteq \{c, d\} \cup \{f, g, h, i, t\}$. So F is a factory from $\{c, d\}$ to $\{t\}$.

4.3 Complexity results

Given a metabolic network $G = (M, R)$ with $\mathcal{S} \subseteq M$ a set of sources and $T \subseteq M$ a set of target metabolites, we address the theoretical complexity of the following three problems:

MinimalPS(T): find a minimal precursor set $X \subseteq \mathcal{S}$ of T.
MinSizePS(T): find a minimum size precursor set $X \subseteq \mathcal{S}$ of T.
AllPS(T): enumerate all minimal precursor sets $X \subseteq \mathcal{S}$ of T.

We also consider the analogous problems where what is searched are precursor *cut* sets: MinimalPCS, MinSizePCS and AllPCS.

Finding a minimal precursor set and a minimal cut set

Given a set $X \subseteq \mathcal{S}$ of sources, we can compute the maximal set of reactions F_{max} that satisfy the first condition of the factory definition (feasibility of reactions). Thus, to decide whether X is a precursor set of a given T, we can compute F_{max} of X and check whether T is included in the products of F_{max}. To obtain this set, we use the following recursion: starting

from the whole set of reactions $F_0 = R$, compute the set $F_{i+1} = \mathcal{R}_{X \cup Prod(F_i)}$ until a fixed point is reached. Defining $K = max_{r \in R}(|Subs(r)| + |Prod(r)|)$, we have the following result.

Theorem 8. *Given a subset $X \subseteq \mathcal{S}$ of sources and a target set $T \subseteq M$, we can decide in polynomial time $O(|M||R| + |R|^2 K)$ whether X is a precursor set of T.*

Proof. We show the maximality of F_{max}. Let F' be another set of reactions such that $Subs(F') \subseteq X \cup Prod(F')$. Clearly if $F' \subseteq F_i$ then $F' \subseteq \mathcal{R}_{X \cup Prod(F')} \subseteq \mathcal{R}_{X \cup Prod(F_i)} = F_{i+1}$. Since we start with $F_0 = R$, we conclude that $F' \subseteq F_{max}$.

The algorithm iterates at most $|R| - |F_{max}|$ times. Computing F_i takes $O(|M| + |R|K)$ time. Therefore, the running time of the whole procedure is $O(|M||R| + |R|^2 K)$. □

This method provides also a way to find a *minimal* precursor set of T. Starting from $X = \mathcal{S}$, we successively check if when removing a metabolite of X, the targets are still produced, maintaining in X only those that are needed to produce T. We obtain a minimal precursor set in $|\mathcal{S}|$ iterations. A similar procedure is also valid to find a minimal cut set starting from $X' = \emptyset$ and adding sources while the target set is not produced. The set $X = \mathcal{S} \setminus X'$ is a minimal cut set.

Corollary 2. *Both* MinimalPS(T) *and* MinimalPCS(T) *can be solved in polynomial time* $O(|M||R||\mathcal{S}| + |R|^2|\mathcal{S}|K)$.

Minimum size precursor set and cut set

Although finding one minimal precursor set of a target set is easy, obtaining a (minimal) precursor set of *minimum size* is NP-hard. This result is proved by a reduction from the NP-complete problem HittingSet (Garey and Johnson, 1990): given a finite set of elements U and a collection of subsets $\mathcal{I} = \{I_1, \ldots, I_n\}$ of U, find a minimum cardinality subset of elements $H \subseteq U$ such that H intersects all the subsets in $\mathcal{I}$.

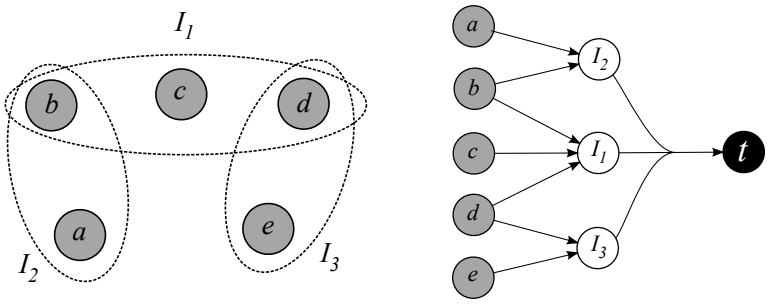


Figure 4.3: Reduction of an instance of the hitting set problem. Each hitting set of $\mathcal{I} = \{I_1, I_2, I_3\}$ corresponds to a precursor set of $\{t\}$ (and *vice versa*).

Theorem 9. *The problem* MinSizePS(T) *is NP-hard.*

Proof. We show hardness by proving completeness of the decision version where we ask if a precursor set of size at most k exists. Theorem 8 implies that this decision version is in NP.

We make a polynomial time reduction from the decision version of HittingSet, asking if there exists a hitting set of size at most k. Consider $H, \mathcal{I}$ and k a hitting set instance with $\mathcal{I} = \{I_1, \ldots, I_n\}$. For each element h in H, we create a vertex h in M, and for each set I_j in $\mathcal{I}$, we create a vertex I_j in M (see Figure 4.3). We create an extra vertex t in M. For each $h \in I_j$,

we create in R an arc r_{hj} going from h to I_j. Moreover, we create the hyperarc r_t having $Subs(r_t) = \{I_1, \ldots, I_n\}$ and $Prod(r_t) = \{t\}$. We define t to be the only target metabolite, and we define the vertices corresponding to the elements of H as the sources $\mathcal{S}$ of G. □

Observe that in the above reduction, there is a one-to-one relation between hitting sets and precursor sets, and a related pair is of the same size. This implies that MinSizePS is as hard to approximate in polynomial time as HittingSet, which is known to be APX-hard (Ausiello et al., 1999). Namely, no polynomial time algorithm for MinSizePS can have approximation ratio $o(\log n)$ unless P=NP (Raz and Safra, 1997).

A similar proof shows NP-hardness for the problem of finding a minimum size cut set. We consider in this case the same reduction but with two modifications (Figure 4.4, left): (a) replace the hyperarc r_t (from $\{I_1, \ldots, I_n\}$ to t) by n separate reactions, from each I_j to t, for $j \in \{1, \ldots, n\}$ and (b) replace, for each I_j, the set of reactions producing I_j by a single reaction r_j producing I_j from the whole set of elements of I_j. In this case, each hitting set corresponds to a cut set. Therefore, MinSizePCS(T) is NP-hard and APX-hard.

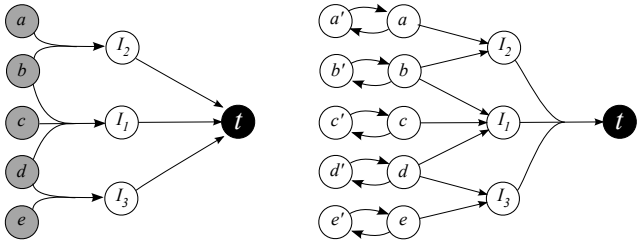


Figure 4.4: Modification of the hitting set reduction to the proof of hardness of MinSizePCS (left) and to the proof of Proposition. 3 (right).

Related hardness results are as follows:

Proposition 3. *Given a precursor set X of T, the following two problems are NP-hard:*

1. *Find a minimum cardinality set of metabolites Z such that Z is an internal supply of X.*
2. *Find a minimum cardinality set of reactions F such that F is a factory from X to T.*

Proof. We modify the reduction presented in the proof of Theorem 9 as follows (Figure 4.4, right): for each element h in H, we create another extra vertex h' in M, and two reactions $r_{hh'}$ and $r_{h'h}$ from h to h' and from h' to h respectively, the set of sources $\mathcal{S}$ is empty and the remaining of the construction stays the same. It is easy to see that the only minimal precursor set of T is the empty set. Using similar arguments as in the previous reduction, we have that any possible set Z corresponds to a hitting set. Analogously, any possible factory F corresponds also to a hitting set. □

Enumerating all minimal precursor sets and cut sets

We showed that MinimalPS$(G, \mathcal{S}, T)$ can be solved in polynomial time. Nevertheless, if we are interested in finding *all* minimal precursor sets of T, the number of solutions can grow exponentially. We are therefore interested in knowing whether AllPS can be solved in polynomial *total* time, that is, polynomial in the size of the input and output (Johnson et al., 1988).

Given a boolean $\wedge, \vee$-formula f (that is, with no negation), a *prime implicant* is a minimal set of variables such that if they are all TRUE then f is TRUE (for instance, $\{p, s\}$ is a prime implicant of $f = (p \vee q) \wedge (r \vee (p \wedge s)) \wedge s)$. Enumerating the set of all prime implicants of f cannot be done in polynomial total time unless P=NP (Gurvich and Khachiyan, 1999). We show that this problem can be reduced to AllPS.

Theorem 10. *The enumeration problem* AllPS *cannot be solved in polynomial total time unless P=NP.*

Proof. Let f be an $\wedge, \vee$-formula. The set M of metabolites corresponds to the set of variables plus one metabolite for each conjunction and disjunction inside the formula (see Figure 4.5). The sources are the metabolites corresponding to each single variable. The set of hyperarcs is as follows: for each metabolite representing a conjunction c in f, there is a single hyperarc from the clauses of c to the metabolite c, and for each metabolite representing a disjunction d, there are arcs from each term of d to the metabolite d. The target set is a singleton containing the metabolite representing f. Clearly, a minimal precursor set of T corresponds to a prime implicant of f and *vice versa*. □

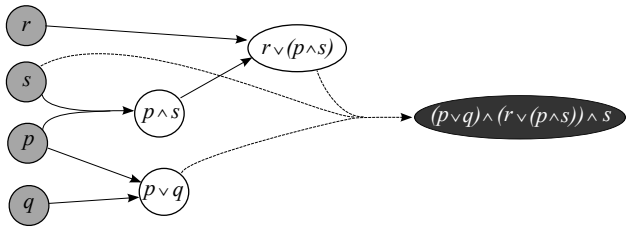


Figure 4.5: Graphical representation of the reduction presented in Theorem 10.

Observe that the reduction holds even in the case of networks without *cycles* (for any reasonable definition of cycle). This result is also valid if we consider the enumeration of all minimal precursor *cut* sets of T. Indeed, in the reduction of Theorem 10, a minimal cut set corresponds exactly to a prime *implicate* of the boolean function f, that is, to a minimal set of variables such that if all are FALSE then f is FALSE. As for prime implicants, enumerating the set of prime implicates cannot be done in polynomial total time unless P=NP (Gurvich and Khachiyan, 1999). Thus, the enumeration problem AllPCS cannot be solved in polynomial total time unless P=NP.

4.4 Algorithms for precursor sets enumeration

In this section, we present our three methods for computing minimal precursor sets for a target set T. To facilitate the exposition, we suppose that the metabolic network has the following properties: (a) each source $x \in \mathcal{S}$ is not produced by any reaction, (b) each reaction belongs to at least one factory from the sources to the target, and (c) the target set is a singleton. It is not difficult to see that by applying the following preprocessing steps, we transform any network in order to satisfy these conditions without changing the collection of precursor sets of a target T:

(a) *Sources are not products:* Rename as x' each x in $\mathcal{S}$ that is the product of at least one reaction. Then, add a new reaction with substrate a new metabolite labelled x and product x'. The set of sources continues to be $\mathcal{S}$.

(b) *All reactions in a factory:* Compute the maximal factory (see proof of Theorem 8) and remove all reactions in the complement. Remove all unconnected metabolites.

(c) *Singleton target set:* If T is not a singleton, add an extra metabolite t and an extra reaction R_t such that $Subs(R_t) = T$ and $Prod(R_t) = t$. The minimal precursor sets of t are the minimal precursor sets to all substrates of R_t, *i.e.*, to the original target set T.

There are two key features that are common to the three algorithms we proposed for precursor sets enumeration. The first one is that we always do a backtracking from the single target, performing a sort of reversed depth-first search on the hypergraph by using reactions in their opposite direction. In this way, the factories that produce T starting from any minimal source sets are covered.

The second key feature of the three algorithms is the decomposition into *subproblems*, where each subproblem has a set M as target set. Starting from $M := T$, two kinds of subproblem decompositions are successively applied:

(a) *Target decomposition:* Given a target set $M = \{m_1, \ldots, m_k\}$ and X a precursor set of M, then X can be written as $X = \cup_{i=1}^{k} X_i$ where each X_i is a precursor set of $M_i = \{m_i\}$. Thus, we can enumerate $\mathbb{P}_M$ by enumerating $\mathbb{P}_{\{m\}}$ (the minimal precursor sets of $\{m\}$) for each metabolite $m \in M$ and taking all the corresponding cross-unions of solution sets (one set from each collection).

(b) *Reaction decomposition:* If the target is a singleton $M = \{m\}$ and $\{r_1, \ldots, r_\ell\}$ is the set of all reactions producing m (which is not empty if m is not a source), then X is a precursor set of M if and only if X is a precursor set of some $M_i = Subs(r_i)$ with $i \in \{1, \ldots, \ell\}$. Thus, to enumerate $\mathbb{P}_{\{m\}}$, we can enumerate $\mathbb{P}_{Subs(r)}$ for all the reactions r that produce m, and then take the union of the collections.

In both decompositions, solutions are obtained after discarding the possible non-minimal sets obtained. Successively alternating these decompositions, the aim is to have subproblems where the target is a singleton *source* $\{s\}$, which has the set $\{s\}$ itself as the only precursor set, that is, $\mathbb{P}_{\{s\}} = \big\{\{s\}\big\}$.

Algorithm PITUFO*: Replacement Tree*

Our first approach to the problem (Cottret et al. (2008)) was the algorithm PITUFO that enumerates all minimal precursor sets by building a *replacement tree*, which represents all paths obtained going from T to the sources by using the reactions in reverse order. The idea of this data structure is that in any two consecutive levels of the tree, we replace a compound at the top level by all the sets of metabolites that produce it at the bottom level, as exemplified in Figure 4.6 that shows a metabolic network and its corresponding replacement tree. This data structure shows exactly the application of a target decomposition followed by as many reaction decompositions as needed.

The replacement tree may be seen as a materialisation of the backtracking and decomposition into a subproblem approach. This hierarchical bipartite tree data structure has two kinds of nodes: metabolites, drawn as circles in Figure 4.6, and reactions, drawn as squares. The root of the tree is the target metabolite and is the only node in the first level of the tree. At the second level, there are as many reaction nodes as there are reactions in the network producing the target, and at the 3rd level there are metabolite nodes corresponding to the substrates of each of such reaction, each one of them connected to their corresponding reaction node. We recall that the idea is that any of these sets of substrates is able to produce the target and, in this sense, we replace the production of the target by the presence of all of such substrates. The structure continues to be built in a recursive manner, with each of the

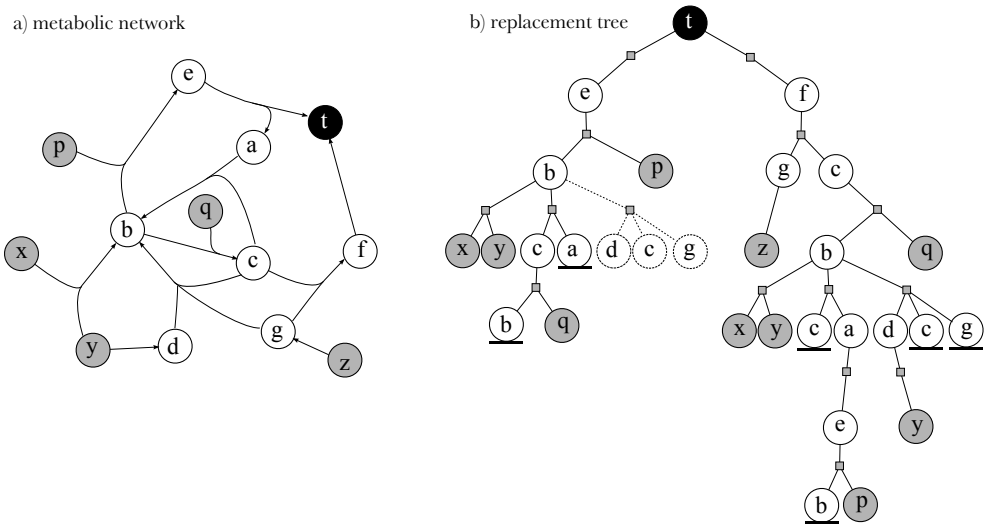


Figure 4.6: Example of a replacement tree (b) corresponding to a metabolic network (a) for the target set $T = \{t\}$. Circles represent metabolites and squares represent reactions. At each level, one metabolite is replaced by the substrates of the reactions that produce it. The target is presented in black, precursors are grey and the remaining metabolites are white. Metabolites identified as internal supply are denoted by an underlying bar. Notice that on the left branch, second reaction producing b, the metabolite a is marked as repeated even if it seems that it does not appear in this branch of the replacement tree but, in fact, a is one of the products of the reaction $e \rightarrow a + t$. Finally, notice that the set of substrates $\{d, c, g\}$ of the 3rd reaction producing b does not need to be analysed since it is not minimal compared to the set of non-repeated substrates of the 2nd reaction, which contains only the metabolite c.

substrates playing the role of the target at each new recursive call. There are two stop conditions for this recursion: finding a precursor or finding a metabolite that is already present on a given branch up to the target. Precursors are naturally expected to be the leaves of such a tree and represent a natural stop condition while repeated metabolites identify a cycle in the network. The metabolites that belong to such cycles are the candidates to be internal supply metabolites and thus are automatically identified by the algorithm. Notice that in Figure 4.6, the precursors are identified in grey while the internal supply metabolites are annotated with an underlying bar.

Finally, it is important to make a distinction between the **or** relationship between a metabolite node and the list of possible reactions that are able to produce it and the **and** relationship between a reaction node and its substrates, since for the reaction to take place all of such substrates must be present. The algorithm basically consists in building recursively the replacement tree and, in a second step, compacting it from the sources to the target until it reaches depth 2, where the solutions are easily recovered as demonstrated in figure 4.7. As the solutions are minimal, there are several steps in which intermediate solutions are eliminated due to non-minimality. For instance, notice that while doing the compression from the replacement tree shown in Figure 4.7(c) to the replacement tree shown in Figure 4.7(d), the reaction $x + y + q \rightarrow c$ is not minimal with respect to $y + q \rightarrow c$, and therefore is not added to the final tree obtained after the compression. The main problem of this method is the huge amount of memory needed to build the replacement tree, which made the algorithm useful only for small networks.

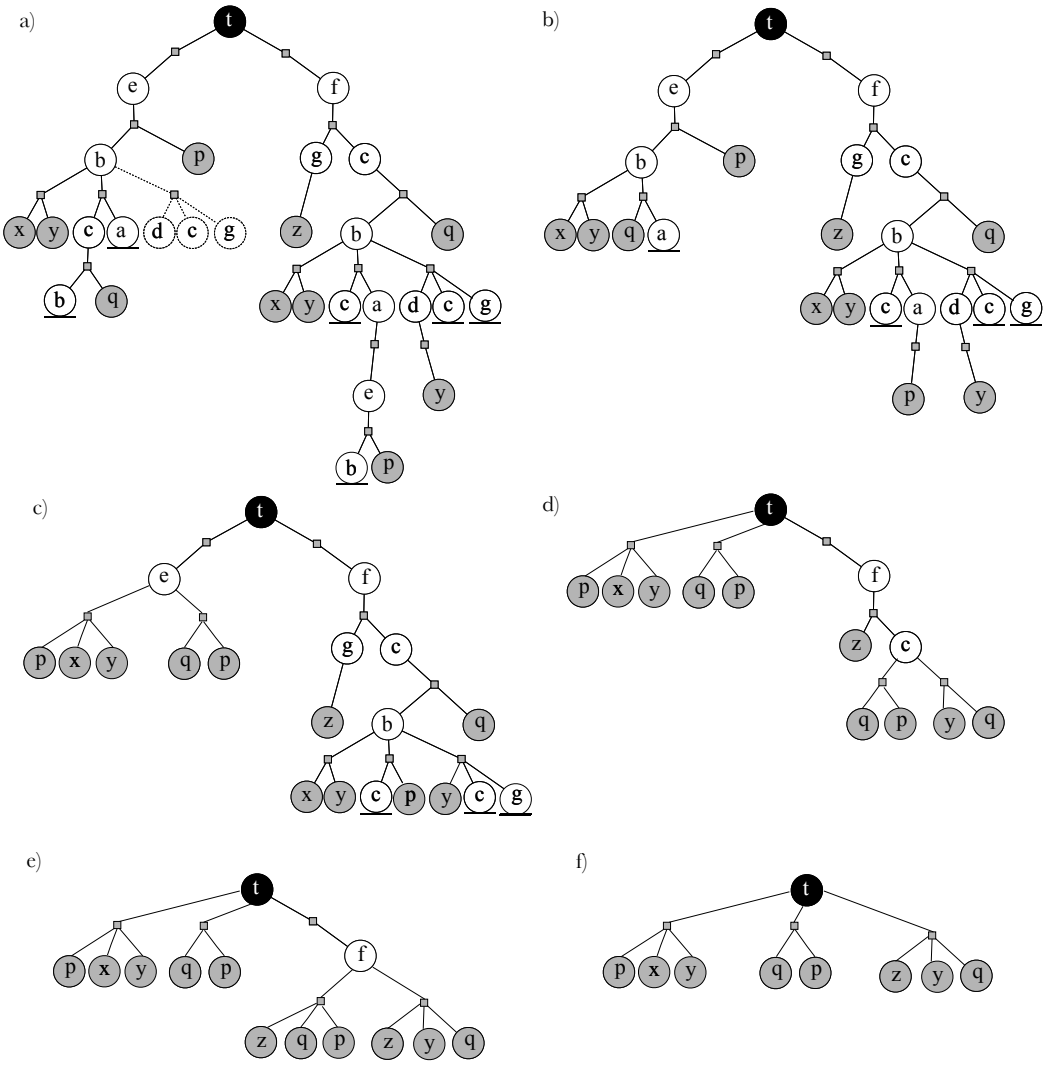


Figure 4.7: Example of the compression steps in which a replacement tree is compressed from bottom to up until the minimal precursor sets are directly connected to the target node.

Including the available metabolites

Before presenting the two newer algorithms, let us explore in more detail the topological information given by the replacement tree, that highlights structural properties of the minimal precursor set enumeration problem, such as the stop condition based on cycles and the possibility of pruning branches due to the expected minimality of the solutions.

One important step of the PITUFO algorithm is to identify a set of metabolites that are in the same branch up to the target in order to identify cycles. For this reason, we must explicitly include in the input of the subproblems the set of *available metabolites*, that is, the metabolites already analysed in previous steps of the algorithm. In this way, we can avoid continuing the search for precursor sets of these metabolites.

Thus, given a set A of available metabolites, we conveniently consider the following generalisation of the precursor set definition: given a set of sources $\mathcal{S}$, a target set M and a set A of available metabolites, we say that a set $X \subseteq \mathcal{S}$ is a *precursor set of M when A is available* if there is a factory from $X \cup A$ to M.

Observe that, if $\mathbb{P}_M(A)$ denotes the collection of all precursor sets of M when A is available, then $\mathbb{P}_T(\emptyset)$ is exactly the collection of all minimal precursor sets of T. Thus, starting from T and having $A = \emptyset$ available, we successively apply the target and reaction decompositions increasing, at each step, the set of available metabolites. The recursion continues solving the subproblems $\mathbb{P}_{\{m\}}(A)$ until one of these two *base* cases appears:

- (a) *m is available*: if m is in A, then $\mathbb{P}_{\{m\}}(A)$ contains only the empty set as element, *i.e.* $\mathbb{P}_{\{m\}}(A) = \big\{\emptyset\big\}$,
- (b) *m is not available but is a source*: if $m \in \mathcal{S} \setminus A$, then $\mathbb{P}_{\{m\}}(A) = \big\{\{m\}\big\}$.

We show how the set of available metabolites can be increased in each subproblem decomposition. Observe that increasing the set A of available metabolites is not only necessary to avoid cycling. The bigger is this set, the shorter are the factories that produce the given target when A is available, that is, we can arrive faster to the base case (a). Thus, in each decomposition, we try to maximise the set of available metabolites that can be added without changing the solution of the original problem.

