Models and algorithms for metabolic networks: elementary modes and precursor sets

Vicente Acuña

N^o d'ordre: 218-2008 Année 2009

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
4 Juin 2010

par

Vicente Acuña

Models and algorithms for metabolic networks: elementary modes and precursor sets

---

Directrice de thèse: Marie-France Sagot
Co-directeur de thèse: Christian Gautier

Jury:	Pierluigi Crescenzi,	Rapporteur
	Khaled Elbassioni,	Rapporteur
	Christian Gautier,	Directeur
	Dominique Perrin,	Examinateur
	Marie-France Sagot,	Directrice
	Alain Viari,	Président

UNIVERSITÉ CLAUDE BERNARD-LYON 1

Président de l'Université	**M. le Professeur L. COLLET**
Vice-Président du Conseil Scientifique	M. le Professeur J. F. MORNEX
Vice-Président du Conseil d'Administration	M. le Professeur G. ANNAT
Vice-Président du Conseil des Etudes et de la Vie Universitaire	M. le Professeur D. SIMON
Secrétaire Général	M. G. GAY

COMPOSANTES SANTÉ

UFR de Médecine Lyon-Est – Claude Bernard	Directeur: M. le Professeur J. ETIENNE
UFR de Médecine Lyon-Sud – Charles Mérieux	Directeur: M. le Professeur F-N. GILLY
UFR d'Ontologie	Directeur: M. le professeur D. BOURGEOIS
Institut des Sciences Pharmaceutiques et Biologiques	Directeur: M. le Professeur F. LOCHER
Institut des Sciences et 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

COMPOSANTES SCIENCES ET TECHNOLOGIE

Faculté des Sciences et Technologies	Directeur: M. le Professeur F. GIERES
UFR Sciences et Techniques des Activités Physiques et Sportives	Directeur: M. C. Collignon
Observatoire de Lyon	Directeur: M. B. Guiderdoni
Institut des Sciences et des Techniques de l'Ingénieur de Lyon	Directeur: M. le Professeur J. LIETO
Institut Universitaire de Technologie A	Directeur: M. le Professeur C. COULET
Institut Universitaire de Technologie B	Directeur: M. le Professeur R. LAMARTINE
Institut de Science Financière et d'Assurances	Directeur: M. le Professeur J-C. AUGROS

Abstract

In this PhD, we present some algorithms and complexity results for two general problems that arise in the analysis of a metabolic network: the search for elementary modes of a network and the search for minimal precursors sets.

Elementary modes is a common tool in the study of the cellular characteristic of a metabolic network. An elementary mode can be seen as a minimal set of reactions that can work in steady state independently of the rest of the network. It has therefore served as a mathematical model for the possible metabolic pathways of a cell. Their computation is not trivial and poses computational challenges. We show that some problems, like checking consistency of a network, finding one elementary mode or checking that a set of reactions constitutes a cut are easy problems, giving polynomial algorithms based on LP formulations. We also prove the hardness of central problems like finding a minimum size elementary mode, finding an elementary mode containing two given reactions, counting the number of elementary modes or finding a minimum reaction cut. On the enumeration problem, we show that enumerating all reactions containing one given reaction cannot be done in polynomial total time unless P=NP. This result provides some idea about the complexity of enumerating all the elementary modes.

The search for precursor sets is motivated by discovering which external metabolites are sufficient to allow the production of a given set of target metabolites. In contrast with previous proposals, we present a new approach which is the first to formally consider the use of cycles in the way to produce the target. We present a polynomial algorithm to decide whether a set is a precursor set of a given target. We also show that, given a target set, finding a minimal precursor set is easy but finding a precursor set of minimum size is NP-hard. We further show that finding a solution with minimum size internal supply is NP-hard. We give a simple characterisation of precursors sets by the existence of hyperpaths between the solutions and the target. If we consider the enumeration of all the minimal precursor sets of a given target, we find that this problem cannot be solved in polynomial total time unless P=NP. Despite this result, we present two algorithms that have good performance for medium-size networks.

Contents

Introduction

The principal aim of this work is a computational study of two problems appearing in the research on metabolism. The model used to address these problems considers the set of all biochemical reactions which can occur in a given organism. The links between those reactions and the compounds (or metabolites) that are used/produced by such reactions constitute the *metabolic network* of the organism. Modelling the whole set of reactions represents a more general approach in comparison to the research that had been done until now on metabolism. Indeed, traditionally biologists and early computational biologists generally concentrated attention on the the study of only a subset of reactions which are able to perform a given task of the cell. This gave rise to the informally defined concept of *metabolic pathways*. Despite its great interest, this approach may fail to explain some more complex behaviours, that emerge only when the system is analysed *as a whole*.

In fact, the metabolic network of an organism defined as its set of reactions and compounds correspond to one level of the complex network of interactions that occur in metabolism. Of course, analysing metabolism only at this level without regard to regulation and other types of relations may be insufficient to describe the mechanisms governing the real processes. Indeed, various approaches in what is called systems biology intend to predict the behaviour of metabolism by modelling all kinds of interactions occurring in it. There are some reasons however that justify our choice of a simpler model without considering such other interactions:

- Insufficient knowledge of the process and lack of data available for other kinds of interactions make difficult to test the accuracy of more complex models.

- A simple model allows a deeper mathematical theory than considering more heterogeneous classes of relations.

- The suitability of a model to predict the behaviour of metabolism will strongly

depend on the biological question posed.

- The problems and solutions proposed can be considered as stepping stones in more general and ambitious analysis that integrates other kinds of data.

Considering thus the metabolic network of an organism as a set of reactions and metabolites, we focused our attention on two problems defined on such set:

- the search for *elementary modes* that correspond to minimal sets of reactions which can work together in steady state independently of the rest of the reactions; and

- the search for *precursors sets*, which are sets of external source compounds whose presence allows the organism to produce a given target metabolite.

The first concept is commonly used in the analysis of metabolism and has indeed been proposed as a formal definition of metabolic pathway. Several algorithms to enumerate elementary modes could be found in the literature but despite the popularity of the problem, no systematic analysis of its complexity had been done until now. For the second problem, we introduce a new approach which is the first that formally considers cycles in the production of the target compounds by the precursor sets. We then give some useful alternative mathematical characterisations and we analyse the time complexity of finding and enumerating precursor sets.

Analysing the time complexity of these problems gives us a notion about the limitations of the algorithms we can aspire to obtain. This serves as a guide on how to continue approaching the problem. For instance, we can know if there is some "hope" of having considerably faster algorithms which find exact solutions or at least a good approximation of them. Or if exact or good solutions were provably hard to find, this would give a good justification to try the use of heuristics.

We show that some of the problems are easy to solve, providing in such cases a polynomial algorithm. Despite the simplicity of the models used, we show that some basic questions are already computationally hard to solve, meaning that finding exact solutions can only be done for small networks under currently believed complexity assumptions. We also show that despite this hardness, in some cases the theoretically slow algorithms run in a reasonable time for real metabolic networks.

We briefly introduce below the two topics covered in this work, including their biological motivation and a summary of our contributions.

Finding and enumerating elementary modes

Studying the dynamics of metabolic networks is usually performed using models based on differential equations whereas structural analyses are mainly based on graph-related formalisms or, as far as metabolism is concerned, on a constraint-based modelling. The latter term is commonly employed in the bioinformatics community following two papers by Palsson (Palsson, 2000; Covert et Palsson, 2003). In the constraint-based framework, the network may still be modelled as an edge-labelled hypergraph, but several types of constraints (stoichiometric, thermodynamic and in some cases regulatory) are added to restrict the possible fluxes through the network. The choice of a particular model heavily depends on the type of question one wishes to address (structural or dynamic) but also on the type of data that is available (qualitative or quantitative). Another type of criterion that may be taken into account is the computational cost of a given analysis, and therefore its scalability to large datasets (such as genome-scale metabolic networks).

In a constraint-based approach, only admissible flux distributions are of interest. An admissible flux distribution corresponds to a set of reactions, which, when taken together in given proportions, perform the transformation of available substrates into removable products with the special property that all intermediate compounds are balanced (steady-state assumption) and irreversible reactions are taken in the appropriate direction (thermodynamic constraint). Such an admissible flux distribution is called a mode.

Even though each mode is potentially interesting, not all of them are generally considered. Classically, two major sub-problems have been introduced. The first one is known as flux balance analysis. It consists in searching for a mode that optimises a given objective function. Examples of objective functions include biomass (usually represented as a pseudo-reaction of the network, in general determined from experimental data) or ATP production. This optimisation problem has several applications (Edwards *et al.*, 2001; Fong et Palsson, 2004) and can be solved using linear programming (LP).

The second sub-problem is the one we discuss in this thesis. In the case where no particular function is to be optimised, all modes are equally interesting. A sensible strategy is then to try to find a set that could generate them all. Such a generating set has been proposed and called the set of elementary modes (Schuster et Hilgetag, 1994), EM for short. Intuitively, an elementary mode is a special mode that has the property of not containing any other mode.

Elementary modes have been said to represent a formalised definition of a biological pathway. Indeed, a biological interpretation can be given to such flux vectors: a mode is a set of enzymes that operate together at steady state (Schuster *et al.*, 2000) and a mode is elementary when the removal of one enzyme causes it to fail.

Another concept we study here is closely related to the notion of elementary mode. This is the concept of a reaction cut set, recently introduced in Klamt et Gilles (2004). In order to avoid any confusion with other types of cuts in graphs or hypergraphs that may be found in the literature (see e.g. Seymour, 1977), we explicitly choose here to use the term *reaction* cut. An elementary mode may be seen as a set of reactions that, when used together, perform a given task while a minimal reaction cut set is a set of reactions one needs to inhibit to prevent a given task, also called *target reaction*, from being performed. As mentioned in Klamt (2006), the task to be silenced can be a combination of reactions. Reaction cut sets have been operationally defined as corresponding to a set of reactions whose deletion from the network stops each elementary mode that contains the target reaction(s).

Previous Work

Surprisingly few results have been established on the complexity of problems concerning detection, counting and enumeration of elementary modes. In Klamt et Stelling (2002), the authors mainly focus on finding an upper bound on the number of elementary modes.

In fact, as mentioned in Fukuda et Prodon (1996), the complexity of the general problem, that is, given a description of a cone (or polytope) in terms of its facets (inequalities), find a description in terms of (enumerate all) its extreme rays (vertices) as a function of the length of the output (number of rays or vertices), is a long-standing open question in computational geometry.

Summary of the main results

The main contribution of this part of our work is in giving a systematic overview of the complexity of optimisation problems related to modes. We first establish results regarding network consistency (Section 4.2). Most consistency problems can be solved in polynomial time (are easy). Most, if not all, of these results have been stated before in the literature. It is in fact easy to formulate these problems as LP-problems, which has the side advantage that computer packages are available to solve them.

We then establish the complexity of finding and enumerating elementary modes

(Sections 4.3 and 4.4). We show in particular that finding one elementary mode is easy but that this task becomes hard when an elementary mode containing two specified reactions is sought. We also examine a number of EM related problems and establish their complexity. We emphasize that the easy problems can be solved by existing software.

Other complexity results show that it is not possible to generate, in polynomial total time, all elementary modes that pass through a given reaction unless P=NP. This result can have biotechnological relevance. Indeed, by knocking-out enzymes and analysing the effect this has on metabolic behaviour, one can identify whether and where a metabolic network is robust or fragile, and ultimately arrive at a better understanding of cellular phenotypes and of their link with the genotype. Enumerating all elementary modes that pass through a given reaction would thus allow determining all possible steady-state behaviours this reaction enables to block.

We also analyse the computational complexity of problems concerning reaction cuts. We prove that finding a *minimum* reaction cut set, one that contains a minimum number of reactions, is hard.

Finding and enumerating precursor sets

The metabolic capacities of an organism are directly defined by the set of its possible biochemical reactions. Once the metabolic network of an organism has been defined, the question of how are produced the essential metabolites for the organism arises. In particular, it is important to know which are the metabolites that the organism needs to obtain from its environment to produce those essential metabolites.

One way to answer this question is to manually inspect the metabolic pathways defined as present in the organism. This is determined by comparing the set of reactions of a reconstructed metabolic network with the set of reactions in the reference metabolic pathways contained in metabolic databases such as METACYC from the BIOCYC database collection (Caspi *et al.*, 2006) or KEGG (Kanehisa *et al.*, 2006). However, each reference metabolic pathway represents a very small part of the whole network and does not consider what occurs upstream of the pathway, nor whether some alternative organism-specific pathways exist.

With the aim to better understand the relationship between an organism and its environment, we propose a model that allows determining which nutrients an organism requires from its environment in order to produce some metabolic compounds that are

essential for its survival.

In particular, the environment could be another living system. This is the case of what is called endosymbiosis where an organism lives within the body or cells of another organism. This concerns mostly bacteria that may inhabit their host intra- or extra-cellularly. Even in the intra-cellular case, a host often accomodates more than one bacterium so that the environment of an endosymbiont is the cell of its host, but also other endosymbionts that may share among them metabolic functions very much like members of a family (should) share chores.

This understanding is in turn important for getting a clearer comprehension of the genotype-phenotype relationship, and ultimately also of the origin of life. Indeed, as is now accepted, one particular endosymbiotic bacterium is the ancestor of an organelle, the mitochondrion, while another evolved into the chloroplast of plants. Understanding how these essential events took place in the course of evolution may be greatly helped by understanding the metabolic dialog between an organism and its living or inanimate environment, what is exchanged between the different partners and how this exchange may evolve through time.

Exploring all the possible ways of producing some metabolic compounds from external metabolites may be important also for bioengineering purposes, where the goal is in this case to directly manipulate the environment so that the controlled organism produces what is desired.

Previous work

The capacity to make progress on these questions has been boosted tremendously in the last few years, and this trend will only intensify in the coming decades, by the huge number of sequencing projects, and by our greater ability to now reconstruct whole metabolic networks from such genomic data (see Francke *et al.*, 2005, for an overview of such metabolic data reconstruction processes). Surprisingly though, the problem of systematically determining all possible routes from a set of externally acquired metabolites (*sources*) to the internally produced compounds of interest, henceforth called *targets*, has not attracted a big literature as is the case for other bioinformatics questions. The issue was first addressed with the purpose to improve the whole network reconstructions by Romero and Karp in the early years of this century (Romero et Karp, 2001), but it is only in 2008 that the first two papers on identifying the sets of metabolites required by an organism to produce some target sets appeared, one by Handorf *et al.* (Handorf *et al.*, 2007) that relied on a previous related work (Handorf *et al.*, 2005),

and our own that was presented at WABI in 2008 (Cottret *et al.*, 2008).

Summary of the main results

An important issue that was not formally treated in the papers of either Romero *et al.* or Handorf *et al.* concerned internal cycles in the network. How some of those cycles could be fired off or maintained was not clear, and indeed the problem was addressed in an ad-hoc manner. One of the main contributions of our work is to mathematically address this problem considering cyclic production of some metabolites, which can have also biological sense.

We further address various important complexity questions related to the search for these precursor sets. This concerns in particular the complexity of enumerating all the minimal ones, for which we provide a proof that the problem is indeed hard, in fact it cannot be solved in polynomial total time unless $P = NP$. We however also show an intriguing relationship with the problem of enumerating the minimal *precursor cut sets* that can be considered dual of this one and, using a previous elegant result of Gurvich and Khachiyan, prove that the two problems taken together can be solved in quasy-polynomial total time.

Despite its theoretical hardness, two algorithms are proposed for enumerating all minimal set of precursor sets of a given target. Both algorithms are based on the enumeration of paths on the hypergraph or *hyperpaths* backtracking from the target to the source. This way to travel the hypergraph defines the *replacement tree* of the network, which was just alluded to in Cottret *et al.* (2008). The first algorithm formalises the manipulation that was done to this structure avoiding to maintaining in memory more than a branch of it. Moreover, this algorithm performs some important ideas to prune this structure and thus economise on space and on the number of operations without loosing any solution. A second algorithm effects some manipulation and transformation directly on the input network to go much further on the number of redundant operations that is saved. Experimental results show that, in comparison with the previous version (Cottret *et al.*, 2008), the gain in space and in time can be dramatic.

Structure of the manuscript

Chapter 1 gives some of basic mathematical concepts used in the model. It includes a very succinct introduction to graphs and hypergraphs which are used all along the

manuscript. It also introduces the concept of hitting set which appears recurrently in the problems studied. Finally monotone boolean functions are presented which are also important in some of the theoretical results on complexity.

Chapter 2 is dedicated to introduce some basic concepts of computational complexity. This is restricted only to the *time* complexity, which is the main interest in our work. In particular, we give some definitions to properly measure time complexity when we want to enumerate all the solutions of certain problems and the number of such solutions can be exponential in the size of the instance.

Chapter 3 presents a brief summary of the biological concepts concerning the metabolism of the cell. We also present some ideas to model metabolic networks from whole set of reactions and metabolites. The models presented in this chapter are used to model the two problems presented in this PhD.

Chapter 4 deals with problems related to the search for elementary modes and reaction cuts. We present algorithms and analyse the complexity of finding and enumerating these minimal sets of reactions, and present a relation between this problem and a fundamental open question in computational geometry.

Chapter 5 presents our new approach to model precursor sets. It gives different characterisations of such sets and shows the relation with hyperpaths. We also define the related concept of precursor cuts. Finally, in this chapter, we present also our results regarding the time complexity of finding and enumerating precursor sets and precursor cuts of a given target.

Chapter 6 presents two methods to enumerate all the precursor sets of a given target. This includes the algorithmic concept of a replacement tree, which is the main structure used to enumerate the solutions. A brief analysis of the performance of such enumeration is presented as well as some extensions to related problems.

Chapter 1

Some Basic Mathematical Definitions

Contents

This chapter presents a concise introduction to the mathematical concepts used in this work. It does not pretend to be an exhaustive introduction to the topics raised but provides instead the minimum necessary definitions to understand the models presented in the following chapters. For a more detailed description and further theory, see the references given in each section.

1.1 Graphs, digraphs and hypergraphs

In this section, we give a short introduction to graph theory and its extension to hypergraphs. For a more extensive theory, see Diestel (2006), Bang-Jensen et Gutin (2008) and Gallo *et al.* (1993).

1.1.1 Graph and digraph definitions

A *graph* G is a pair $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a finite set and $\mathcal{E}$ is a set whose elements are subsets of $\mathcal{V}$ of cardinality two, that is, $\mathcal{E} \subseteq \mathcal{P}_2(\mathcal{V})$. The elements of $\mathcal{V}$ are called *vertices* or *nodes*, and the elements of E are called *edges*.

If $G = (\mathcal{V}, \mathcal{E})$ is a graph and $e = \{v_1, v_2\} \in \mathcal{E}$, then we say that v_1 and v_2 are the *ends* of e. The *degree* of a vertex v is the number of edges having v as an end.

A *directed graph* or *digraph* is a a pair $D = (\mathcal{V}, \mathcal{A})$ such that $\mathcal{V}$ is a finite set and $\mathcal{A}$ is a set of *ordered* pairs of different elements of $\mathcal{V}$, that is, $\mathcal{A} \subseteq \{(u, v) \in \mathcal{V} \times \mathcal{V} \mid u \neq v\}$. The elements of $\mathcal{V}$ are also called *vertices*, and the ordered pair of vertices in $\mathcal{A}$ are called *arcs*.

If $D = (\mathcal{V}, \mathcal{A})$ is a digraph and $a = (v_1, v_2) \in \mathcal{A}$, then we say that a is an arc *from* v_1 *to* v_2. The vertices v_1 and v_2 are called, respectively, the *tail* and the *head* of a, and both are called the *ends* of e. The *indegree* of a vertex v is the number of arcs having v as head, and the *outdegree* of v is the number of arcs having v as tail.

1.1.2 Graphical representation and labels

The sets $\mathcal{V}$ and $\mathcal{E}$ (or $\mathcal{A}$) represent the basic structure that defines G (or D). In general, this topology can be graphically represented by the set of elements in $\mathcal{V}$ connected by lines (or arrows for digraphs) between the related pairs (see Figure 1.1 cases (a) and (b)).

Depending on what we are modelling, we can also add some extra information to the vertices or edges (arcs) of G (D). This information is included as functions called *labels*. For instance, vertices can have *colours* depending on some classification given to them, and edges/arcs can have *weights* or *costs* associated to them.

1.1.3 Walks, paths, cycles and hamiltonian cycles

A *walk* in a graph $G = (\mathcal{V}, \mathcal{E})$ is a sequence of vertices $p = (v_1, v_2, \ldots, v_k)$ with $k \geq 1$, such that $[v_j, v_{j+1}] \in \mathcal{E}$ for $j = 1, \ldots, k-1$. The walk is *closed* if $k > 1$ and $v_1 = v_k$. A walk without any repeated nodes in it is called a *path*. A closed walk with no repeated nodes other than its first and last ones is called a *circuit* or *cycle*.

A *directed walk* in a digraph $D = (\mathcal{V}, \mathcal{A})$ is a sequence of vertices $p = (v_1, v_2, \ldots, v_k)$ with $k \geq 1$, such that $(v_j, v_{j+1}) \in \mathcal{E}$ for $j = 1, \ldots, k-1$. A directed walk is *closed* if $k > 1$ and $v_1 = v_k$. A directed walk without any repeated nodes in it is called a

directed path. A closed walk with no repeated nodes other than its first and last one is called a *directed circuit* or *directed cycle*.

A *hamiltonian* (directed) cycle is a (directed) cycle that visits each vertex (see Figure 1.1 (a)).

In general, we can also identify paths and cycles by the set of edges (or arcs) connecting the successive vertices they contain.

1.1.4 Adjacency and incidence matrix

For a given order of the elements of $\mathcal{V} = \{v_1, \ldots, v_n\}$ and of $\mathcal{E} = \{e_1, \ldots, e_m\}$, a graph $G = (\mathcal{V}, \mathcal{E})$ can be represented by its *adjacency matrix* $M \in \mathcal{R}^{n \times n}$. This matrix is such that $M_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $M_{ij} = 0$ otherwise. Hence M is symmetric. We can define in the same way the adjacency matrix of a digraph, although in this case it is not necessarily symmetric.

A graph G can also be represented by the *incidence matrix* $I \in \mathcal{R}^{n \times m}$. This matrix is such that $I_{ij} = 1$ if v_i is an end of e_j and $I_{ij} = 0$ otherwise. In a similar way, we define the incidence matrix $I \in \mathcal{R}^{n \times m}$ of a digraph D as a matrix such that $I_{ij} = -1$ if v_i is the tail of a_j, $I_{ij} = 1$ if v_i is the head of a_j and $I_{ij} = 0$ otherwise.

1.1.5 Induced subgraph, bipartite graphs and trees

Given two graphs $G = (\mathcal{V}, \mathcal{E})$ and $G' = (\mathcal{V}', \mathcal{E}')$, we say that G' is a *subgraph* of G if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. We say that G' is an *induced subgraph* of G if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}'$ contains all the edges of $\mathcal{E}$ connecting vertices of $\mathcal{V}'$.

A graph $G = (\mathcal{V}, \mathcal{E})$ is called a *bipartite* graph if the set of vertices $\mathcal{V}$ can be partitioned into two sets U and W, and each edge in $\mathcal{E}$ has one end in U and the other end in W (see Figure 1.1 (c)). It is easy to see that a graph is bipartite if and only if it has no cycle of odd length. A *matching* of a bipartite graph is a subset of edges $E \subseteq \mathcal{E}$ such that any pair of edges in E have no ends in common.

Another interesting class of graphs is the class of *trees*. A graph is *connected* if for any two nodes in it there is a path between them. A graph $T = (\mathcal{V}, \mathcal{E})$ is a *tree* if it is a connected graph without cycles (see Figure 1.1 (b)). Any vertex of T with degree one is called a *leaf*. Sometimes we define a vertex of the tree as the *root* of the tree, indicating that there is a relevant meaning for the paths between this node and the leaves of the tree. In this case, we say that T is rooted. Each vertex other than the root has as parent the next vertex on the path to the root. A child of a vertex u is a

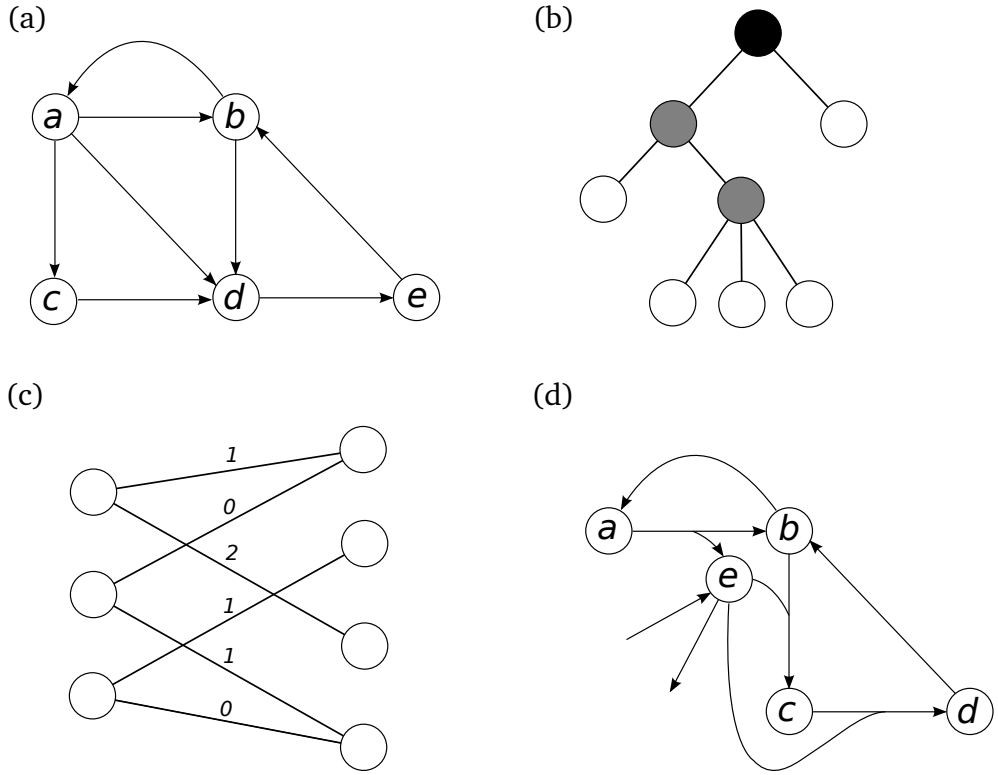

Figure 1.1: Examples of graphs, digraphs and hypergraphs. (a) A digraph containing the hamiltonian directed cycle (a, c, d, e, b, a). (b) A tree coloured having five leaves. If it is rooted at the black vertex, the depth of the tree is three. (c) A bipartite graph where edges are labelled by weights. Edges with weight one forms a matching. (d) A directed hypergraph with hyperarcs $(\emptyset, \{e\}), (\{e\}, \emptyset), (\{a\}, \{b, e\}), (\{b\}, \{a\}), (\{b, e\}, \{c\}), (\{c, e\}, \{d\})$ and $(\{d\}, \{b\})$.

vertex v such that v is the parent of u. Therefore, leaves are the only vertices without children. The depth of the tree is the length of the longest path between a leaf and the root.

1.1.6 Directed hypergraphs

We present a generalisation of the definition of digraphs to the case where arcs, here called *hyperarcs*, are defined from a set of vertices to another set of vertices.

A *directed hypergraph* H is a pair $H = (\mathcal{C}, \mathcal{R})$, where $\mathcal{C}$ is a finite set of *vertices* and $\mathcal{R} \subseteq \mathcal{P}(C) \times \mathcal{P}(C)$ is a set of *hyperarcs*. Each hyperarc $r \in \mathcal{R}$ is an ordered pair of disjoint sets $r = (Tail(r), Head(r))$, both subsets of $\mathcal{C}$ (see Figure 1.1 (d) for a graphical representation). For any hyperarc r, one of the sets $Tail(r)$ or $Head(r)$ can be empty but not both. For a given order of the elements of $\mathcal{C} = \{c_1, \ldots, c_n\}$ and of

$\mathcal{R} = \{r_1, \ldots, r_m\}$, a hypergraph $H = (\mathcal{C}, \mathcal{R})$ can be represented by its *incidence matrix* $I \in \mathcal{R}^{n \times m}$. This matrix is such that $I_{ij} = -1$ if $c_i \in Tail(r_j)$, $I_{ij} = 1$ if $c_i \in Head(r_j)$ and $I_{ij} = 0$ otherwise.

1.2 Hitting set

A concept very often used in this work is that of the hitting set of a collection. Given a finite set $\mathcal{U}$ and a collection $\mathcal{I} = \{I_1, \ldots, I_n\}$ of subsets of $\mathcal{U}$ (that is, $I_i \subseteq \mathcal{U}$ for all $i = 1, \ldots, n$), we say that a set $H \subseteq \mathcal{U}$ is a *hitting set* of $\mathcal{I}$ if $I_i \cap H \neq \emptyset$ for all $i = 1, \ldots, n$. In other words if H intersects all the sets in the collection.

The set H is a *minimal* hitting set of $\mathcal{I}$ if H is a hitting set of $\mathcal{I}$ and for any $H' \subset H$, H' is not a hitting set of $\mathcal{I}$ (H does not contain other hitting sets of $\mathcal{I}$). There is an interesting property of symmetry between the collection $\mathcal{I}$ and the collection of all minimal hitting sets of $\mathcal{I}$ (see e.g. Berge, 1989).

Property 1.1. *Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a collection of subsets where I_i is not a subset of I_j for any $i \neq j$ (no set of $\mathcal{I}$ is contained in another set of $\mathcal{I}$). Let $\mathcal{H}$ be the collection of all minimal hitting sets of the collection $\mathcal{I}$. Then, $\mathcal{I}$ is the collection of all minimal hitting sets of $\mathcal{H}$.*

Proof. Let $I \in \mathcal{I}$. We show that I is a hitting set of the collection $\mathcal{H}$. Let H be in $\mathcal{H}$. Since H is a hitting set of $\mathcal{I}$, H and I are not disjoint sets. Therefore, I is a hitting set of $\mathcal{H}$.

For the minimality, take I' a subset of I. By hypothesis, for any $i \in \{1, \ldots, n\}$ the set I_i of the collection is not contained in I' and therefore $I_i \setminus I'$ is not empty. Consider the set $H' = \cup_{i \in \{1,\ldots,n\}} I_i \setminus I'$. Clearly H' is a hitting set of $\mathcal{I}$. Therefore, there exists $H'' \subseteq H'$ in the collection $\mathcal{H}$. Since I' does not intersect H'', we conclude that I' is not a hitting set of $\mathcal{H}$ and then I is a *minimal* hitting set.

Now we show that any minimal hitting set of $\mathcal{H}$ belongs to $\mathcal{I}$. Let J be a minimal hitting set of $\mathcal{H}$. Suppose, by contradiction, that J is not in $\mathcal{I}$. In this case J do not contain any I_i (because J and I_i are both minimal hitting sets of $\mathcal{H}$). The set $I_i \setminus J$ is not empty for any $i \in \{1, \ldots, n\}$. The set $H' = \cup_{i \in \{1,\ldots,n\}} I_i \setminus J$ is a hitting set of $\mathcal{I}$. Therefore, there exists $H'' \subseteq H'$ in the collection $\mathcal{H}$. Since J does not intersect H'', we conclude that J is not a hitting set of $\mathcal{H}$ which is a contradiction. We conclude that J is a set of the collection $\mathcal{I}$. □

1.3 Boolean functions

A *Boolean function* is a function of the form $f : \{0,1\}^k \to \{0,1\}$, where k is a positive integer called the *arity* of f. For example, the function $f_1 : \{0,1\}^3 \to \{0,1\}$ defined by $f_1(x) = x_1(1 - x_2) + x_2x_3$ is Boolean. Alternatively, the set $\{0,1\}$ is also represented by the set {FALSE, TRUE}.

A *propositional formula* or *Boolean expression* in k Boolean variables is a well-formed expression that uses: *variables* $x_1, \ldots, x_k$, *conjunctions* (represented by AND or $\wedge$), *disjunctions* (represented by OR or $\vee$) and *negations* (represented by NOT or $\neg$) and parentheses. For instance, the following are Boolean expressions:

(i) $(x_1 \wedge \neg x_2) \vee (x_2 \wedge x_3)$

(ii) $(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3)$

(iii) $\neg(\neg x_1 \vee x_2) \vee (x_2 \wedge x_3)$

Every Boolean function of arity k can be expressed as a propositional formula in k variables (for instance, the Boolean function f_1 defined above, can be expressed as the propositional formula (i)). Two Boolean expressions are *logically equivalent* if and only if they express the same Boolean function (in the example expressions (i), (ii) and (iii) are all equivalent). We say that a propositional formula is *satisfiable* if there exists an assignment of TRUE and FALSE values to the variables such that the propositional formula evaluates TRUE. For instance, the expressions (i), (ii) and (iii) are satisfiable for $x_2 = x_3 =$ TRUE.

Given a propositional formula, we define some of its parts. A *literal* is either a variable or the negation of a variable. A *clause* is a disjunction of literals and a *term* is a conjunction of literals. For instance, expression (iii) contains the clause: $\neg x_1 \vee x_2$ and the term $x_2 \wedge x_3$.

We say that a propositional formula is in *conjunctive normal form* (CNF) if it is a conjunction of clauses. An important result states that any Boolean function can be expressed as a logically equivalent CNF (for instance, expression (ii) is a CNF of function f_1). We say that a propositional formula is in *disjunctive normal form* (DNF) if it is a disjunction of terms. Again, any Boolean function can be expressed as a logically equivalent DNF (for instance, expression (i) is a DNF of function f_1).

1.3.1 Monotone Boolean functions

We say that a Boolean function $f : \{0,1\}^k \to \{0,1\}$ is *monotone* if for any pair $x, y \in \{0,1\}^k$ such that $x_i \leq y_i$ for all $i \in \{1,\ldots,k\}$ we have that $f(x) \leq f(y)$.

For example the function $f_2(x) : \{0,1\}^3 \to \{0,1\}$ defined by

$$f_2(x) = \begin{cases} 1 & \text{if } x_1(x_1 + x_2 + x_3) > 1 \\ 0 & \text{otherwise} \end{cases}$$

is a monotone Boolean function.

A propositional formula is called a $\wedge, \vee$-*formula* if it does not contain any negation. Any monotone Boolean function can be expressed as a $\wedge, \vee$-*formula*, that is, it can be expressed using variables, conjunctions, disjunctions and parentheses. For instance, f_2 can be expressed by

$$f_2(x) = (x_1 \wedge x_2) \vee (x_1 \wedge x_3).$$

Let f be a monotone Boolean function. A *prime implicant* of f is a minimal set of variables such that if they are all TRUE then f has to be also TRUE. A *prime implicate* of f is a minimal set of variables such that if they are all FALSE then f has to be also FALSE. In the example, $\{x_1, x_2\}$ and $\{x_1, x_3\}$ are the prime implicants of f_2; and the sets $\{x_1\}$ and $\{x_2, x_3\}$ are the prime implicates of f_2.

Any monotone Boolean function f can be expressed as a CNF where the clauses correspond exactly to the set of prime *implicants*. Any monotone Boolean function f can also be expressed as a DNF where the terms correspond exactly to the set of prime *implicates*. In the example, f_2 can be expressed in CNF and DNF:

$$f_2(x) = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) = x_1 \wedge (x_2 \vee x_3).$$

Finally, it is easy to see that a prime implicate has to contain at least one variable of each prime implicant. In other words, a prime implicate is a minimal hitting set of the collection of prime implicates. Considering Property 1.1, we have the following relation:

Property 1.2. *Given a Boolean function f, the collection of its prime implicants corresponds to the minimal hitting sets of its prime implicates, and vice versa.*

Chapter 2

Basic Concepts of Time Complexity Analysis

Contents

In this chapter, we introduce the reader to some basic concepts in the theory of computational complexity. We restrict the analysis to *time complexity*, although similar concepts are also defined for *space complexity*. For further theory, see Papadimitriou (1994); Ausiello *et al.* (1999).

2.1 Defining a problem

We define a *problem* as a relation $\mathcal{P} \subseteq I_{\mathcal{P}} \times S_{\mathcal{P}}$, where $I_{\mathcal{P}}$ is the set of *problem instances* and $S_{\mathcal{P}}$ is the set of *problem solutions*. The set of instances represents the particular cases of the generic problem. If $(x, y) \in \mathcal{P}$, we say that y *is a solution of* x.

For a given instance $x \in I_{\mathcal{P}}$ of a problem $\mathcal{P}$, we are interested in finding one or several solutions of x. We can classify a problem depending on the characteristics of the sets $I_{\mathcal{P}}$ and $S_{\mathcal{P}}$ and depending on which solutions we want to find.

2.1.1 Decision problems

A *decision problem* is a problem $\mathcal{P}$ such that:

1. $S_{\mathcal{P}} = \{\text{NO}, \text{YES}\}$ (or $S_{\mathcal{P}} = \{0, 1\}$)
2. For each $x \in I_{\mathcal{P}}$ there exists one and only one $y \in \{\text{NO}, \text{YES}\}$ such that $(x, y) \in \mathcal{P}$

That is, in a decision problem we want to determine if a given instance x satisfies some condition. In this case, the set of instances can be partitioned into $I_{\mathcal{P}} = Y_{\mathcal{P}} \cup N_{\mathcal{P}}$ such that $x \in Y_{\mathcal{P}}$ if and only if $(x, \text{YES}) \in \mathcal{P}$. Therefore, we are interested to know whether x belongs to the set $Y_{\mathcal{P}}$.

We present some examples of decision problems. Some of them are classical examples used in this field and some are important for the complexity analyses presented in the next chapters.

- HITTING SET: Given a set of integers I and a collection of subsets $I_1, \ldots, I_n$, does there exist a hitting set of $I_1, \ldots, I_n$ of size at most k?
- SATISFABILITY (SAT): Given a Boolean expression f, does there exist some assignment to the Boolean variables such that f evaluates TRUE?
- *st*-PATH: Given a digraph D and two vertices s and t, does D contain a directed path from s to t?

- Directed Hamiltonian Cycle (DHC): Given a digraph D, does D contain a Hamiltonian cycle, i.e. a directed cycle that visit each vertex of D exactly once?

- Perfect Matching: Given a bipartite graph $G = (\mathcal{V}, \mathcal{E})$, does $\mathcal{E}$ contain a subset $\mathcal{E}'$ of edges such that all vertices of the subgraph $G' = (\mathcal{V}, \mathcal{E}')$ have degree one?

2.1.2 Optimisation problems

In some cases, although an instance of a problem can have many solutions, there are some that are better than others. In an *optimisation problem* $\mathcal{P}$, there is a function $m : I_\mathcal{P} \times S_\mathcal{P} \to \mathbb{R}$ which measures, for an instance $x \in I_\mathcal{P}$ the cost or benefit of a solution y of x. We are interested in finding the "best" solution(s) which minimise or maximise m or at least a good approximation to it.

Some examples of optimisation problems are:

- Minimum Hitting Set (minHS): Given a set of integers I and a collection of subsets $I_1, \ldots, I_n$, find a set $H \subseteq I$ of minimum size, such that $I_i \cap H \neq \emptyset$ for all $i = 1, \ldots, n$.

- Linear Programming (LP): Given matrices $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, find a vector $x \in \mathbb{R}^n$ such that it satisfies $Ax \leq b$ and maximises a linear objective function $c^T x$.

- Travelling Salesman (TSP): Given a complete graph G with distances on its edges, find a Hamiltonian cycle of minimum total distance.

- Vertex Cover: Given a graph $G = (\mathcal{V}, \mathcal{E})$, find a set $C \subseteq \mathcal{V}$ of minimum size, such that each edge of G is incident to at least one vertex in C.

2.1.3 Enumeration and counting problems

For a given $\mathcal{P}$, we are interested in finding not one but *all* solutions of a given instance $x \in I_\mathcal{P}$, that is, we want to *enumerate* all the solutions $y \in S_\mathcal{P}$ such that $(x, y) \in \mathcal{P}$.

Some examples of enumeration problems are:

- All Minimal Hitting Sets (AllHS): Given a set of integers I and a collection of subsets $I_1, \ldots, I_n$, find all minimal sets $H \subseteq I$, such that $I_i \cap H \neq \emptyset$ for all $i = 1, \ldots, n$.

- All Prime Implicants: Given a Boolean $\wedge, \vee$-formula f, find all prime implicants of f.
- All Negative Directed Cycles: Given a digraph D, with integer weights on its arcs, find all negative cycles of D.

In some cases we are interested in *counting* the number of solutions of a given instance x. In other words, for a given $x \in I_{\mathcal{P}}$, we want to obtain $k = |\{y \in S_{\mathcal{P}} \mid (x, y) \in \mathcal{P}\}|$. This is the case of the following example:

- Count Perfect Matchings: Given a bipartite graph G, how many perfect matchings does G contain?