As mentioned before, to enumerate the collection $\mathbb{P}_M(A)$, we can enumerate the collections $\mathbb{P}_{\{m_i\}}(A)$ for each $m_i \in M$ and compute all possible cross-unions of its elements (one from each collection). In fact, in the enumeration of the solutions of $\mathbb{P}_{\{m_i\}}(A)$, we can include as available any other metabolite in M different from m_i, that is, $\mathbb{P}_{\{m_i\}}\big(A \cup (M \setminus \{m_i\})\big)$. Indeed, we know that these metabolites will be produced by the precursor sets given by the other *parallel* subproblem solutions. In the next lemma, for a given collection of sets $\mathbb{X}$, *minimal*$[\mathbb{X}]$ is the collection of all sets of $\mathbb{X}$ that are not supersets of any other set of $\mathbb{X}$.

Lemma 3. *Given the sets $M = \{m_1, \ldots, m_\ell\} \subseteq M$ and $A \subseteq M$, we have the following relation:*

$$\mathbb{P}_M(A) = \mathit{minimal}\Big[\Big\{ \bigcup_{i=1}^{\ell} X_i \text{ s.t. } X_i \in \mathbb{P}_{\{m_i\}}\big(A \cup (M \setminus \{m_i\})\big) \Big\}\Big]$$

Proof. Given $X \in \mathbb{P}_M(A)$, there is a factory F such that $M \cup \mathit{Subs}(F) \subseteq X \cup A \cup \mathit{Prod}(F)$. Therefore, $\{m_i\} \cup \mathit{Subs}(F) \subseteq X \cup A \cup \mathit{Prod}(F)$ for all $m_i \in M$. Adding $(M \setminus \{m_i\})$ to the right side, we conclude that F is a factory from $X \cup (M \setminus \{m_i\}) \cup A$ to $\{m_i\}$ for all $m_i \in M$.

Conversely, given for all $m_i \in M$ the sets $X_i \in \mathbb{P}_{\{m_i\}}((M \setminus \{m_i\}) \cup A)$, there exist sets F_i such that $\{m_i\} \cup \mathit{Subs}(F_i) \subseteq X_i \cup (M \setminus \{m_i\}) \cup A \cup \mathit{Prod}(F_i)$ for all $m_i \in M$. This implies that $m_i \in X_i \cup A \cup \mathit{Prod}(F_i)$, and also implies that $M \cup \mathit{Subs}(F) \subseteq X \cup M \cup A \cup \mathit{Prod}(F)$ where $F = \cup_i F_i$ and $X = \cup_i X_i$. These two relations in turn imply $M \cup \mathit{Subs}(F) \subseteq X \cup A \cup \mathit{Prod}(F)$. □

In the case of *reaction decomposition*, computation of $\mathbb{P}_{\{m\}}(A)$ (when we are not in one of the base cases) requires to compute $\mathbb{P}_{\mathit{Subs}(r)}(A)$ for any reaction r producing m. Clearly, since r produces m, we can include m as available in the subproblems, that is $\mathbb{P}_{\mathit{Subs}(r)}(A \cup \{m\})$ (which avoids getting into an endless loop). Furthermore, we can also include any other product of r, that is $\mathbb{P}_{\mathit{Subs}(r)}(A \cup \mathit{Prod}(r))$.

Lemma 4. *Given $m \in M$ and $A \subseteq M$, if $m \notin \mathcal{S} \cup A$, then we have the following relation:*

$$\mathbb{P}_{\{m\}}(A) = \mathit{minimal}\Big[\bigcup_{\forall r \text{ producing } m} \mathbb{P}_{\mathit{Subs}(r)}\big(A \cup \mathit{Prod}(r)\big) \Big]$$

Proof. Consider $X \in \mathcal{S}$ and $F \subseteq R$ such that $Subs(r) \cup Subs(F) \subseteq A \cup Prod(r) \cup X \cup Prod(F)$. Since $m \in Prod(r)$, we have $\{m\} \cup Subs(\{r\} \cup F) \subseteq A \cup X \cup Prod(\{r\} \cup F)$. Hence, $\{r\} \cup F$ is a factory from $A \cup X$ to $\{m\}$. Conversely, consider $X \in \mathcal{S}$ and $F \subseteq R$ a factory from $X \cup A$ to the target $\{m\}$. Then F must contain a reaction r that produces m. Therefore, $Subs(r) \cup Subs(F) = Subs(F) \subseteq A \cup X \cup Prod(F) \subseteq A \cup Prod(r) \cup X \cup Prod(F)$. F is also a factory from $X \cup A \cup Prod(r)$ to $Subs(r)$.

Hence, the collection of sets X having a factory from $X \cup A$ to the target $\{m\}$ is the same as the collection of sets of X having a factory from $X \cup Prod(r) \cup A$ to $Subs(r)$ for any r producing m. By considering minimality on each side, we conclude the proof. □

Pruning solutions by minimality

While performing reaction decomposition, there might exist reactions that can be *a priori* discarded because they do not give any *minimal* solution. Indeed, if r and r' produce m, and furthermore if the set of substrates of r that are not in A is a subset of the set of substrates of r' that are not in A, then for any solution to the subproblem defined on $Subs(r')$, we have a solution smaller or equal on $Subs(r)$. In other words, any solution given by r' is not minimal or is included in the solutions given by r. Therefore, we can avoid computing $\mathbb{P}_{Subs(r')}(Prod(r') \cup A)$ without losing minimal precursor sets.

Lemma 5. *Let r and r' be two reactions producing m such that $Subs(r) \setminus A \subseteq Subs(r') \setminus A$. Then for any solution $X' \in \mathbb{P}\big(Subs(r'), Prod(r') \cup A\big)$, there is a solution $X \in \mathbb{P}\big(Subs(r), Prod(r) \cup A\big)$ such that $X \subseteq X'$.*

Proof. Since $X' \in \mathbb{P}_{Subs(r')}(Prod(r') \cup A)$, there exists X' such that $Subs(r') \cup Subs(F) \subseteq Prod(r') \cup A \cup X' \cup Prod(F)$. Therefore $Subs(F \cup \{r'\}) \cup A \subseteq A \cup X' \cup Prod(F \cup \{r'\})$. By hypothesis, $Subs(r) \subseteq Subs(r') \cup A$, and then we can add $Subs(r)$ to the left side of the previous equation: $Subs(r) \cup Subs(F \cup \{r'\}) \cup A \subseteq A \cup X' \cup Prod(F \cup \{r'\})$. By removing the union of A on the left and adding the union of $Prod(r)$ on the right, we obtain $Subs(r) \cup Subs(F \cup \{r'\}) \subseteq A \cup Prod(r) \cup X' \cup Prod(F \cup \{r'\})$. In other words, $F \cup \{r'\}$ is a factory from $A \cup Prod(r) \cup X'$ to $Subs(r)$. We conclude that there exists $X \subseteq X'$ such that $X \in \mathbb{P}_{Subs(r)}(Prod(r) \cup A)$. □

Thus, if we compute the solutions of $\mathbb{P}_{\{m\}}(A)$ by using Lemma 4, we can first compute $Subs(r) \setminus A$ for all reactions r producing m and not consider those reactions where this set is not minimal.

Algorithm Pitufina*: Target-Reaction Decomposition*

Our second algorithm, called Pitufina, consists in successively applying target and reaction decompositions using the procedures *TargetDecomp* and *ReactionDecomp* defined below until reaching the base cases. The first method uses the subroutine *CrossUnions*($\mathbb{U}$, $\mathbb{P}_m$) that computes the collection of all unions of one set of $\mathbb{U}$ and one set of $\mathbb{P}_m$. The idea is exactly the same as the one explored by Pitufo but avoiding to build the replacement tree, *i.e.*, now the algorithm traverses directly the metabolic network, keeping track of the path from the target to the sources. Running *TargetDecomp*(M,A), we obtain exactly all the minimal precursor sets of M when A is available. Therefore, Pitufina algorithm is simply given by the execution of *TargetDecomp*(T,$\emptyset$).

The idea of the *TargetDecomp* procedure is to find all minimal precursor sets that produce all metabolites contained in the set M, considering that the metabolites in the set A are available. For each metabolite $m \in M$, the algorithm first checks if $m \in A$ or if $m \in \mathcal{S}$

Algorithm TargetDecomp($M \subseteq M$, $A \subseteq M$)
Require: A set M of metabolites and a set A of available metabolites.
Ensure: Minimal precursor sets that produces M.

```
𝕌 := [{}]
for all metabolite m ∈ M do
    if m is in A then
        ℙ_m := [{}]
    else
        if m is in 𝒮 then
            ℙ_m := [{m}]
        else
            ℙ_m :=ReactionDecomp(m, A ∪ (M \ {m}))
    𝕌 := CrossUnions(𝕌, ℙ_m)
return minimal(𝕌)
```

because these are the two stop conditions. In the first case, a cycle involving m was identified and in the second case m is a precursor set. If both checks fail, then the algorithm has to dig deeper to find the minimal precursor sets that allow the production of m and it calls the procedure *ReactionDecomp* to obtain such a response, updating the set of available metabolites by adding the siblings of m in the set M. The algorithm finishes by returning a cross union of the solutions obtained for each $m \in M$, *i.e.*, the minimal precursor sets for producing the whole set M of metabolites.

Algorithm ReactionDecomp($m \in M$, $A \subseteq M$)
Require: A target metabolite m and set A of available metabolites.
Ensure: Minimal precursor sets that produce m.

```
ℙ := [ ];
for all reaction r producing m with Subs(r) \ A minimal do
    𝕌_r :=TargetDecomp(Subs(r), A ∪ Prod(r) )
    ℙ := ℙ ∪ 𝕌_r
return minimal(ℙ)
```

The procedure *ReactionDecomp*, on its turn, has to solve the problem of finding the minimal precursor sets for one metabolite m, considering that the metabolites in the set A are available. The algorithm consists in identifying the reactions r that are able to produce m and to filter out the ones that are not minimal. An important remark is that the minimality test is made considering only non-available metabolites, *i.e.*, ignoring the ones contained in the set A. For each reaction r producing m, as all its substrates also need to be produced for the reaction to take place and for m to be produced, we have a new instance for the *TargetDecomp* algorithm considering $Subs(r)$ as the set M of target metabolites and including $Prod(r)$ in the set A of available metabolites. The procedure finishes by returning the minimal sets obtained after processing all of such reactions.

Algorithm Papa Pitufo*: Including factories as pseudo-reactions*

A limitation of Pitufina is its dependency on the procedure *ReactionDecomp*(m, A), that outputs all precursor sets of $\{m\}$ when A is available. Therefore, if X is a set included in the output, we know that there exists a factory from $X \cup A$ to $\{m\}$. This dependency is a problem in the cases where *ReactionDecomp* is called more than once for the same metabolite m and with a similar (or equal) set of available metabolites A'. Indeed, then most of the successive decompositions will be repeated again until the base cases are reached and a result very similar (or equal) to X will be produced as output. In this sense, the algorithm has no memory about the factories previously computed.

We propose a new algorithm that, each time that a decomposition is finished, includes this information in the network by adding *pseudo-reactions* representing the previously computed factories. For instance, if in the network, there is a factory from $X \cup A$ to $\{m\}$ given by reactions r_1 and r_2, then we include a pseudo-reaction $\bar{r}_{1+2}$ with $Subs(\bar{r}_{1+2}) = X \cup A$ and $Prod(\bar{r}_{1+2}) = \{m\}$. Clearly this operation is *safe*: the precursor sets of T do not change.

Moreover, we do not want to lose the information about the remaining of the metabolites produced by the factory, which are used to increment the set A of available metabolites. For this reason, we associate to each pseudo-reaction $\bar{r}$ a set $Int(\bar{r})$ of *internal available metabolites* which contains any metabolite produced by the reactions represented by $\bar{r}$. Thus, if we use this reaction in a future decomposition, we can include the metabolites in this set in the current set A of available metabolites. If we define $Int(r) = Prod(r)$ for any original reaction r of the network, then we do not need to distinguish between reactions and pseudo-reactions. In the previous example, we have then that $Int(\bar{r}_{1+2}) = Int(r_1) \cup Int(r_2)$.

Adding new pseudo-reactions to the network decreases the number of reactions of the factories from X to $\{m\}$, since it creates shortcuts that come directly from the sources to m. However, to really decrease the execution time, we need to ensure that the algorithm will not consider again the original factory. Otherwise, if m is revisited, the algorithm would analyse both the original factory and the new one containing the new added reactions. To avoid this, we delete the original reaction producing m. Of course this operation must be done in a way that guarantees that the collection of minimal precursor sets of T is maintained.

Suppose that we want to delete a reaction r producing m. Notice that any factory from X to m that contains r must also contain at least one reaction producing each substrate of r (except the sources). Thus, if r is merged with each set of such reactions, then r can be removed without modifying the minimal precursor sets.

More formally, given a reaction r, we say that a set of reactions R is a *predecessor reaction set* of r, if R produces all the substrates of r that are not sources, that is, $Prod(R) \supseteq Subs(r) \backslash \mathcal{S}$. Let $\mathbb{R}_{min}(r)$ be the collection of all minimal predecessor reaction sets of r. Clearly, any factory containing r must also contain a set $R \in \mathbb{R}_{min}(r)$. The method *Replace*$(r)$, shown below, explains how to remove reaction r and add pseudo-reactions corresponding to the merge of r with every reaction set $R \in \mathbb{R}_{min}(r)$. Figure 4.8 provides an example of how the *Replace* method works.

Algorithm Replace($r \in R$)

Require: A reaction r to be replaced.
Ensure: Removal of r and its replacement by pseudo-reactions that merge r with reactions producing $Subs(r)$.

Compute $\mathbb{R}_{min}(r) = min\{M \subseteq R \mid Prod(M) \supseteq Subs(r) \setminus \mathcal{S}\}$
for all set $M \in \mathbb{R}_{min}$ **do**
  Add a new reaction $\bar{r}_M$ to the network
  $Prod(\bar{r}_M) := Prod(r)$
  $Int(\bar{r}_M) := Int(M) \cup Int(r)$
  $Subs(\bar{r}_M) := (Subs(M) \cup Subs(r)) \setminus Int(\bar{r}_M)$
Remove r from the network

Lemma 6. *Let $r \in R$ be a reaction of the network G and let G' be the network that results from applying the procedure* Replace(r). *Then, X is a precursor set of T in G if and only if X is a precursor set of T in G'.*

Proof. The factories in G which do not include r are factories also in G'. Let $F \subseteq R$ be a factory from X to T in G which contains r. Clearly, F must contain a set $R \in \mathbb{R}_{min}(r)$. Thus, the set $F' = F \cup \{\bar{r}_R\} \setminus \{r\}$ is a factory from X to T in G'. Conversely, if F' is a factory from

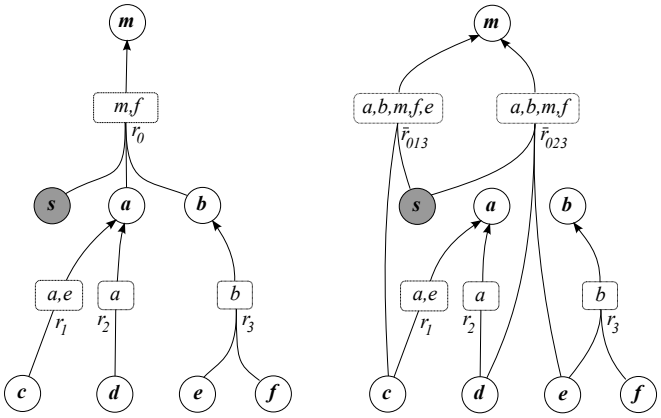


Figure 4.8: Example of the application of *Replace* to reaction r_0. *Left:* Reaction r_0 has internal production m and f (enclosed in a rectangle). The substrates of r_0 are s (which is a source), a and b. The collection $\mathbb{R}_{min}(r_0)$ contains the minimal sets of reactions that produce a and b, that is, $\mathbb{R}_{min}(r_0) = [\{r_1, r_3\}, \{r_2, r_3\}]$. *Right:* We replace r_0 by new reactions corresponding to the merge of r_0 to each set of reactions of $\mathbb{R}_{min}(r_0)$. Thus, reaction r_0 is replaced by reactions $\bar{r}_{013}$ and $\bar{r}_{023}$. Notice that the substrates of $\bar{r}_{013}$ do not include substrates of r_3 since they are internally produced by r_1 and r_0.

X to T in G' containing the set $R_{\text{new}} = \{\bar{r}_{R_1}, \ldots, \bar{r}_{R_k}\}$ of new reactions added by *Replace(r)*, then $F = (F' \setminus R_{\text{new}}) \cup \{r\} \cup \bigcup_{i=1}^{k} R_i$ is a factory from X to T in G. □

We define a new algorithm Papa Pitufo to compute all precursor sets of a target T based on the idea of reaction replacement. The following preprocessing of the network is required: a new metabolite t and a new reaction r_t are created, such that $Subs(r_t) = T$ and $Prod(r_t) = \{t\}$. Clearly, the minimal precursor sets of $\{t\}$ are exactly the minimal precursor sets of T.

Starting from $r := r_t$, Papa Pitufo traverses the network in the same way that Pitufina does. However, instead of computing the minimal solutions at each step down, Papa Pitufo goes deep in the recursion until finding a reaction r satisfying the following two conditions: (a) *not all* substrates of r are in the base cases, and (b) *all* substrates of all reactions in the next level of the recursion are in the base cases.

When such a reaction is found, then it is replaced by new reactions. Successively removing and adding reactions in this way, we decrease the size of the factories from $\mathcal{S}$ to $\{t\}$. Finally, the last reaction removed is r_t which is replaced by new reactions producing t and having only sources as substrates, in a process very similar to the compression of the replacement tree data structure. The substrates of each reaction correspond exactly to a minimal precursor set of $\{t\}$.

Running Papa Pitufo$(r_t, \emptyset)$ we obtain a network where the minimal precursor sets are exactly the substrate sets of all the reactions that produce t. The network can also contain many other reactions, but they are not even connected to t.

Algorithm PapaPitufo($r_0 \in R$,$A \subseteq M$)
Require: A reaction r producing the target and a set A of available metabolites.
Ensure: All factories from X to t will replace r_0, thus the minimal precursor sets of t may be easily recovered.

```
M := Subs(r_0)
if M contains a metabolite not in A ∪ S then
    for all metabolite m ∈ M \ (A ∪ S) do
        NewA := A ∪ (M \ {m})
        for all r producing m with Subs(r) \ NewA minimal do
            PapaPitufo(r, NewA ∪ Int(r))
    Replace(r_0)
```

4.5 Performance analysis

Extensive tests were performed in order to measure the performance of the different algorithms on real metabolic networks. These algorithms were compared for several different singleton target sets (for instance, amino-acids, metabolites related to the synthesis of the cell wall, DNA, RNA, membranes, etc.) in 7 networks of different sizes and topologies downloaded from MetExplore (Cottret et al., 2010b). Ubiquitous metabolites were filtered out and the split reactions using pairs of co-factors option was chosen.

We adopted an automatic process to define the set of sources based on the topology of the network. A metabolite m is considered a source if it satisfies one of these two conditions: (a) m is not the product of any reaction, or (b) m is involved in only two reactions corresponding to both directions of an originally reversible reaction (*i.e.* m is substrate in one and product in the other). The target sets were chosen based on their role: amino-acids, metabolites related to the synthesis of the cell wall or DNA, etc.

Removing void cycles

There are some cycles that we know *a priori* that are not realistic since they are able to produce compounds outside the cycle without the need of any input. In particular, the two directions of an originally reversible reaction form a cycle which can produce its metabolites without using any external source. These *void cycles* must be avoided in factories since they may create fake solutions in which an empty set of metabolites produces the target.

In order to avoid void cycles in factories, we preprocess the input network breaking this kind of cycles by removing some reactions. Specifically, starting from a set M of metabolites containing only the target and an empty set R of reactions, we include in R a randomly chosen reaction producing a metabolite of M unless its inclusion generates a void cycle. The substrates of the added reactions are included in M. Successively repeating this process we obtain a network with no void cycles. Notice that this process corresponds to a heuristic whose result depends on the order in which reactions are chosen to be included in R.

Benchmarks

Table 4.1 presents an extract of the results for PITUFO, PITUFINA and PAPA PITUFO. For the two last ones, we also tried a variation turning on and off a minimality test that had a cost to be checked but that could save effort on processing by pruning non-minimal branches. The targets presented are those for which finding the minimal precursor sets required more time considering the times obtained by PITUFINA and PAPA PITUFO with the minimality test turned on. The table shows, for each network, the size of the sets of metabolites and reactions, and for each target, the size of the preprocessed network, the number of precursor

sets found and the time in seconds that each algorithm spent.

All algorithms have been implemented in Java, more details on the modelling and coding aspects will be given in the Appendix, and the running times were collected using a cluster for the computation and setting a limit of 1GB of RAM memory for each process. Although PITUFO may be fast for small networks, its use is limited since, as the size of the networks grows, the method takes a long time to finish, and for some targets it does not finish in the given time limit of 24 hours. This already justifies the other two methods, since they do not present the same behaviour for larger networks.

Table 4.1: Runtime (in seconds) for computing minimal precursor sets of three singleton targets in 7 different networks using 5 methods: PITUFO, PITUFINA without and with the minimality pruning (resp. *All* and *Min*) and PAPA PITUFO without and with the minimality pruning (resp. *All* and *Min*). All methods were applied to the same preprocessed network on each target. Notice that the same instance (and not a random one) was used for collecting running time for the different methods.

Network ($\|M\|/\|R\|$)	PITUFO	PITUFINA		PAPA PITUFO	
Target ($\|M\|/\|R\|$ after preprocess)		*All*	*Min*	*All*	*Min*
S. muelleri (75 / 65)					
L-Arginine (33/22)	0.017	0.062	0.02	**0.015**	0.018
L-Isoleucine (32/21)	**0.008**	0.069	0.02	0.015	0.016
L-Lysine (31/20)	**0.014**	0.084	0.019	0.021	0.016
Carsonella Ruddii (114/126)					
L-Leucine (86/56)	**0.005**	0.106	0.046	0.035	0.047
L-Isoleucine (83/49)	0.055	0.105	**0.032**	0.036	0.040
L-Valine (83/49)	0.037	0.091	0.030	**0.028**	0.035
B. cicadellinicola (236/229)					
Octapremyl diphos. (149/160)	0.726	0.283	0.209	0.221	**0.195**
Tetrahydrofolate (148/149)	0.337	0.227	**0.170**	0.237	0.179
Heme-O (150 / 161)	1.164	0.319	0.208	0.217	**0.172**
B. aphidicola (396/338)					
Pyruvate (219/87)	**0.082**	0.131	0.105	0.105	0.104
dGTP (206/76)	**0.099**	0.138	0.126	0.118	0.101
UTP (219/87)	0.113	0.117	**0.099**	0.148	0.104
Yeast (703/1010)					
FADH2 (444/314)	*	14.39	**5.55**	7.27	14.55
L-Histidine (415/269)	*	5.55	**4.80**	5.02	6.62
L-Aspartate (410/ 274)	176.40	**4.53**	4.65	4.82	4.66
Human (997/1225)					
L-Alanine (710/359)	5,058.27	5.15	**3.34**	10.76	10.78
Seriapterine (698/329)	*	3.19	2.96	6.85	**2.88**
L-Cysteina (150/161)	5,579.85	3.32	3.32	4.22	**3.17**
E. coli (1010/1164)					
L-Aspartate (714/507)	*	2,139.01	**3.32**	10.57	47.72
L-Metionine (737/545)	*	632.20	**13.62**	14.08	14.17
Glycine (706/503)	*	553.21	11.55	**11.01**	13.90

In the cases marked '*', the algorithm did not finish within 24 hours.

For each target, the number of metabolites and reactions after void cycle deletion is indicated.

Concerning the minimality check, we may observe that it is not necessarily true that it improves the running time. In some cases, doing the check may even lead to worse results (ex: *Yeast*, target FADH2, Papa Pitufo method), while in others it may have a strong positive impact on the execution time of the algorithm which becomes 700 times faster (ex: *E. coli*, target L-aspartate, Pitufina method).

Notice also that as the size and complexity of the networks increase, the number of different minimal precursor sets found increases also, and it does this at a rate faster than the increase of the time needed to compute them.