2.2 Analysis of algorithms

We give only an intuitive definition of what is an algorithm. For a rigourous definition, we would need to introduce the concept of a *Turing machine*. This would be out of the scope of this introduction. We thus just say that an algorithm A for a problem $\mathcal{P}$ is a sequence of well-defined instructions such that for any instance $x \in I_{\mathcal{P}}$ given as input, it executes a list of *basic operations* in order to return one or more solutions of x.

Given a decision problem $\mathcal{P}$, we say that an algorithm A *decides* $\mathcal{P}$ if for any instance $x \in I_{\mathcal{P}}$, $A(x)$ returns YES if and only if $x \in Y_{\mathcal{P}}$. In other words, A computes correctly if x satisfies the condition defined by $\mathcal{P}$. Turing showed that there exists some decision problems for which there is no algorithm that can give the correct answer for all the instances of the problem (for example see the *halting problem*). We say that a decision problem $\mathcal{P}$ is *decidable* if there exists an algorithm that decides $\mathcal{P}$.

Of course, most of the "real problems" that we face are decidable, although this does not mean that the problem will be "easy" to be solved. In general, it is not difficult to find, for a given problem $\mathcal{P}$, an algorithm that finds a desirable solution by checking all possible solutions for the given input. For example, we can solve the problem st-Path by checking all the possible directed walks from s in the digraph until a path to t is found or decide that such path does not exist. However, this method is in general impracticable, since it will take an enormous amount of time, specially when the digraph has a considerable size.

In general, for a given problem, we want to have a *good* algorithm in the sense that it does not take *too much time* in finding a desirable solution. However, there are many famous problems for which, despite years of work, the best known algorithms to solve

them still take an "exponential amount of time". This concerns algorithms that for some (or all) possible inputs, need to do an exponential number (in terms of the size of the instance) of basic operations to output the solution. Thus, they are methods that can be used only for very small instances of the problem. A natural question that arises in this case is: does there exist an algorithm that solves $\mathcal{P}$ in a polynomial number of steps or is $\mathcal{P}$ intrinsically "hard"? Can we classify each problem in terms of its intrinsic "complexity"?

With the aim of giving a coherent theory to try to answer this question, we introduce some basic concepts.

2.2.1 Input size

Since problems can be defined over a very dissimilar variety of data, we suppose that the input and the output of an algorithm are encoded as sequences of bits. In other words, if we define the set $\{0,1\}^*$ as the set of all finite sequences of 0's and 1's, we suppose that $I_{\mathcal{P}} \subseteq \{0,1\}^*$ and $S_{\mathcal{P}} \subseteq \{0,1\}^*$.

For a given instance $x \in I_{\mathcal{P}}$, we define the *size* of x, denoted by $|x|$, to be the length of the sequence x.

We suppose that the way of encoding is *reasonable* in the sense that it does not introduce an artificially redundant information. In general, we can also suppose that for two different reasonable ways of encoding a problem and for any instance x, the size of both codifications of x are not too different (the length of the sequences are polynomially related).

2.2.2 Worst case analysis

Given an algorithm A for a problem $\mathcal{P}$ and an instance $x \in I_{\mathcal{P}}$ of size $|x| = n$, we want to measure, in terms of n, the time that $A(x)$ takes to stop. Of course, this time should depend not only on the size of x but on x itself. A way to be conservative, is to consider the *worst case*, that is, we consider the maximum time that A takes for all the inputs of size n.

2.2.3 Asymptotic analysis

Clearly, the real running time of an algorithm depends on the technology of the machine. Therefore, if we want to have a measure independent of the machine, we should avoid to consider the time used in executing each basic operation. We can do this

by defining a class of functions that expresses the time *asymptotically* in terms of the input, independent of the technological characteristics of the machine.

Let A be an algorithm whose running time, in the worst case, is $t_A(n)$ (where n is the size of the input). Let $g : \mathbb{N} \to \mathbb{N}$ be a function. We say that the running time of A is $O(g(n))$, if there exist constants c and n_0 such that, for all $n \geq n_0$

$$t_A(n) \leq cg(n).$$

2.3 Complexity classes of decision problems

2.3.1 The class P

Let $\mathcal{P}$ be a decision problem. We say that $\mathcal{P}$ belongs to the complexity class P, if $\mathcal{P}$ can be decided in polynomial time. In other words, there exists an algorithm A such that $\mathcal{P}$ is decided by A and the running time of A is $O(n^k)$ for some constant $k \in \mathbb{N}$.

Basically to show that a problem is in P, we just need to find a polynomial algorithm which solves it. For example, the problems *st*-Path and Perfect Matching are known to be in P since there are polynomial algorithms that solve them.

2.3.2 The class NP

There are many important problems for which we do not know whether they belong to P. Can we define a class of decision problems such that we could still have a *hope* that they belong to P? In effect, most of the problems that we do not know whether they belong to P have at least the property to be *easy to check*. This means that given an instance x of $\mathcal{P}$ such that $x \in Y_{\mathcal{P}}$ and a mathematical object c called *certificate* of x, we can *verify* in polynomial time that $x \in Y_{\mathcal{P}}$. For instance, consider the problem Directed Hamiltonian Cycle (DHC). We can easily check that a digraph has a Hamiltonian cycle by showing as certificate c the set of arcs that forms the cycle. Indeed, we can easily verify that c effectively corresponds to a Hamiltonian cycle.

Problems that, given a certificate, are easily checkable form the class NP. Formally, a problem $\mathcal{P}$ belongs to the complexity class NP, if there exists a polynomial time algorithm A (the verifier) such that:

- For each instance $x \in Y_{\mathcal{P}}$, there exists a certificate $c(x)$ (of polynomial size with respect to x) such that $A(x, c(x)) = 1$

- For each instance $x \in N_{\mathcal{P}}$, we have $A(x, c) = 0$ for any certificate c.

Clearly, any problem in P is also in NP since a verifier is just an algorithm that solves the problem without need to use the certificate. Hence P $\subseteq$ NP.

Apart from DHC and the P problems *st*-PATH and PERFECT MATCHING, other examples of problems in NP are HITTING SET and SAT. All of them are easily checkable by an appropriate certificate.

Note that we said that P $\subseteq$ NP and not P $\subset$ NP. Indeed, we cannot discard the possibility that the two sets are equal since there is no proof that there exists a problem in NP which *cannot* be solved in polynomial time.

2.3.3 Reducibility among problems

A very useful idea when we analyse the complexity of problems, is the reducibility among them. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two decision problems with $I_{\mathcal{P}_1}$ and $I_{\mathcal{P}_2}$ the respective sets of instances. We say that $\mathcal{P}_1$ is *reducible* to $\mathcal{P}_2$ if there is a way to transform (reduce) any instance x of $\mathcal{P}_1$ to an instance $R(x)$ of $\mathcal{P}_2$ such that $x \in Y_{\mathcal{P}_1}$ if and only if $R(x) \in Y_{\mathcal{P}_2}$.

This concept is applied to show that, if R is an algorithm that reduces in *polynomial time* any instance of $\mathcal{P}_1$ to an instance of $\mathcal{P}_2$, we have that if $\mathcal{P}_2$ is in the class P then $\mathcal{P}_1$ is also in the class P. In effect, since there is an algorithm A that solves $\mathcal{P}_2$ in polynomial time, then we can construct an algorithm A' which solves $\mathcal{P}_1$ in polynomial time by applying R and A.

2.3.4 NP-complete problems

Stephen Cook showed (Cook, 1971) that the problem SAT has the interesting property that any other problem in NP is polynomial time reducible to it. Hence, if SAT can be solved in polynomial time, then any other NP problem can be solved in polynomial time. This motivates the following definition: A problem $\mathcal{P}$ in the class NP is called NP-*complete* if any other problem in NP is polynomial time reducible to $\mathcal{P}$.

Note that, given a problem $\mathcal{P}_1$ in NP and an NP-complete problem $\mathcal{P}_2$, if $\mathcal{P}_2$ can be reduced in polynomial time to $\mathcal{P}_1$, then $\mathcal{P}_1$ is also NP-complete. Using this fact, many problems have been shown to be NP-complete. Apart from SAT, the problems HITTING SET and DHC are other examples of NP-complete problems.

NP-complete problems are considered the *hardest* problems in the class NP. Indeed, if we are able to find a polynomial algorithm that solves one NP-complete problem,

then *all* the problems in NP can be solved in polynomial time, that is, $NP = P$. However, despite many years of research, nobody has found an algorithm that solves any single NP-complete problem in polynomial time.

The interest of this concept is that, if we are studying a problem $\mathcal{P}$ such that we are not able to find a polynomial time algorithm that solves it, we can alternatively try to find a *polynomial time reduction* of one NP-complete problem to $\mathcal{P}$. We can then show that $\mathcal{P}$ is a new NP-complete problem and therefore, considering all the unsuccessful work spent through the years, that finding a polynomial algorithm to solve it should be very unlikely.

2.4 Complexity classes of optimisation problems

In an optimisation problem, for a given instance $x \in I_{\mathcal{P}}$, the set $f(x)$ of *feasible solutions* of x is the set of all possible solutions y of x, that is, $f(x) = \{y \in S_{\mathcal{P}} \mid (x, y) \in \mathcal{P}\}$. As we said before, in an optimisation problem $\mathcal{P}$ we have an additional function $m : I_{\mathcal{P}} \times S_{\mathcal{P}} \rightarrow \mathbb{R}$ that indicates the cost (or benefit) of a given solution of x. For a given instance $x \in I_{\mathcal{P}}$, we call *optimal solution* any solution y that minimises the cost (maximises the benefit), and *optimal value* the measure of an optimal solution. We are therefore interested in finding a feasible solution y that is an optimal solution or at least that approximates the value of an optimal one.

2.4.1 The classes PO and NPO

Similar to NP for decision problems whose certificates can be checked in polynomial time, for optimisation problems also we can define the class NPO of problems that satisfy some minimal conditions to be considered *tractable*. Without giving a formal definition, we can say that NPO consists of all the optimisation problems for which there exists a constant $k > 0$ such that: the instances x can be recognised in polynomial time, the size of any feasible solution y of x is $O(|x|^k)$, the cost function can be computed in polynomial time, and for any instance x and sequence y of size $O(|x|^k)$ we can decide in polynomial time if $(x, y) \in \mathcal{P}$.

As for decision problems, we can define the subclass of optimisation problems that are easy to be solved. Given a problem $\mathcal{P}$ in the class NPO, we say that $\mathcal{P}$ is in the class PO if there exists a polynomial time algorithm A such that, for any instance $x \in I_{\mathcal{P}}$, it returns an optimal solution $\bar{y}$ and the optimal value $c(x, \bar{y})$.

Examples of problems in NPO are MINIMUM HITTING SET, TRAVELLING SALESMAN and LINEAR PROGRAMMING. In particular, we know that LINEAR PROGRAMING can be solved in polynomial time, hence it is in PO.

2.4.2 NP-hard optimisation problems

As for decision problems, there are many important NPO problems for which we do not know whether there exists a polynomial algorithm. Moreover, many of them are NP-*hard* in the sense that if they were in PO, then any decision problem in NP could be solved in polynomial time, that is P = NP. Indeed, it has been proved that PO = NPO if and only if P = NP. Therefore, to find a polynomial algorithm that exactly solves an NP-hard optimisation problem should be extremely difficult if not impossible.

One way to prove that an optimisation problem is NP-hard is to consider the *decision version* of the problem. Indeed, we can transform the question of "given x, find a solution with maximum $m(x)$" into "given x and k, *decide* if there exists a feasible solution of x with $m(x)$ greater than k". If the latter problem is NP-complete, then the optimisation problem is clearly NP-hard. Problems VERTEX COVER, MINIMUM HITTING SET and TRAVELLING SALESMAN are examples of NP-hard optimisation problems.

2.4.3 Approximation algorithms

Despite the difficulty in finding exact solutions, many NP-hard optimisation problems have polynomial algorithms that, although they cannot give an optimal solution, guarantee a feasible solution *close* to the optimal one.

Let $\bar{y}$ be any optimal solution of an optimisation problem $\mathcal{P}$. We say that A is an $\varepsilon(x)$-approximation algorithm for some $\varepsilon(x) > 0$ if and only if

$$\frac{|c(x, A(x)) - c(x, \bar{y})|}{c(x, \bar{y})} \leq \varepsilon(x)$$

for any instance $x \in I_{\mathcal{P}}$. The value $\varepsilon(x)$ is called the *approximation ratio*.

This definition gives some measure about the accuracy of an algorithm to approach the optimal solution. For instance, we know that MINIMUM HITTING SET has an $O(\log n)$-approximation where n is the number of sets.

2.4.4 The classes APX and APX-hard

Some classes of problems have been defined depending on the existence and characteristics of the approximation algorithms that can solve them. For instance, APX is the class of optimisation problems for which there exists an ε-approximation algorithm for some *fixed* (independent of the input) $\varepsilon > 0$. For example, VERTEX COVER has a 2-approximation algorithm, and therefore it is in APX. Unfortunately, MINIMUM HITTING SET and TRAVELLING SALESMAN are not in APX unless P = NP.

The class APX-hard corresponds to the problems whose approximability is bounded unless P = NP. In other words, a problem is APX-hard if, under the assumption that P $\neq$ NP, there is $b > 0$ such that there is no polynomial ε-approximation algorithm for any $\varepsilon < b$. Of course, MINIMUM HITTING SET and TRAVELLING SALESMAN are APX-hard, but also VERTEX COVER is APX-hard. Indeed, it is not approximable within a factor of 1.3606 unless P = NP.

2.5 Complexity of counting solutions

2.5.1 ♯P and ♯P-complete

Suppose that, for a given instance x of a problem $\mathcal{P}$, we want to count the number of solutions y of x. Again, we restrict the problems to those having some minimum conditions. Therefore, we define ♯P to be the set of counting problems such for any instance $x \in I_{\mathcal{P}}$ and sequence $y \in \{0,1\}^*$, we can decide in polynomial time if $(x, y) \in \mathcal{P}$.

The complexity class ♯P contains the counting problems associated with decisions problems in NP: for instance, counting the number of Hamiltonian cycles in a digraph is in ♯P. Since if we can count objects, we can decide the existence of at least one of them, a counting problem in ♯P must be at least as hard as the corresponding decision problem. Like the class NP, also ♯P has complete problems, the hardest problems within the class. Solving any of the ♯P-complete problems in polynomial time would prove that any problem in ♯P can be solved in polynomial time, and therefore that P = NP.

There are some ♯P-complete problems that corresponds to some easy decision problems. For instance, although PERFECT MATCHING is in P, the problem COUNT PERFECT MATCHINGS is ♯P-complete (Valiant, 1979).

2.6 Complexity of enumerating all the solutions

Given a problem $\mathcal{P}$, we study the time complexity of enumerating all the solutions of any instance $x \in I_{\mathcal{P}}$ (we suppose that the number of solutions of any x is finite). However, we must redefine how we measure the time. Indeed, consider, for example, the problem of enumerating all the Hamiltonian cycles of a graph. Note that, in the worst case, the number of solutions can grow exponentially in terms of the size of the graph (consider, for instance, the complete graph that has $\frac{(n-1)!}{2}$ Hamiltonian cycles). Then, if we measure the running time of some algorithm A for enumerating all the solutions, although A can find each solution in polynomial time, it will be exponential only because it needs exponential time to *print* all the solutions. For that reason, it is natural to analyse the complexity of enumeration problems with respect to the size of the input and the output.

Moreover, we can consider *how* an algorithm A that solves $\mathcal{P}$ returns the set of solutions. Algorithm A could have the ability of finding one solution and then of repeating the process to find another one, (in which case we can already have some solutions before the end of the complete process) or it may need to compute the total set of solutions at the same time.

Taking into account these considerations, we can classify an enumeration problem $\mathcal{P}$ depending on the existence of efficient algorithms that solve it.

2.6.1 Time delay, incremental time and total time

We define three time complexity classes of enumeration problems that have been proposed in Johnson *et al.* (1988).

- An enumeration problem can be solved with *polynomial delay* if given a set of elements already enumerated, the time needed for generating another element or asserting that no other element exists can be done within a time bounded by a polynomial function of the input size only. We call this class of problems PD.

- An enumeration problem can be solved in *incremental polynomial time* if given a set of elements already enumerated, the time needed for generating another element or asserting that no other element exists can be done within a time bounded by a polynomial function of the input size and the number of already enumerated elements. We call this class of problems PI.

- An enumeration problem can be solved in *polynomial total time* if an algorithm exists with running time bounded by a polynomial function of the combined size of the input and the output. We call this class of problems PT.

It is easy to show that PD ⊆ PI ⊆ PT.

Concerning the enumeration problems introduced, Gurvich et Khachiyan (1999) showed that ALL PRIME IMPLICANTS is not in PT unless P=NP. Khachiyan *et al.* (2008) showed the same result for the ALL NEGATIVE DIRECTED CYCLES enumeration problem. Therefore, these two problems are hard to enumerate.

On the other hand, concerning the ALL MINIMAL HITTING SETS enumeration problem, Gurvich et Khachiyan (1999) showed, from a result of Fredman et Khachiyan (1996), that given a subset of solutions of this problem (that is, minimal hitting sets), we can compute a new solution (or asserting that no other minimal hitting set exists) in time $o(k^3) + k^{o(logk)}$ with k the combined size of the input and the already enumerated solutions. Therefore, ALL MINIMAL HITTING SETS can be enumerated in incremental *quasi-polynomial* time.

Chapter 3

Metabolic Networks

Contents

In this chapter we present the biological concepts that motivate the mathematical model adopted.

According to Webster's Unabridged Dictionary, *Metabolism* can be defined as "the sum of the physical and chemical processes in an organism by which its material substance is produced, maintained, and destroyed, and by which energy is made available". In other words, metabolism corresponds to the set of processes and transformations occurring in living organisms in order to maintain life. This comprises, among others, obtaining energy from the degradation of nutrients (*Catabolism*) and producing the molecules needed to accomplish specific cellular functions (*Anabolism*). The set of all biochemical reactions and of all biochemical compounds that are consumed and produced by these reactions forms a network of relations which is called a *metabolic network*.

Understanding how the metabolic network of an organism performs the needed transformations is not an easy task. Indeed, the set of reactions are also *regulated* by

the cell in order to obtain what it needs for each given particular condition. Thus, a metabolic network is just one level of a very complex and heterogeneous network of relations between the set of entities involved in metabolism. We present a brief description of the main entities and levels of interactions, which are defined depending on their general functions and on the nature that each one has.

3.1 Entities involved in metabolism

3.1.1 Biochemical reactions and metabolites

The *reactions* are the basic transformations of the metabolism. They transform a set of chemical compounds by reordering the atoms that compose them. The set of compounds involved in the reactions are called *metabolites*. Each reaction transforms a set of metabolites called *substrates* into another set of metabolites called *products* of the reaction. For instance, the synthesis of acetolactate transforms the substrate $C_3H_4O_3$ to produce $C_5H_8O_4$ and CO_2.

Stoichiometry

Sometimes we need to consider not only which compounds are transformed by the reaction but also the amount of each metabolite that is consumed and produced. We call *stoichiometry* of a reaction the quantitative relations between the metabolites involved.

We can add the stoichiometric values to the reaction representation. For instance,

$$2C_3H_4O_3 \rightarrow C_5H_8O_4 + CO_2$$

indicates that two molecules of the substrate $C_3H_4O_3$ are needed to produce a molecule of $C_5H_8O_4$ and a molecule of CO_2.

Reversibility of reactions

In theory, all reactions can occur in both directions but many of them are considered *irreversible* when the transformation represented by the reaction happens exclusively or preferentially in only one direction. If a reaction can occur in both directions, we say that it is *reversible* and a double arrow ($\leftrightarrow$) is used to represent it. Depending on the kind of analysis, the two directions of a reversible reaction can be considered

as two different reactions. In this way, we avoid confusion on which compounds are defined as substrates or products of it.

3.1.2 Enzymes and genes

Almost all the reactions in the cell need the presence of some proteins called *enzymes* to occur at a significant rate (amount of metabolites reacting in a unit of time). Each enzyme acts as a catalyser of a reaction, decreasing its activation energy and thus providing the necessary conditions to accelerate dramatically its rate up to a million times.

Although there is great specificity in the relation between enzymes and reactions, this is not a general rule. A single enzyme may catalyse several reactions and one reaction may be catalysed by several enzymes. Since enzymes are not modified during the reaction, they are not considered as a substrate or product in the equation of a given reaction.

The *genome* of a living organism contains all the information about the hereditary characteristics of an organism including the codification of all the machinery for life. It is composed of one or more *chromosomes* that correspond to chains of *nucleotides* of four types (a, c, g, t). These long sequences contain the elementary units of information called *genes*.

Genes correspond among others to the coded sequence of each protein of the organism and the mechanisms to regulate the protein production. In particular, the protein or protein complex that forms each enzyme is coded by one or more genes.

3.1.3 Metabolism regulation

Not all the possible reactions that an organism can perform occur at any time or in any cell of the organism. Many factors of different nature can control and modulate the possible reactions according to the particular needs of the cell. This mechanism of regulation is essential in giving to the cell its adaptability to the different conditions of the medium and, in the case of multicellular organisms, it allows *cellular differentiation* and *morphogenesis*.

We mention succinctly two ways of regulation:

Regulation of gene expression

The rate at which a protein is produced is called *gene expression*. Regulation of gene expression is modulated at each step between copying the gene information and producing the enzyme. The mechanisms are varied and complex and in general not very well-known.

Regulation of enzyme concentration through regulation of gene expression is essential in metabolism since the presence and absence of the enzymes in a given time period can define the capacities of the metabolic network.

Presence of cofactors

The catalysis of a reaction by enzymes sometimes is affected by the presence of some small molecules, called *cofactors*. By binding the enzyme, cofactors can enhance or decrease the activity of the enzyme. Depending on the effect produced, they are called *allosteric activators* or *allosteric inhibitors*.

3.1.4 Reconstructing a metabolic network

Reconstructing the metabolic network of a particular organism consists in inferring the relations between genes, proteins (enzymes), and reactions in a given metabolic system. It is principally done by *sequencing* its genome and using *comparative genomics*, that is, identifying the enzymes coded in it by comparing the genome sequence with the databases of coded enzymes of other organisms. In recent years, this process has been highly automated, although it still requires a slow manual expert validation supported by carefully collected data from the literature.

The quality of the reconstruction by comparative genomics depends highly on the quality of this annotation and, to a lesser extent, on the taxonomic position of the organism. Thus, for instance, the reconstruction for organisms closer to *E.coli*, for which the relation between genes and metabolic functions are well known, are highly likely to be of better quality.

A metabolic reconstruction can be also complemented by using metabolomic data, consisting in inferring data about the type and quantity of metabolites present in the metabolism of the organism.

Other kinds of metabolic relations, like for instance allosteric effects on enzymes, may be more difficult to study and therefore are in general not considered in the metabolic reconstruction of an organism.

3.2 Modelling metabolic networks

When we consider a set of reactions and metabolites as the input data, the most classical mathematical models used to describe a metabolic network are graphs (or hypergraphs), constraint-based methods and differential equations (Stelling, 2004). The last is mainly used for describing the dynamic of the network (Szallasi *et al.*, 2006). We shall focus on the first two models, since they describe the metabolic network from its structural characteristics and are the framework of the results presented in this thesis. Of course, the choice of the appropriate model to use depends mainly on the kind of questions that we are interested in answering.

3.2.1 Graph and hypergraphs models

We present a brief description of the most commonly used graphs to model a metabolic network (Lacroix *et al.*, 2008):

1. *Compound graph*: Nodes correspond to compounds and there is an edge between two compounds if there is a reaction where one is a substrate and the other is a product.

2. *Reaction graph*: Nodes correspond to reactions and there is an edge between two reactions if there exists a compound that is produced by one and consumed by the other.

3. *Reaction-compound bipartite graph*: The set of nodes is divided into the set of compounds and the set of reactions. There is an edge between a compound and a reaction if the compound is substrate or product of the reaction.

Note that all these models are ambiguous in the sense that two different sets of reactions could be represented by the same graph. In other words, they do not maintain all the information given by the reaction set. These ambiguities can be solved by adding appropriate labels to the edges and/or nodes (Lacroix *et al.*, 2008).

In particular, the direction of the irreversible reactions are not represented in these graphs. In the case where all reactions are irreversible (or when reversible reactions are represented by two irreversible reactions of opposite direction), we can consider the same representations but using *digraphs*, that is, we can use arcs instead of edges according to the direction of the reactions involved.

It is easy to see that, regardless of the stoichiometric values, the only representation which maintains complete information on how the compounds are involved in the reactions is the reaction-compound bipartite *digraph*.

Directed hypergraph modelling

Directed hypergraphs are an alternative representation equivalent to the reaction-compound bipartite digraph presented above. Indeed, directed hypergraphs seem to be a very natural object to represent metabolic networks where all reactions are irreversible: nodes correspond to compounds and hyperarcs correspond to reactions.

3.2.2 Including stoichiometry

Stoichiometric values can be added as numerical labels in graph and hypergraph representations (Lacroix *et al.*, 2008). For instance, in the reaction-compound bipartite graph, we can add the stoichiometric values to the edges of the graph. Moreover, we can use positive and negative values to indicate if the compounds are, respectively, produced or consumed. In this case, the signs made unnecessary the use of directed edges.

If we number from 1 to $|\mathcal{C}|$ the set of metabolites and from 1 to $|\mathcal{R}|$ the set of reactions, then we can define a *stoichiometric matrix* S with $|\mathcal{C}|$ rows and $|\mathcal{R}|$ columns containing all the stoichiometric values of the network. In other words, S is defined by

$$S_{a,r} = \begin{cases} k & \text{if } r \text{ produces } k \text{ units of } a \\ -k & \text{if } r \text{ consumes } k \text{ units of } a \\ 0 & \text{otherwise} \end{cases}$$

3.2.3 Assuming steady state

In constraint-based modelling, the aim is to analyse the distribution of mass fluxes through the reactions when the system is in *steady state*. This means that the amount produced of every metabolite in one unit of time is equivalent to the amount of it that is consumed. In order to analyse the dynamics of a network, we consider: a) the vector c of concentrations of each metabolite and b) the *flux* vector v of fluxes of each reaction (how many times each reaction happens in one unit of time). Note that we allow negative fluxes for reversible reactions, meaning that the transformation is done in the opposite direction (from products to substrates). Clearly we have the following

relation:

$$dv/dt = Sv$$

Assuming that the system has reached a *steady state*, that is the concentration of each metabolite remains constant, the vector v of rates satisfies the equation $Sv = 0$. That is, the flux vector v belongs to the nullspace of S. However, not any vector in the nullspace is a feasible flux of the system since it may violate the irreversibility of some reactions.

The two constraints define a portion of the flux space represented by a convex polyhedral cone containing all admissible flux vectors. Given this framework, two main problems have been addressed. The first is called *Flux Balance Analysis* (FBA) and is concerned with finding an admissible flux vector that optimises a given objective function like, for instance, biomass or ATP production. FBA has been shown to have a good phenotypic predictive power (Edwards et Palsson, 2000) and can also be applied to predict the phenotypic effect of a perturbation of the system, like gene deletions.

When all the admissible flux vectors are of interest, we can try to find a subset of the vectors able to generate all of them. Several approaches have been presented having in common the fact that they are descriptions of the flux cone. The most widely used, in the context of metabolic networks, are the concepts of *elementary modes* and *extreme pathways* (Schilling *et al.*, 2000). Recently, minimal behaviours have been introduced (Larhlimi et Bockmayr, 2009) as an alternative description of the flux cone.

1. *Elementary (flux) modes:* They correspond to the smallest set of reactions that can function together in steady state. Any flux of the steady state flux cone can be decomposed as the sum of elementary modes. They also have been proposed as a formal definition of a metabolic pathway.

2. *Extreme pathways:* Very similar to elementary modes, extreme pathways are a subset of them. Again, any flux of the cone can be decomposed as a sum of extreme pathways. The difference is that, unlike elementary modes, this decomposition is always *unique*.

3. *Minimal behaviours:* Instead of being an inner description of the steady state flux cone by a set of generating fluxes (like elementary modes and extreme pathways), minimal behaviours correspond to an *outer* description of the cone by a set of non-negativity constraints. This description is more compact than the previously presented ones, although biological interpretation of the corresponding objects has not yet been clearly established.

The concepts of elementary modes and extreme pathways only differ in the way reversible reactions are treated. In fact, if the set of reversible reactions is empty, both notions coincide. In particular, they are the same notion in the case where reversible reactions are considered as two different reactions. For a detailed comparison of both approaches, see Klamt et Stelling (2003).

As outlined in Schuster *et al.* (2002b), the concept of minimal T-invariants used in Petri Nets is also closely related to the concept of elementary modes. Both notions coincide in the case where all reactions are irreversible. For completeness sake, we can also mention that the extreme currents defined by Bruce Clarke (Clarke, 1981) coincide with elementary modes in the irreversible case. Unlike extreme pathways and elementary modes, minimal T-invariants and extreme currents have only been defined in the case of a network of irreversible reactions.

Because of its easy biological interpretation, we focus only on elementary modes as a model of the dynamics of a metabolic network in steady state. Moreover, we assume that all reactions are irreversible, and therefore the results are valid for most of the concepts.

Chapter 4

Complexity of Computing Elementary Modes

Contents

4.1 Modelling metabolic network in steady state

4.1.1 Definitions

A metabolic network is modelled as a set $\mathcal{C}$ of *metabolites* (also called *compounds*) and a set $\mathcal{R}$ of *reactions*. Each reaction transforms a subset of metabolites, the *substrates*, into another set of metabolites, the *products* of the reaction. The set of reactions is partitioned into two subsets: Rev and $Irrev$, which correspond to the sets of reversible and irreversible reactions, respectively. The matrix S contains the stoichiometric values of the reactions.

Assuming that the system has reached a steady state, any flux vector v satisfies the equation $Sv = 0$ and the thermodynamic constraints (positive flux in irreversible reactions). The vectors v in the nullspace that satisfy these constraints form the set of the a priori possible *modes* of a metabolic network.

Definition 4.1. *A* mode *is a vector $v \in \mathbb{R}^m$ such that:*

1. $Sv = 0$
2. $v_j \geq 0 \; \forall j \in Irrev$

In other words, the modes of a system cover the set of feasible flux vectors that maintains the system in steady state.

Given a mode $v \neq 0$, we define $R(v)$ the *support* of v as the set $R(v) = \{j \mid v_j \neq 0\}$, *i.e.*, the set of reactions participating (with non-zero flux) in v.

In a similar way that any vector v of the nullspace can be described as a *linear* combination of vectors of a basis of the nullspace, we can describe the set of modes of S as a *positive* combination of modes called *elementary modes*.

Definition 4.2. *We say that a mode $v \neq 0$ is an* elementary mode *if its support is minimal, that is, if there is no other mode $w \neq 0$ such that: $R(w) \subset R(v)$.*

Note that if v is an elementary mode (EM) then αv is also an EM for any real $\alpha > 0$. We consider that they correspond to the same EM (they are identical up to scaling). If two EMs differ by a negative factor then they are considered different since they correspond to fluxes of opposite directions that should represent different biological functions.

Property 4.3. *If two elementary modes v and w have the same support $R(v) = R(w)$, then there exists $\alpha \neq 0$ such that $v = \alpha w$ (i.e. they are collinear vectors).*

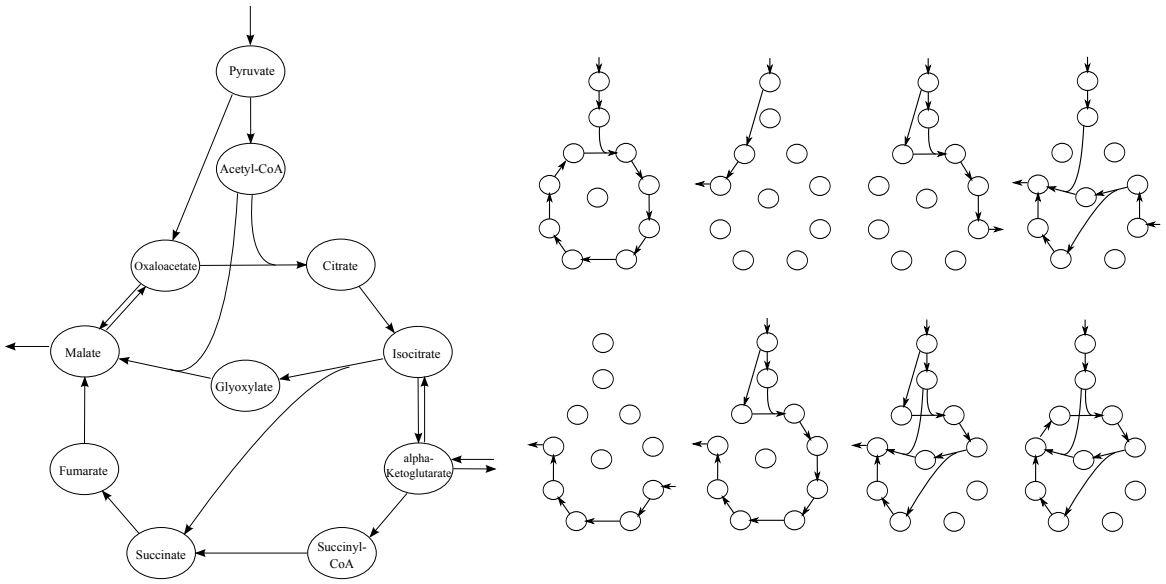

Figure 4.1: An example of elementary modes analysis. *Left:* A simplified model of the Citric Acid Cycle (including some anaplerotic reactions and the glyoxylate cycle). Some ubiquitous compounds were excluded from the model. The stoichiometric matrix has values -1, 1 and 0. *Right:* The eight elementary modes of this metabolic network (trivial cycles of two reactions are excluded).

Proof. By contradiction, suppose that $R(v) = R(w)$ and v and w are not collinear vectors. Let k be such that

$$\frac{v_k}{w_k} = \max_{i \in R(v)} \frac{v_i}{w_i}.$$

Then the flux

$$u = \frac{v_k}{w_k} w - v$$

is such that $Su = 0$. Since v and w are not collinear, $u \neq 0$. Moreover, $u_\ell \geq 0$ for any irreversible reaction ℓ in $R(v)$. In effect,

$$u_\ell = \frac{v_k}{w_k} w_\ell - v_\ell = (\frac{v_k}{w_k} - \frac{v_\ell}{w_\ell}) w_\ell \geq 0$$

Therefore u is a mode such that $R(u) \subset R(v)$ which contradicts the fact that v is an EM. We conclude that v and w are collinear vectors. Therefore they correspond to the same EM or they correspond to the two opposite directions of a reversible flux. □

Corollary 4.4. *If all reactions are irreversible, then an elementary mode is completely defined by its support.*

4.1.2 Relation between elementary modes and extreme rays

Modes and elementary modes can be given a geometrical interpretation. Indeed, the set of vectors $\{v \geq 0 \mid Sv = 0\}$ defines a convex polyhedral cone in the flux space. When all reactions are irreversible, the EMs exactly correspond to the *extreme rays* of this cone (Schuster et Hilgetag, 1994). An extreme ray is a ray of the cone that can not be expressed as a convex combination of other rays of the cone.

Lemma 4.5. *If all reactions are irreversible, then the set of EMs corresponds one-to-one to the set of extreme rays of the cone* $\{v \geq 0 \mid Sv = 0\}$.

Proof. Clearly the definition of an EM implies that it is an extreme ray of the cone. To see that it is also the other way round, suppose we have two vectors $v \geq 0$ and $w \geq 0$ with $Sv = 0$ and $Sw = 0$ such that $R(v) \subset R(w)$. Then clearly w is not an EM and we show that w is not either an extreme ray of the cone.

Let k be such that

$$\frac{v_k}{w_k} = \max_{i \in R(v)} \frac{v_i}{w_i}.$$

Then the flux

$$u = \frac{v_k}{w_k} w - v \geq 0 \text{ and } Su = 0.$$

Thus u is a mode. Moreover, $R(u) \subset R(w)$ and $u_k = 0$ and most importantly,

$$\frac{v_k}{w_k} w = v + u,$$

proving the claim. □

4.1.3 Reversibility of reactions

In the particular case when all reactions are reversible ($Irrev = \emptyset$), an elementary mode corresponds to a minimally dependent set of columns of the stoichiometric matrix. Hence the EMs are exactly the *circuits* of a *linear matroid* (for definitions of matroids and circuits we refer to Oxley, 1992; Schrijver, 2003).

In the more general case when there are some irreversible reactions, Gagneur et Klamt (2004) observed that we can define a pointed cone in a higher dimensional space by representing each reversible reaction by two irreversible reactions in the obvious way: suppose the reaction r is represented in S by the column s_r, then we add the column $s_{\overline{r}} = -s_r$ to S, yielding S^+, and require both v_r and $v_{\overline{r}}$ to be non-negative.

The cone defined by the matrix S^+ has extreme rays that consist of those of S and the vectors v with $v_r = v_{\overleftarrow{r}} = 1$ and $v_i = 0$ otherwise, corresponding to length-2 cycles consisting of the two reactions making up for a reversible reaction. We can easily detect and simply ignore these length-2 cycles. A consequence of this observation is that **all the complexity results and the proposed algorithms for the irreversible case can be extended to the general case.**

From now on we assume that all reactions are irreversible unless explicitly stated otherwise.

4.2 Checking consistency of the stoichiometric matrix

One of the applications of constraint-based modelling is in checking the consistency of reconstructed metabolic networks (Schuster *et al.*, 2000). A network is said to be consistent if each reaction belongs to at least one elementary mode, or equivalently, there exists a mode such that all reactions belong to it. When a network is consistent, we say equivalently that its stoichiometric matrix is consistent: the stoichiometric matrix S is *consistent* if $Sv = 0$ has a solution with $v_j > 0 \ \forall j$.

We give an overview of some problems related to the consistency of stoichiometric matrices. If a matrix S is not consistent, this may indicate a case of incomplete modelling of the metabolic network. In that sense, detecting inconsistency is a valuable tool for finding deficiencies in the metabolic network description.

In the following theorems, we explicitly state that the problems can be solved using LP. Since the LP-formulations have a size that is bounded by a polynomial function of the stoichiometric matrix, we implicitly state that these problems are easy (in P). We choose for stating solvability through LP to emphasize that they are not only theoretically tractable but that in fact off-the-shelf computer packages can be used to solve the problems.

Theorem 4.6. *Given a stoichiometric matrix S, checking the consistency of S can be done using LP.*

Proof. Consider the following LP, where we insert a bound on the sum of the values of

the v_j's to avoid unboundedness of the problem.

$$\begin{aligned} \max \quad & z \\ \text{s.t.} \quad & v_j \geq z \ \forall j \\ & Sv = 0 \\ & \textstyle\sum_j v_j \leq 1 \end{aligned} \tag{4.1}$$

S is consistent if the optimal value is strictly positive, otherwise it is not. □

In case of inconsistency, it is also easy to find a consistent submatrix containing a maximum number of reactions.

Theorem 4.7. *Given a stoichiometric matrix S, detecting a minimum number of reactions to be deleted to make S consistent can be done using LP.*

Proof. For each reaction h, solve the LP

$$\begin{aligned} \max \quad & v_h \\ \text{s.t.} \quad & Sv = 0 \\ & \textstyle\sum_j v_j \leq 1 \\ & v \geq 0 \end{aligned}$$

If for reaction h, the optimal value is strictly positive, then h is part of some mode, and one such mode is given by the optimal solution. Otherwise there is no mode in which reaction h appears and it must be deleted to make S consistent. This is a safe operation: since h belongs to no mode, eliminating h will not eliminate any existing mode. For the same reason, the order of elimination is indifferent. □

Unfortunately, a problem complementary to the previous one is hard.

Theorem 4.8. *Given a stoichiometric matrix S, and some other set of reactions represented by a stoichiometric matrix S', find a subset of reactions of S' of minimum cardinality such that the corresponding submatrix added to S yields a consistent matrix is* NP*-hard.*

This is of practical interest as in general, when a stoichiometric matrix is not consistent, it is because some enzymes, and therefore some reactions, were not detected as present due to the lack of a strong enough similarity with the enzymes in a known

network, usually that of *Escherichia coli*, from which the one for a newly sequenced organism was inferred.

Proof. Taking as S an empty matrix and as S' the stoichiometric matrix of the network, the problem is a special case of finding an elementary mode with a minimum number of reactions in its support. NP-hardness of the latter problem will be established in Theorem 4.12. □

4.3 Finding elementary modes

4.3.1 Finding an elementary mode

As observed already in Klamt *et al.* (2005), just by substitution we can check that $Sv = 0$ in order to decide if $v \geq 0$ is a mode. It is also easy to decide if a given mode $v \geq 0$ is an elementary mode by calculating the rank of the submatrix of S consisting of the reactions in the support of v. If this is equal to the number of reactions in the support of v minus 1, then vector v represents an elementary mode (Klamt *et al.*, 2005). Finding one EM is also easy.