Computing solutions for several preprocessed networks

As mentioned before, the network free of void cycles that is given by the heuristic proposed depends on the order in which reactions are added to R. Thus, different orders can generate different minimal precursor sets. To recover as many solutions as possible, we can repeat the search for precursors on several different results of the preprocessing part. In order to analyse the effect of this heuristic on the algorithms, we successively repeated this random process while computing, at each repetition, the number of new precursor sets obtained. The process stops when no new precursor set is recovered in 10 consecutive repetitions. Analysing the results for three different targets of *E. coli*, this convergence criterium was reached in less than 100 iterations. In the three cases, more than 50% of the solutions were recovered in the first 6 repetitions and more than 80% in the first 50 iterations (see Table 4.2).

Table 4.2: Computation of minimal precursor sets of the *E. coli* network for three targets, using several different preprocessed networks .

	Convergency		Prec.	Iteration reaching X% of Prec. sets			
Target	Iteration	Time (sec)	Sets	25%	50%	80%	95%
L-Aspartate	71	3128	267	1	6	49	57
L-Metionine	94	2738	399	1	5	46	80
Glycine	73	2693	242	1	4	19	56

Each iteration corresponds to repeating the heuristical random preprocessing and computing the minimal precursor sets using Pitufina with minimality. For each target, we show the iteration where the convergence is reached, the time required, the total number of different minimal precursor sets, and the iterations in which a given percentage of the total number of solutions are recovered.

4.6 Biological application

There are illustrations on how to apply our algorithms to enumerate precursor sets both in Cottret et al. (2008) and Cottret et al. (2010a). We will here focus on briefly covering the results presented in the latter publication as we consider it is a more appealing example and is more illustrative of the potential the methods have.

Endocytobiont bacteria from different species are bacteria that live permanently inside the cells of a pluricellular organism and often bring an adaptative advantage to their host by synthetising compounds that cannot be produced or are not found in the diet of the host (Charles and Nardon (1999); Bourtzis and Miller (2008)). In such cases, many metabolic functions of the bacterium are provided by the host and, inversely, the metabolism of the endocytobiont appears specialised into functions that are absent in the metabolism of the

host. The presence of one or more bacteria inside the cells of a host leads to the idea that metabolic exchanges happen in such a system both between the host and the bacteria, and between the different bacteria.

In order to better understand the metabolites that are good candidates for such exchanges, we chose a symbiotic system for which metabolic network reconstructions of the endocytobionts were available, namely *Sulcia muelleri* and *Baumannia cicadellinicola*, that live inside specific cells of the sharpshooter *Homalodisca coagulata*. Both metabolic networks were obtained from the MaGe database (Vallenet et al. (2006)). The downloaded versions of these networks were initial drafts and a manual curation was performed in order to improve the quality of the analysis. In this process, disconnected reactions or reactions connected to the network only through co-factors were removed, metabolic pathways predicted to be present in the metabolism of the organisms led to the inclusion of missing reactions that filled the gaps for the complete pathway to be present, and the reversibility of some reactions was corrected, among other modifications that are explained in detail in Cottret et al. (2010a). Once the metabolic network reconstruction of the two symbionts was complete, we applied some filters since our goal was to study the metabolism involving transfers of carbon atoms and to concentrate only on the small molecules. First, the metabolites that do not exchange carbon atoms, called *side compounds*, were removed. Inorganic compounds such as water, proton, ammonia and oxygen were also eliminated. Finally, reactions that do not imply a transfer of carbon atoms were filtered out. A complete description of the filtered reactions and metabolites, as well as the final datasets obtained are described in Cottret et al. (2010a).

The complete genome annotation of the two endocytobionts revealed that their metabolic capacities are broadly complementary (Wu et al. (2006); McCutcheon and Moran (2007)). The metabolism of *B. cicadellinicola* is globally devoted to cofactor and vitamin biosynthesis whereas the metabolism of *S. muelleri* is specialised in the biosynthesis of essential amino acids that the sharpshooter cannot produce nor find in its diet. Nevertheless, the partition of these metabolic roles is not so perfect: *B. cicadellinicola* produces two essential amino acids, methionine and histidine, that *S. muelleri* cannot produce while the latter appears to be able to synthesise menaquinone, a vitamin. Moreover, the complementarity between the two metabolisms also concerns the biosynthesis of some metabolites not needed by the insect host, such as the fatty acid biosynthesis pathway, supplied by *B. cicadellinicola*, except for one step which is provided by *S. muelleri* (McCutcheon and Moran (2007)). However, these previous analyses were essentially manually performed by comparing the lists of annotated metabolic genes. Even if the size of the networks is not so big, these metabolic networks are complex enough to discourage manual approaches.

Figure 4.9 presents a schema we used to study the possible metabolic exchanges between the host and its two endocytobionts. Basically, we automatically determined the set of metabolites potentially exogenously acquired (potential precursor sets or *seeds*) for both metabolic networks using a topological approach for identifying strongly connected components that have no incoming arcs of the graph representation of the metabolic network (Borenstein et al. (2008)). Then we selected as target sets essential compounds that must be produced for each of the bacteria and computed their minimal precursor sets. By analysing the intersection between the seed sets and minimal precursor sets, it is possible to infer possible metabolic exchanges between the two endocytobionts. Considering additional information on the host diet allows us to eliminate some of the candidates since some precursor sets were more probably supplied directly by the host.

The first step of the experiment is the automatic identification of the seeds. After this, we may observe that both bacteria work on a very reduced number of seeds, which indicates that

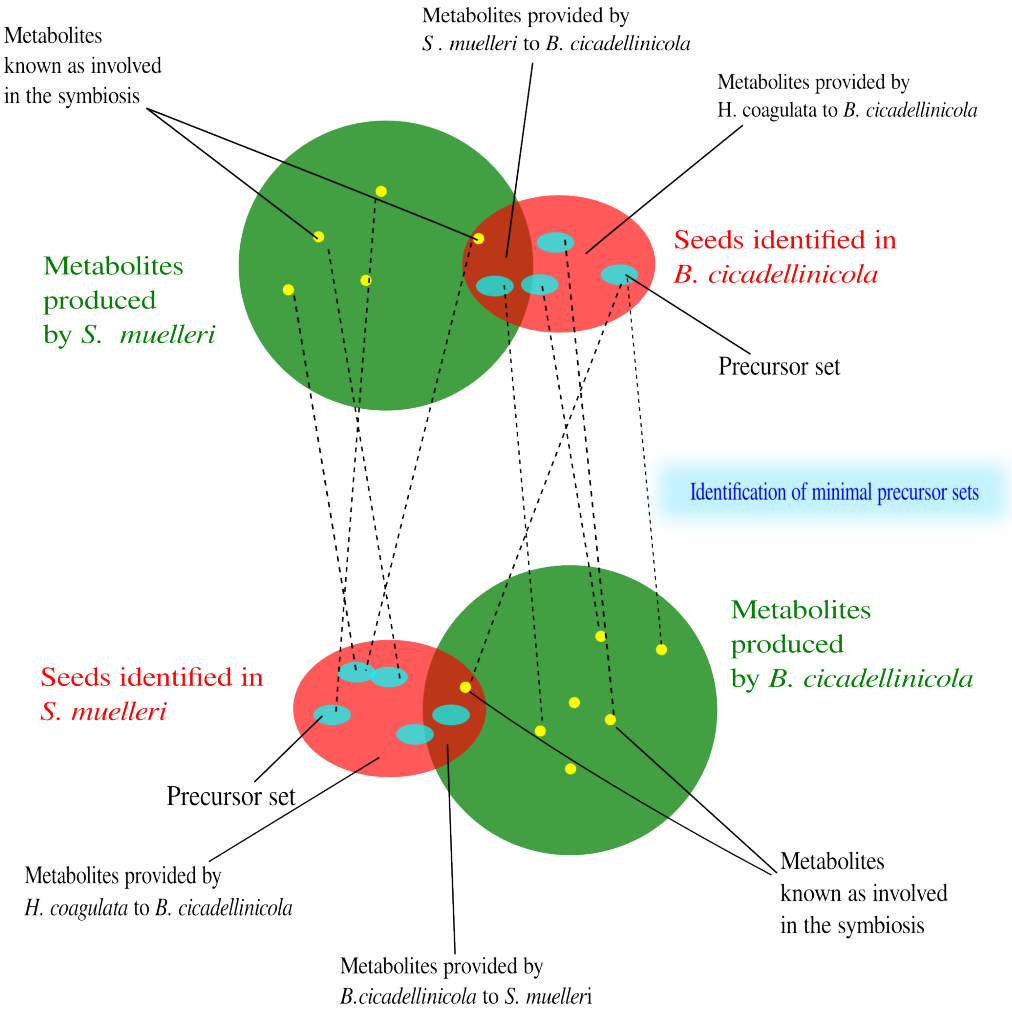


Figure 4.9: **Identification of the metabolic exchanges in the symbiotic system involving the two endocytobionts *B. cicadellinicola*, *S. muelleri* and their insect host, the sharpshooter *Homalodisca coagulata*.** Seeds are identified in the metabolic network of each bacterium (red sets). By comparing with the metabolites produced in the metabolic network of the other bacterium (green sets), we determined which metabolites are potentially provided by the cosymbiont or by the insect host. From the identified seeds, we determined which sets of seeds (precursor sets, blue sets) lead to the synthesis of metabolites important in the symbiotic association (yellow points). Obs: Figure extracted from Cottret et al. (2010a).

their metabolism has been highly reduced, as is usually the case with endocytobionts (Brinza et al. (2009)). By comparing the sets of seeds of both networks, we may see that the small intersection (only three metabolites) indicates a high complementary of the two metabolisms. The analysis also indicates that the carbon metabolism of *S. muelleri* may be completely independent on the metabolic network of *B. cicadellinicola*. On the contrary, the carbon metabolism of the latter appears dependent on the metabolism of *S. muelleri*, at least for two essential amino acids, threonine and lysine. Figure 4.10 displays the set of seeds identified in the metabolic network for each bacterium. Coloured arrows mark those produced in the metabolic network of the co-endocytobiont and those potentially provided by the insect host according to the literature. Seeds that correspond to annotated transport reactions are also tagged.

The next step in the analysis was to identify which subsets of the seeds (potential precursor sets) are sufficient to produce the metabolites chosen as target. We first put as targets

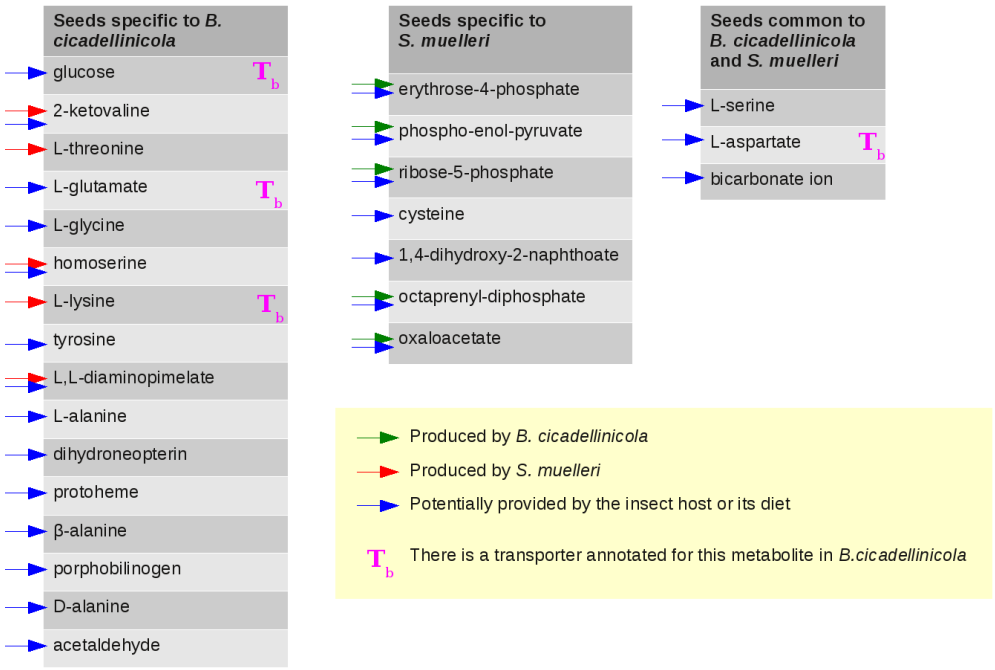


Figure 4.10: **Seeds identified in the metabolic graph of *B. cicadellinicola* and *S. muelleri*. Obs: Figure extracted from Cottret et al. (2010a).**

the metabolites reported as involved in the symbiotic association by McCutcheon *et al.* (McCutcheon and Moran (2007)). We then added erythrose-4-phosphate, phosphoenolpyruvate, oxaloacetate and ribose-5-phosphate to the list of target compounds for *B. cicadellinicola* because of their presence both in its metabolic network and in the precursor sets identified for *S. muelleri*. These additional targets are particularly interesting since they could directly correspond to metabolic pathways shared between the two metabolic networks. For the same reasons, we added homoserine and 2-ketovaline to the list of target compounds for *S. muelleri*. To compute the minimal precursor sets, we used PITUFO, which was the only of our algorithms available at the time the study was done and that works well for the size of the studied networks. The complete list of targets and the minimal precursor sets identified to produce them are presented in Figures 4.11 and 4.12 for *S. muelleri* and *B. cicadellinicola*, respectively.

For *S. muelleri*, apart from menaquinone, all targets are amino acids, which explains the uniformity of the results.Two seeds are present in all minimal precursor sets computed for these amino acids (except homoserine): erythrose-4-phosphate and phosphoenolpyruvate. Both are potentially provided by *B. cicadellinicola*. We found oxaloacetate and aspartate as alternative precursors for the synthesis of isoleucine and lysine. Indeed, each one can produce the other by the same reversible reaction in the metabolic network of *S. muelleri*. On the one hand, aspartate is one of the primary components of the food source of the host and it is thus reasonable to think that this compound should be provided by the host. On the other hand, aspartate is involved in other reactions that in particular participate in the synthesis of other amino acids in the metabolic network of *S. muelleri*. Since *S. muelleri* is able to produce aspartate from oxaloacetate, and since the former is not used in other reactions, the import of oxaloacetate by *S. muelleri* seems to be more realistic than the import of aspartate. Moreover, *B. cicadellinicola* could provide oxaloacetate while the bacterium is able to synthesise it from

aspartate (see Figure 4.12).

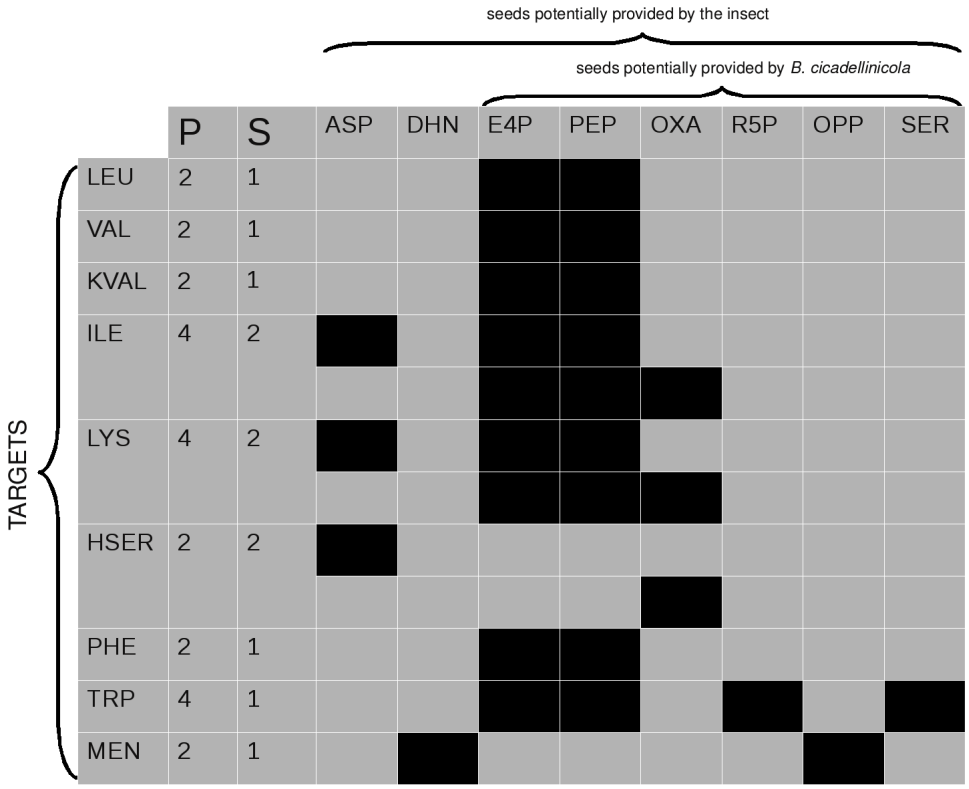


Figure 4.11: **Precursor sets of important metabolites in the metabolic network of *S. muelleri*.** Rows correspond to target metabolites and columns to seeds. Column P indicates the total number of precursors and column S the total number of solutions for the corresponding target. A black square means that a seed is present in a solution. **LEU**: L-leucine; **ILE**: L-isoleucine; **HSER**: homoserine; **KVAL**: 2-ketovaline; **LYS**:L-lysine; **PHE**:L-phenylalanine; **TRP**:L-tryptophane; **VAL**:L-valine; **MEN**:menaquinone; **ASP**:L-aspartate; **E4P**:erythrose-4-phosphate; **PEP**:phosphoenolpyruvate; **OXA**:oxaloacetate; **R5P**:ribose-5-phosphate; **OPP**:octaprenyl-diphosphate; **SER**: serine; **DHN**:1,4-dihydroxy-2-naphthoate. Obs: Figure extracted from Cottret et al. (2010a).

Some seeds appear as obligatory in the synthesis of several targets in *B. cicadellinicola*. Glucose and aspartate, reported as provided by the insect cell, thus appear as obligatory for the synthesis of, respectively, twelve and five target compounds. Serine, glycine and threonine have been detected as alternative seeds by the Borenstein method (Borenstein et al. (2008)) and they appear in the minimal precursor sets for methionine, coenzyme A, glutathione and thiamine. McCutcheon *et al.* suggested that homoserine and 2-ketovaline, potentially provided by *S. muelleri*, could be precursors of metabolites supplied by *B. cicadellinicola*. Homoserine was reported as a precursor of methionine and 2-ketovaline as a precursor of coenzyme A (McCutcheon and Moran (2007)). Our results confirm these hypotheses. For methionine, our method adds precision by indicating also the alternative precursors serine-glycine-threonine. For coenzyme A, our method further suggests this triplet and also β-alanine as obligatory precursors. Interestingly, we observed that only methionine and coenzyme-A require metabolites provided by *S. muelleri*. Moreover, the metabolites needed by the other targets could be all potentially acquired from the host cell by *B.cicadellinicola*.

The precursor sets identified in both bacteria could be made available by the insect host and our results suggest both *B. cicadellinicola* and *S. muelleri* may be completely independent

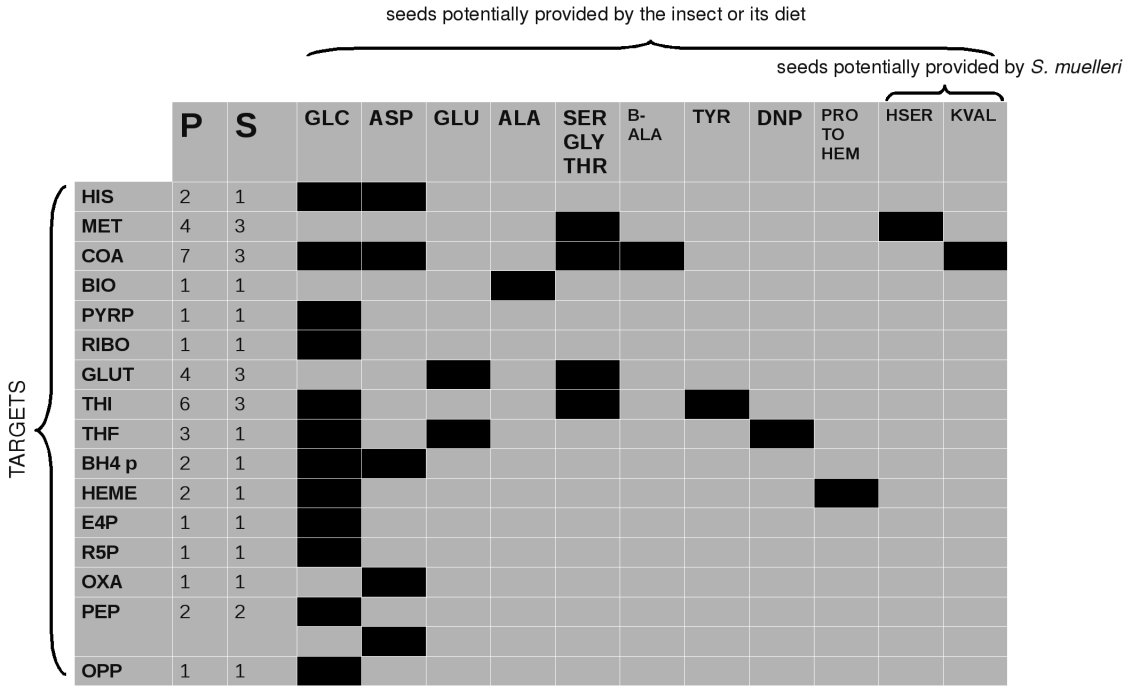


Figure 4.12: **Precursor sets of important metabolites in the metabolic network of *B. cicadellinicola*.** Rows correspond to target metabolites and columns to seeds. Column P indicates the total number of precursors and column S the total number of solutions for the corresponding target. A black square means that a seed is present in a solution. **HIS**:L-histidine; **MET**:L-methionine; **COA**:coenzyme A; **BIO**:biotin; **PYRP**:pyridoxal-5'-phosphate; **RIBO**:riboflavin; **GLUT**:glutathione; **THI**:thiamine; **THF**:tetrahydrofolate; **BH4 p**:BH4 precursor; **E4P**:erythrose-4-phosphate; **R5P**:ribose-5-phosphate; **OXA**:oxaloaceatate; **PEP**:phosphoenolpyruvate; **OPP**:octaprenyl-diphosphate; **GLC**:glucose; **ASP**:L-aspartate; **ALA**:L-alpha-alanine; **SER**:serine; **GLY**:glycine; **THR**:threonine; **HSER**:homoserine; **KVAL**:2-ketovaline; **B-ALA**:Beta-alanine; **DNP**:dihydroneopterin. Obs: Figure extracted from Cottret et al. (2010a).

of the metabolites provided by the co-resident endocytobiont to produce the carbon backbone of the metabolites provided to the symbiotic system. However, notice that the two essential amino acids (threonine and lysine) identified as seeds for *B. cicadellinicola* are certainly not produced by the insect host nor present in its diet and must be provided by *S. muelleri*. There is thus a dependence of the carbon metabolism of *B. cicadellinicola* on the metabolism of *S. muelleri*, but threonine and lysine seem to be exploited by *B. cicadellinicola* to produce its proteins and not to provide metabolites to the symbiotic system.

4.7 Open problems and perspectives

There are two main open modelling problems concerning the enumeration of minimal precursor sets for a set of target metabolites. The first one is that the current formal definition does not deal explicitly with void cycles and, for this reason, the three proposed algorithms are subject to finding the empty set as a solution, which is not biologically meaningful. The second one is that the method explores the topology of the metabolic network but not quantitative aspects that could be integrated, namely the stoichiometric coefficients of the metabolites in the reactions. Indeed, the factories from a subset of the sources to the targets are expected to

correspond to a feasible state of the cell and, in this sense, an approach considering steady-state factories is desirable. Both directions are interesting and, ideally, they may be managed together to design a model and an algorithm that fit even better the biological needs.

Recently, Carbonell et al. (2012) proposed two methods for the enumeration of minimal metabolic pathways, with a metabolic network also modelled as a hypergraph and a minimal metabolic pathway as a minimal hyperpath according to the definition of Nielsen et al. (2005). Even if the biological application of finding all possible ways of producing target metabolites is almost the same, the mathematical problem and the models used differ considerably. Their definition of a minimal hyperpath is close to our definition of reaction factories, but they impose an ordering of the reactions in the definition of hyperpaths that makes it difficult to deal with cycles, which in fact is something they do not do. Instead of using the internal supply metabolites directly in the definition of their minimal hyperpaths, the authors propose to do a pre-processing step to identify the set of internal supply compounds, that they call *supplements*, and to add them to the set of sources. However, it is not clear how minimality of the solutions is kept after this addition. For instance, observe Figure 4.13, which is basically the same as Figure 4.1 with the addition of a new reaction, r_4, that produces c from a. Notice that z_1 and z_2 are internal supply metabolites and $\{a, b, z_1\}$ or $\{a, b, z_2\}$ are precursor sets of the target c. However, now they are not minimal since $\{a\}$ alone is able to produce c through reaction r_4, which is therefore a minimal factory. It is not clear how Carbonell et al. (2012) deal with this imaginary situation, since z_1 and z_2 are going to be identified as supplements by their pre-processing step starting from the source set $\{a, b\}$. The source set is then increased to be $\{a, b, z_1, z_2\}$ and the reaction r_3 that produces c through z_1 is now considered a minimal hyperpath producing the target.

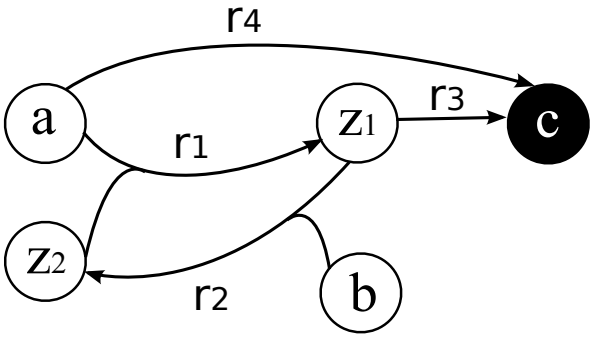


Figure 4.13: A metabolic network containing 4 reactions and 5 metabolites. Considering c as a target metabolite, the only minimal precursor set is $\{a\}$ and the only minimal factory is $\{r_4\}$.

Finding precursor sets or minimal factories adding the stoichiometry coefficients to the topological procedure is a nice and natural follow up of the work done. Figure 4.14 shows an example in which the same topology of a cyclic network may be feasible or not depending on the stoichiometric coefficients. In that same paper (Carbonell et al. (2012)), the authors introduce an approach to find minimal hyperpaths taking stoichiometry into account. The idea is to find minimal hyperpaths connecting the target to subsets of the sources that satisfy the steady-state assumption, *i.e.*, the internal metabolites in the hyperpath must be balanced. To address the problem, they include artificial input reactions to produce the sources and artificial output reactions to consume the targets, and compute the set of elementary modes for this modified network. For their definition of hyperpaths, the solutions will be a subset of the elementary modes, for instance they are interested in acyclic elementary modes. In practice, their results show that this approach is slower than the topological one for enumerating

minimal hyperpaths, and this is probably due to the fact that the number of elementary modes is much bigger than the number of acyclic hyperpaths they are interested in. Another related method is the one proposed in (Urbanczik and Wagner (2005)) to enumerate so-called elementary conversions, which correspond to the extreme rays of a cone of concentration distributions in stationary state, which can be seen as sets of inputs and outputs of the network that are co-related through one or more elementary mode connecting them. In fact, this seems to be a possible strategy, with some small adaptations, to enumerate minimal precursor sets taking stoichiometry into account, but most probably a method like ours, that also considers the topology of the network, could provide much better running times.

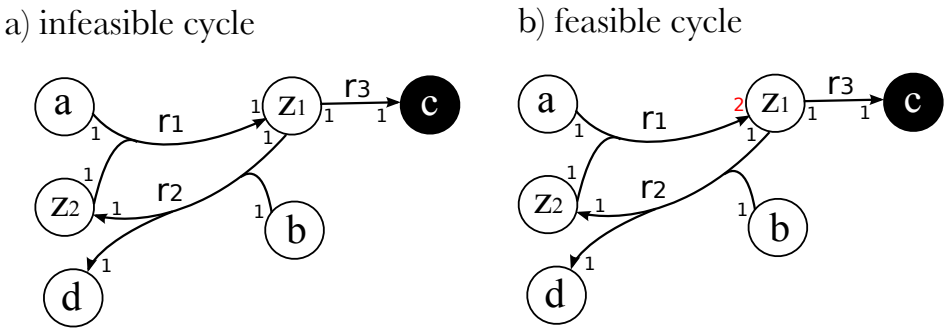


Figure 4.14: a) The cycle involving reactions r_1 and r_2 is infeasible as a precursor set of c, since the regeneration of z_1 is enough to keep the cycle running but not for the production of c. b) Considering that the stoichiometric coefficient of the production of z_1 in r_2 is 2 the cycle is now able to keep running and to produce the target c at the same time.

For our enumeration problem, it is also true that elementary modes correspond to factories, however it is not clear how to explore a similar approach in order to efficiently solve the problem of enumerating minimal precursor sets since different factories may correspond to exactly the same precursor set. Nevertheless, we started addressing the problems and we had some preliminary ideas that however need to be further investigated. Our current approach to the problem of adding stoichiometry to the model was simply to proceed with our topological approach working directly on the network (Pitufina), and every time a stop condition of the recursive algorithm is reached (because we reach a cycle or a precursor), to validate the branch from the identified precursor set to the target. A first idea to validate it, was to explore the fact that unbalanced fluxes would be produced only in the case a cycle is present, since precursors are assumed to be in infinite supply. We could therefore simply verify if the amount needed of internal supply compounds is enough to keep the cycle running, with some leftover for the production of the target. This corresponds simply to checking if the production coefficient is bigger than the consumption one. Unfortunately, this idea does not work as exemplified in Figure 4.15, in which two branches of the traversal should be combined to identify that the cycle is viable. Indeed, this simple validation fails, in this example, because the consumption of a in both branches is 1 and its production is also 1, suggesting that the cycle may not run while using a to produce t at the same time. Notice, however, that both branches use reaction r_2 to produce compound b and that, in this case, combining the two branches would consume only one molecule of a to produce 2 molecules of b, allowing in the end the production of 2 molecules of a instead of 1, which is enough to say that the cycle may run and produce t. This example illustrates a major difference between the topological and the stoichiometric approaches, since in the first case only one reaction producing any compound in the hyperpath was chosen while now that we need to satisfy production/consumption constraints, we may have to combine more than one reaction producing the same metabolite, as is the case for

compound a in the example. Currently, we are adapting a different validation step that seems promising. The idea is to validate through a linear-programming formulation whether the cycle on the identified branch from the precursor sets to the target is feasible. For this goal we may impose a positive flux on the reactions consuming the identified precursors or cyclic compounds and also a positive flux on the reaction producing the target. The validation is then performed by checking if a feasible flux distribution is possible assuming the system to be in steady-state or growing-state, *i.e.*, with the internal accumulation of compounds allowed. For instance, in the example of Figure 4.15, a positive flux over r_1 and r_2 is imposed once the recursion is finished and a valid flux distribution may be found, which validates the precursor set $\{p\}$ as a stoichiometric valid precursor set of t with the use of $\{a\}$ as internal supply.

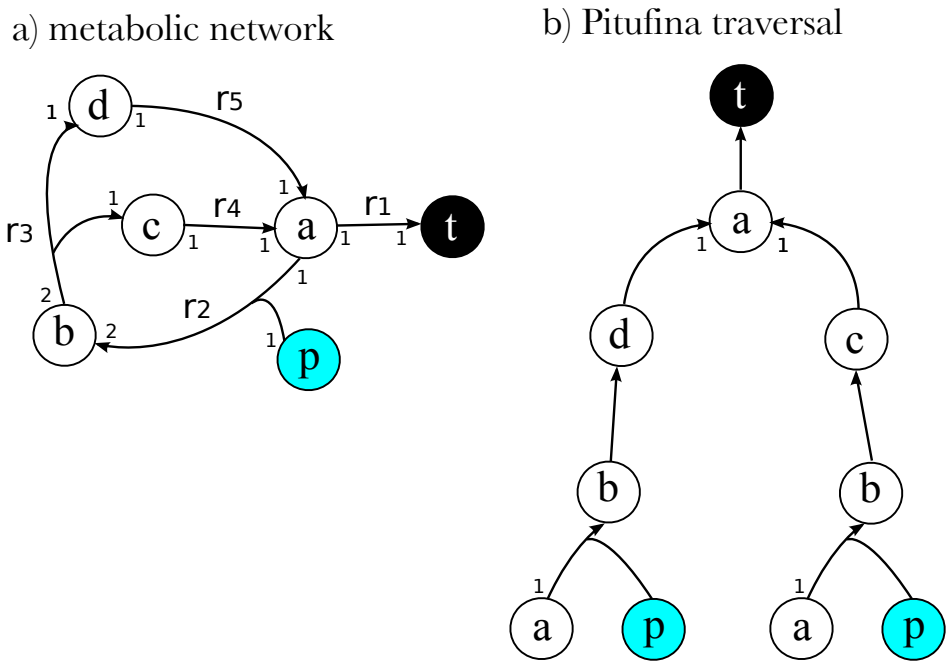


Figure 4.15: a) A simple network with 5 reactions and 6 compounds. Compound t is a target and compound p is a potential precursor. b) A traversal of the network by Pitufina corresponds to the shown replacement tree rooted in t.

Chapter 5

Chemical Organisations

Contents

5.1 Introduction

The results presented in this chapter are strongly based on our paper Milreu et al. (2010). Until recently, metabolism was analysed via the pathways composing it, which were traditionally established in a non automatic way by experts interested in some specific function (glycolysis for instance, or anaerobic respiration). The pathways were studied independently from each other, even though molecules could be shared. The advent of full genome sequences now enables to infer genome-scale metabolic networks, as briefly explained in Section 1.2. The study of these networks has revealed extensive cross-talk between traditionally defined pathways, as well as the use by different organisms of alternative pathways, that is, different metabolic routes to a same final overall product goal. While the notion of a metabolic pathway remains useful as a reference definition of functional modules, the exact frontiers between pathways can now be questioned, and other definitions of the concept of a metabolic module can be proposed. While several formal definitions have already been suggested, none of them is able to capture all the expert knowledge that led to the first pathways while at the same time providing an insight into alternative functional ones. The context of this work is therefore the same as the one in Chapter 3. The motivation of this chapter, however, is to explore another model for metabolic modules called chemical organisations, both in terms of the complexity of enumerating such modules and of exact algorithms for performing the enumeration. The results obtained constitute a solid algorithmic ground for the study of chemical organisations, a necessary first step to widen the use of this notion for the computational analysis of metabolic networks.

Several formal definitions of pathways and modules can be found in the literature on metabolism, the best known of which may be elementary modes (Schuster and Hilgetag

(1994)) or any of its close cousins (see Lacroix et al. (2008); Papin et al. (2004) for surveys). Elementary modes may be informally described as metabolic subnetworks that can function at steady state, meaning that all internal metabolites are produced and consumed in equal rates (that is, nothing accumulates internally). This is a fine definition, but has at least one drawback: it is restricted to the analysis of the system at steady state and does not allow to describe states of the system where metabolites can accumulate. However, such states are relevant as they could correspond to intermediary steps in the evolution of metabolism, or temporary states in the dynamics of metabolism.

A relatively recent model called *chemical organisations* was introduced in 2005 by Peter Dittrich and his group (Dittrich and di Fenizio (2007)) and is able to deal with the growing states of a network. The concept was introduced, building on earlier work by Fontana and Buss (Fontana and Buss (1994)) and can be used not only for metabolism, but also for any kind of reaction system, including regulatory networks.

Chemical organisations are sets of molecules that are self-maintaining and closed (we use the terms metabolite and molecule with no distinction). Informally, a self-maintaining set is a set where molecules can accumulate – the feature we were seeking – provided no molecule vanishes, *i.e.*, its abundance goes to zero. A set is closed if all metabolites produced from reactions for which all the inputs are present in the set will also be present and thus part of the set. By convention, this includes all reactions that take their input from the environment, *i.e.* are external. All external inputs are therefore considered as being available *and used*. This introduces a second contrast with elementary modes (EMs). Indeed, EMs may use only part of the externally available inputs. More generally, EMs are not closed. Intuitively, the concept of a chemical organisation may be illustrated by means of the following experiment. Suppose one put a set of metabolites into a vessel. If these set of metabolites is a chemical organisation, at any time one look again to the vessel, none of the original metabolites put into the vessel vanished (self-maintaining) and no new metabolite appeared (closed). These two properties do not guarantee that no change on the metabolites will happen, even because accumulation is allowed, but the lack of one of them is enough to say that changes on the original set will occur, at least if one waits long enough. The usefulness of such a concept for biological problems has been illustrated in studies of sugar metabolism in *Escherichia coli* (Centler et al. (2007)) and for the curation of metabolic network reconstructions (Kaleta et al. (2009)). In addition, the relation with elementary modes was established (Kaleta et al. (2006)).

Finally, as we have already said, the theory of chemical organisations has been proposed for general reaction systems. Its application to metabolic networks raises new specific questions, as the networks have specific properties. They are indeed expected to be flux-consistent (each reaction belongs to at least one elementary mode) and also mass-consistent (the transformations imposed by the set of reactions should respect conservation of mass).

In this chapter, we will first revisit chemical organisations in the context of mass- and flux-consistent networks. In particular, finding a chemical organisation was shown to be hard in Centler et al. (2008). A legitimate and non-trivial question is whether this remains true in biologically more realistic mass- and flux-consistent networks. Section 5.2 presents the main definitions of chemical organisations and consistency of networks. Section 5.3 shows that even for consistent networks the enumeration problem is hard. We go however further by identifying the specific structural properties of the network that account for this hardness. Those are discussed in Section 5.4, while Section 5.5 describes a new algorithm that takes advantage of such properties to obtain an exact method that is in all cases theoretically more efficient for consistent networks than the enumeration algorithms presented in Centler et al.

(2008) because, at best, a smaller part of the solution space needs to be explored.

5.2 Definitions

A metabolic network may be modelled as a *weighted directed hypergraph* $G = (M, R)$ with M the set of *vertices* corresponding to the metabolites and R the set of *hyperarcs* corresponding to the reactions. A directed hyperarc (*i.e.* a reaction) $r \in R$ is an ordered pair of sets of vertices (*i.e.* metabolites) $r = (subs(r), prod(r))$ where $subs(r)$ is the set of substrates of r and $prod(r)$ is the set of products of r. For each x in $subs(r)$ (in $prod(r)$) the weight of x with respect to r denotes the stoichiometric coefficient of x in r, that is, the number of units of x consumed (or produced) when r fires. Notice that x can belong to both $subs(r)$ and $prod(r)$; in this case there are two weights associated to x w.r.t. r. Observe also that, according to the above definitions, the set of substrates of a reaction r can be empty: in this case, we say that the metabolites in $prod(r)$ are *inputs* of the network.

Metabolic networks have also been often modelled using matrices (e.g., Schuster et al. (1999)). The *stoichiometric matrix* S has $|M|$ rows and $|R|$ columns where $S_{i,j}$ is the stoichiometric coefficient of molecule i in reaction j. $S_{i,j}$ is negative if i is consumed and is positive if i is produced. We notice here that while the stoichiometric matrix can always be derived from the weighted hypergraph, the reverse is not true. Indeed, metabolites involved as substrates and products of the same reaction cannot be handled in the matrix representation.

For some of the results presented, we also use the concept of the *underlying graph* of G, which is a directed graph with the same set of vertices of G and arcs $x \to y$ for every pair of vertices x, y for which there is an hyperarc r such that $x \in subs(r)$ and $y \in prod(r)$. A reaction is said to be in a path/cycle of the underlying directed graph if any of its (substrate,product)-pairs is an arc of the path/cycle. Notice that the defined underlying graph corresponds exactly to the compound graph introduced in Chapter 2.

In the context of metabolic networks, we say that a *flux* over the network is the rate at which each reaction occurs. A flux can be represented as a *flux vector* $v \in \mathbb{R}^{|R|}$ with $v[i]$ denoting the rate of reaction i. We also define a *mass vector* $m \in \mathbb{R}^{|M|}$ with $m[j]$ denoting the mass of metabolite j.

A metabolic network is **flux-consistent** if there exists a flux vector $v > 0$, *i.e.* $\forall i \in R$ the flux $v[i] > 0$ is such that $Sv = 0$ (Acuña et al. (2009)). This is the same as saying that every reaction of the network belongs to at least one elementary mode, thus checking for the usefulness of each reaction. For more information on elementary modes, we refer to Schuster and Hilgetag (1994) and Schuster et al. (1999).

A metabolic network is **mass-consistent** if there exists a mass vector $m > 0$, *i.e.* $\forall i \in M$, $m[i] > 0$, such that $m^T S = 0$, where m^T denotes the transposed vector. This is the same as saying that there exists some mass distribution for all metabolites such that the whole set of reactions is mass balanced.

An example of a network with two reactions $r_1 : a + c \to b$ and $r_2 : b + d \to d + c$ that is not mass-consistent is shown in Figure 5.3. Since reaction r_2 requires that b and c have the same mass, then the mass of a in r_1 has to be 0, which is inconsistent. If we replace r_1 by $r_1' : a + c \to 2b$, then the system becomes mass-consistent (indeed, every positive mass vector with equal components is consistent).

We denote by $\mathcal{R}_A \subseteq R$ the subset of reactions that can be fired when the metabolites in the set $A \subseteq M$ are present, *i.e.*, $\mathcal{R}_A = \{r \in R | subs(r) \subseteq A\}$.

A set $C \subseteq M$ is said to be **closed** (Dittrich and di Fenizio (2007)) if, for all reactions $r \in \mathcal{R}_C$, $prod(r) \subseteq C$. Moreover, given a set $C \subseteq M$, the **closure** of C, denoted by Cl_C, is

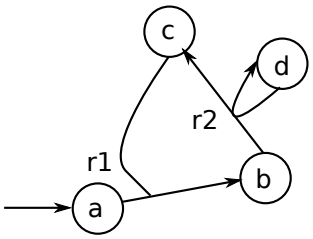


Figure 5.1: Metabolic network whose mass-consistence depends on the stoichiometric coefficients of the first reaction.

the smallest closed set H that contains C. Notice that if C is a closed set of molecules, then C must contain all inputs of the network (since the empty set is a subset of C, the inputs of the network must belong to $\mathcal{R}_C$). In particular, the closure of the empty set will contain all inputs and whatever can be produced from them.

A set $C \subseteq M$ is said to be **self-maintaining** (Dittrich and di Fenizio (2007)) if there is a flux vector v such that:

1. for all reactions $r \in \mathcal{R}_C$, $v[r] > 0$;
2. for all reactions $r \notin \mathcal{R}_C$, $v[r] = 0$;
3. for all molecules $i \in C$, the production rate $(Sv)[i] \geq 0$.

A set of molecules is self-maintaining if there exists a flux vector such that the molecules present in the set can accumulate ($(Sv)[i] > 0$) or be consumed and produced at the same rate ($(Sv)[i] = 0$)(first and second conditions) but none of them may disappear (third condition). Conditions 1 and 2 basically specify that all reactions that can fire with molecules from the set will fire. In particular, reactions which produce molecules outside the set will also fire. A self-maintaining set is therefore really self-maintaining, even in the presence of "leaks".

Finally, a set of molecules $O \subseteq M$ is said to be an **organisation** (Dittrich and di Fenizio (2007)) if it is both closed and self-maintaining. O is said to be **reactive connected** if:

- (reactive) each metabolite in O takes part as substrate or product in at least one reaction inside $\mathcal{R}_O$;
- (connected) for any two molecules x and y in O, there is a path from x to y in the underlying undirected graph.

We present a network in Figure 5.2 that has 3 connected components and 8 organisations but only 2 of them are reactive connected organisations. The others are just combinations of organisations which cannot directly interact among them. Notice also that some sets are not organisations, such as for instance the sets $\{s, p\}$ and $\{b, c, d\}$, since they are not closed, or the sets $\{s, q, r\}$ and $\{c, d\}$ that are neither closed nor self-maintaining. On the other hand, the closure of the empty set is $\{s\}$ and it is an organisation. This organisation must be present in all organisations even if there is no possible interaction between their molecules as in the case of organisation $\{s, a\}$. Clearly, any set of disconnected nodes will form an organisation, as long as we consider its union with the closure of the empty set. In the following, we shall ignore such organisations and focus on organisations where the molecules interact among them. This is our motivation to find only reactive connected organisations.

For this reason, from now on we restrict our networks to have only one connected component and some input and output metabolites. For general networks, indeed, we can without

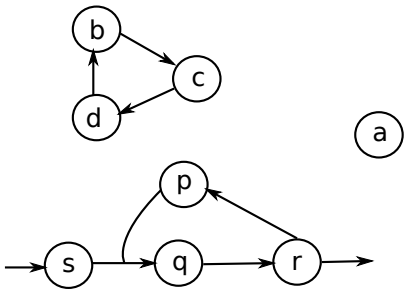


Figure 5.2: A metabolic network with (a) 3 connected components, (b) 8 organisations: $\{\{s\},\{s,p,q,r\},\{s,a\},\{s,b,c,d\},\{s,a,b,c,d\},\{s,a,p,q,r\},\{s,b,c,d,p,q,r\},\{s,a,b,c,\ d,p,q,r\}\}$, and (c) 2 reactive connected organisations: $\{\{s\},\{s,p,q,r\}\}$.

loss of generality work on each connected component separately and then combine the results. In order to compute closures of sets of metabolites, we recall the forward propagation algorithm (Romero and Karp (2001)) that was already presented in Chapter 4 on precursor sets. Informally, it consists in starting from a set C, adding $prod(r)$ for every $r \in \mathcal{R}_C$, and repeating this procedure until no new metabolites are added to C.

As already mentioned, all inputs of the network need to be considered together in order to compute organisations. This is a modelling choice that implies that if one wished to compute organisations for different subsets of the inputs, then it would be necessary to edit the network and recompute the organisations for the subsets of interest.

5.3 Chemical organisations in consistent networks

It was shown that deciding whether a network contains at least one organisation is NP-complete (Centler et al. (2008)). However the proof was based on a network that was not mass- and flux-consistent. We now characterise organisations in consistent networks. First of all, we observe that it is easy to check whether a set C is an organisation by inspecting the reaction rules to check closure and to check self-maintenance using linear programming.

The following theorem indicates how to compute two possible organisations.

Theorem 11. *If a network is flux-consistent then the whole network and the closure of the empty set are organisations.*

Proof. The whole network is always closed. By definition of flux-consistency, we have a flux vector v that covers the whole network, satisfying the condition of self-maintenance. Therefore the whole network is an organisation. Analogously, if the closure of the empty set produces the whole network then it is an organisation. Otherwise, since every metabolite is produced from the empty set, we can easily obtain a valid flux vector v satisfying the condition of self-maintenance. □

Notice that the closure of the empty set is the smallest possible organisation since it has to be contained in all other organisations. In the following, we say that the whole network and the closure of the empty set are **trivial organisations**. Observe also that the closure of the empty set may not always produce the whole network. An example is given in Figure 5.3 since the closure of the empty set for that network is $\{a\}$.

Theorem 12. *If the network is flux-consistent and acyclic, i.e. the underlying directed graph of the hypergraph is acyclic, then the whole network is the only organisation.*

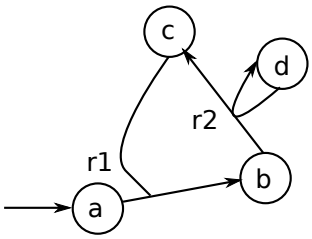


Figure 5.3: Network in which the closure of the empty set does not produce the whole network.

Proof. The smallest organisation is given by the closure of the empty set, which can be obtained by applying the forward propagation algorithm to the empty set. As the network is flux-consistent and acyclic, from the inputs any metabolite can be reached, *i.e.*, produced. Hence, the closure of the empty set is the entire network. From the flux-consistency of the network and from Theorem 11, it follows that the smallest organisation is the whole network. □

Notice that both previous theorems hold considering that the network is flux-consistent only. The next result shows that the problem of finding a non-trivial organisation in a mass- and flux-consistent network is NP-hard. The proof is based on a reduction from the 3-SAT problem, which is an appropriate modification of the original reduction given in Centler et al. (2008), that showed that finding a reactive organisation in a general reaction system is NP-hard.