Theorem 4.9. *Given a stoichiometric matrix S, an elementary mode can be found in polynomial time.*

Proof. We "slice" the cone $\{v \geq 0 \mid Sv = 0\}$ by the inequality $\sum_j v_j \leq 1$ and solve the LP for an arbitrary reaction h:

$$\begin{aligned} \max \quad & v_h \\ \text{s.t.} \quad & Sv = 0 \\ & \textstyle\sum_j v_j \leq 1 \\ & v \geq 0. \end{aligned} \tag{4.2}$$

In case of a consistent matrix, there is an optimal solution which is a non-all-0 vertex of the polytope $\{v \geq 0 \mid Sv = 0, \sum_j v_j \leq 1\}$ satisfying the inequality $\sum_j v_j \leq 1$ with equality. Let v_h^* be the optimal solution value.

Using interior point methods to find an optimal solution does not necessarily yield a vertex of the polytope. However, if a vertex is not found, the objective function is set equal to the optimal value, $v_h = v_h^*$, and is added as a constraint. As an auxiliary objective, maximisation of one of the decision variables (other than v_h) is chosen.

In this way, iteratively applying an interior point method, in each such iteration the dimension of the optimal solution set is diminished by at least one. Thus, after a number of iterations less than the number of variables a vertex will be obtained, and we conclude that an elementary mode can be found in polynomial time. □

Of course, we can use also any simplex method-based LP-package for solving LP (4.2), since, although this is not in the worst-case a polynomial time method, it is very fast in practice. Moreover, it has the advantage that it will always produce directly a non-all-0 vertex of the polytope as an optimal solution.

4.3.2 Finding elementary modes with support containing a given set of reactions

An optimal solution of the LP in the proof of Theorem 4.9 gives an elementary mode that contains reaction h. In general, it is easy to detect if there exists a *mode* whose support contains a given set of reactions T_{IN}, and does not contain any of the reactions of another set T_{OUT}: simply add the restrictions:

$$v_j = 0 \ \forall j \in T_{OUT} \tag{4.3}$$

to LP (4.1), replace the first restriction of LP (4.1) by:

$$v_j \geq z \ \forall j \in T_{IN},$$

and check if the optimal solution is positive or 0.

The existence of an *elementary mode* with the same properties for any set T_{IN} is NP-complete in general, which may (partly) explain the difficulties we encounter in enumerating EMs.

Theorem 4.10. *Given a stoichiometric matrix S, sets of reactions T_{IN} and T_{OUT}, deciding if an elementary mode v exists that has positive value in all its coordinates corresponding to T_{IN}, and has value 0 in all its coordinates corresponding to the set T_{OUT} is* NP*-complete.*

Proof. We start by observing that this decision problem is in NP because, if we give a flux vector as certificate, we can check in polynomial time if it is an elementary mode with the desired properties. We observe also that the set T_{OUT} has no influence on

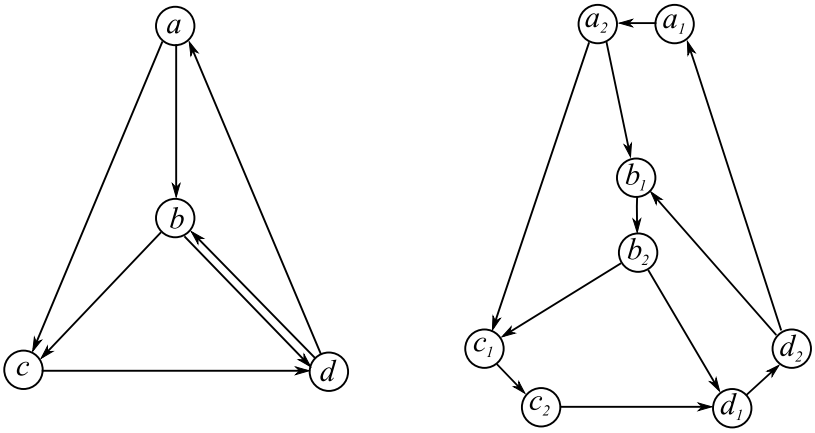

Figure 4.2: Graphical illustration of the DHC reduction.

the complexity of the problem. Indeed, we just need to delete from S the columns corresponding to the reactions in T_{OUT} and solve the problem on the reduced matrix.

NP-completeness is proved by a reduction from the DIRECTED HAMILTONIAN CYCLE problem (DHC). The intuition behind the proof is to build, in polynomial time, from a general DHC instance, a specific instance of our problem, that is a network, with the following characteristic: each elementary mode in the network that contains all reactions in T_{IN} corresponds to a hamiltonian cycle in the directed graph and vice versa. A solution to our problem therefore provides a solution to the NP-complete problem DHC.

Given a directed graph G that is a general DHC instance, for each vertex u in G, create two compounds u_1, u_2 and create a reaction from u_1 to u_2. For each edge (u, w) of G, create a reaction from u_2 to w_1 (see Figure 4.2). Let H be this network that can be built in linear time from G. The corresponding stoichiometric matrix is simply the $\{-1, 0, +1\}$ incidence matrix of this directed bipartite graph. Choose T_{IN} to be the set of all reactions corresponding to (derived from) vertices in G and $T_{OUT} = \emptyset$.

Notice that any directed cycle C in H corresponds to an elementary mode: just setting all values for the reactions corresponding to the arcs in C equal to 1 and the rest to 0 gives a mode. It is clear that no subset of the arcs can give rise to a mode, hence it must be an EM. Notice also that because of the absence of outputs in this network, any mode in H has to contain a directed cycle in its support. However, a directed cycle is, in fact, the support of an EM and therefore any EM must be a single directed cycle of H. Since directed cycles in H and G have a one-to-one correspondence, any EM corresponds to a directed cycle in G and vice versa.

In particular, there is a one-to-one correspondence between elementary modes of H that contain all of T_{IN} and directed cycles in G that contain all vertices of G, i.e., hamiltonian cycles. □

Although we showed the hardness of finding an elementary mode with support in T_{IN} in the general case, this does not solve the question for a fixed size of T_{IN}. Clearly, the problem is easy if $|T_{IN}| = 1$. Indeed, the proof follows from selecting h in the LP (4.2) as the only reaction in T_{IN}. On the other hand, we can observe that it becomes trivial when $|T_{IN}| > rank(S) + 1$. Indeed, according to Lemma 4 in Schuster *et al.* (2002a), no EM can have as many non-zero elements. This leaves an interesting and rather fundamental question: what is the complexity of the problem if $|T_{IN}| = k$ for any fixed $k, 1 < k \leq rank(S)+1$? In fact, we can show that it is NP-hard already for the case $k = 2$. This result is a corollary of Theorem 4.26 presented in Section 4.7. In that section, some complexity results are presented by establishing a relation between the extreme rays of a cone (and therefore EMs) and the cycles in directed weighted graphs.

Theorem 4.11. *Given two reactions r_i and r_j, deciding if there exists an elementary mode that has both r_i and r_j in its support is* NP*-complete.*

Proof. This is a direct consequence of Theorem 4.26 of Section 4.7.

□

Note that the complexity is the same for the problem of deciding if there is an elementary mode that passes through two given *compounds* (or through a given compound and a given reaction) instead of two given reactions. Indeed, both problems are equivalent: we can reduce one formulation to the other by just breaking the given reactions (respectively compound) in two steps and putting an extra compound (respectively reaction) connecting both. Analogously, deciding if there is an EM that passes through a given compound and a given reaction is also NP-hard.

4.3.3 Finding the shortest elementary modes

We present the next result about the complexity of detecting elementary modes with a support of small size.

Theorem 4.12. *Given a matrix S and a number k, deciding the existence of an elementary mode with at most k reactions in its support is* NP*-complete.*

Proof. Clearly this decision problem is in NP because, if we give a flux vector as certificate, we can check in polynomial time if it is an elementary mode with at most k reactions in its support. The proof of the hardness is a reduction from the NP-complete 3-DIMENSIONAL MATCHING problem (3DM) (see Garey et Johnson, 1979): Given a set of elements $X = \{x_1, \ldots, x_{3n}\}$ partitioned into three sets of n elements each, and given a collection of 3-element subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$ each subset containing exactly one element from each set of the partition, does there exist a subcollection of $\mathcal{S}$ of n subsets that covers all elements of X?

The reduction is depicted in Figure 4.3. For each element and each 3-element set of the 3DM instance, a compound vertex is created. The first reaction is an input reaction that has as output all elements of the 3DM instance, the grey vertices in Figure 4.3; *i.e.*, the first column of the stoichiometric matrix has 1-entries at all element compounds and 0 at all element set compounds. For each 3-element set of the 3DM instance a reaction is created with input the compounds corresponding to the three elements of the set and output the compound corresponding to the 3-element set, the s_i-nodes in Figure 4.3; *i.e.*, a column in the stoichiometric matrix with -1-entries at the three element compounds, 1 at the element set compound and 0's elsewhere. For each 3-element set there is also an output reaction that has the 3-element set compound as its only input. Finally we choose $k = 2n + 1$.

The vector of reactions which has a 1 at the positions of the first reaction and the two reactions corresponding to each 3-element set of any 3-dimensional matching and 0's elsewhere, clearly forms an elementary mode with $2n + 1$ reactions in its support. On the other hand, any mode must contain the first reaction. Hence, any EM must have a positive value in the first position, and therefore has as output exactly one copy of each element, all of which must have the same value. For every 3-element-set-reaction that we choose, we have to add the corresponding output reaction. Thus to cover all $3n$ elements from the first reaction, we have to choose exactly n reactions that correspond to 3-element sets. Such a set of reactions corresponds to a 3-dimensional matching. □

This theorem shows that finding a *shortest* elementary mode (one with a minimum number of reactions) is NP-hard. Observe that in the theorem, k is considered to be part of the input. For fixed values of k, the problem is trivially solvable in polynomial time by complete enumeration. In practice, enumerating EMs with at most k reactions may therefore be possible for small values of k. To the best of our knowledge, there is no current application of short EMs, but it should become interesting if size were

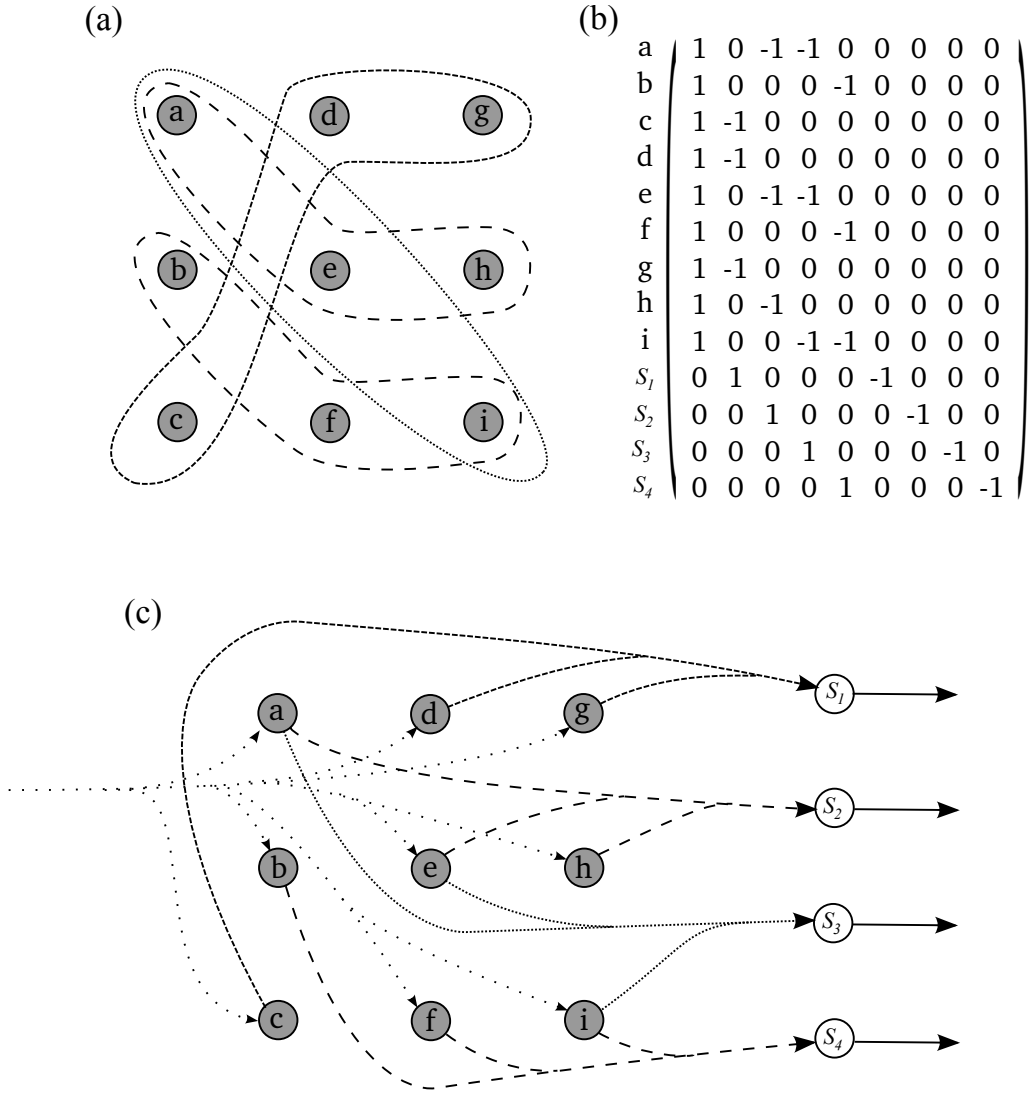

Figure 4.3: Graphical illustration of the 3DM reduction. (a) An instance of 3DM with $X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}\}$ partitioned in three sets $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, $\{\mathtt{d}, \mathtt{e}, \mathtt{f}\}$ and $\{\mathtt{g}, \mathtt{h}, \mathtt{i}\}$. The collection $\mathcal{S}$ contains four 3-element-subset. (b) The stoichiometric matrix of the reduction. (c) Graphical representation of the network. Every EM with at most 7 reactions (for instance, the EM containing the input reaction and the reactions using the compounds S_1, S_2 and S_4) corresponds to a 3-dimensional matching.

considered as a relevant criterion to classify EMs. Short EMs may also be seen as good seeds for a motif detection algorithm such as in Lacroix *et al.* (2006): two (or more) short EMs that represent connected sets of equivalent chemical transformations (equivalent enzymatic functions) may help to understand how metabolism evolved. In any case, Theorem 4.12 is also interesting in itself for the further insight it provides in the hardness of elementary mode computations.

As a final example to illustrate the intricacies in detecting elementary modes, we define the notion of a *simple elementary mode* as an EM v such that $\forall j,\ v_j \in \{0,1\}$. The reduction in the proof of Theorem 4.12 shows that it is hard to find simple EMs. Though it is unlikely that any biological relevance will ever be found for the notion of simple EM, this result shows again the subtlety of EM computations, all the more so as the hardness can be extended to any fixed interval of integers.

Corollary 4.13. *Given a matrix S, deciding the existence of a simple elementary mode is* NP*-complete.*

4.4 Counting elementary modes

System biologists are interested in enumerating all elementary modes of a metabolic network. Before turning to that problem, we show that merely counting EMs is hard. In Klamt et Stelling (2002) the authors show that the number of EMs can be bounded by $\binom{m}{n+1}$, but they did not give the complexity of computing the exact number.

Counting elementary modes is essentially a problem of counting the rays of a polyhedral cone, which in its turn is equivalent to a problem of counting vertices of a polytope, which is known to be ♯P-complete (Dyer, 1983)[1] in general.

Not surprisingly, given its connection with the extreme rays of a convex cone, counting elementary modes turns out to be also ♯P-complete.

Theorem 4.14. *Given a matrix S counting the number of elementary modes is* ♯P*-complete.*

Proof. The proof follows by a reduction from the ♯P-complete problem COUNT PERFECT MATCHINGS, that is, counting perfect matchings in a bipartite graph (Valiant, 1979). Given a bipartite graph $G = (U, V, E)$ with two colour classes U and V, each of size n, we construct the following hypergraph H. First, we create an input compound vertex s, which we connect with one hyperedge r_s to all vertices in U, and direct this

[1] In fact, Dyer (1983) only claims NP-hardness, but the proof establishes #P-completeness.

hyperedge from s into U. We direct all edges of E from U to V. Finally, we create an output compound vertex t which we connect with one hyperedge r_t to all vertices of V, and direct this hyperedge from V into t. This relates in the obvious way to a $\{-1, 0, +1\}$-stoichiometric matrix.

Clearly any perfect matching in G corresponds to an EM. In fact, this relation is one-to-one. In effect, let x be an elementary mode with $R(x)$ its support. Clearly the flux over the reaction r_s is equal to the flux over r_t and w.l.o.g we can suppose this flux equal to one. For any set $\bar{U} \subseteq U$ we define the set $R_x(\bar{U})$ of reactions in the support of x that consume compounds of $\bar{U}$, that is,

$$R_x(\bar{U}) = \{r \in R(x) \mid r = (\{u\}, \{v\}) \text{ and } u \in \bar{U}\}.$$

For the set $\bar{U}$ we also define the set $N_x(\bar{U}) \subseteq V$ of compounds produced by $R_x(\bar{U})$. If we prove that for any set $\bar{U}$ we have that $|N_x(\bar{U})| \geq |\bar{U}|$ then we prove that $R(x)$ contains a perfect matching between U and V (see Hall (1935)). Thus, by minimality of elementary modes, we conclude that $R(x)$ corresponds exactly to a perfect matching plus the reactions r_s and r_t.

We show that $|N_x(\bar{U})| \geq |\bar{U}|$. Since the flux over r_s is equal to one, the sum of the fluxes over reactions in $R_x(\bar{U})$ must be exactly $|\bar{U}|$. Then, by the stoichiometric values, the total production of these reactions (that is, the total production of the compounds $N_x(\bar{U})$ by the reactions in $R_x(\bar{U})$) is also equal to $|\bar{U}|$. But the production of each compound in $N_x(\bar{U})$ cannot be more than one (since the flux over r_t is equal to one). Therefore we conclude that $|N_x(\bar{U})| \geq |\bar{U}|$.

□

4.5 Enumerating elementary modes

Enumerating the elementary modes of a metabolic network with stoichiometric matrix A for some $m \times n$ matrix A, corresponds exactly to enumerating the extreme rays of the cone $\{x \in \mathbb{R}^n \mid Ax = 0, x \geq 0\}$. An extreme ray of a cone is a vector of the cone that cannot be expressed as a convex combination of any two other vectors of the cone. The cone is pointed in the origin 0 of $\mathbb{R}^n$. Therefore, its extreme rays correspond one-to-one to the vertices of the bounded polyhedron $\{x \in \mathbb{R}^n \mid Ax = 0, \underline{1}^T x = 1, x \geq 0\}$, with $\underline{1}$ denoting the all-1 vector in $\mathbb{R}^n$. As a result, enumerating the extreme rays of the cone is not harder than enumerating the vertices of a bounded polyhedron (polytope). Since the number of objects to be enumerated can be exponential in the size of the

input, the complexity in terms of running time is measured as a function of the size of the input and of the output.

4.5.1 Enumerating elementary modes with a given reaction in its support

The complexity of enumerating vertices of polytopes is a famous and long-standing open question (see e.g., Dyer et Proll, 1977). We do not solve this question but present an intriguing related result: Given a coordinate i, enumerating all extreme rays r of the cone that have $r_i > 0$ cannot be done *in polynomial total time* (that is, polynomial in the size of the input and of the output) unless P = NP. This in terms of elementary modes, can be written as:

Theorem 4.15. *Given a reaction r, enumerating all elementary modes that have r in the support cannot be done in polynomial total time unless* P=NP.

Proof. This is a direct consequence of Theorem 4.25 of Section 4.7. □

Clearly, the complexity of this problem remains the same if we consider enumeration of elementary modes passing through a given *compound* instead of reaction (both problems are equivalent).

4.5.2 Analysis of the complexity result

It may seem strange that enumerating the elementary modes passing through a given reaction is hard while the complexity of enumerating *all* elementary modes remains unknown. This apparent contradiction comes from the fact that time is measured in terms of the output size. Given the "normalisation" effect introduced by this, enumerating a smaller subset of objects could therefore be harder than enumerating the whole set. Nevertheless, the hardness of enumerating a specific subset of the elementary modes gives some intuition on the difficulty of enumerating the whole set of them.