Theorem 13. *Deciding if a mass- and flux-consistent network contains a non-trivial organisation is NP-hard.*

Proof. We reduce our problem from 3-SAT. Given a boolean formula F in 3-CNF with n boolean variables and ℓ clauses, we construct a mass- and flux-consistent reaction network for which the existence of a non-trivial reactive (connected) organisation implies a positive answer to the 3-SAT and vice-versa.

In the reaction network we define the $2n$ literal metabolites $x_1, x_2, ..., x_n$, $\neg x_1$, $\neg x_2$, $\ldots, \neg x_n$ ($\neg x$ is the negation of x), clause metabolites $C_1, C_2, ..., C_\ell$ and additional metabolites $key, a, input$. The hyperarcs (reactions) are defined as:

- **Influx**: A reaction $\emptyset \to input$
- **Clause reactions**: For each clause $C_h = (A_1 \vee A_2 \vee A_3)$, we introduce 3 reactions: $key + A_j \to C_h + A_j,\ j = 1, 2, 3.$
- **Key reaction**: $C_1 + C_2 + ... + C_l + input \to (l + 1)\ key.$
- **Key outflux**: A reaction $key \to \emptyset$
- **Variable reactions of two types**: For each variable x_j, we introduce a type 1 reaction $x_j + \neg x_j \to a$, and a type 2 reaction $a + input \to x_j + \neg x_j + key$, $j = 1, \ldots, n$.

An illustration of this transformation is given in Figure 5.4. The transformation can obviously be done in polynomial time.

Let us first prove that the network is mass- and flux-consistent. To prove the latter, we construct a flux vector $v > 0$ such that $Sv = 0$ (Acuña et al. (2009)).

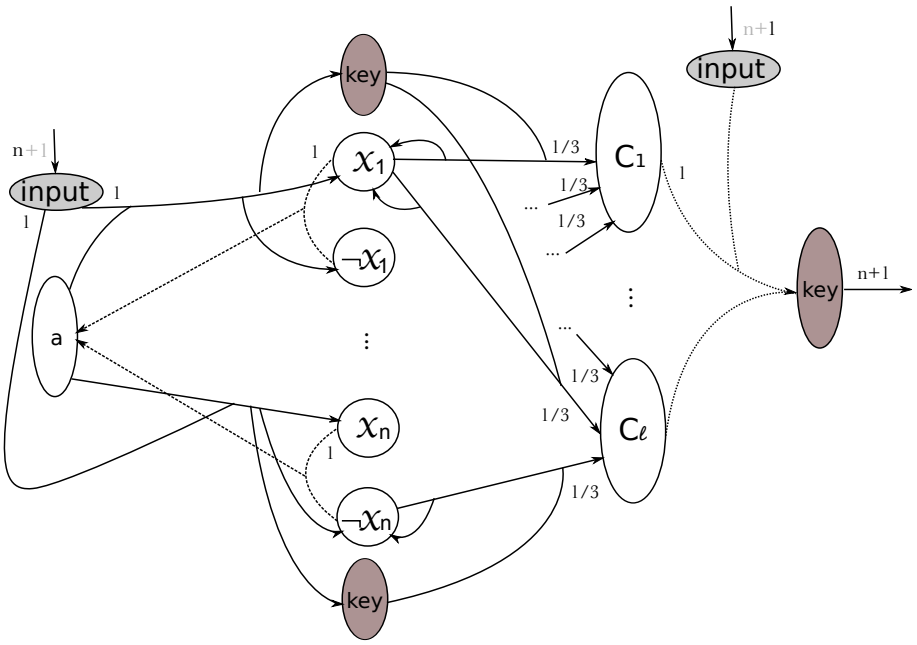


Figure 5.4: Finding a non-trivial reactive (connected) organisation is NP-hard. The *key* and *input* metabolites are presented more than once in the figure for convenience. Edge labels do not stand for stoichiometries but fluxes for a valid flux distribution that covers the whole network.

1. Setting the flux for each of the $2n$ variable reactions to 1, all literal metabolites and a are balanced, n of the metabolites *input* are consumed and n of *key* are produced.

2. Next we set the flux through each of the 3ℓ *clause reactions* to $1/3$, producing 1 of each clause metabolite, and consuming ℓ of *key*.

3. These clause metabolites are balanced by setting the flux of the *key reaction* to 1, while consuming 1 of *input* and producing $\ell + 1$ units of *key*.

4. The last step is to balance the *input* and *key* metabolites, by setting the flux of the *influx* and *outflux* reactions both to $n + 1$.

The balanced flux distribution v constructed has positive flux on every reaction of the network.

Mass-consistency of the network is easily verified by the mass vector that has mass 1 for all metabolites except for a, for which the mass is 2.

Suppose we have a satisfying truth assignment to F. Let K be the set of literals that are TRUE, then we claim that the set containing all metabolites corresponding to K, all clauses, *input* and *key* is a non-trivial organisation. Closedness is easily verified. We now show self-maintenance. Let k_i be the number of TRUE literals in C_i, $i = 1, \ldots, \ell$. For each TRUE literal in C_i we set a flux of $1/k_i$ on the corresponding *clause reaction*. Together with a flux of at least 1 in the *key reaction*, the *influx* and *outflux* secure self-maintenance.

Reversely, suppose we have a non-trivial reactive (connected) organisation O. First, notice that $\{input\}$ (the closure of $\emptyset$) is a trivial organisation and that it is part of every organisation, hence $input \in O$. As we now explain, this implies that if any organisation contains both x_i and $\neg x_i$ for any i, then it must contain the whole network, and hence be trivial. The presence of x_i and $\neg x_i$ triggers a *variable type 1 reaction*, producing a, which on its turn triggers all *variable type 2 reactions* producing all literal metabolites and *key*. All *clause reactions* are

therefore triggered and henceforth the *key reaction* and the *outflux reaction*. Hence, O does not contain both literals corresponding to the same variable, and none of the variable reactions will have positive flux.

Since O is reactive, *input* must be the substrate of some active reaction, hence the *key reaction* must be active, implying that $key \in O$. Because of self-maintenance, all clause metabolites must be produced. To produce a clause metabolite, at least one of its three *clause reactions* must be fired. Let K be the set of literal metabolites that are part of our reactive (connected) organisation. By closure, they are able to produce all clause metabolites, hence setting their value to TRUE yields a satisfying truth assignment for F. □

5.4 Enumerating chemical organisations

Theorem 13 immediately implies that it is not possible to enumerate all organisations in a mass- and flux-consistent network in polynomial-time-delay in the size of the network.

We now observe that Theorem 12 indicates that for flux-consistent networks, the difficulty of finding non-trivial organisations comes from the presence of cycles in the network. Indeed, as shown in Figure 5.5(b), cycles may interrupt the forward propagation procedure if there exists a reaction that can produce a new metabolite a but needs for this a metabolite b which is not available and therefore blocks the reaction.

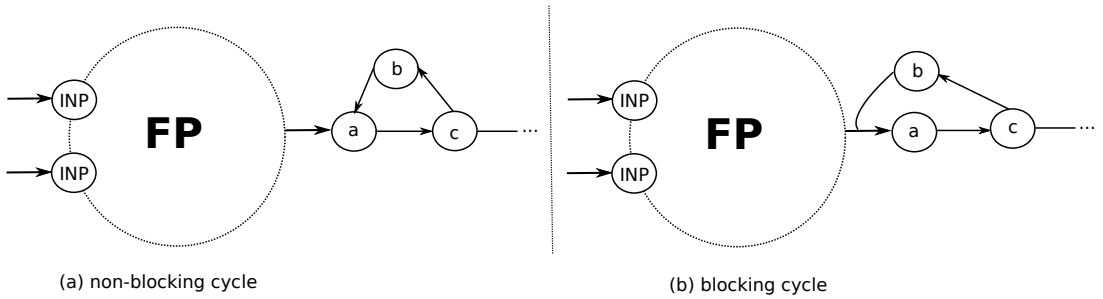


Figure 5.5: (a) Non blocking cycle that will be traversed by the forward propagation procedure. (b) Forward propagation blocked by a unreached metabolite b.

In order to find the reactive connected organisations, we need to process cycles every time the forward propagation procedure stops. This simple observation gives an upper bound of 2^k on the number of reactive connected organisations in flux-consistent networks, where k is the number of cycles in the network. In order to proceed, we formally define a **cycle** in the metabolic network as a simple *directed* cycle in the underlying graph. Self-loops are also considered as cycles. We also define a **hitting set** of a set of cycles as a set of metabolites such that each cycle contains at least one element of the hitting set.

Theorem 14. *Let H be a hitting set of all the cycles of a directed hypergraph. The set of all reactive connected organisations, denoted by $\mathcal{O}$, is such that*

$$\mathcal{O} \subseteq \bigcup_{C \subseteq H} \{Cl_C\}$$

Proof. It is sufficient to show that if A is a reactive connected organisation then $A = Cl_C$, where $C = A \cap H$.

First observe that, since A is closed and C is a subset of its metabolites, it follows that $Cl_C \subseteq A$.

Let us suppose that A contains vertices which are not in Cl_C. We colour these vertices white and the vertices of Cl_C black. Consider any white metabolite a_1. Since A is an organisation, a_1 cannot be vanishing. Moreover, it is not an input of the network as otherwise it would be black. Therefore, there exists a reaction r fired by A that has a_1 as product and has a white substrate a_2, $a_2 \neq a_1$, otherwise a_1 would again be black by closure.

By iterating the above reasoning, it follows that the subgraph of the underlying directed graph induced by white vertices has minimum in-degree at least 1 and contains a directed cycle. This contradicts the fact that H hits all the cycles. The set of white vertices is therefore empty and $A = Cl_C$. □

The bound of the previous theorem is tight. Indeed, an example where the number of organisations reaches $2^{|H|}$ is given in Figure 5.6 where from the input, k metabolites are produced by k independent reactions and all of them are blocked by cycles. Any combination of these independent paths can be de-blocked and produce a new organisation, and therefore we have 2^k organisations.

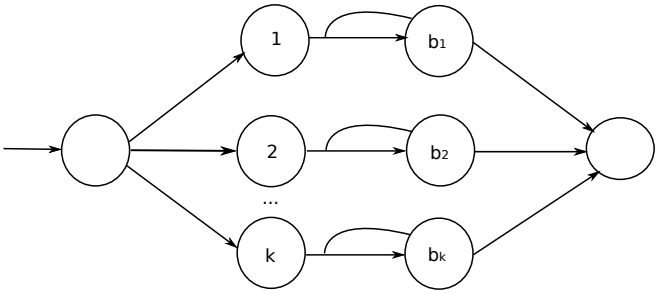


Figure 5.6: Example with k parallel blocking cycles and 2^k organisations.

However, some cycles never interrupt the forward propagation procedure as illustrated in Figure 5.5(a). Other cycles, on the other hand, exhibit structural properties that may lead to a blocking situation. Such cycles are called **potentially blocking cycles**. A basic solution to find all organisations is to know how to unblock all cycles of the network. However, for cycles which are non blocking it is a waste of time. Therefore, instead of finding a hitting set for all cycles of the network, it is enough to break all potentially blocking cycles to compute all reactive connected organisations. In order to prove this, we first introduce a more formal definition of potentially blocking cycle.

A **potentially blocking cycle** is a cycle such that there exists a reaction $r = (\{s_1, \ldots, s_h\}, \{p_1, \ldots, p_k\})$ in the network satisfying the following two conditions: (1) there exists i and j such that (s_i, p_j) is an edge of the cycle, and (2) there exists ℓ such that s_ℓ is not in the cycle.

A potentially blocking cycle may or may not interrupt the forward propagation depending on the metabolites that were produced by the procedure once the cycle is reached. Figure 5.7(a) shows an example in which the cycle will be traversed, while Figure 5.7(b) shows an example in which it will block the forward propagation algorithm.

Theorem 15. *Let H be a hitting set of all the potentially blocking cycles of a directed hypergraph. The set of all reactive connected organisations, denoted by $\mathcal{O}$, is such that*

$$\mathcal{O} \subseteq \bigcup_{C \subseteq H} \{Cl_C\}$$

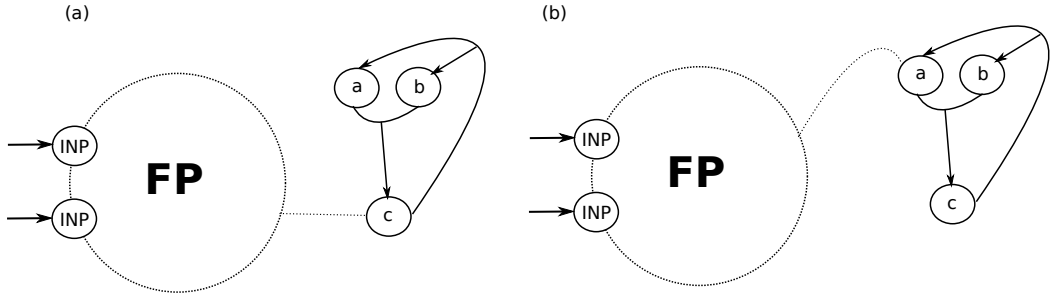


Figure 5.7: Example of a potentially blocking cycle formed by the vertices a and c and reactions $a + b \to c$ and $c \to a + b$ (note that vertices b and c also form a symmetric cycle). If the forward propagation procedure (FP) reaches the cycle through c, the cycle is traversed, but if it reaches it through a, it is blocked.

Proof. As in the proof of Theorem 14, it is sufficient to show that, given a reactive connected organisation A, $A = Cl_C$, where $C = A \cap H$ (in the following, all paths and cycles are meant directed in the underlying directed graph). Once again, it is easy to see that $Cl_C \subseteq A$. Let us then suppose that A contains vertices which are not in Cl_C and let us colour them white and those of Cl_C black.

Let a_i $(i = 1, 2, \ldots, k)$ be the set of white vertices such that there exists a reaction r_i having a_i as product and at least one black substrate. Then r_i has also at least one white substrate, which we denote by w_i, as otherwise a_i would be black by closure. Notice that a white vertex does not have to belong to the set of the a_i's, but that this set is not empty as the organisation is connected and the set of white vertices is not empty.

If $w_i = a_i$, the self-loop induced by r_i is a potentially blocking cycle that contains no vertex of H, leading to a contradiction. Thus we may assume that w_i is distinct from a_i.

For $1 \leq i \leq k$, define T_i as follows: a vertex w is in T_i if either $w = w_i$ or there exists a white path starting from w and ending in w_i. Up to a reordering, we may assume that $|T_1| \leq |T_i|$ for $2 \leq i \leq k$. As T_1 is not empty, it has to contain at least one vertex that is a product of a reaction having a black substrate. If that vertex is a_1, there exists a path from a_1 to w_1, which yields a white cycle with the edge (w_1, a_1). That cycle is a potentially blocking cycle (because of the reaction r_1) which contains no vertex in H, leading to a contradiction.

In other words, T_1 contains a vertex among $(a_2, \ldots, a_k)$, say a_2. This implies that every path ending in w_2 can be extended to a path ending in w_1 and thus $T_2 \subseteq T_1$. Therefore, by minimality of T_1, $T_1 = T_2$.

This implies that $w_1 \in T_2$: hence, there exists a white path from w_1 to w_2. As $a_2 \in T_1$, there also exists a white path from a_2 to w_1. Thus, we can construct a white path from a_2 to w_2. Considering the edge (w_2, a_2), we again obtain a white cycle, which is potentially blocking (because of the reaction r_2) and contains no vertex in H, leading to a contradiction.

Thus, the set of white vertices is empty and $A \subseteq Cl_C$. □

Even in the case of the previous theorem, the bound is tight. Indeed, an example where the number of organisations is $2^{|H|}$ is, once again, given in Figure 5.6, since all cycles presented in the example are potentially blocking.

5.5 Hitting set approach to enumerate organisations

Two exact algorithms were proposed in Centler et al. (2008) to enumerate organisations. The first one consisted in enumerating all closed sets and then checking for their

self-maintenance, while the second one consisted in enumerating all self-maintaining sets and then checking linear combinations of them in order to obtain closed sets. A third approach was also proposed that was based on the second one but avoided enumerating all self-maintaining sets. This algorithm however was a heuristic not guaranteed to find all self-maintaining sets (and thus organisations). Finally, a variation of the first algorithm was proposed in order to enumerate only reactive connected organisations. This algorithm comes closer to the one we describe later and works as follows. First, the forward propagation of the empty set is computed. Once the procedure is blocked, all possible combinations of metabolites that are connected to the produced set X are considered for addition in order to obtain further closed sets that include X. At this point, the algorithm recursively continues.

Observe that in the above procedures, no concept of blocking cycles has been formally identified and used. Now that we know that the hardness comes from such cycles, two different approaches can be applied in mass- and flux-consistent networks. In fact, this approach may be used in general, *i.e.*, even in networks that do not fulfill the consistency features, since its idea is a general approach for enumerating closed sets. The first idea is to find a **global hitting set** for all cycles of the network and then, following Theorem 14, to apply the forward propagation procedure on each subset of the hitting set to produce closed sets which together form all candidate organisations and, finally, to check through linear programming (LP) techniques if the candidates are self-maintaining. However, the problem of finding a minimum hitting set for all cycles of a directed graph is NP-hard as indeed it corresponds to the **feedback vertex set (FVS)** problem Karp (1972). Nevertheless approximation algorithms such as the one described in Seymour (1995) can be used in order to perform this step.

A second possibility is to find a **local hitting set**. According to Theorem 15, only potentially blocking cycles need to be considered. This is a superset of the blocking cycles that can be identified when the forward propagation procedure stops because it is at this moment that we know we are dealing with actually blocking cycles. A more efficient algorithm to enumerate reactive connected organisations is thus the following one: apply the forward propagation algorithm and once blocked, identify the set B of metabolites that are blocking the closure and find a hitting set that unblocks only the cycles which directly or indirectly involve these blocking metabolites.

In Even et al. (1998), the authors presented an approximation algorithm to a generalisation of the FVS problem, called SUBSET-FVS, in which only a subset of the directed cycles in the graph is considered interesting, more specifically the ones that intersect a set of special vertices. In our case, the set of special vertices would be the blocking metabolites locally identified as described in the previous paragraph. The authors in Even et al. (1998) gave two approximation algorithms for the SUBSET-FVS problem. The first algorithm achieves an approximation factor of $O\left(\log^2 |B|\right)$. The second achieves an approximation factor of $O\left(\min\{\log T \log\log T, \log n \log\log n\}\right)$, where T is the value of the optimum fractional solution of the problem at hand, and n is the number of vertices in the graph.

Before proving that this idea can be correctly used to exactly solve our problem, we need to define the concept of a blocking cycle in relation to a given set C of metabolites. Let C be a set of metabolites. A **C-blocking cycle** is a cycle of vertices which are not in C such that there exists a reaction r in the cycle whose set of substrates contains at least one metabolite in C. Hence, C-blocking cycles correspond to those which actually stop the forward propagation procedure.

Theorem 16. *Let C be a closed set and H a hitting set of the C-blocking cycles of a metabolic network. Let A be a reactive connected organisation whose metabolites contain C. Then either*

$C = A$ or there exists a non empty subset B of H such that the closure of $C \cup B$ is still a subset of A.

Proof. Let A be a reactive connected organisation containing C as a subset of its metabolites and let $B = A \cap H$. Let us suppose that $C \neq A$. To prove the theorem, it is then sufficient to prove that $B = A \cap H$ is not empty.

We colour the vertices of $A \setminus C$ in white and those of C in black. Since A is a reactive connected organisation, there exist edges between the white and the black metabolites, and some of them go from a black to a white vertex, as otherwise, white vertices would be vanishing. Let $(a_1, \ldots, a_k)$ be the set of white vertices reached by at least one edge coming from a black vertex.

The same argument of the proof of Theorem 15 can now be applied, showing that, as the set of white vertices is not empty, it contains a white C-blocking cycle. Therefore, B is not empty. □

Corollary 3. *Every reactive connected organisation is included in the set $\mathcal{CO}$ returned by the procedure given in Algorithm CCO.*

Proof. Let A be a reactive connected organisation. It has to contain C_0 as every organisation contains the closure of the empty set. Let C be maximum among the elements of $\mathcal{CO}$ which are subsets of A. Then Theorem 16 implies, by maximality of C, that $A = C$. □

Algorithm CCO(G)

Require: a metabolic network represented as a hypergraph $G = (M, R)$;
Ensure: the set $\mathcal{CO}$ of all candidates for being organisations.
  $\mathcal{CO} \leftarrow \{C_0\}$ where C_0 is the closure of the empty set ($Cl_{\{\}}$)
  for all elements C in $\mathcal{CO}$ which have not been treated before **do**
    Compute a hitting set H of the C-blocking cycles
    for every $B \subset H$ **do**
      Compute $Cl_{C \cup B}$ and add it to $\mathcal{CO}$ if it was not present already
  return $\mathcal{CO}$

Notice that the size of the hitting set computed by the algorithm is never greater than the number of blocking metabolites. Thus we can guarantee that our algorithm is theoretically better than existing algorithms which consider all blocking metabolites and then test all subsets. An illustration of the local hitting set approach is given in Figure 5.8. In this example, the forward propagation procedure produces the set C and is blocked by two metabolites, w_1 and w_2. The goal is to find a hitting set that intersects the blocking cycles for these metabolites. In this case, the hitting set has size 1 which is better than checking the combination of C with each of the blocking metabolites.

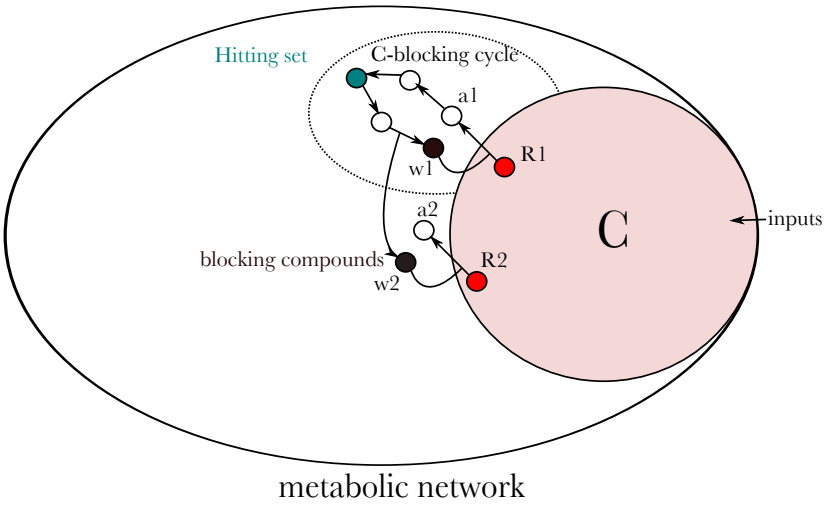


Figure 5.8: How a local hitting set approach for the blocking metabolites may improve the efficiency of an algorithm to enumerate closed sets. In the example, the forward propagation procedure started from the inputs and possibly other metabolites in the set C until it blocked. Two reactions r_1 and r_2 are in the border of C, i.e., need a metabolite outside C in order to proceed. The metabolites w_1 and w_2 that may unblock these reactions must be involved in some cycle. In this case, a single cycle unblocks both reactions and this may be done by adding to C a metabolite on the hitting set of the cycles involving w_1 and w_2.

5.6 Open problems and perspectives

The algorithms presented in this work for the enumeration of reactive connected organisations are based on the enumeration of closed sets as potential chemical organisations. A different approach is to focus first on the enumeration of self-maintaining sets and then check if these candidates are closed. As far as we know, such an approach has not been done for the cases of mass- and flux-consistent metabolic networks. Indeed, not even the algorithms here presented take any specific advantage of the fact that the network should be mass-consistent, and exploiting this might lead to better algorithms for enumerating self-maintaining sets.