This result is in fact rather surprising. Although nobody has enough confidence to call it a conjecture, most people who have done theoretical research in this field guess that enumerating vertices of polytopes should be achievable in *polynomial total time* (see for a definition Section 2.6). If, contrary to this guess, enumerating vertices of polytopes will appear to be hard, it will be caused by degeneracy, since enumerating vertices of non-degenerate polytopes can be done in polynomial total time by a Local Reverse Search method (Dyer, 1983). Cones corresponding to real-life stoichiometric

matrices appear to be highly degenerate (see Terzer, 2009). Therefore, for enumerating extreme rays of the cone, variations of the *double description method* of Motzkin *et al.* (1953) are the most popular ones in the analysis of stoichiometric metabolic networks (Terzer, 2009). Where local reverse search methods suffer from degeneracy, double description methods suffer from generating intermediate (candidate) vectors that do not appear in the output.

4.5.3 Case when all reactions are reversible

Despite the difficulty of establishing the complexity of enumerating elementary modes for the general case, there is a positive result in the particular case when all reactions are *reversible*. Indeed, in this case an elementary mode corresponds to a minimally dependent set of columns of the stoichiometric matrix. Hence the elementary modes are exactly the *circuits* of a *linear matroid* (for definitions of matroids and circuits we refer to Oxley (1992) or Schrijver (2003)). It was shown in Khachiyan *et al.* (2005) that k circuits of a matroid (with k no greater than the total number of circuits) can be computed in polynomial time in k and n, with n the number of elements in the ground set of the matroid; in our case the number of reactions, that is, columns of the stoichiometric matrix. As a result, elementary modes of a completely reversible network, can be enumerated in incremental polynomial time.

Theorem 4.16. *In case all reactions in a metabolic network are reversible, the elementary modes can be enumerated in incremental polynomial time.*

4.6 Reaction cuts

In this section, we focus on Reaction Cut Sets. The notion of minimal cut sets in a metabolic network represented as a hypergraph was first introduced by Klamt et Gilles (2004). The motivation is to study so-called "failure modes" that render the functioning of a given target reaction r_t impossible. A minimal cut set is a set of reactions that must be cut (removed) in order to prevent a flux through the target reaction r_t. Operationally, this has been defined as a set of reactions whose deletion from the network stops each elementary mode that contains r_t.

Before proceeding we mention that the notion of a s, t-cut of a hypergraph, i.e., a cut that separates nodes s and t, has been proposed and studied for directed hypergraphs. In Gallo *et al.* (1998), it was observed that finding s, t-cuts in unweighted directed

hypergraphs can be done in polynomial time if all hyperedges are defined by a subset of input nodes and a single destination node; in the context of metabolic networks this would model the situation in which each reaction is irreversible and produces a single metabolite. We also refer to Ausiello *et al.* (2001) for a survey of related results on directed hypergraphs.

In what follows, we study two problems: finding a reaction cut of minimum cardinality, which we call Min Reaction Cut, and enumerating all minimal reaction cuts. We prove that Min Reaction Cut cannot be approximated within any constant approximation ratio unless P = Np.

4.6.1 Finding minimal and minimum reaction cuts

The first basic problem about reaction cuts is recognising them.

Theorem 4.17. *Given a stoichiometric matrix S, some target reaction r_t, and a subset F of reactions, deciding if F is a reaction cut of r_t can be done using LP.*

Proof. Consider the following LP:

$$\begin{aligned} \max \quad & v_{r_t} \\ \text{s.t.} \quad & Sv = 0 \\ & v_j = 0 \ \forall j \in F \\ & \textstyle\sum_j v_j \leq 1 \\ & v_j \geq 0 \ \forall j \notin F \cup r_t. \end{aligned}$$

The optimal solution value is positive if and only if F is not a reaction cut of r_t. □

Finding an optimal cut is a lot more difficult.

Theorem 4.18. Min Reaction Cut *is* Np*-hard.*

Proof. We prove Np-completeness of the decision version of Min Reaction Cut. By the previous theorem this problem is in Np. Completeness is proved through a reduction from Hitting set: Given a set of elements $X = \{x_1, \ldots, x_n\}$, a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$, and an integer K, does there exist a subset $Y \subset X$ of at most k elements such that $S_i \bigcap Y \neq \emptyset \ \forall i = 1, \ldots, m$.

We construct a metabolic network whose stoichiometric matrix contains only entries with values −1,0, or +1. For each element x_j and for each set S_i, we create a compound

vertex, which we also denote by x_j and S_i, respectively. We create three additional compounds s, t and t'. The compounds s and t' are considered as external and we create the reaction $(t \to t')$ as the target reaction r_t. For each x_j, we create a reaction $(s \to x_j)$. Similarly, for each set $S_i = \{x_{i_1}, \ldots, x_{i_k}\}$, we create a reaction $(S_i \to t)$ and another with multiple input compounds $x_{i_1}, \ldots, x_{i_k}$ and output compound S_i. We select for the decision version of Min Reaction Cut the same integer K as in the Hitting Set instance.

To each set $S_i = \{x_{i_1}, \ldots, x_{i_k}\}$ corresponds an elementary mode consisting of the reactions $(s \to x_{i_1}), \ldots, (s \to x_{i_k}), (x_{i_1}, \ldots, x_{i_k} \to S_i), (S_i \to t), (t \to t')$. Indeed, it is easy to check that the vector that assigns a 1 to each of these reactions and a 0 otherwise is indeed a mode. Removing any reaction from this set gives a submatrix which does not have any mode.

Moreover, suppose that some mode contains reactions corresponding to two sets, that is, $v(S_i \to t) = a_i > 0$, $v(S_j \to t) = a_j > 0$ and $v(S_l \to t) = 0\ \forall l \notin \{i, j\}$. Then this mode should also have $v(x_{i_1}, \ldots, x_{i_k} \to S_i) = a_i$ and $v(x_{j_1}, \ldots, x_{j_l} \to S_j) = a_j$, and also $v(t \to t') = a_i + a_j$ and $v(s \to x_\ell) = a_i\ \forall x_\ell \in (S_i \backslash S_j)$, $v(s \to x_\ell) = a_j$ $\forall x_\ell \in (S_j \backslash S_i)$, $v(s \to x_\ell) = a_i + a_j\ \forall x_\ell \in (S_i \cap S_j)$, and $v(s \to x_\ell) = 0$ otherwise. Hence this is the linear combination of two elementary modes of the above type, and therefore by itself not an elementary mode. Clearly, the same reasoning holds if a mode were to correspond to more than two sets. If we suppose the existence of an elementary mode containing k set nodes, with $2 < k \leq m$, we can similarly show that it can be written as a linear combination of the k corresponding elementary modes of the above type.

Thus, the elementary modes corresponding to the sets of $\mathcal{S}$ are exactly all the elementary modes, and from each of them some reaction must be selected in the reaction cut. Selecting $(s \to x_\ell)$ cuts all the elementary modes whose corresponding set contains x_ℓ. This immediately implies that given a hitting set of size at most K, the reactions from s to the x's of this hitting set cut all elementary modes and therefore form a reaction cut of size at most K.

On the other hand, any reaction $(x_{i_1}, \ldots, x_{i_k} \to S_i)$ or $(S_i \to t)$ in a reaction cut can be replaced by one reaction $(s \to x_j)$ (with $x_j \in S_i$), giving another reaction cut. Thus for any reaction cut of size at most K, there exists a reaction cut of the same size consisting only of reactions of type $(s \to x_j)$, hence corresponding to a hitting set of size at most K. □

The above reduction yields a one-to-one correspondence between minimal reaction

cuts of size K and hitting sets of size K. Therefore it is approximation preserving (see for a precise definition of an approximation preserving reduction *e.g.* Ausiello *et al.*, 1999). Because of its equivalence to SetCover in which elements have to be covered by sets, no polynomial time algorithm for HittingSet can have approximation ratio $O(\log n)$ unless P = Np (Raz et Safra, 1997), with n the number of elements. The following inapproximability result follows directly.

Theorem 4.19. *Any polynomial time approximation algorithm for* Min Reaction Cut *cannot have approximation ratio* $o(\log n)$*, with* n *the number of reactions, unless* P = Np. □

4.7 Proof of Theorem 4.11 and Theorem 4.15

Both results are based on a reduction to the decision problem on the existence of negative simple cycles in directed graphs and are inspired by the work of Khachiyan *et al.* (2008), who proved that enumerating vertices of any (possibly unbounded) polyhedron cannot be achieved in polynomial total time unless P=Np. Of course, Khachiyan et al.'s result does not apply to *polytopes*, which could still be easier than the general case.

Given a directed graph, or network, $G = (V, E)$, each column of its node-arc incidence matrix M corresponds to an arc $(u, v) \in E$ and contains exactly one -1 in the row of its tail-node u, and exactly one $+1$ in the row of its head-node v, and otherwise 0 entries. We call a cycle simple if it does not contain subcycles.

Lemma 4.20. *Let* $G = (V, E)$ *be a directed graph with the node-arc incidence matrix* M*, then the extreme rays of the cone* $\{x \in \mathbb{R}^{|E|} \mid Mx = 0, x \geq 0\}$ *correspond one-to-one to the directed simple cycles of* G*.*

Proof. In graph optimisation, a vector $x \geq 0$ that satisfies $Mx = 0$ is called a *flow circulation*. The flow decomposition lemma (see e.g. Schrijver, 1986) states that any such vector x can be written as a positive linear combination of characteristic vectors of directed simple cycles of G. Thus, the set of all characteristic vectors of directed simple cycles of G contains all extreme rays of the cone. It is also obvious that the characteristic vector of any simple cycle cannot be written as a positive linear combination of vectors of other simple cycles. □

In a *weighted directed graph*, a function $w : E \to \mathbb{R}$ assigns weights to arcs. The total weight of a set of arcs is the sum of the weights of the arcs. We say that a cycle

C is negative if its total weight is negative. We create a matrix M' by appending an extra row to the node-incidence matrix M of G. In the column corresponding to arc $e \in E$, the entry in the extra row is $w(e)$. The extra row could be seen as corresponding to a dummy node d, and a column as representing a directed hyperarc with a weight on the extra node. In terms of stoichiometry, this would correspond to a reaction transforming 1 molecule of compound u into 1 molecule of compound v and $w(u, v)$ molecules of compound d in case $w(u, v) > 0$, or transforming 1 molecule of compound u and $w(u, v)$ molecules of compound d into 1 molecule of compound v in case $w(u, v) < 0$. We append two extra columns to M': the first one is the unit vector of the dummy node and the second is its negative. In stoichiometric terms, these can be regarded, respectively, as a reaction that produces 1 molecule of compound d from external nutrients and a reaction that excretes 1 molecule of compound d. To facilitate the exposition, we denote the two arcs between the dummy node and "some invisible external node", respectively, e^+ and e^-. We call the resulting matrix M^+. In Figure 4.4, we present an example of the matrix M^+.

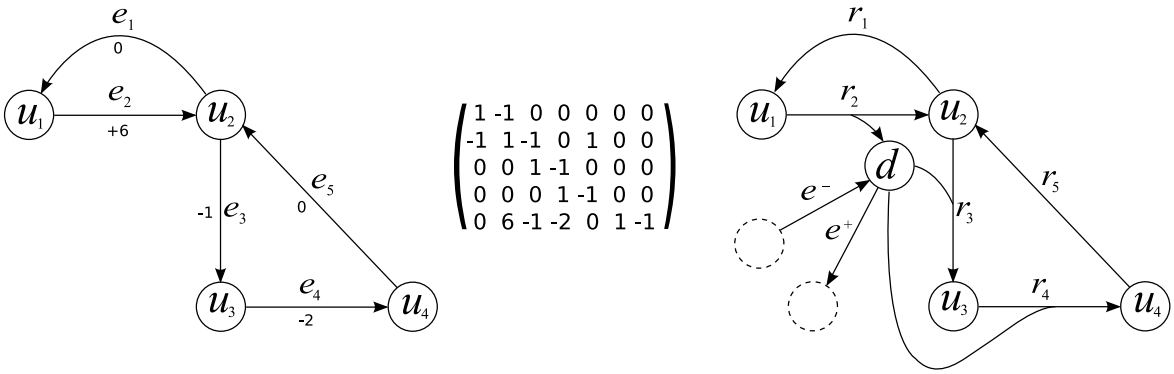

Figure 4.4: Transformation of a weighted directed graph into a metabolic network with stoichiometry. *Left:* A weighted directed graph G. *Center:* The matrix M^+ of G. *Right:* The set of reactions that represent the stoichiometric matrix M^+ (stoichiometry is not shown).

Given a vector x we denote by $R(x)$ its support, i.e., the set of non-zero coordinates of x. As observed by many researchers, the extreme rays of the cone $\{x \in \mathbb{R}^n \mid Ax = 0, x \geq 0\}$ are exactly the vectors in the cone with minimal support and they are uniquely characterised by their support, up to a positive scalar multiplication.

In the context of the cone related to directed graphs, we index the coordinates of the vectors by the arcs to which they correspond and write the support of a vector as a subset of arcs. The following relations exist between directed simple cycles of G and extreme rays of the cone $\Gamma =: \{x \in \mathbb{R}^{|E|+2} \mid M^+x = 0, x \geq 0\}$.

Lemma 4.21. *Let $G = (V, E, w)$ be a directed weighted graph. For every extreme ray x of the cone Γ, either $R(x) = \{e^+, e^-\}$, or $R(x) \setminus \{e^+, e^-\}$ is the union of simple directed cycles of G.*

Proof. Let x be an extreme ray of the cone Γ such that $R(x) \neq \{e^+, e^-\}$. Let $x' \in \mathbb{R}^{|E|}$ be the truncated vector x without the values corresponding to the arcs e^+ and e^-. Then, $x' \neq 0$ and $Mx' = 0$ where M is the node-arc incidence matrix of G. The vector x' belongs to the cone $\{x \in \mathbb{R}^{|E|} \mid Mx = 0, x \geq 0\}$ and therefore $x' \neq 0$ is a positive linear combination of extreme rays of this cone. By Lemma 4.20, the support of x' is the union of simple directed cycles of G. Therefore, $R(x) \setminus \{e^+, e^-\}$ is the union of simple directed cycles of G. □

Lemma 4.22. *Let $G = (V, E, w)$ be a directed weighted graph. Let C be a directed simple cycle.*

- *if C is negative then $C \cup \{e^-\}$ is the support of an extreme ray of Γ;*
- *if C is positive then $C \cup \{e^+\}$ is the support of an extreme ray of Γ;*
- *otherwise C is the support of an extreme ray of Γ.*

Proof. If C is negative, that is $w(C) = \sum_{e \in C} w(e) < 0$, we consider the vector x defined by

$$x_i = \begin{cases} 1 & \text{if} \quad i \in C \\ -w(C) & \text{if} \quad i = e^- \\ 0 & \text{otherwise} \end{cases}$$

Clearly, x is in the cone $\Gamma =: \{x \in \mathbb{R}^{|E|+2} \mid M^+x = 0, x \geq 0\}$ and has support $R(x) = C \cup \{e^-\}$. We show that x is an extreme ray of Γ. Suppose it is not. Then it is a positive linear combination of extreme rays of Γ. Thus, there must exist an extreme ray x' such that $R(x') \subseteq R(x)$ and $R(x') \ni e^-$. By Lemma 4.21, $R(x') \setminus \{e^-\}$ is the union of directed simple cycles of G. But $R(x') \setminus \{e^-\} \subseteq R(x) \setminus \{e^-\} = C$, a directed simple cycle C. We conclude that $R(x') = R(x) = C \cup \{e^-\}$. The proof is analogous for the case that C is positive.

Now, if the cycle has weight 0, then we choose in $\Gamma(G)$ the vector x defined as

$$x_i = \begin{cases} 1 & \text{if} \quad i \in C; \\ 0 & \text{otherwise} \end{cases}$$

Arguing in a similar way as in the previous case, we conclude that x is an extreme ray of Γ with $R(x) = C$. □

Lemma 4.23. *Let $G = (V, E, w)$ be a directed weighted graph. Let x be an extreme ray of Γ. Then exactly one of the following possibilities is true:*

- $R(x) = \{e^+, e^-\}$;
- $R(x)$ *is a union of directed simple cycles;*
- $R(x) = C \cup \{e^-\}$ *where C is a negative directed simple cycle of G;*
- $R(x) = C \cup \{e^+\}$ *where C is a positive directed simple cycle of G.*

Proof. If $R(x)$ contains both e^+ and e^- then, because of minimality of support, $R(x) = \{e^+, e^-\}$. If $R(x)$ and $\{e^+, e^-\}$ are disjoint sets, then by Lemma 4.21 the set $R(x)$ is a union of simple cycles.

Suppose that $R(x)$ contains e^- but does not contain e^+. Let us consider $F = R(x) \setminus \{e^-\}$. By Lemma 4.21, $R(x) \setminus \{e^-\}$ is a union of simple cycles. Let C be one of them. We show that C is negative and $C = R(x) \setminus \{e^-\}$.

If not and C has weight 0, then by Lemma 4.22, C is the support of an extreme ray that does not contain e^-. This contradicts the fact that x is an extreme ray.

If not and C is positive, $C \cup \{e^+\}$ is the support of an extreme ray, say x'. Let $\alpha = x'_{e^+} > 0$ be the value of the coordinate e^+ of x'. Let $\beta = x_{e^-} > 0$ be the value of the coordinate e^- of x. Let $z \in \mathbb{R}^{|E|+2}$ be the vector such that $z_i = 1$ if $i \in \{e^-, e^+\}$ and $z_i = 0$ otherwise. The vector $y = (1/\alpha)x' + (1/\beta)x - z$ is non-negative and $M^+y = 0$. Therefore, $y \in \Gamma$ and $R(y) = C \subseteq R(x) \setminus \{e^-\}$. This contradicts the fact that x is an extreme ray. We conclude that C is a negative simple cycle and therefore $C \cup \{e^-\} = R(x)$.

Analogously, if $R(x)$ contains e^+ but does not contain e^-, then the support of x is $C \cup \{e^+\}$ for some positive simple cycle C. □

As a corollary, we obtain the following crucial observation for our results.

Theorem 4.24. *Let $G = (V, E, w)$ be a directed weighted graph and let $F \subset E$. Then the following two statements are equivalent:*

- F *is a negative simple directed cycle;*
- $F \cup \{e^-\}$ *is the support of an extreme ray of Γ.*

Our first result follows directly from this theorem in combination with a result from Khachiyan *et al.* (2008).

Theorem 4.25. *Given a cone* $\{x \in \mathbb{R}^n \mid Ax = 0, x \geq 0\}$ *and a coordinate* i*, enumerating all extreme rays that contain* i *in their support is not in PT unless* P=NP.

Proof. Khachiyan *et al.* (2008) showed that enumerating negative cycles of a weighted directed graph G is not in PT unless P=NP. By Theorem 4.24, searching for negative cycles in G is equivalent to searching for extreme rays of Γ having e^- in their support. Since, given G, we can construct Γ in polynomial time, the theorem follows. □

The proof of the hardness of enumerating all negative cycles in a directed weighted graph made by Khachiyan *et al.* (2008) is done by a reduction from the CNF satisfiability problem. The proof also shows (this is not explicitly mentioned in the paper) that the problem of *deciding if there exists a negative cycle that uses a given arc* u is NP-hard. In fact, from an instance ϕ of the CNF satisfiability problem, the authors construct a weighted directed graph G such that there is a one-to-one correspondence between a positive assignment of ϕ and a negative cycle that uses a particular arc of G (called (u_{m+n}, u_0) in the proof). We use this result to prove the following theorem.

Theorem 4.26. *Given a cone* $\{x \in \mathbb{R}^n \mid Ax = 0, x \geq 0\}$ *and two coordinates* i *and* j*, deciding if there exists an extreme ray of the cone that has both* i *and* j *in its support is* NP*-complete.*

Proof. Verifying that a vector $x \in \mathbb{R}^n$ is an extreme ray can be done in polynomial time, hence the problem is in NP. On the other hand, we know from Khachiyan *et al.* (2008) that the problem of *deciding if there exists a negative cycle in a graph that uses a given arc* u is NP-hard. By Theorem 4.24, this is equivalent to deciding if there exists an extreme ray of Γ that contains e^- and u in its support. Since, given G, we can construct Γ in polynomial time, the theorem follows. □

Chapter 5

Modelling Precursor Sets in Metabolic Networks

Contents

5.1 Definitions and Characterisations

5.1.1 Modelling a metabolic network

As seen in Chapter 3, a metabolic network is modelled as a *directed hypergraph* $G = (\mathcal{C}, \mathcal{R})$ with $\mathcal{C}$ the set of *vertices* corresponding to metabolites and $\mathcal{R}$ the set of *hyperarcs* corresponding to reactions. Again, reactions are supposed to be irreversible: each originally reversible reaction is therefore considered as two different irreversible reactions of opposite direction.

The set $\mathcal{S}$ denotes a particular subset of the compounds, called *sources*, that are *potentially available* in infinite supply (for instance, from the environment). Sources used as substrates of reactions produce other metabolites, thereby increasing the set of available ones. On the other hand, set T denotes the *target set*, that is a set of compounds that it is interesting to produce.

Given a specific target set T of metabolites, the aim is to find a subset of the sources, which is able to produce all the metabolites of T.

5.1.2 Forward propagation

We want to define which metabolites become available when a subset of the sources is available. To this aim, we introduce first the notion of *next available compounds* that corresponds to the set of all compounds that can be produced from a set M of metabolites using one reaction. For any set S we use $\mathcal{P}(S)$ to denote its power set.

Definition 5.1. (Next available compounds) *Given $M \in \mathcal{P}(\mathcal{C})$ a set of metabolites, the* next available compounds *from M, denoted by $Next(M) \in \mathcal{P}(\mathcal{C})$, is the set of all metabolites y for which there exists $r \in \mathcal{R}$ with $Subs(r) \subseteq M$ and $y \in Prod(r)$.*

Let f be a function $f : \mathcal{P}(\mathcal{C}) \to \mathcal{P}(\mathcal{C})$. We say that f is *order-preserving* if, for any pair of sets $A, B \in \mathcal{P}(\mathcal{C})$ such that $A \subseteq B$ we have $f(A) \subseteq f(B)$. We say that f is *increasing* (resp. *decreasing*) if for all $A \in \mathcal{P}(\mathcal{C})$, $f(A) \supseteq A$ (resp. $f(A) \subseteq A$). We say that f is *monotone* if it is increasing or decreasing.

Given a monotone function $f : \mathcal{P}(\mathcal{C}) \to \mathcal{P}(\mathcal{C})$, we define the function $f^k(M) = f(f^{k-1}(M))$ (with $f^1(M) = f(M)$). We define f^* as the successive application of f until a fixed point is reached, that is, $f^*(M) = f^k(M)$ for any k such that $f^k(M) = f^{k+1}(M)$.

Definition 5.2. (Forward propagation) *Given $M \in \mathcal{P}(\mathcal{C})$, a set of metabolites, and the monotone increasing function f such that $f(M) = M \cup Next(M) \in \mathcal{P}(\mathcal{C})$, the* forward propagation *of M is defined as $FP(M) = f^*(M)$, that is, as the successive application of f until a fixed point is reached.*

For instance, in the network of Figure 5.1, the forward propagation of the sets $\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ and $\{\mathsf{B}, \mathsf{C}, \mathsf{D}\}$ are $FP(\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}) = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{E}, \mathsf{H}, \mathsf{I}, \mathsf{T}\}$ and $FP(\{\mathsf{B}, \mathsf{C}, \mathsf{D}\}) = \{\mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}$.

5.1.3 Definition of precursor sets considering cycles

Using the definition of forward propagation, Romero and Karp considered a subset X of the sources $\mathcal{S}$ as a precursor set of a target T, when $T \subseteq FP(X)$ (Romero et Karp, 2001). This iterative way to calculate what is available from X may however not be enough to model the real process. Indeed, the network could have cycles whose metabolites need to be consumed and produced *all at the same time*. These cycles will not be reached using the forward propagation definition.

Let us consider the network of Figure 5.1 with A, B, C and D as sources and $T = \{\mathsf{T}\}$ as target set. The set $\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ is a precursor set of T because if A, B and C are available, then T can be produced by the iterative use of reactions r_4, r_5 and r_6. However, in the restricted definition given initially, the set $\{\mathsf{C}, \mathsf{D}\}$ is not able to generate T. Indeed, reaction r_1 needs also the unavailable metabolite F to produce G. In the new model we propose, the set $\{\mathsf{C}, \mathsf{D}\}$ is considered as a precursor set of the target $\{\mathsf{T}\}$ because they are able to fire off the cycle r_1, r_2, maintaining F and G available as long as C and D are, and producing successively H and T.

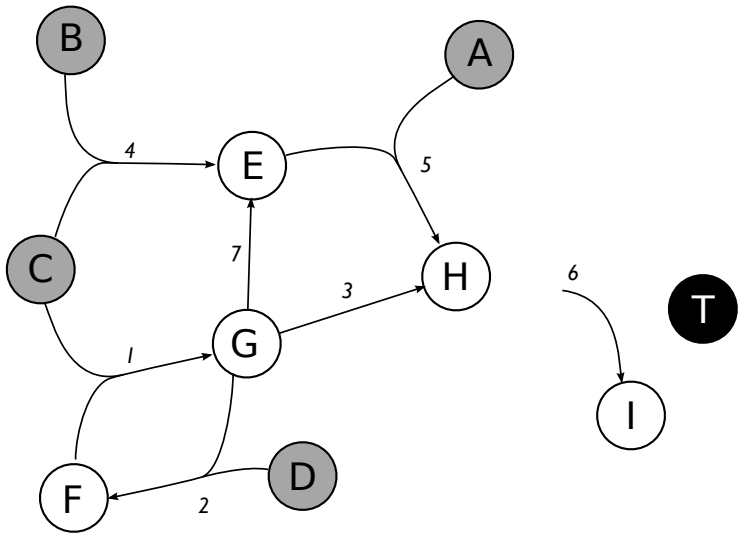

Figure 5.1: A metabolic network G with set of metabolites $\mathcal{C} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{F}, \mathsf{G}, \mathsf{H}, \mathsf{I}, \mathsf{T}\}$ and set of reactions $\mathcal{R} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$

One way to formulate this notion is to consider that some set Z of metabolites not in X (for instance $Z = \{\mathsf{F}\}$ in the example of Figure 5.1), are available at the beginning of the process. Then, we check if with this extra supply, the target is produced and Z regenerated (to maintain the cycles working). In other words, the forward propagation of X with Z needs to produce T and Z. To regenerate Z means, in terms of the concept as defined, that $Z \subseteq Next(FP(X \cup Z))$. If it does and $T \subseteq FP(X \cup Z)$ (or

equivalently $T \subseteq X \cup Next(FP(X \cup Z)))$, then X is a precursor set of the target and we call Z an *internal supply* of the solution.

Definition 5.3. (Precursor set) *A set of sources $X \in \mathcal{P}(\mathcal{S})$ is a* precursor set of *$T \in \mathcal{P}(\mathcal{C})$ if and only if there exists a set $Z \in \mathcal{P}(\mathcal{C})$ such that*

$$T \cup Z \subseteq X \cup Next(FP(X \cup Z))$$

In that case, we call Z an internal supply *of the precursor set X.*

In the network of Figure 5.1, if $X = \{\mathsf{C}, \mathsf{D}\}$ and $Z = \{\mathsf{F}\}$, then $FP(X \cup Z) = \{\mathsf{C}, \mathsf{D}, \mathsf{F}, \mathsf{G}, \mathsf{E}, \mathsf{H}, \mathsf{I}, \mathsf{T}\}$ and $Next(FP(X \cup Z)) = \{\mathsf{F}, \mathsf{G}, \mathsf{E}, \mathsf{H}, \mathsf{I}, \mathsf{T}\}$. Since this last set contains T and Z, then X is a precursor set of T.

We call attention to the fact that this definition leads to a larger number of precursor sets of T than those identified by the methods available in the literature which do not allow for the possibility of an internal supply. Of course, the internal supply may not be unique for a given precursor set. In Figure 5.1, we may verify that both $Z = \{\mathsf{F}\}$ and $Z = \{\mathsf{G}\}$ are internal supplies for the precursor set $X = \{\mathsf{C}, \mathsf{D}\}$ of target $\{T\}$. To identify a set of precursors as being a precursor set of a given target set, it suffices to find one set of internal supplies.

5.1.4 Alternative characterisation of precursor set

There is a simpler way to characterise a precursor set of T. Indeed, if Z is contained in the set $X \cup Next(FP(X \cup Z))$, then the whole set $FP(X \cup Z)$ is also contained in it. This motivates the next property that gives an alternative and equivalent definition of precursor set which will be useful for simplifying several arguments in the sequel.

Lemma 5.4. *A set of sources $X \in \mathcal{P}(\mathcal{S})$ is a precursor set of $T \in \mathcal{P}(\mathcal{C})$ if and only if there exists a set $A \in \mathcal{P}(\mathcal{C})$ such that*

$$T \subseteq A \subseteq X \cup Next(A).$$

Proof. Note that for any set $C \in \mathcal{P}(\mathcal{C})$, we have that $FP(C) = C \cup Next(FP(C))$. If X is a precursor set of T, then by definition there exists Z such that $T \cup Z \subseteq X \cup Next(FP(X \cup Z))$. Let $A = FP(X \cup Z)$, then A is a superset of X and of $Next(FP(X \cup Z))$, hence A is a superset of T. Since, by definition of FP, $A = (X \cup Z) \cup Next(A)$ and since $Z \subseteq Next(A) \cup X$ we have $A \subseteq X \cup Next(A)$.

Conversely, suppose that there is a set A such that $T \subseteq A \subseteq X \cup Next(A)$. Since $Next$ is order-preserving and FP is order-preserving and monotone increasing, then if there is a set A such that $T \subseteq A \subseteq X \cup Next(A)$ then

$$T \cup A = A \subseteq X \cup Next(A) \subseteq X \cup Next(FP(A)) \subseteq X \cup Next(FP(A \cup X)).$$

□

In the example of Figure 5.1, the set $A = \{\mathsf{C}, \mathsf{D}, \mathsf{F}, \mathsf{G}, \mathsf{H}, \mathsf{T}\}$ is such that $A \subseteq \{\mathsf{C}, \mathsf{D}\} \cup Next(A)$. Since A contains the target $\{\mathsf{T}\}$, the set $X = \{\mathsf{C}, \mathsf{D}\}$ is a precursor set of $\{\mathsf{T}\}$. Intuitively, A has to contain all metabolites in the path used from X and Z to produce T and regenerate Z.

5.1.5 Maximal target

Clearly, a set of sources X that is a precursor set of T, is also a precursor set of any subset of T. Moreover, if X is also a precursor set of another target T', then it is precursor set of $T \cup T'$. Thus, it is easy to see that, given X, there exists a *maximal target set* $T_{max}(X)$, such that X is precursor of any target T if and only if T is contained in $T_{max}(X)$.

Definition 5.5. (Maximal target) *Given a set of sources $X \in \mathcal{P}(\mathcal{S})$, $T_{max}(X) \in \mathcal{P}(\mathcal{C})$ is the set of all metabolites t for which X is a precursor set of $\{t\}$.*

For instance, in the network of Figure 5.1, the maximal target of $\{\mathsf{C}, \mathsf{D}\}$ is the set $T_{max}(\{\mathsf{C}, \mathsf{D}\}) = \{\mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{F}, \mathsf{G}, \mathsf{H}, \mathsf{I}, \mathsf{T}\}$.

Given X, if a set A is such that $A \subseteq X \cup Next(A)$, then X is a precursor set of any subset T of A. Thus, A has to be included in $T_{max}(X)$. Therefore $T_{max}(X)$ is exactly the maximal set that satisfies this property. The next lemma is useful to find this maximal set.

Lemma 5.6. *Given $X \in \mathcal{P}(\mathcal{S})$ a set of sources and given the monotone decreasing function g_X defined by $g_X(M) = M \cap (X \cup Next(M))$, the set $g_X^*(\mathcal{C})$ is exactly the maximal target of X.*

Proof. Clearly $g_X(M) \subseteq M$ and therefore g_X is, indeed, monotone decreasing. Then g_X^* is well-defined. By definition, $g_X^*(\mathcal{C}) = g_X^*(\mathcal{C}) \cap (X \cup Next(g_X^*(\mathcal{C})))$ or equivalently $g_X^*(\mathcal{C}) \subseteq X \cup Next(g_X^*(\mathcal{C}))$. We show that it is the maximal set that satisfies this property. Let A such that $A \subseteq X \cup Next(A)$. Then $A \subseteq A \cap (X \cup Next(A)) = g_X(A)$.

Since g_X is order-preserving, we can apply g_X to each side obtaining $g_X(A) \subseteq g_X^2(A)$. But g_X is monotone decreasing, therefore $g_X(A) = g_X^2(A) = g_X^*(A)$. Since g_X^* is order-preserving, we conclude that $A \subseteq g_X^*(A)$. □

5.1.6 Hyperpaths from sources to the target

Computing the maximal target of X gives an easy test to check if X is a precursor set of T. However, it is less useful for *finding* precursor sets of a given target. Therefore we adopted a different approach that takes advantage of the topology of the network. Indeed, if X is a precursor set of T, then there is a path of reactions, or *hyperpath* from X to T. To search for precursors of T, we therefore backtrack along hyperpaths from T to the sources.

Given $H \subseteq \mathcal{R}$, we define the sets $Subs(H) = \bigcup_{r \in H} Subs(r)$ and $Prod(H) = \bigcup_{r \in H} Prod(r)$. We now provide a formal definition of the notion of hyperpath.

Definition 5.7. (Hyperpath) *Let $X \subseteq \mathcal{C}$ and $T \subseteq \mathcal{C}$ be two sets of metabolites of a network G. A set of reactions $H \subseteq \mathcal{R}$ is called a* hyperpath from X to T, *if*

$$T \cup Subs(H) \subseteq X \cup Prod(H)$$

and no other subset of H satisfies this condition (H is minimal). In this case, if $(T \cup Subs(H)) \setminus Prod(H) = X$, then we say that H is a **precise** *hyperpath from X to T.*

Note that the minimality condition is over the set of reactions, not over the set X. In the network example of Figure 5.1 for the target set $T = \{\mathsf{T}\}$, the set $H = \{r_1, r_2, r_7, r_5, r_6\}$ is a *precise* hyperpath from $\{\mathsf{A}, \mathsf{C}, \mathsf{D}\}$ to T and the set $H' = \{r_1, r_2, r_3, r_6\}$ is a precise hyperpath from $\{\mathsf{C}, \mathsf{D}\}$ to T. Of course, H' is also a hyperpath from $\{\mathsf{A}, \mathsf{C}, \mathsf{D}\}$ to T but it is not a precise hyperpath from this set of sources. Finally, the set $H'' = \{r_1, r_2, r_3, r_7, r_5, r_6\}$ is not a hyperpath from $\{\mathsf{A}, \mathsf{C}, \mathsf{D}\}$ to T since it is not minimal.

Lemma 5.8. *A set of sources $X \subseteq \mathcal{S}$ is a precursor set of T if and only if there exists a hyperpath H from X to T.*

Proof. If X is a precursor set of T, then $T \subseteq T_{max}(X)$. Consider the set $W = \{r \in \mathcal{R} \mid Subs(r) \subseteq T_{max}(X)\}$. Hence, $Subs(W) \subseteq T_{max}(X)$. Since $T_{max}(X) \subseteq X \cup Next(T_{max}(X))$, we have that $T \cup Subs(W) \subseteq X \cup Next(T_{max}(X)) = X \cup Prod(W)$

and therefore W satisfies the first condition of the hyperpath definition. Any minimal subset of W that satisfies this condition is a hyperpath from X to T.

For the reverse direction, let H be a hyperpath from X to T. Then, the set $Subs(H) \cup T$ is such that $Subs(H) \cup T \subseteq X \cup Prod(H) \subseteq X \cup Next(Subs(H) \cup T)$. By Lemma 5.4, we conclude that X is a precursor set of T. □

Corollary 5.9. *If a set of sources $X \subseteq \mathcal{S}$ is a minimal precursor set of T, then there exists a precise hyperpath H from X to T.*

5.1.7 Precursor cut set

We defined the concept of precursor set as a set of sources that is able to produce the target. Suppose now that the target is a set of compounds whose production we want to avoid. In this case, we can define a *precursor 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 of the sources. This concept has a biological application, for instance, in the case where we want a bacterium to avoid producing some given compound while providing it with a maximal set of resources that enables it to continue doing its specific tasks.

Definition 5.10. *A set of sources $X \in \mathcal{P}(\mathcal{S})$ is a* precursor cut set *of $T \in \mathcal{P}(\mathcal{C})$ if and only if the set $\mathcal{S} \setminus X$ is* **not** *a precursor of T.*

5.2 Complexity results

In this part, we address the theoretical complexity of three problems related to the search for precursor sets:

> **Problem** MINIMAL-PS$(G, \mathcal{S}, T)$: given a metabolic network $G = (\mathcal{C}, \mathcal{R})$ with $\mathcal{S} \subset \mathcal{C}$ the set of sources and $T \subseteq \mathcal{C}$ the set of target metabolites, find a minimal precursor set $X \subseteq \mathcal{S}$ of T in G.
>
> **Problem** MINIMUM-PS$(G, \mathcal{S}, T)$: given a metabolic network $G = (\mathcal{C}, \mathcal{R})$ with $\mathcal{S} \subseteq \mathcal{C}$ the set of sources and $T \subseteq \mathcal{C}$ the set of target metabolites, find a minimum size precursor set $X \subseteq \mathcal{S}$ of T in G.
>
> **Problem** ALLMINIMAL-PS$(G, \mathcal{S}, T)$: given a metabolic network $G = (\mathcal{C}, \mathcal{R})$ with $\mathcal{S} \subseteq \mathcal{C}$ the set of sources and $T \subseteq \mathcal{C}$ the set of target metabolites, enumerate all minimal precursor sets $X \subseteq \mathcal{S}$ of T in G.

Given the network of Figure 5.1, let $T = \{\mathsf{T}\}$ be the target set and $\mathcal{S} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$ the sources. A solution to the MINIMAL-PS(G, P, T) problem is $X_1 = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ or $X_2 = \{\mathsf{C}, \mathsf{D}\}$, whereas the precursor set $X_3 = \{\mathsf{A}, \mathsf{C}, \mathsf{D}\}$ is not a solution because it is not minimal. The solution to the MINIMUM-PS(G, P, T) problem is $\{\mathsf{C}, \mathsf{D}\}$. Finally, the solution to the ALLMINIMAL-PS(G, P, T) problem is given by $\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ and $\{\mathsf{C}, \mathsf{D}\}$.

5.2.1 Deciding if a set of sources is a precursor set

We present an algorithm to find the maximal target of a set of sources X by computing $g^*_X(\mathcal{C})$. We introduce the notation $k_{max} := max_{r\in\mathcal{R}}(|Subs(r)| + |Prod(r)|)$.

Gstar(X, M):
Starting from $S := M$, iterate the following steps:

Compute $S_{next} := S \cap (X \cup Next(S))$;

If $S_{next} \supseteq S$ then stop and return S;

Otherwise, reset $S := S_{next}$ and iterate.

Lemma 5.11. *Given a metabolic network G and a subset X of the sources $\mathcal{S}$, the algorithm* Gstar$(X, \mathcal{C})$ *produces the maximal target of X in time $O(|\mathcal{C}|^2 + |\mathcal{C}||\mathcal{R}|k_{max})$.*

Proof. Clearly *Gstar(X, $\mathcal{C}$)* computes $g^*_X(\mathcal{C})$, the set that results from the iterative application of the function g_X to $\mathcal{C}$. The algorithm iterates at most $|\mathcal{C}| - |T_{max}(X)|$ times. Computing $S \cap (X \cup Next(X \cup S))$ takes $O(|\mathcal{C}| + |\mathcal{R}|k_{max})$ time. Therefore, the running time of the whole procedure is $O(|\mathcal{C}|^2 + |\mathcal{C}||\mathcal{R}|k_{max})$. □

By Lemma 5.6, it is possible to check in polynomial time if a subset X of the sources is a precursor set of T.

Corollary 5.12. *Given a metabolic network G, a subset X of the sources $\mathcal{S}$ and target T, we can decide in time $O(|\mathcal{C}|^2 + |\mathcal{C}||\mathcal{R}|k_{max})$ whether X is a precursor set of T.*

5.2.2 Finding a minimal and a minimum precursor sets

Theorem 5.13. *Given a metabolic network G, set of sources $\mathcal{S}$ and target set T, the problem* MINIMAL-PS$(G, \mathcal{S}, T)$ *can be solved in time $O(|\mathcal{C}|^2|\mathcal{S}| + |\mathcal{C}||\mathcal{R}||\mathcal{S}|k_{max})$.*

Proof. Using the procedure *Gstar*$(\mathcal{S}, \mathcal{C})$, we determine whether $\mathcal{S}$ is a precursor set of T. If the answer is negative, then there is clearly no precursor set and we stop. Otherwise, we set $X := \mathcal{S}$ and all compounds in $\mathcal{S}$ as unmarked. For each compound $u \in \mathcal{S}$ in some fixed but arbitrary order, the algorithm sets $X' = X - \{u\}$. Using

Gstar$(X', \mathcal{C})$, we determine whether X' is a precursor set of T. If not, then u remains in X and it is marked, and we continue with the next compound in $\mathcal{S}$. Otherwise, there exists at least one precursor set that does not contain u, and therefore u is deleted from X and we proceed with the next compound in $\mathcal{S}$. Since, at the end, there are only marked compounds left in X and these are all essential for the obtained precursor set, they form a minimal precursor set of T in G.