Another consideration is that these results are mainly theoretical since they are based on an approximation algorithm to solve the SUBSET-FVS subproblem that, in practice, is not so useful since it uses an sphere-growing algorithmic technique that hides a fractional multicut problem and complex data structures to keep track of the solution space yet analysed that would lead to difficult implementation problems. Since there are several possible applications for the enumeration of chemical organisations in metabolic networks, proposing practical algorithms for such networks is an important task and, therefore, this is an interesting open problem.

One yet not well-defined but interesting perspective is to further investigate a possible biological role that the blocking compounds identified for our enumeration algorithm may have. In the context of the evolution of metabolic networks, a possibility is that they could correspond to nutrients that the organism could not yet deal with and that could be linked to the acquisition of new enzymatic capabilities which may improve its metabolic functions and possibly enable it to explore new biological environments.

Conclusion

In this thesis, we presented three different methods for enumerating special subnetworks contained in a metabolic network: metabolic stories, minimal precursor sets and chemical organisations. For each of the three methods, we gave theoretical results, and for the two first ones, we further provided an illustration on how to apply them in order to study the metabolic behaviour of living organisms. At the end of each chapter, open problems and perspectives were discussed, providing the reader with some of our ideas for following up on the developed methods and also for potential biological applications in the case of chemical organisations.

As concerns metabolic stories, we covered the work from its original motivation raised by the need to treat metabolomics experimental data to an actual analysis of a metabolomics experiment concerning yeast cells exposed to the heavy metal cadmium applying the method developed during the thesis. We also presented all intermediate results which included the modelling aspects, formal definition of the computational problem, proposed algorithms both to find one story and to enumerate them all, as well as theoretical results for related problems. We showed a polynomial algorithm to find one story based on growing a more simple structure, that we called a pitch, that can be easily obtained through a total ordering of the nodes of the input graph. We then described how to explore the relationship between total orderings of the nodes and pitches to propose an algorithm that enumerates all stories. We also showed that the related problem of finding one story with a fixed set of sources and targets is NP-complete and that the enumeration of path-stories is probably hard, since it is as hard as the enumeration of minimal hitting sets, which is an important problem in computer science whose theoretical complexity has been open since long. Nevertheless, we used our method, called Gobbolino, to try both to interpret the outcome of metabolomics data and to automatically recover metabolic pathways, which represented an application of the method in a different context than the one it was designed for. In the future, we believe that the method as it is may be further explored in the context of metabolomics analyses, and that it may be also very useful in the alternative context of automatic metabolic pathway inference, in particular our method proposes alternative pathways. A simple first idea of improvement which is not directly related to our method and that would benefit both applications is to replace the computation of lightest paths by an atom mapping route approach, which may enable to identify more realistic reaction chains between the pairs of black nodes, and thus lead to better results in biological terms. As future perspectives for the topic, we believe that a first step is to review the formal definition of a story in a way that internal cycles may be considered, which will have as consequence a decrease in the quantity of stories found. A second important point to be introduced in the definition of a story is some degree of incertitude on the black/white node status, since a strict dichotomy does not always appear as clear in the results of the experiments.

As concerns minimal precursor sets, we also covered the work from the original motivation of providing a formal definition for the problem that included cyclic solutions to the application

of the developed method in which the possible metabolic exchanges in a symbiotic systems were studied. We presented a first algorithm called Pitufo based on the creation of a data structure called a replacement tree, then we moved to Pitufina which is an improved version that completely avoids building this memory-consuming structure while keeping the basic recursive algorithm which backtracks from the target to the minimal precursor sets and, finally, we described the Papa Pitufo algorithm that through local modifications in the network can “save" solutions in a branch in a way that enables those to be reused by other branches with similar chemical dependencies. Theoretical results were provided on the problems of finding minimum precursor sets or cut sets, as well as on enumerating all minimal precursor sets. The two first were shown to be NP-Complete while the enumeration problem cannot be solved in polynomial total time unless P=NP. As a main future improvement, we may cite the introduction of a stoichiometric validation on the algorithm.

As concerns now chemical organisations, we considered the complexity of finding and enumerating such objects considering their use in the context of metabolic networks which are expected to be both mass and flux-consistent. We could prove that for such a class of networks, there always exist two chemical organisations corresponding to the closure of the empty set (that in this model are all the inputs of the network) and the whole network. We called these two chemical organisations trivial and we also showed that finding non-trivial organisations is NP-hard. We presented an algorithm to perform the enumeration of closed sets based on finding local hitting sets of so-called blocking compounds. These closed sets are a superset of the chemical organisations and we may filter them through linear programming by checking whether they are self-maintaining or not. The mentioned blocking compounds play a key role in the algorithm and in the computational complexity of the problem, and we believe that they may also play an important biological role. In this sense, we believe that it will be interesting to try to analyse them in a network evolution perspective, trying to identify if they may be linked to the acquisition of new enzymes or to the adaptation of an organism to different niches in which some new metabolites are present.

From a biological perspective, we believe that the developed methods may be useful for investigating metabolic properties in several different ways, as we tried to illustrate in this work. All softwares developed are publicly available and their application is immediate for problems such as analysing metabolomics data and identifying nutrient-related metabolic requirements. From a computer science perspective, we believe that at least the formal problem defined by the metabolic stories is general enough to be applied and studied in different contexts, probably even outside of biology, since it is a natural generalisation of the problem of enumerating minimal feedback arc sets, or their complement, namely maximal directed acyclic subgraphs.

Bibliography

Acuña, V., Chierichetti, F., Lacroix, V., Marchetti-Spaccamela, A., Sagot, M.-F., and Stougie, L. (2009). Modes and cuts in metabolic networks: Complexity and algorithms. *Biosystems*, 95(1):51–60.

Acuña, V., Birmelé, E., Cottret, L., Crescenzi, P., Jourdan, F., Lacroix, V., Marchetti-Spaccamela, A., Marino, A., Milreu, P. V., Sagot, M.-F., and Stougie, L. (2012a). Telling stories: Enumerating maximal directed acyclic graphs with a constrained set of sources and targets. *Theoretical Computer Science*, 457(0):1–9.

Acuña, V., Milreu, P. V., Cottret, L., Marchetti-Spaccamela, A., Stougie, L., and Sagot, M.-F. (2012b). Algorithms and complexity of enumerating minimal precursor sets in genome-wide metabolic networks. *Bioinformatics*.

Antonov, A. V., Dietmann, S., Wong, P., and Mewes, H. W. (2009). Ticl – a web tool for network-based interpretation of compound lists inferred by high-throughput metabolomics. *FEBS Journal*, 276(7):2084–2094.

Arita, M. (2004). The metabolic world of Escherichia coli is not small. *Proc Natl Acad Sci U S A*, 101(6):1543–1547.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (1999). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, Berlin.

Ausiello, G., Franciosa, P., and Frigioni, D. (2001). Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In *Theoretical Computer Science*, volume 2202 of *Lecture Notes in Computer Science*, pages 312–328. Springer Berlin Heidelberg.

Bang-Jensen, J. and Gutin, G. (2010). *Digraphs: Theory, Algorithms and Applications*. Springer Monographs in Mathematics. Springer.

Barve, A., Rodrigues, J. F. M., and Wagner, A. (2012). Superessential reactions in metabolic networks. *Proceedings of the National Academy of Sciences*.

Berge, C. (1976). *Graphs and Hypergraphs*. North-Holland Mathematical Library. North-Holland Publishing Company.

Betzler, N. (2005). Steiner tree problems in the analysis of biological networks. Master's Thesis, Universitat Tubingen, Germany.

Blum, T. and Kohlbacher, O. (2008). Using Atom Mapping Rules for an Improved Detection of Relevant Routes in Weighted Metabolic Networks . *Journal of Computational Biology*, 15(6):565–576.

Bondy, J. and Murty, U. (1976). *Graph theory with applications*. American Elsevier Pub. Co.

Borenstein, E., Kupiec, M., Feldman, M. W., and Ruppin, E. (2008). Large-scale reconstruction and phylogenetic analysis of metabolic environments. *Proc Natl Acad Sci U S A*, 105(38):14482–14487.

Bourtzis, K. and Miller, T. A., editors (2008). *Insect Symbiosis*, volume 3. CRC Press.

Boyer, F. and Viari, A. (2003). Ab initio reconstruction of metabolic pathways. In *ECCB*, pages 26–34.

Boyle, N. and Morgan, J. (2009). Flux balance analysis of primary metabolism in chlamydomonas reinhardtii. *BMC Systems Biology*, 3(1):4.

Brinza, L., Viñuelas, J., Cottret, L., Calevro, F., Rahbé, Y., Febvay, G., Duport, G., Colella, S., Rabatel, A., Gautier, C., Fayard, J.-M., Sagot, M.-F., and Charles, H. (2009). Systemic analysis of the symbiotic function of *Buchnera aphidicola*, the primary endosymbiont of the pea aphid *Acyrthosiphon pisum*. *C R Biol*, 332(11):1034–1049.

Carbonell, P., Fichera, D., Pandit, S., and Faulon, J.-L. (2012). Enumerating metabolic pathways for the production of heterologous target chemicals in chassis organisms. *BMC Systems Biology*, 6(1):10.

Caspi, R., Altman, T., Dale, J. M., Dreher, K., Fulcher, C. A., Gilham, F., Kaipa, P., Karthikeyan, A. S., Kothari, A., Krummenacker, M., Latendresse, M., Mueller, L. A., Paley, S., Popescu, L., Pujar, A., Shearer, A. G., Zhang, P., and Karp, P. D. (2010). The metacyc database of metabolic pathways and enzymes and the biocyc collection of pathway/genome databases. *Nucleic Acids Research*, 38(suppl 1):D473–D479.

Centler, F., Fenizio, P. S. d., Matsumaru, N., and Dittrich, P. (2007). Chemical organizations in the central sugar metabolism of escherichia coli. In Deutsch, A., Brusch, L., Byrne, H., Vries, G. d., and Herzel, H., editors, *Mathematical Modeling of Biological Systems, Volume I*, Modeling and Simulation in Science, Engineering and Technology, pages 105–119. Birkhauser Boston.

Centler, F., Kaleta, C., di Fenizio, P. S., and Dittrich, P. (2008). Computing chemical organizations in biological networks. *Bioinformatics*, 24(14):1611–1618.

Charles, H. and Nardon, P. (1999). *Enigmatic Microorganisms and Life in Extreme Environments*, volume 1 of *Cellular Origin and Life in Extreme Habitats*, chapter Intracellular symbiotic bacteria within insects, pages 651–660. Kluwer Academic Publishers, Dordrecht, Netherlands.

Consortium, T. U. (2009). The universal protein resource (uniprot) 2009. *Nucleic Acids Research*, 37(suppl 1):D169–D174.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). *Introduction To Algorithms*. MIT Press.

Cottret, L., Milreu, P. V., Acuña, V., Marchetti-Spaccamela, A., Martinez, F. V., Sagot, M. F., and Stougie, L. (2008). Enumerating Precursor Sets of Target Metabolites in a Metabolic Network. In *WABI'2008*, pages 233–244.

Cottret, L., Milreu, P. V., Acuña, V., Marchetti-Spaccamela, A., Stougie, L., Charles, H., and Sagot, M.-F. (2010a). Graph-based analysis of the metabolic exchanges between two co-resident intracellular symbionts, aumannia cicadellinicola and sulcia muelleri, with their insect host, homalodisca coagulata. *PLoS Comput Biol*, 6(9):e1000904.

Cottret, L., Wildridge, D., Vinson, F., Barrett, M. P., Charles, H., Sagot, M.-F., and Jourdan, F. (2010b). Metexplore: a web server to link metabolomic experiments and genome-scale metabolic networks. *Nucleic Acids Research*, 38(suppl 2):W132–W137.

Croes, D., Couche, F., Wodak, S. J., and van Helden, J. (2006). Inferring meaningful pathways in weighted metabolic networks. *Journal of Molecular Biology*, 356(1):222 – 236.

Deitel, H. M. and Deitel, P. J. (2007). *Java How to Program*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 7th edition.

Dittrich, M. T., Klau, G. W., Rosenwald, A., Dandekar, T., and Muller, T. (2008). Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):i223–i231.

Dittrich, P. and di Fenizio, P. S. (2007). Chemical organisation theory. *Bull. Math. Biol.*, 69(4):1199–1231.

Dupont, P., Callut, J., Dooms, G., Monette, J.-N., and Deville, Y. (2006). Relevant subgraph extraction from random walks in a graph. *Research Report RR 380167*, 2006-07(2006-07).

Eiter, T., Makino, K., and Gottlob, G. (2008). Computational aspects of monotone dualization: A brief survey. *Discrete Appl. Math.*, 156:2035–2049.

Elliott, W. H. and Elliott, D. C. (2001). *Biochemistry and Molecular Biology*. Oxford University Press, Oxford, 2001, 2 edition.

Even, G., Naor, J., Schieber, B., and Sudan, M. (1998). Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20:151–174.

Fauchon, M., Lagniel, G., Aude, J.-C., Lombardia, L., Soularue, P., Petat, C., Marguerie, G., Sentenac, A., Werner, M., and Labarre, J. (2002). Sulfur sparing in the yeast proteome in response to sulfur demand. *Molecular Cell*, 9(4):713–723.

Faust, K., Dupont, P., Callut, J., and van Helden, J. (2010). Pathway discovery in metabolic networks by subgraph extraction. *Bioinformatics*, 26(9):1211–1218.

Feist, A. and Palsson, B. (2010). The biomass objective function. *Current Opinion in Microbiology*, 13(3):344 – 349.

Feist, A. M., Herrgard, M. J., Thiele, I., Reed, J. L., and Palsson, B. O. (2009). Reconstruction of biochemical networks in microorganisms. *Nat Rev Micro*, 7(2):129–143.

Fong, S. S., Burgard, A. P., Herring, C. D., Knight, E. M., Blattner, F. R., Maranas, C. D., and Palsson, B. O. (2005). In silico design and adaptive evolution of escherichia coli for production of lactic acid. *Biotechnology and Bioengineering*, 91(5):643–648.

Fontana, W. and Buss, L. (1994). The Arrival of the fittest: towards a theory of biological organization. *Bull. Math. Biol.*, 56:1–64.

Gabaldon, T., Pereto, J., Montero, F., Gil, R., Latorre, A., and Moya, A. (2007). Structural analyses of a hypothetical minimal metabolism. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1486):1751–1762.

Gagneur, J., Krause, R., Bouwmeester, T., and Casari, G. (2004). Modular decomposition of protein-protein interaction networks. *Genome Biology*, 5(8):R57.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gevorgyan, A., Poolman, M. G., and Fell, D. A. (2008). Detection of stoichiometric inconsistencies in biomolecular models. *Bioinformatics*, 24(19):2245–2251.

Gurvich, V. and Khachiyan, L. (1999). On generating the irredundant conjunctive and disjunctive normal forms of monotone boolean functions. *Discrete Applied Mathematics*, 96-97:363 – 373.

Handorf, T., Christian, N., Ebenhˆh, O., and Kahn, D. (2008). An environmental perspective on metabolism. *J Theor Biol*, 252(3):530–537.

Handorf, T., Ebenhöh, O., and Heinrich, R. (2005). Expanding metabolic networks: scopes of compounds, robustness, and evolution. *J Mol Evol*, 61(4):498–512.

Holme, P., Huss, M., and Jeong, H. (2003). Subnetwork hierarchies of biochemical pathways. *Bioinformatics*, 19(4):532–538.

Hucka, M., Finney, A., Bornstein, B. J., Keating, S. M., Shapiro, B. E., Matthews, J., Kovitz, B. L., Schilstra, M. J., Funahashi, A., Doyle, J. C., and Kitano, H. (2004). Evolving a lingua franca and associated software infrastructure for computational systems biology: the systems biology markup language (sbml) project. *Systems Biology*, 1(1):41–53.

Jacobson, I., Booch, G., and Rumbaugh, J. (2012). *The Unified Software Development Process*. Addison-Wesley Object Technology. Pearson Education, Limited.

Johnson, D. S., Yannakakis, M., and Papadimitriou, C. H. (1988). On generating all maximal independent sets. *Information Processing Letters*, 27(3):119 – 123.

Joyce, A. R. and Palsson, B. O. (2008). Predicting gene essentiality using genome-scale in silico models. In Osterman, A. L. and Gerdes, S. Y., editors, *Microbial Gene Essentiality: Protocols and Bioinformatics*, volume 416 of *Methods in Molecular Biology*, pages 433–457. Humana Press.

Kaleta, C., Centler, F., and Dittrich, P. (2006). Analyzing molecular reaction networks. *Mol. Biotechnology*, 34(2):117–123.

Kaleta, C., Richter, S., and Dittrich, P. (2009). Using chemical organization theory for model checking. *Bioinformatics*, 25(15):1915–1922.

Kanehisa, M., Araki, M., Goto, S., Hattori, M., Hirakawa, M., Itoh, M., Katayama, T., Kawashima, S., Okuda, S., Tokimatsu, T., and Yamanishi, Y. (2008). Kegg for linking genomes to life and the environment. *Nucleic Acids Research*, pages 480–484.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Kauffman, K. J., Prakash, P., and Edwards, J. S. (2003). Advances in flux balance analysis. *Current Opinion in Biotechnology*, 14(5):491 – 496.

Klamt, S., Haus, U.-U., and Theis, F. (2009). Hypergraphs and cellular networks. *PLoS Comput Biol*, 5(5):e1000385.

Klein, C., Cottret, L., Kielbassa, J., Charles, H., Gautier, C., Vasconcelos, A. T., Lacroix, V., and Sagot, M.-F. (2012a). Exploration of the core metabolism of symbiotic bacteria. *BMC Genomics*, 13(1):438.

Klein, C., Marino, A., Sagot, M.-F., Vieira Milreu, P., and Brilli, M. (2012b). Structural and dynamical analysis of biological networks. *Briefings in Functional Genomics*.

Koschutzki, D. and Schreiber, F. (2008). Centrality analysis methods for biological networks and their application to gene regulatory networks. *Gene Regulation and Systems Biology*, 2:193–201.

Lacroix, V., Cottret, L., Thébault, P., and Sagot, M. F. (2008). An Introduction to Metabolic Networks and Their Structural Analysis. *TCBB*, 5(4):594–617.

Lafaye, A., Junot, C., Pereira, Y., Lagniel, G., Tabet, J.-C., Ezan, E., and Labarre, J. (2005). Combined proteome and metabolite-profiling analyses reveal surprising insights into yeast sulfur metabolism. *Journal of Biological Chemistry*, 280(26):24723–24730.

Larhlimi, A. and Bockmayr, A. (2009). A new constraint-based description of the steady-state flux cone of metabolic networks. *Discrete Applied Mathematics*, 157(10):2257 – 2266.

Leader, D. P., Burgess, K., Creek, D., and Barrett, M. P. (2011). Pathos: A web facility that uses metabolic maps to display experimental changes in metabolites identified by mass spectrometry. *Rapid Communications in Mass Spectrometry*, 25(22):3422–3426.

Li, Z.-S., Lu, Y.-P., Zhen, R.-G., Szczypka, M., Thiele, D. J., and Rea, P. A. (1997). A new pathway for vacuolar cadmium sequestration in saccharomyces cerevisiae: Ycf1-catalyzed transport of glutathionato cadmium. *PNAS*, 94(1).

Madalinski, G., Godat, E., Alves, S., Lesage, D., Genin, E., Levi, P., Labarre, J., Tabet, J.-C., Ezan, E., and Junot, C. (2008). Direct introduction of biological samples into a ltq-orbitrap hybrid mass spectrometer as a tool for fast metabolome analysis. *Analytical Chemistry*, 80(9):3291–3303.

Marashi, S.-A. and Bockmayr, A. (2011). Flux coupling analysis of metabolic networks is sensitive to missing reactions. *Biosystems*, 103(1):57 – 66.

Mason, O. and Verwoerd, M. (2007). Graph theory and networks in biology. In *IET Systems Biology, 1:89 – 119*, pages 89–119.

McCutcheon, J. P. and Moran, N. A. (2007). Parallel genomic evolution and metabolic interdependence in an ancient symbiosis. *Proc Natl Acad Sci U S A*, 104(49):19392–19397.

Milreu, P., Acuña, V., Birmelé, E., Crescenzi, P., Marchetti-Spaccamela, A., Sagot, M.-F., Stougie, L., and Lacroix, V. (2010). Enumerating chemical organisations in consistent metabolic networks: Complexity and algorithms. In Moulton, V. and Singh, M., editors, *Algorithms in Bioinformatics*, volume 6293 of *Lecture Notes in Computer Science*, pages 226–237. Springer Berlin / Heidelberg.

Nielsen, L. R., Andersen, K. A., and Pretolani, D. (2005). Finding the k shortest hyperpaths. *Comput. Oper. Res.*, 32(6):1477–1497.

Orth, J. D. and Palsson, B. (2010). Systematizing the generation of missing metabolic knowledge. *Biotechnol. Bioeng.*, 107(3):403–412.

Papin, J. A., Stelling, J., Price, N. D., Klamt, S., Schuster, S., and Palsson, B. O. (2004). Comparison of network-based pathway analysis methods. *Trends in Biotechnology*, 22(8):400 – 405.

Pavlopoulos, G., Secrier, M., Moschopoulos, C., Soldatos, T., Kossida, S., Aerts, J., Schneider, R., and Bagos, P. (2011). Using graph theory to analyze biological networks. *BioData Mining*, 4(1):10.

Pharkya, P., Burgard, A. P., and Maranas, C. D. (2004). Optstrain: A computational framework for redesign of microbial production systems. *Genome Research*, 14(11):2367–2376.

Poolman, M. G., Sebu, C., Pidcock, M. K., and Fell, D. A. (2007). Modular decomposition of metabolic systems via null-space analysis. *Journal of Theoretical Biology*, 249(4):691 – 705.

Price, N. D., Papin, J. A., Schilling, C. H., and Palsson, B. O. (2003). Genome-scale microbial in silico models: the constraints-based approach. *Trends in Biotechnology*, 21(4):162 – 169.

Raz, R. and Safra, S. (1997). A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 475–484, New York, NY, USA. ACM.

Reed, J. L., Famili, I., Thiele, I., and Palsson, B. O. (2006). Towards multidimensional genome annotation. *Nat Rev Genet*, 7(2):130–141.

Romero, P. R. and Karp, P. (2001). Nutrient-related analysis of pathway/genome databases. *Pac Symp Biocomput*, pages 471–482.

Satish Kumar, V., Dasika, M., and Maranas, C. (2007a). Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics*, 8(1):212.

Satish Kumar, V., Dasika, M., and Maranas, C. (2007b). Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics*, 8(1):212.

Scheer, M., Grote, A., Chang, A., Schomburg, I., Munaretto, C., Rother, M., Sohngen, C., Stelzer, M., Thiele, J., and Schomburg, D. (2011). Brenda, the enzyme information system in 2011. *Nucleic Acids Research*, 39(suppl 1):D670–D676.

Schilling, C. H., Letscher, D., and Palsson, B. O. (2000). Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *Journal of Theoretical Biology*, 203(3):229 – 248.

Schmidt, J. (2009). Enumeration: algorithms and complexity. Research report, Université de la Méditerranée.

Schuster, S., Dandekar, T., and Fell, D. A. (1999). Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *Trends in Biotechnology*, 17(2):53–60.

Schuster, S. and Hilgetag, C. (1994). On elementary flux modes in biochemical reaction systems at steady state. *J. Biol. Syst.*, 2(2):165–182.

Schwikowski, B. and Speckenmeyer, E. (2002). On enumerating all minimal solutions of feedback problems. *Discrete Applied Mathematics*, 117(1-3):253 – 265.

Scott, M. S., Perkins, T., Bunnell, S., Pepin, F., Thomas, D. Y., and Hallett, M. (2005). Identifying regulatory subnetworks for a set of genes. *Molecular and Cellular Proteomics*, 4(5):683–692.

Seymour, P. D. (1995). Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288.

Stelling, J. (2004). Mathematical models in microbial systems biology. *Current Opinion in Microbiology*, 7(5):513 – 518.

Suthers, P. F., Zomorrodi, A., and Maranas, C. D. (2009). Genome-scale gene/reaction essentiality and synthetic lethality analysis. *Mol Syst Biol*, 5.

Takahashi, H. and Matsuyama, A. (1980). An approximate solution for the steiner problem in graphs. *Math Japan*, 24:573–577.

Thiele, I. and Palsson, B. O. (2010). A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nat. Protocols*, 5(1):93–121.

Urbanczik, R. and Wagner, C. (2005). Functional stoichiometric analysis of metabolic networks. *Bioinformatics*, 21(22):4176–4180.

Vallenet, D., Labarre, L., Rouy, Z., Barbe, V., Bocs, S., Cruveiller, S., Lajus, A., Pascal, G., Scarpelli, C., and Médigue, C. (2006). Mage: a microbial genome annotation system supported by synteny results. *Nucleic Acids Res*, 34(1):53–65.

van Helden, J., Wernisch, L., Gilbert, D., and Wodak, S. J. (2002). Graph-based analysis of metabolic networks. *Ernst Schering Res Found Workshop*, 38:245–74.

Varma, A. and Palsson, B. O. (1994a). Metabolic flux balancing: Basic concepts, scientific and practical use. *Nat Biotech*, 12(10):994–998.

Varma, A. and Palsson, B. O. (1994b). Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type escherichia coli w3110. *Applied and Environmental Microbiology*, 60(10):3724–3731.

Verwoerd, W. (2011). A new computational method to split large biochemical networks into coherent subnets. *BMC Systems Biology*, 5(1):25.

Wu, D., Daugherty, S. C., Aken, S. E. V., Pai, G. H., Watkins, K. L., Khouri, H., Tallon, L. J., Zaborsky, J. M., Dunbar, H. E., Tran, P. L., Moran, N. A., and Eisen, J. A. (2006). Metabolic complementarity and genomics of the dual bacterial symbiosis of sharpshooters. *PLoS Biol*, 4(6):e188.

Yoon, J., Si, Y., Nolan, R., and Lee, K. (2007). Modular decomposition of metabolic reaction networks based on flux analysis and pathway projection. *Bioinformatics*, 23(18):2433–2440.

Appendix: Metabolic Network Software Library

The purposes of this appendix are twofold: first to concentrate on a documentation of the code done to address the studied problems covered in this work, and second to highlight the general features that were carefully designed to allow this code to be extended in order to possibly address different problems than the original ones for which it was developed. In this work, we have used Java to code our algorithms. For an introduction to the Java language, we refer to Deitel and Deitel (2007). Java uses the object-oriented paradigm, which means that the code is written as classes of objects that have attributes and that may perform some tasks, called methods. Programs in object-oriented languages may be seen as a natural interaction between objects of these classes, that exchange messages in order to accomplish some task.

We may divide the software engineering results of this work into three libraries *i.e.*, collection of classes: **MetNetwork**, **Pitufoland** and **Gobbolino**. The first one contains all the classes developed to deal with some of the different ways there exists to store data of a metabolic network in a computer. The *MetNetwork* library contains classes that allow to deal with a metabolic network as a compound graph, reaction graph, or hypergraph, and to extract its stoichiometric matrix. More interestingly, the classes are able to deal with the common SBML format in which metabolic network data are usually available but are general enough to allow extensions dealing with different formats. *Pitufoland* contains the classes responsible for implementing the three methods proposed for the enumeration of minimal precursor sets, namely the *Pitufo*, *Pitufina* and *Papa Pitufo* algorithms. *Gobbolino* contains the classes responsible for implementing our algorithm for enumeration of metabolic stories based on sampling the space of orderings of the vertices of the graph. Both *Pitufoland* and *Gobbolino* have application classes designed for providing to the final user command-line settings for exploring several different possibilities of parameterisation. *Gobbolino* provides also a *Cytoscape* plug-in in which users have access to a visual representation of the metabolic network, its compressed version and the complete list of metabolic stories.

Unified Modelling Language (UML)

Before introducing the software engineering aspects of the three above-mentioned libraries, we first provide a short introduction to the **Unified Modelling Language (UML)** that has been proposed during the 90s in order to unify the existing modelling languages that proliferated at that time. UML has been successful in its goal and now it has become the most used language to describe object-oriented models, such as a Java program. For a more comprehensive guide to UML, we refer to Jacobson et al. (2012).

Basically, UML is a collection of diagrams describing different aspects of an object-oriented model. The main entities that are part of such a model are the classes of an object-oriented

software and their relationships. There are three classes of diagrams in UML: Structural diagrams, such as the Class diagram, Behaviour diagrams, such as the Use Case diagram, and Interaction diagrams, such as the Sequence diagram. These three cited diagrams are the three most used from the whole collection of diagrams described by the UML Language. In this work, we will use only Class diagrams to explore our libraries, as indeed this is the main building block of any object-oriented model.

In object-oriented languages, a **class** represents a set of objects that share the same attributes, behaviour and relationships. Classes are drawn, in a Class diagram, as a rectangle divided in three parts: name, set of attributes and set of methods. Classes may have different relationships with other classes. An **association** is a binary relationship that models a static association between objects of a class. For instance, we may think that buildings have flats and flats have rooms. These are examples of associations between classes of objects. A special kind of association is an **aggregation** which models a "whole-part" relationship, that happens when an object of a class is made of objects of other classes, that in their turn exist to be part of the first object. As an example, a Vehicle class that is an aggregation of Chassis, Wheels, and so on. Another kind of association is the **composition** relationship, which may be seen as an ownership relationship. It is like an aggregation with the additional information that the objects share the same life-cycle, *i.e.*, the owned object usually does not exist out of the scope of the owner. The Room object, for instance, makes sense only when inside some building or house, there is no "room" on its own. All kinds of association may be annotated with a multiplicity that tells us how many objects are possibly involved in that relationship, with possible values zero, one, or indeterminately many objects (denoted by an asterisk or by the letter n). Finally, there are hierarchical relationships between classes that specify the richer aspects of an object-oriented model. A class may be part of a generalization-specialization relationship, that models the fact that one class is a descendent of another, sharing with its ancestral class all of its attributes and methods but possibly providing additional ones or even modifying the original parent behaviour. An example of a simple UML model capturing all these properties is shown in Figure 6.1.

Notice, in Figure 6.1, that classes Residence and Room have their names written in italic. This is the case because they are *abstract classes*. Abstract classes represent a generic concept, with description purposes only and they are not instantiated. In other words, there exist no objects of such classes but only of their descendants and, thus, abstract classes must be specialized by more specific concrete classes that derive from them. Another entity that UML introduces to deal with abstract and description purposes are **Interfaces**, which define a list of methods that should be further implemented by some class. An interface basically defines a contract or protocol, specifying how a class may interact with any other class that agrees with the contract.

Moving a bit to the Software Engineering aspect of an object-oriented language, a nice feature of both abstract classes and interfaces is that this allows a natural extension of a set of classes in such a way that implementation details of a class are not needed. This eases the task of collaborative programming, integration and extension of software, among several other advantages. Plug-in developments that allow programmers to extend softwares by integrating their own code to them is possible mainly due to the right use of interfaces. Once defined what is the protocol to interact with the software, *i.e.*, what is the possible information you may recover or provide to it, it becomes possible to extend it by plugging in your own piece of code.

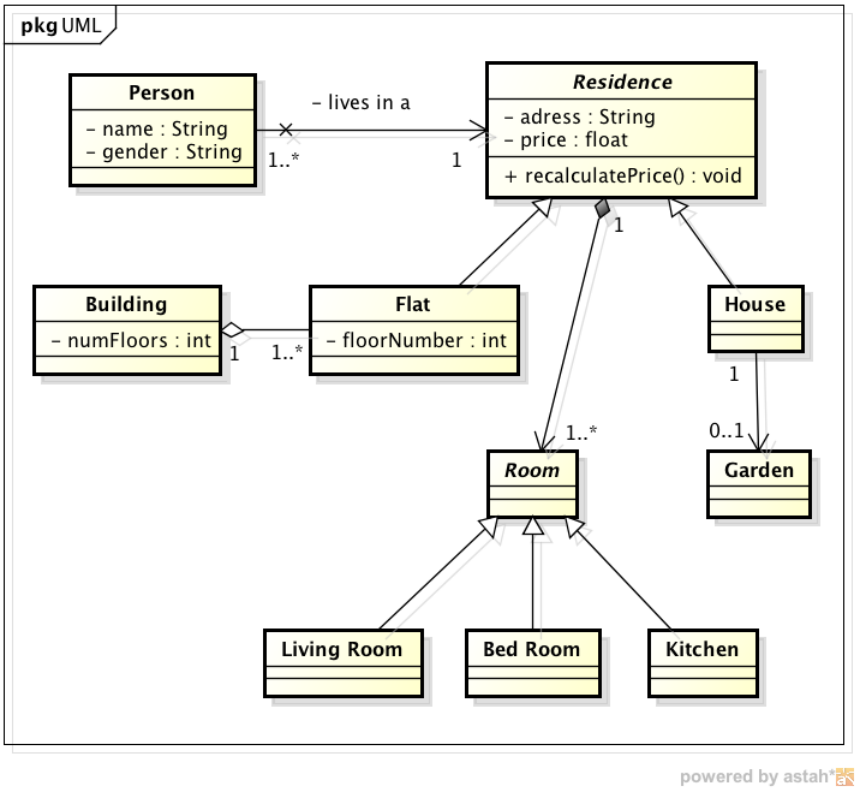


Figure 6.1: UML class diagram with classes Person, Residence, Building, Flat, House, Room, Garden, Living Room, Bed Room and Kitchen. Examples of attributes are the name of a person or the price of a residence. An example of an operation is the *recalculatePrice* method that updates the price of the residence. A person has an association with a residence with the name "lives in a", modelling the fact that people live in residences. The multiplicity of such a relationship is $1 \ldots *$ at the Person endpoint and 1 at the Residence endpoint, considering that one or more persons may live in one residence. Building has an aggregation association with Flat, denoting that a building is made of several flats. Flat and House are specializations of Residence, *i.e.*, both of them are different kinds of residences. Residence has a composition relationship with Room, that is a generalization of Living Room, Kitchen and Bed Room. A house may or may not have a Garden.

MetNetwork Library

In this section, we will present the MetNetwork library. First we present the UML Class diagram, then the main role of each class and an example of utilisation.

UML diagram

Let us now focus on the classes we have developed for dealing with metabolic networks: the *MetNetwork* library. A UML Class diagram for the classes related to a metabolic network representation are shown in Figure 6.2. A chemical compound and a chemical reaction are basically modelled by its name and a unique identifier for the classes *Compound* and *Reaction*, respectively. *Reaction* objects also have a boolean variable to indicate if they are reversible or not. A *Stoichiometry* class is introduced to store the stoichiometric coefficients of the compounds in the reactions. Notice that a *Stoichiometry* object is associated with a *Compound* object in a "has a" relationship. A *Reaction*, on its turn, is composed of two

lists of *Stoichiometry* objects, one for its substrates and the other for its products. *Reaction* provides also methods for inclusion, deletion and retrieval of products and substrates as well as information on their consumption or production stoichiometric coefficients. The *Metabolic Network* interface lists some common operations one may be interested in for a metabolic network object, such as: loading a network from a file, modifying the network by adding and removing compounds and reactions, recovering its list of compounds and reactions, getting the list of reactions that consume or produce a given compound, computing an induced subnetwork from a list of compounds or reactions, or computing the union of two networks. There are three concrete classes that implement the *Metabolic Network* interface: *Compound Network*, *Reaction Network* and *Hypergraph*. They are also specializations of *DefaultDirectedGraph*, for the first two mentioned classes, and *SetHypergraph*, for the third one. These are two classes from the *JGraphT* open source project that provide a large library of Java code to build graph and hypergraph objects.

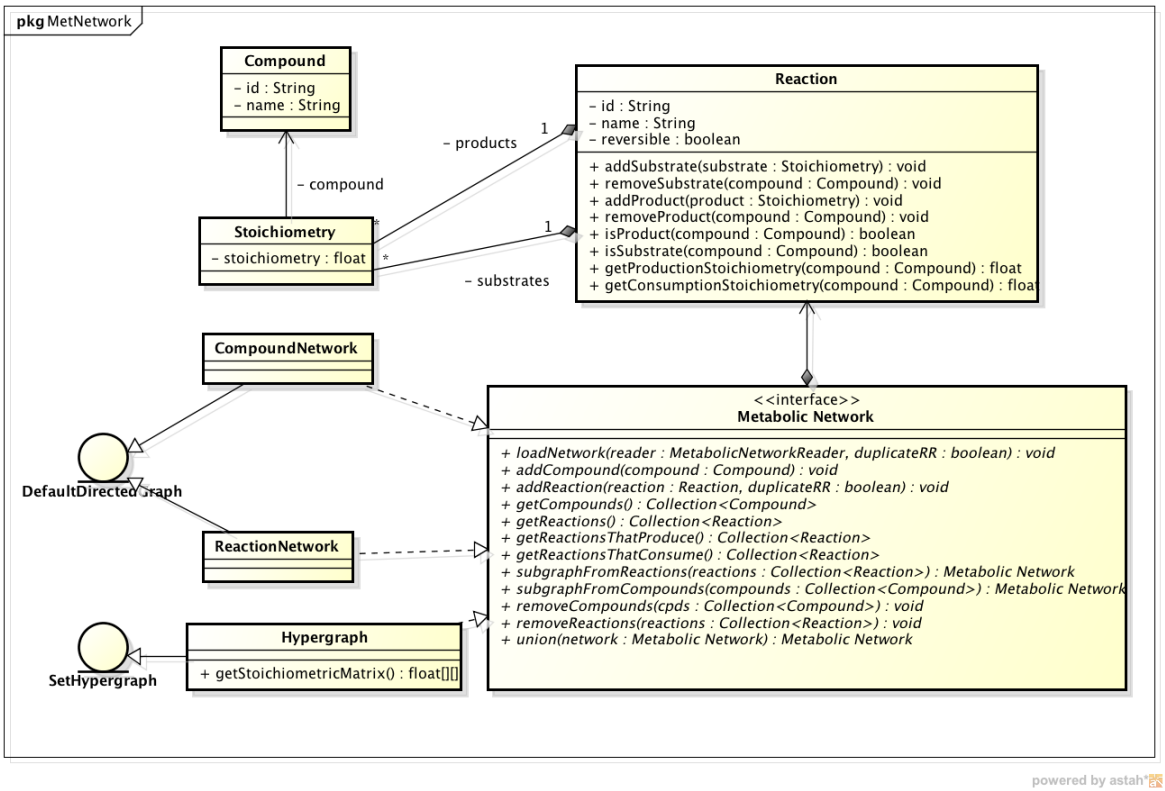


Figure 6.2: UML class diagram for the MetNetwork library with the main classes: Metabolic Network, Compound Network, Reaction Network, Hypergraph, Compound, Reaction and Stoichiometry.

Description of the classes

- – **Class Compound**: Represents a compound of the metabolic network.
  - – **Attributes**: Compound objects have only two string attributes: the *id* and the *name* of the chemical compound.
- – **Class Stoichiometry**: Represents the stoichiometric coefficient of each compound participating in a reaction. Therefore, it is an intermediate class to link reaction

objects to compound objects.

- **Attributes**: The single object is a float number to store the *stoichiometry* of a compound.

- **Class Reaction**: Represents a reaction of the metabolic network.
- **Attributes**: Reaction objects have three simple attributes. Two strings to store their *id* and *name* and a boolean value to indicate whether the reaction is *reversible* or not. Reaction objects have also two one-to-many associations with the class Stoichiometry, to store a list of its *substrates* and a list of its *products*.
- **Methods**: The Reaction class provides service methods to add and remove compounds from its list of substrates and products. There are also methods to answer if a given compound is a product or a substrate of the reaction and methods to get their stoichiometric coefficients.

- **Interface Metabolic Network**: Defines a list of services that metabolic networks are expected to provide, independent of their internal representations.
- **Methods**: Any metabolic network object should be able to be loaded (from a file, a memory stream, an URL, etc.) by means of an object of the class MetabolicNetworkReader, that will not be detailed in the context of this appendix. The parameter *duplicateRR* indicates whether the metabolic network should duplicate a reversible reaction in two or if it should keep only one object. The class also introduces several utilitary methods for adding and removing compounds and reactions, and to get a list of the reactions that consume or produce a given compound. There are some methods that return another metabolic network object. *SubgraphFromReactions* and *subgraphFromCompounds* extract and return a metabolic network which is a subgraph induced by a list of reactions or compounds, respectively. Finally, the *union* method computes the union between a metabolic network and a second one passed as parameter, returning a new metabolic network corresponding to this union.

- **Class Compound Network**: This class is a compound graph representation of a metabolic network. It is implemented as a directed graph (class *DefaultDirectedGraph* from the jGraphT library) whose nodes are compound objects and arcs are reaction objects.
- **Methods**: This class is an implementation of the *Metabolic Network* interface and thus provides implementations for all the methods defined in the interface. No specific methods are added.

- **Class Reaction Network**: This class is a reaction graph representation of a metabolic network. It is implemented as a directed graph (class *DefaultDirectedGraph* from the jGraphT library) whose nodes are reaction objects and arcs are instances of the *DefaultEdge* class defined in the jGraphT library, which corresponds to a simple relationship between two reactions.
- **Methods**: This class is an implementation of the *Metabolic Network* interface and thus provides implementations for all the methods defined in the interface. No specific methods are added.

- – **Class Hypergraph**: This class is a hypergraph representation of a metabolic network. It is implemented as a hypergraph (class *SetHypergraph* from the jGraphT library) whose nodes are compound objects and hyperarcs are reaction objects.
  – **Methods**: This class is an implementation of the *Metabolic Network* interface and thus provides implementations for all the methods defined in the interface. A new *getStoichiometricMatrix* method is provided that returns a float matrix corresponding to the stoichiometric matrix representation of the metabolic network.

Pitufoland Library

In this section, we will present the *Pitufoland* library. First we present the UML Class diagram, then the main role of each class and an example of utilisation.

UML diagram

Let us now focus on the classes we have developed for finding minimal precursor sets for a given set of target compounds in a metabolic network, the *Pitufoland* library. A UML Class diagram for the classes related to this task are shown in Figure 6.3. The main class is *PrecursorFinder* that is a generalization for any class solving the problem of finding minimal precursor sets for a given target set. We provide three different specializations of such a class: *Pitufo*, *Pitufina* and *Papa Pitufo* that correspond to the three methods presented in Chapter 4.

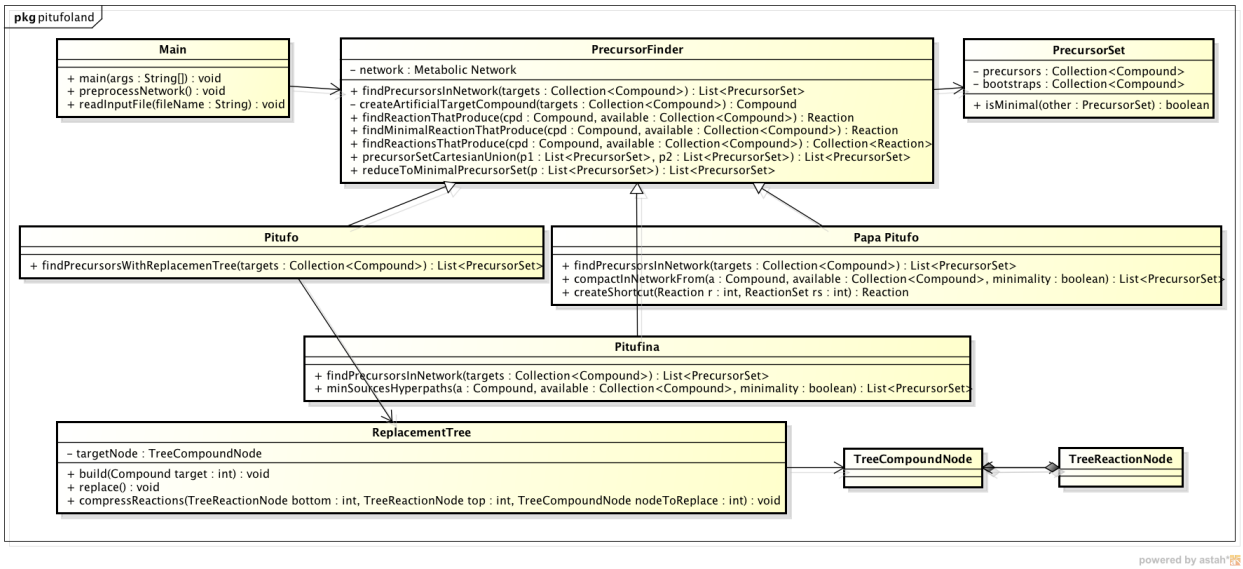


Figure 6.3: UML class diagram for the *Pitufoland* library with the main classes: *PrecursorSet*, *PrecursorFinder*, *Pitufo*, *Pitufina*, *Papa Pitufo* and *ReplacemenTree*.

Description of the classes

- – **Class PrecursorSet**: represents a solution of the problem, *i.e.*, a list of compounds that are the minimal precursor sets to produce a given target.

  - **Attributes**: A list of compounds representing the *precursors* and a list of *bootstrap* compounds representing the metabolites that are internally supplied and are further recycled by the cell.

  - **Class Main**: Represents the user interface. A main object is needed for dealing with the command-line arguments, reading the input data and calling the appropriate methods for finding the solution and printing it out to the user.
  - **Methods**: The *main* method is the starting point of the application. It basically parses the command-line options and calls the appropriate method to deal with the input problem. Next subsection explains the main options available for using the *Pitufoland* library. Two parameters are mandatory: the name of a metabolic network input file (currently, it must be in the SBML format) and the name of an input file containing the list of targets and, optionally, the list of user-defined potential precursors (also a XML format defined by our library). The method *preprocessNetwork* does the preprocessing steps described in Section 4.4, such as transforming user-defined precursors into topological precursors, and also the preprocessing steps described in Section 4.5, namely the removal of void cycles.

- **Class PrecursorFinder**: represents a generic class to solve the problem of finding all minimal precursor sets for a given set of target metabolites.
  - **Attributes**: The single attribute of this class is the metabolic network for which the enumeration of minimal precursor sets will be performed.
  - **Methods**: The main method is *findPrecursorsInNetwork* that has as parameter a collection of Compound objects representing the targets and returns the list of precursor sets found to produce these targets. There are some support methods that are useful for descendant classes: *createArtificialTargetCompound* implements the *singleton target set* preprocessing step described in Section 4.4; *findReactionThatProduce*, *findMinimalReactionThatProduce* and *findReactionsThatProduce* are used to recover a reaction (the first two methods) or a list of reactions (the third one) that produces a given compound considering that a list of other compounds is available; the method *precursorSetCartesianUnion* computes and returns the cartesian union of two lists of precursor sets and is important for intermediate steps of both PITUFINA and PAPA PITUFO; finally, the method *reduceToMinimalPrecursorSet* extracts and returns the minimal precursor sets among the ones passed as argument to the method.

- **Class Pitufo**: Implements the PITUFO method for finding minimal precursor sets building a secondary structure called a *ReplacementTree*.
  - **Methods**: The unique method of this class is *findPrecursorSetsWithReplacementTree*, that gets a list of *Compound* objects as parameter and returns a list of minimal precursor sets that are able to produce them. For finding the solutions, it builds a *ReplacementTree* object for the singleton target compound (if there are more than one target compound in the argument, an artificial one is created using the *createArtificialTargetCompound* method defined in the parent class) and inspecting the leaves of the tree once the building process is finished.

- **Class ReplacementTree**: Represents the replacement tree object described in Section 4.4.

  - **Methods**: The main methods of this class are *build* and *replace*. The method *build* is the starting point that creates a *TreeCompoundNode* for the target parameter and then calls recursively the method *replace* in order to replace it by the substrates of the reactions that are able to produce the target. The method *compressReactions* is used to compress reactions once the recursion stopped, moving the leaves closer to the root of the replacement tree.

- – **Class Pitufina**: Implements the PITUFINA method for finding minimal precursor sets through a recursive traversal of the metabolic network, backtracking from the singleton target to the minimal precursor sets.
  - **Methods**: The main method of this class is *minSourcesHyperpaths* that is an implementation of the Algorithm REACTIONDECOMP, presented in Section 4.4, *i.e.*, it is a reaction decomposition recursive approach used by PITUFINA to traverse the network from the target metabolite to the minimal precursor sets. The method *findPrecursorsInNetworkForTarget* defined in the parent class is overwritten in order to call *minSourcesHyperpaths* for the singleton target compound (if there are more than one target compound in the argument, an artificial one is created using the *createArtificialTargetCompound* method defined in the parent class). Notice that the boolean parameter *minimality* changes the behaviour of the method, since a *true* value will restrict the selection of reactions to the minimal ones.

- – **Class Papa Pitufo**: Implements the PAPA PITUFO method for finding minimal precursor sets through local modifications of the metabolic network, creating shortcuts in the form of artificial reactions.
  - **Methods**: The main method of this class is *compactInNetworkFrom* that is an implementation of the Algorithm PAPAPITUFO, presented in Section 4.4, *i.e.*, it is a reaction decomposition recursive approach used by PAPA PITUFO to traverse from the target metabolite to the minimal precursor sets and, once the recursion stops, to include an artificial reaction creating a shortcut from precursor sets to local target metabolites. The method *findPrecursorsInNetworkForTarget* defined in the parent class is overwritten in order to call *compactInNetworkFrom* for the singleton target compound (if there are more than one target compound in the argument, an artificial one is created using the *createArtificialTargetCompound* method defined in the parent class). Notice that the boolean parameter *minimality* changes the behaviour of the method, since a *true* value will restrict the selection of reactions to the minimal ones. Finally, the method *createShortcut* is an implementation of the Algorithm REPLACE also presented in Section 4.4.

Example of utilisation

The list of arguments that the user has to explore the *Pitufoland* library are:

- *-s=filename*: Metabolic network file in the SBML format.
- *-i=filename*: Input file in a XML application-defined format containing the list of targets and an optional list of user-defined potential precursor compounds.
- *-o*: Look separately at each target compound, instead of dealing with them as a set.
- *-mT*: Use the PITUFO method.

- *-mP*: Use the PITUFINA method.
- *-mPP*: Use the PAPAPITUFO method.
- *-mPMin*: Use the PITUFINA method with the minimality condition for choosing the reactions at each step.
- *-mPPMin*: Use the PAPAPITUFO method with the minimality condition for choosing the reactions at each step.
- *-ecc*: Eliminate void cycles from the network as a preprocessing step.
- *-random=off*: By default, the -ecc option uses a randomized approach to find the void cycles. This feature may be turned off, mainly for debugging purposes if one wants to repeat the same choices at each run.

As an example of a XML input file for the program, we present Figure 6.4 that shows an input file that specifies L-aspartate as a target compound and glucose as a user-defined potential precursor. Suppose that this file is called *input.xml* and that a SBML metabolic network input file called *network.xml* is also available. Suppose also that the *Pitufoland* library is packaged in a *pitufoland.jar* Java file. In this case, a command-line to compute the minimal precursor sets is: *java -jar pitufoland.jar -s="network.xml" -i="input.xml"*.

```
<?xml version="1.0"?>
<inputs date="2008-01-24" comment="input file designed to search for the precursors in the whole network of E.
  <input-for-model id="ecoli_metabolic_network" name="E. coli" />
    <precursor-compounds>
      <species id="GLC-6-P" comment="glucose" />
    </precursor-compounds>

    <target-compounds>
      <species id="L-ASPARTATE" comment="" />
    </target-compounds>
</inputs>
```

Figure 6.4: Example of the XML input file for defining the targets and the potential precursors.

Project URL

The *Pitufoland* source code is available at the following URL: *http://pitufo.gforge.inria.fr/*.

Gobbolino Library

In this section, we will present the Gobbolino library. First we present the UML Class diagram, then the main role of each class and an example of utilisation.

UML diagram

Let us now focus on the classes we have developed for enumerating metabolic stories given a set of black nodes in a metabolic network, the *Gobbolino* library. A UML Class diagram for the classes related to this task are shown in Figure 6.5. As the number of classes in this package is large, we preferred to put less details on the diagram. In the figure, we do not include the attributes and methods for the class to highlight mainly the relationships between the classes of the library. The main class for the enumeration process is *OrderBasedEnumerator*

that implements our metabolic stories enumeration algorithm presented in Chapter 3. The library also includes the class *ComputeBlackNodeScenario* that is able to transform the input metabolic network in the SBML format into the internally defined NEL format (stored in a *NELNetwork* object) and to apply the simplification preprocessing steps introduced in Section 3.4. The class *RandomBNSimulation* runs the enumeration algorithm without the need of supplying a list of black nodes, because it selects randomly a user-defined number of black nodes. This class was designed mainly for statistical analysis and was used to measure the efficiency of the preprocessing steps. The class *PathwayCoveringSimulation* implements the second biological application described in Section 3.6, namely the application of metabolic stories for automatic pathway recovery. The general purpose class *ComputeStatistics*, that was originally designed for computing statistics both on the input file and the output files of the enumeration process, was further adapted to produce an anthology (union of several stories) while reading the output file.

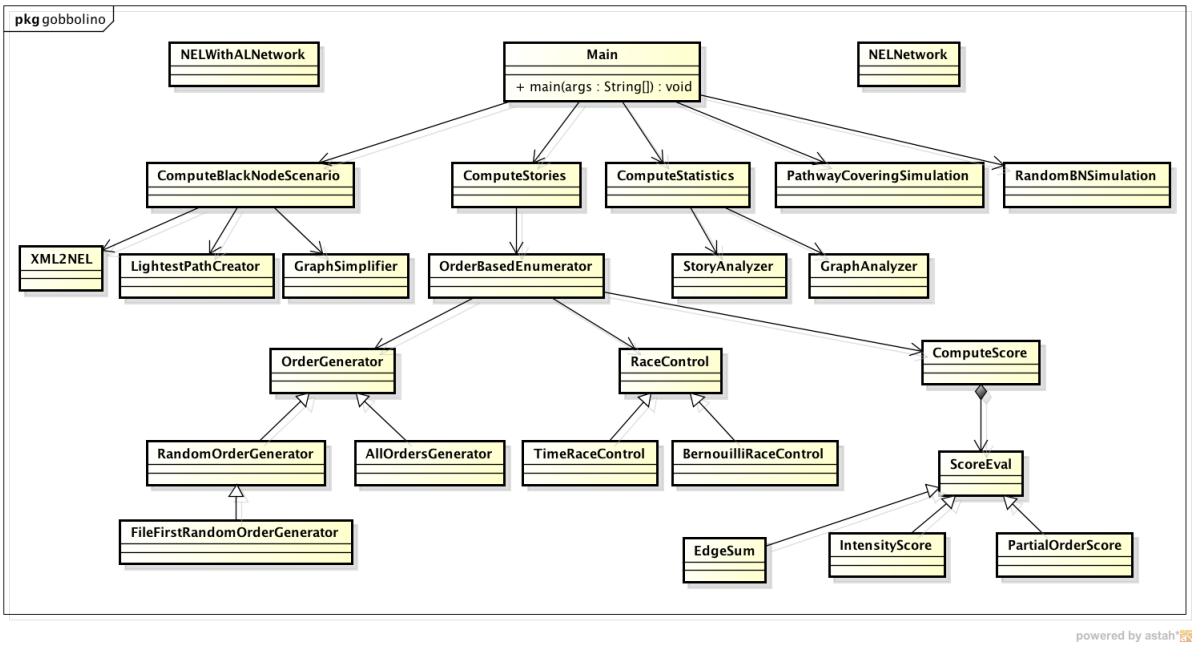


Figure 6.5: UML class diagram for the Gobbolino library with the main classes: ComputeBlackNodeScenario, ComputeStories, ComputeStatistics, PathwayCoveringSimulation and RandomBNSimulation.

Description of the classes

- – **Class NELNetwork**: represents a directed graph stored as an adjacency matrix with additional information on which ones among the vertices are the black nodes.
  – **Attributes**: The main attributes of this class is *adjacency* that is a boolean matrix of *nn* dimension, where *nn* is the number of nodes in the graph. There are two integer value vectors of *nn* dimension, to store the in-degree and the out-degree of the nodes, and a boolean vector of the same size that stores the information whether the node is black or not.
  – **Methods**: The main method of this class is *readFile* that reads an input file in the NEL format and builds an *NELNetwork* object.

- – **Class NELWithALNetwork**: represents a directed graph stored as an adjacency list. It fulfills the same purpose of the *NELNetwork* class but with a different internal storage.

  – **Class Main**: Represents the user interface. A main object is needed for dealing with the command-line arguments, reading the input data files and calling the appropriate methods for enumerating stories or using the other operation modes available: compression of the SBML input file into a lightest-path simplified NEL file, computing statistics both on the input and output files, generating random simulations or performing the metabolic pathway automatic recovery.

  – **Methods**: The *main* method is the starting point of the application. It basically parses the command-line options and calls the appropriate method to deal with the input problem. Next subsection explains the main options available for using the *Gobbolino* library.

  – **Class ComputeBlackNodeScenario**: This class is responsible for dealing with the basic input file, a metabolic network in the SBML format, and preprocessing it to generate a NEL file containing the union of all lightest paths between the black nodes and applying all preprocessing steps as specified in Section 3.4. For performing these tasks, it makes use of the class *XML2NEL*, that converts the SBML input file into the NEL format; the *LightestPathCreator* class that computes all lightest paths between the black nodes and generates the corresponding network; and the *GraphSimplifier* class that applies the simplification operations and generates the final compressed network.

  – **Class ComputeStories**: This class is responsible for calling the enumeration of metabolic stories according with the user-defined parameters. To this goal, an object of the class *OrderBasedEnumerator* is created.

  – **Class OrderBasedEnumerator**: This class is responsible for actually enumerating the metabolic stories.

  – **Attributes**: This class is a specialization of *NELWithALNetwork*, inheriting its attributes. Therefore, this class *is* a network built from the input file. Additional attributes are: *control*, a *RaceControl* object used to decide if the enumeration should continue or be interrupted; *orderGenerator*, an *OrderGenerator* object used to generate the next total ordering of the nodes; *score*, a *ComputeScore* object used to assign a value to each story computed; *sasSet*, which is a map containing all stories generated so far, storing the minimal story arc set corresponding to the story as its identifier in the map.

  – **Methods**: The main method is *run*, that is called by a *ComputerStories* object. This method contains a main loop that generates a new ordering of the nodes through the *orderGenerator* object, computes a new story based on this ordering through the method *computeSAS* and then checks the stop conditions. A first stop condition may be reached by the *control* object, that is called after each new story is generated, if the control does not stop the computation, the method checks that the maximum number of stories to be generated is already reached. If this is not the case, the method checks if all permutations of the nodes were explored. If none of these stop conditions is reached, the computation of stories continues.

- **Class ComputeStatistics**: This class is responsible for computing statistics on the input and output files. For the output file, an anthology with the same name of the output file with an additional suffix "-anthology" is generated, containing the highest scoring stories.

- **Class PathwayCoveringSimulation**: This class is responsible for performing the experiment of metabolic pathway automatic recovery.
- **Methods**: The main method is *run*, that is called by a *PathwaySimulation* object (class not presented in the diagram). This method iterates over the input metabolic pathway files and for each one of them, computes a compressed network and enumerates metabolic stories. At the end of the processing, the method performs an analysis of the results and prints the accuracy obtained by the best story for each metabolic pathway.

- **Class RandomBNSimulation**: This class is responsible for enumerating metabolic stories through random selection of black nodes.
- **Methods**: The main method is *run*, that is called by a *Simulation* object (class not presented in the diagram). This method runs from a minimum to a maximum user-defined value that corresponds to the number of black nodes to be randomly selected. Once the list of black nodes is chosen, the method computes a compressed network and enumerates metabolic stories, outputting text files containing detailed information on the compression rates at each step of the processing.

Example of utilisation

The list of arguments that the user has to explore the *Gobbolino* library are divided in five different modes of operation. The first one, called *BNS*, is used to generate a compressed network. It consists in (optionally) converting a SBML (XML) metabolic network into a NEL graph format file, selecting (all/shortest/lightest) paths between black nodes and, finally, compacting the network by applying the compression rules (bottleneck removals/dead-end removals/self-loop removals) described in Section 3.4. The second operation mode, called *STORIES*, is the default one and is used to enumerate metabolic stories. The third operation mode, called *STAT*, is used to compute statistics both on the input network and on the output file containing the enumerated stories, as well as to generate an anthology of stories that have a score close or equal to the best score in the output file. The fourth mode, called *RANDOM*, computes stories for all networks contained in a given directory by choosing black nodes randomly. Finally, the last operation mode, called *PATHWAY*, that is used for enumerating stories for a given set of metabolic pathways and one metabolic network from which these pathways were extracted. Individual input files with the list of black nodes for each metabolic pathway are required. The user can select in which of the five operations mode he/she wants to operate by using the *-mode=NAME* option, where name is a string with the name of the mode. The *mode* parameter is optional and if not provided, the *STORIES* mode is assumed.

The parameters of *Gobbolino* for the *BNS* mode are:

- *-n=filename*: Input file name with no extension. If there is a XML file, it will be first converted to the NEL format. Then the lightest paths are going to be extracted (generating a file *filename-LP.NEL*) and simplified (generating a file *filename-LP-S*).

- *-paths=all | shortest | LIGHTEST*: Select the paths between black nodes to be preserved. The options are all paths, only the shortest paths (weight of arcs all equal to one) or only the lightest paths (weight of arcs equal to the out-degree of the target node), which is the default option.
- *-reversibility=ON | off*: The user may choose to ignore the reversibility of the reactions.

The parameters of *Gobbolino* for the *STORIES* mode are:

- *-n=filename*: Input file name with no extension (NEL format is assumed).
- *-o=filename*: Output file that will contain the stories generated. This parameter is optional and the filename *"./output/stories/stories.sto"* is the default output.
- *-q= 10 | all*: Maximum number of stories to be generated. 10 is the default value for this parameter. The value *all* indicates that there is no limit on the number of stories to be generated.
- *-order=RANDOM | all*. Chooses between randomly inspecting the space of orders or to generate all orders in a deterministic way. The randomized approach is the default option.
- *-score=EDGESUM | partialOrder | metabolomics*: Chooses the score function to be used: sum of edge weights, partial order (needs an additional file with the preferred partial orders) and metabolomics intensities (needs an additional information on the black nodes file with the concentrations of black nodes). Scores may be combined using semicolon (;) as a separator. For instance, the option *-score=METABOLOMICS;EDGESUM* will use two score functions to classify the stories.
- *-scoreFile=filename*: Input filename for the score functions that need additional data.
- *-maxScore*: Outputs only the maximum scored stories. Default is to output all of them.
- *-orderFile=filename*: Input filename containing total order of the nodes (1 per line, node ids separated with 1 space) to be used before passing to the random enumeration.
- *-stop=MAXSTORIES | bernouilli | time*: By default the enumeration of stories stops only when all stories are computed or the defined maximum number of stories is reached. Two other options available are to stop based on a time limit (specified in minutes) or on statistical measurements of the number of new stories generated trying to estimate the number of new stories still to be found.
- *-stopTime=10*: Sets how many minutes the enumeration should run. By default, the time stop condition is set to 10 minutes.
- *-stopSampleSize=10000*. Parameter for the Bernouilli stop condition, specifying the size of the sample after which each estimation of how well the space of solutions is explored is performed. By default, the sample size is set to 10000.");

All input files are searched in the *input* folder and the generated stories are saved in the *output/stories* folder.

The parameters of *Gobbolino* for the *STAT* mode are:

- *-n=filename*: Input file name with no extension (NEL format is assumed). If the file is present, several statistics will be performed.
- *-o=filename*: Output file containing the stories generated. This parameter is optional and the filename *"./output/stories/stories.sto"* is the default output. If the file is present, several statistical analyses will be performed. An anthology will also be generated.

The parameters of *Gobbolino* for the *RANDOM* mode are the same ones defined for the *STORIES* mode, with the addition of the following parameters:

- *-rbnMin=5*: Minimum number of random black nodes to be chosen.
- *-rbnMax=15*: Maximum number of random black nodes to be chosen.
- *-rbnIncrement=5*: Sets the Increment at each iteration from the minimum to the maximum number of black nodes specified.
- *-rbnMode=PERCENT | absolut*: Defines how the minimum and maximum parameters are going to be interpreted. The default is to treat them as percentages of the number of nodes of the network, *i.e.*, a value of 5 will be interpreted as an order to select 5% of the nodes of the network as black nodes. The value *absolut* changes the behaviour so that it will be interpreted as an order to select 5 black nodes.

The parameters of *Gobbolino* for the *PATHWAY* mode are the same ones defined for the *STORIES* mode. An important difference, however, is that here the input file must be the original one (SBML format), since several metabolic pathway-specific NEL files will be generated in the process. The metabolic pathway files also in the SBML format are expected to be found in the *input/pathway/pathways* folder. The input files containing, for each metabolic pathway, the list of corresponding black nodes are expected to be found in the folder *input/pathway/BNs* and to have the same name as the metabolic pathway file, but using a *BN* extension instead of XML. The score function automatically associated with the *PATHWAY* option is the partial ordering score and the input files describing the preferred orders for each pathway are expected to be found in the folder *input/pathway/score* and to have the same name as the metabolic pathway file, but using an *order* extension instead of XML. The additional parameters specific to the *PATHWAY* computation mode are:

- *-autoBN=sourceTarget | borensteinAll | borensteinOne | borensteinRG*: By default, the system will use the black nodes defined in the predefined folders. The user may want also to automatically choose black nodes using the *autoBN* option. The first option, *sourceTarget*, corresponds to choosing as black nodes the topological source/targets of the metabolic pathway. The other 3 options apply the *Borenstein et. al* method (Borenstein et al. (2008)) identifying source/target strongly connected components. The first one takes all compounds inside a strongly connected component as black nodes, the second one chooses one compound to represent the strongly connected component while the third applies the method on a reaction graph representation of the network, choosing as black nodes the compounds involved in the source/target reaction nodes.
- *-tas=10*: Defines a threshold on the number of black nodes. For pathways with less than *tas* black nodes, all stories are going to be computed (avoiding the randomized approach). For the others, the maximum number of stories defined (with the selected stop condition) is searched.

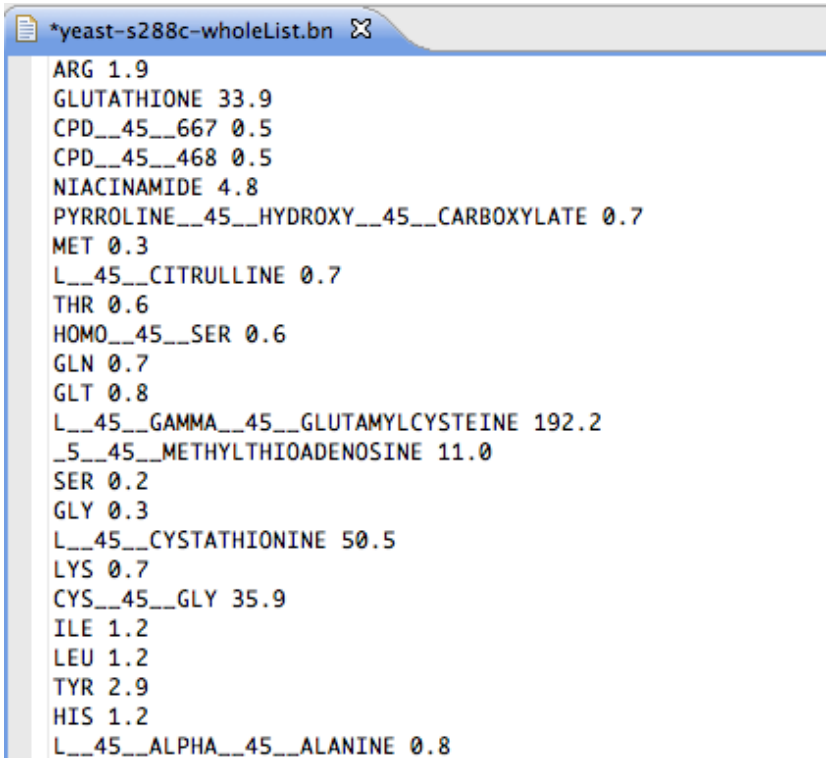


Figure 6.6: Example of the input file for defining the black nodes for the metabolic stories enumeration.

As an example of a black nodes input file, we present Figure 6.6 that shows the input file containing the whole list of black nodes used in our first biological application, described in Section 3.6. Suppose that this file is called *yeast-s288c.bn* and that a SBML metabolic network input file called *yeast-s288c.xml* is also available. Suppose also that the *Gobbolino* library is packaged in a *gobbolino.jar* Java file. In this case, a command-line to compute the compressed network is: *java -jar gobbolino.jar -mode=BNS -n="yeast-s288c"* and it will generate, in the input folder, a file named *yeast-s288c-LP-S.NEL*. In order to perform the same experiment we have described in Section 3.6, the command-line is: *java -jar gobbolino.jar -n="yeast-s288c-LP-S" -o="stories-yeast-s288c-wholeList.sto" -q=500000 -score=METABOLOMICS -scoreFile="yeast-s288c.bn" -stop=TIME -stopTime=1.*

Project URL

The *Gobbolino* source code is available at the following URL: *http://gobbolino.gforge.inria.fr.*

TITRE en français

Énumération de Sous-Structures Fonctionnelle dans des Réseaux Métaboliques Complets: Histoires Métaboliques, Précurseurs et Organisations Chimiques

RÉSUMÉ en français

Dans cette thèse, nous avons présenté trois méthodes différentes pour l'énumération de sous-réseaux particuliers d'un réseau métabolique: les histoires métaboliques, les ensembles minimaux de précurseurs et les organisations chimiques. Pour chacune de ces trois méthodes, nous avons présenté des résultats théoriques, et pour les deux premières, nous avons en outre fourni une illustration sur comment les appliquer afin d'étudier le comportement métabolique des organismes vivants. Les histoires métaboliques sont définies comme des graphes acycliques dirigés maximaux dont les ensembles de sources et de cibles sont limités à un sous-ensemble des nœuds. La motivation initiale de cette définition était d'analyser des données expérimentales de métabolomique, mais la méthode a également été explorée dans un contexte différent. Les ensembles de précurseurs métaboliques sont des ensembles minimaux de nutriments qui permettent de produire des métabolites d'intérêt. Nous présentons trois méthodes différentes pour l'énumération de tels ensembles minimaux de précurseurs, et nous illustrons leur application dans une étude des échanges métaboliques dans un système symbiotique. Les organisations chimiques sont des ensembles de métabolites qui à la fois sont fermés et s'auto-maintiennent, ce qui reflète des caractéristiques de stabilité dans le sens où aucun nouveau métabolite ne peut être produit et qu'aucun des métabolites déjà présents dans le système ne peut disparaître.

MOTS-CLEFS en français
algorithme; complexité; graphes; exact exponentiel; randomisation; biologie computationelle; symbiose; métabolisme; réseau métabolique; énumération; ensemble d'arcs de rétroaction; graphe acyclique dirigé maximal; histoire métabolique; ensemble de précurseurs; organisation chimique

TITRE en anglais

Enumerating Functional Substructures of Genome-Scale Metabolic Networks: Stories, Precursors and Organisations

RÉSUMÉ en anglais

In this thesis, we presented three different methods for enumerating special subnetworks contained in a metabolic network: metabolic stories, minimal precursor sets and chemical organisations. For each of the three methods, we gave theoretical results, and for the two first ones, we further provided an illustration on how to apply them in order to study the metabolic behaviour of living organisms. Metabolic stories are defined as maximal directed acyclic graphs whose sets of sources and targets are restricted to a subset of the nodes. The initial motivation of this definition was to analyse metabolomics experimental data, but the method was also explored in a different context. Metabolic precursor sets are minimal sets of nutrients that are able to produce metabolites of interest. We present three different methods for enumerating minimal precursor sets and we illustrate the application in a study of the metabolic exchanges in a symbiotic system. Chemical organisations are sets of metabolites that are simultaneously closed and self-maintaining, which captures some stability feature in the sense that no new metabolite may be produced and that none of the present metabolites vanishes.

MOTS-CLEFS en anglais

algorithm; complexity; graphs; exact exponential; randomisation; computational biology; symbiosis; metabolism; metabolic network; graph; enumeration; feedback arc set; maximal directed acyclic graph; metabolic story; precursor set; chemical organisation