If there is no precursor set then this method gives the answer in $O(|\mathcal{C}|^2+|\mathcal{C}||\mathcal{R}|k_{max})$ by the previous lemma. Otherwise, the algorithm requires to check for $|\mathcal{S}|$ sets if they are precursor sets. Therefore, the running time of this algorithm is $O(|\mathcal{C}|^2|\mathcal{S}| + |\mathcal{C}||\mathcal{R}||\mathcal{S}|k_{max})$. □

The running time of the algorithm can be improved slightly by eliminating some unnecessary iterations in the procedure *Gstar*. Instead of calling *Gstar*$(X', \mathcal{C})$ for $X' = X - \{u\}$, we can call *Gstar*$(X', T_{max}(X))$. The time in this case is $O(|\mathcal{C}|^2|\hat{X}| + |\mathcal{C}||\mathcal{R}||\hat{X}|k_{max})$ where $\hat{X}$ is the minimal precursor set found (if there are no minimal precursor sets, the time remains $O(|\mathcal{C}|^2 + |\mathcal{C}||\mathcal{R}|k_{max})$). The complexity is therefore reduced in the mean case, although in the worst case, it remains the same.

The following result is proved by a reduction from the NP-complete problem HITTING SET: given a set of elements $H = \{1, \ldots, m\}$ and a collection of subsets $\mathcal{I} = \{I_1, \ldots, I_n\}$ of H, find a minimum cardinality subset of elements $Y \subseteq H$ such that $I_i \bigcap Y \neq \emptyset$, $\forall i = 1, \ldots, n$.

Theorem 5.14. MINIMUM-PS$(G, \mathcal{S}, T)$ *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. Corollary 5.12 implies that the decision version is in NP.

We make a reduction from the decision version of HITTING SET, asking if there exists a hitting set of size at most k. Consider a hitting set instance with $H = \{1, \ldots, m\}, \mathcal{I} = \{I_1, \ldots, I_n\}$ and k. For each element h in H, we create a vertex h in $\mathcal{C}$, and for each set I_j in $\mathcal{I}$, we create a vertex I_j in $\mathcal{C}$. We create an extra vertex t in $\mathcal{C}$. For each $h \in I_j$, we create in $\mathcal{R}$ 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 compound, and we define the vertices corresponding to the elements of H as the sources $\mathcal{S}$ of G. See Figure 5.2. The transformation clearly takes polynomial time.

Clearly, if $H' \subseteq H$ is a hitting set then the set of vertices corresponding to H' is a precursor set of t, since all I_j, $j = 1, \ldots, n$ can be produced. Conversely, suppose that

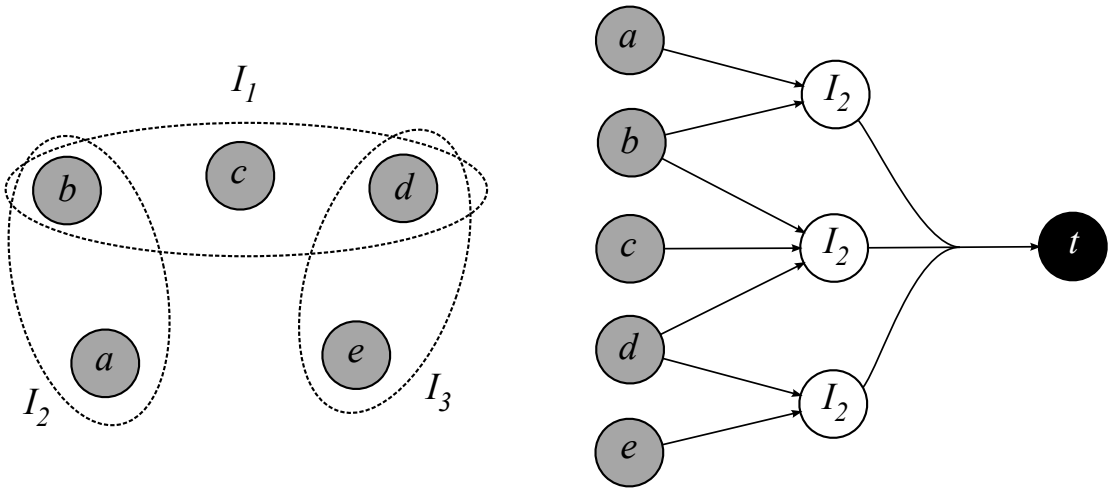

Figure 5.2: Reduction of an instance of the hitting set problem. Each hitting set of $\mathcal{I} = \{I_1, I_2, I_3\}$ corresponds to a precursor set of $\{t\}$ (and *vice versa*).

G has $X \subset \mathcal{S}$ as precursor set of t. Then, each I_j has to be in $X \cup Next(FP(X \cup Z))$. Since vertex I_j does not belong to the sources, it must be generated by some reaction r_{hj}, with $h \in I_j$ and vertex $h \in X \cup Next(FP(X \cup Z))$. Since there is no vertex producing h, it has to belong to X. Therefore, for each set I_j, there exists $h \in I_j$ with $h \in X$. Hence X corresponds to a hitting set. □

We observe that in the above transformation there is a one-to-one relation between hitting sets and precursor sets, and a related pair is of the same size. This immediately implies that MINIMUM-PS is as hard to approximate in polynomial time as HITTING SET, which is known to be APX-hard (see Ausiello *et al.*, 1999). Specifically, under the assumption that P $\neq$ NP, no polynomial time algorithm for MINIMUM-PS can have approximation ratio $o(\log n)$ (see Raz et Safra (1997) for the SET-COVER problem, an equivalent version of HITTING SET).

A similar reduction shows NP-hardness of the problem of searching for a minimum precursor cut set. Indeed, we need in this case to consider the same reduction as in the proof of Theorem 5.14, but with two modifications: (a) replace the hyperarc r_t (from $\{I_1, \ldots, I_n\}$ to t) by n separated 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 precursor cut set. Therefore, finding a minimum size precursor cut is NP-hard.

Another NP-hard minimisation problem emerges from our definition of precursor set, though this is arguably less interesting from a biological point of view.

Proposition 5.15. *Given a precursor set X of T, to find a minimum cardinality set Z such that $T \cup Z \subseteq X \cup Next(FP(X \cup Z))$ is NP-hard.*

Proof. Consider the same reduction of the hitting set problem presented in the proof of Theorem 5.14, with a slightly different hypergraph construction: For each element h in H, we create another extra vertex h' in $\mathcal{C}$ and two reactions $r_{hh'}$ and $r_{h'h}$ from h to h' and from h' to h respectively. The rest of the construction remains the same. Now we define $\mathcal{S} = \emptyset$. We show that for any minimum hitting set of size k, there exists a minimum cardinality set Z such that $T \cup Z \subseteq Next(FP(\emptyset \cup Z)) \cup \emptyset$ and $|Z| = k$, and viceversa.

If H' is a hitting set, we consider $X = \emptyset$ and the set Z of vertices h corresponding to H'. Clearly t can be produced and Z regenerated, that is $T \cup Z \subseteq Next(FP(Z))$ and $|Z| = |H'|$. Conversely, if Z is a minimum cardinality set such that $T \cup Z \subseteq Next(FP(Z))$, then each I_j has to be in $Next(FP(Z))$. For each I_j, there is $h \in I_j$ such that at least one of the two vertices h and h' belongs to Z. Therefore, the set $H' = \{h \in H \mid h \in Z \text{ or } h' \in Z\}$ is a hitting set with $|H'| \leq |Z|$. By the previous part, there exists Z' satisfying the condition with $|Z'| = |H'| \leq |Z|$. Since Z is a minimum cardinality set, then $|Z| = |H'|$. □

5.2.3 Enumerating all minimal precursor sets

We know that MINIMAL-PS$(G, \mathcal{S}, T)$ can be solved in polynomial time with respect to the size of the input. 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 ALLMINIMAL-PS can be solved in polynomial *total* time, that is, in time proportional to the size of the input and output (Johnson *et al.*, 1988).

Theorem 5.16. *The enumeration problem* ALLMINIMAL-PS *cannot be solved in polynomial total time unless P=NP.*

Proof. Let $\mathcal{F}$ be the class of all boolean $\wedge, \vee$-formulae. Given a formula $f \in \mathcal{F}$, enumerating the set of all prime implicants of f cannot be done in polynomial total unless P=NP (Gurvich et Khachiyan, 1999). We show that this problem can be easily reduced to the problem of enumerating minimal precursor sets. Let $f \in \mathcal{F}$ be a $\wedge, \vee$-formula. We can construct (see Figure 5.3) a hypergraph G_f, with sources $\mathcal{S}$ corresponding to the set of variables of f and a target set T, such that a set I of variables is a prime implicant of f if and only if I is a minimal precursor set of T. The set $\mathcal{C}$ of metabolites corresponds to the set of variables plus one metabolite for each

conjunction and disjunction inside the formula (in the given example, $\mathcal{C}$ contains eight metabolites: p, q, r, s, $p \vee q$, $p \wedge s, r \vee (p \wedge s)$ and $(p \vee q) \wedge (r \vee (p \wedge s)) \wedge s)$. The set of hyperarcs is composed 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 reactions from each term of d to the metabolite d. The target set is a singleton containing the metabolite representing f (the most external conjunction or disjunction). Clearly, a minimal precursor set of T in G_f corresponds to a prime implicant of f and *vice versa*. □

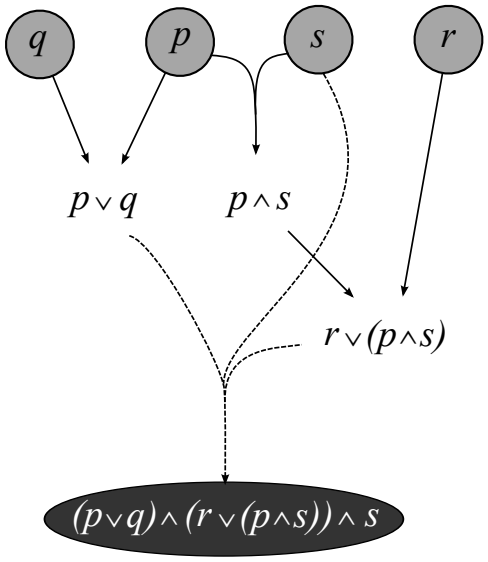

Figure 5.3: Graphical representation of the reduction

This result shows that enumerating all precursor sets in polynomial total time is NP-hard even in the case of networks without *cycles* (for any reasonable definition of cycle).

Again, this result is also valid if we consider the enumeration of all minimal precursor *cut* sets of T. Indeed, in the reduction of the proof of Theorem 5.16, the minimal precursor cut sets correspond exactly to the set of prime *implicates* of the boolean function f. As for prime implicants, enumerating the set of prime implicates cannot be done in polynomial total time unless P=NP (Gurvich et Khachiyan, 1999).

Although enumeration of minimal precursor sets and enumeration of minimal precursor cut sets are both hard problems, we can show a more positive result if we consider the enumeration of both problems simultaneously. Indeed, enumeration of all minimal precursor cut sets is *dual* to ALLMINIMAL-PS in the following sense: the collection of all minimal precursor cut sets of T corresponds to the collection of all minimal hitting

sets of the collection of minimal precursors sets of T and vice versa. Given a collection $\mathbb{X}$ of subsets of $\mathcal{S}$, we denote by $mhs(\mathbb{X})$ the collection of all minimal hitting sets of $\mathbb{X}$.

Lemma 5.17. *Let $\mathbb{P}^T_{min}$ be the collection of all minimal precursor sets of T and $\mathbb{C}^T_{min}$ the collection of all minimal precursor cut sets of T. We have*

$$\mathbb{P}^T_{min} = mhs(\mathbb{C}^T_{min}).$$

Proof. Let $X \in \mathbb{P}^T_{min}$ be a precursor set of T. We show that X is a hitting set of the collection $\mathbb{C}^T_{min}$. Let Y be in $\mathbb{C}^T_{min}$. By contradiction, suppose that Y and X are disjoint sets. Therefore, $X \subseteq \mathcal{S} \setminus Y$, that is, a precursor set of T is subset of a non-precursor set of T which is contradictory. We conclude that X is a hitting set of $\mathbb{C}^T_{min}$. Now take any hitting set X of $mhs(\mathbb{C}^T_{min})$. We show that X is a precursor set of T. By contradiction, if X is not a precursor set, then $\mathcal{S} \setminus X$ is a precursor cut set which contains a minimal precursor cut set, say Y', that belongs to $\mathbb{C}^T_{min}$. Since $X \cap Y' = \emptyset$ then X' is not a hitting set of $\mathbb{C}^T_{min}$ which is contradictory. We conclude that X is a precursor set of T.

Therefore, the collection of precursor sets is exactly the collection of hitting set of $\mathbb{C}^T_{min}$. Taking the minimal sets we conclude that $\mathbb{P}^T_{min} = mhs(\mathbb{C}^T_{min})$. □

Consider $X \subseteq \mathcal{S}$ a set of sources. We can represent X as a vector of $\{0,1\}^{|\mathcal{S}|}$ that we still denote by X. We define the function $F : \{0,1\}^{|\mathcal{S}|} \to \{0,1\}$ as $F(X) = 1$ if X is a precursor set of T and $F(X) = 0$ otherwise. Clearly, F is a monotone boolean function (although it is not explicitly expressed as conjunction and disjunction of literals). Moreover, it is easy to see that the collection $\mathbb{P}^T_{min}$ corresponds to the set of all *prime implicants* of F and, by Lemma 5.17, the collection $\mathbb{C}^T_{min}$ corresponds to the set of all *prime implicates* of F.

In Gurvich et Khachiyan (1999), the authors show that for any monotone boolean functions, enumerating both prime implicants and prime implicates simultaneously can be, in some sense, easier than enumerating only one set of them. They prove the following result:

Theorem 5.18. (Gurvich and Khachiyan, 1999). *Let $f : \{0,1\}^{|\mathcal{S}|} \to \{0,1\}$ be a monotone boolean function whose value at any point $x \in \{0,1\}^{|\mathcal{S}|}$ can be determined in time t, and let C and D be the sets of, respectively, prime implicates and prime implicants of f. Given two subsets $C_0 \subseteq C$ and $D_0 \subseteq D$ of total size $m = |C_0| + |D_0| < |C| + |D|$, a new element in $(C \setminus C_0) \cup [(D \setminus D_0)$ can be found in time $O(n(t+n)) + m^{O(logm)}$.*

We know, by Lemma 5.12 that for any $X \subseteq \mathcal{S}$, we can compute $F(X)$ in time $O(|\mathcal{C}|^2 + |\mathcal{C}||\mathcal{R}|k_{max})$. Therefore, applying Theorem 5.18, there is an algorithm that, given a solution of $\mathbb{P}^T_{min}$ and $\mathbb{C}^T_{min}$, provides a new solution in $\mathbb{P}^T_{min} \cup \mathbb{C}^T_{min}$ in time $O(|\mathcal{S}||\mathcal{C}|^2 + |\mathcal{S}||\mathcal{C}||\mathcal{R}|k_{max} + |\mathcal{S}|^2) + m^{O(logm)}$ where m is the number of solutions already found.

Corollary 5.19. *The collections $\mathbb{P}^T_{min}$ and $\mathbb{C}^T_{min}$ can be jointly enumerated in quasi-polynomial incremental (and hence total) time.*

Chapter 6

Algorithms to Enumerate All Minimal Precursor Sets

Contents

In this chapter, we present two algorithms that compute the collection of minimal precursor sets of a target set T.

6.1 Preprocessing the network

To facilitate the exposition, we suppose that the metabolic network has the following properties: (a) the target set is a singleton set $\{t\}$, (b) each source $x \in \mathcal{S}$ is not the product of any reaction, and (c) each compound belongs to the maximal target of $\mathcal{S}$. We can pre-process any network in order to satisfy these conditions without changing the collection of precursor sets of T:

(a) *Singleton target:* Add a new reaction to $\mathcal{R}$ with substrate the metabolites in T and producing a new metabolite $\{t\}$. Then redefine the target as $T := \{t\}$.

(b) *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 compound labelled x and product x'. The set of sources continues to be $\mathcal{S}$.

(c) *All compounds in the maximal target of $\mathcal{S}$:* Compute $T_{max}(\mathcal{S})$ and remove all compounds in the complement. Remove all reactions having had their substrates or products removed.

It is not difficult to see that if X is (or is not) a precursor set of T, then it will remain so after these transformations.

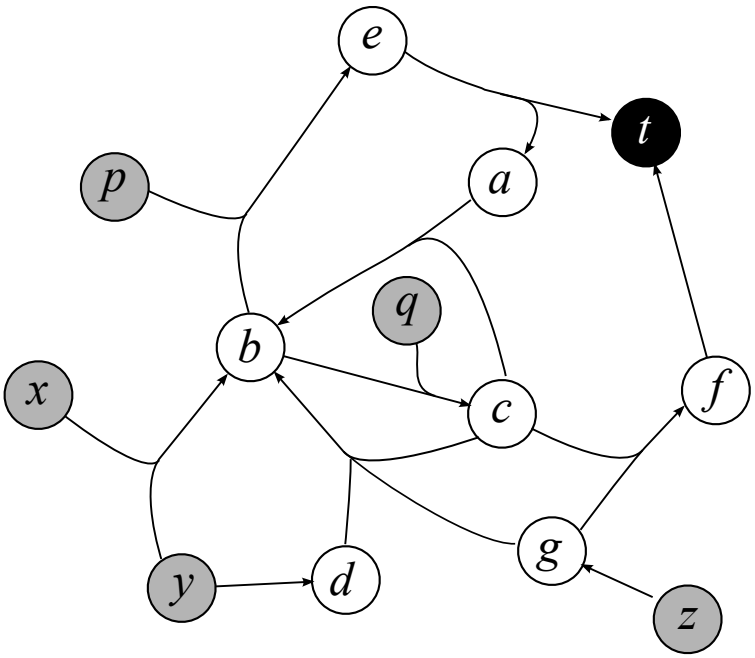

Figure 6.1: Example of network. Black node correspond to the target. Grey nodes correspond to the sources.

In order to find precursor sets that can produce the target t, the algorithm searches for hyperpaths from source sets to t but starting at t and using the reactions of G in reverse direction. Enumerating the hyperpaths obtained we get, by Lemma 5.8, the minimal precursor sets of t.

6.2 The replacement tree

In order to explain how the algorithm traverses the hypergraph, we represent all the hyperpaths arriving at t in an equivalent tree structure, which we call the *replacement tree*. This tree is such that, if X is a minimal precursor set of t, then there is a subtree (satisfying some conditions) from some leaves labelled by X to the root labelled t.

The replacement tree is rooted. Each node different from the root has as *parent* the next node on the path towards the root. Therefore, nodes (except for the root) have only one parent, whereas they can have several *children*. An *ancestor* of a node is any other node on the path towards the root.

Nodes of the tree are labelled either by a metabolite or by a reaction, and are called, respectively, *metabolite nodes* and *reaction nodes*. We denote by $\alpha(m)$ the label of a metabolite node m and $\rho(u)$ the label of a reaction node u. The children of a metabolite node are reaction nodes labelled by those reactions that produce the metabolite while the children of a reaction node are metabolite nodes labelled by the substrates of the reaction.

Given the network G, the target t and the set of sources$\mathcal{S}$, we define R_t the replacement tree of G by the following construction. The root is a metabolite node *root* which is labelled t (that is, $\alpha(root) = t$). For each reaction producing this metabolite, we create a reaction node, which has as parent the root, and as children new metabolite nodes labelled by its substrates. In this way, we obtain a tree (of depth 2) whose leaves are metabolites. This process is then iterated for each new metabolite node.

For any metabolite node m, we define the *cover set* of m to be the set of substrates and products of all the reaction nodes ancestor of m, that is,

$$Cover(m) = \bigcup_{\substack{v \text{ ancestor} \\ \text{reaction} \\ \text{node of } m}} Prod(\rho(v)) \cup Subs(\rho(v))$$

Let n be a newly created metabolite node. If $\alpha(n)$ is in the cover set of its grandparent (i.e. $\alpha(n)$ is a substrate or product of an ancestor reaction of n which is not its parent) then we say that n is *repeated*. Along any branch of the tree, the process of expanding the tree stops at n when at least one of the following two conditions holds:

1. n is *repeated*

2. $\alpha(n)$ is a source ($\alpha(n) \in \mathcal{S}$).

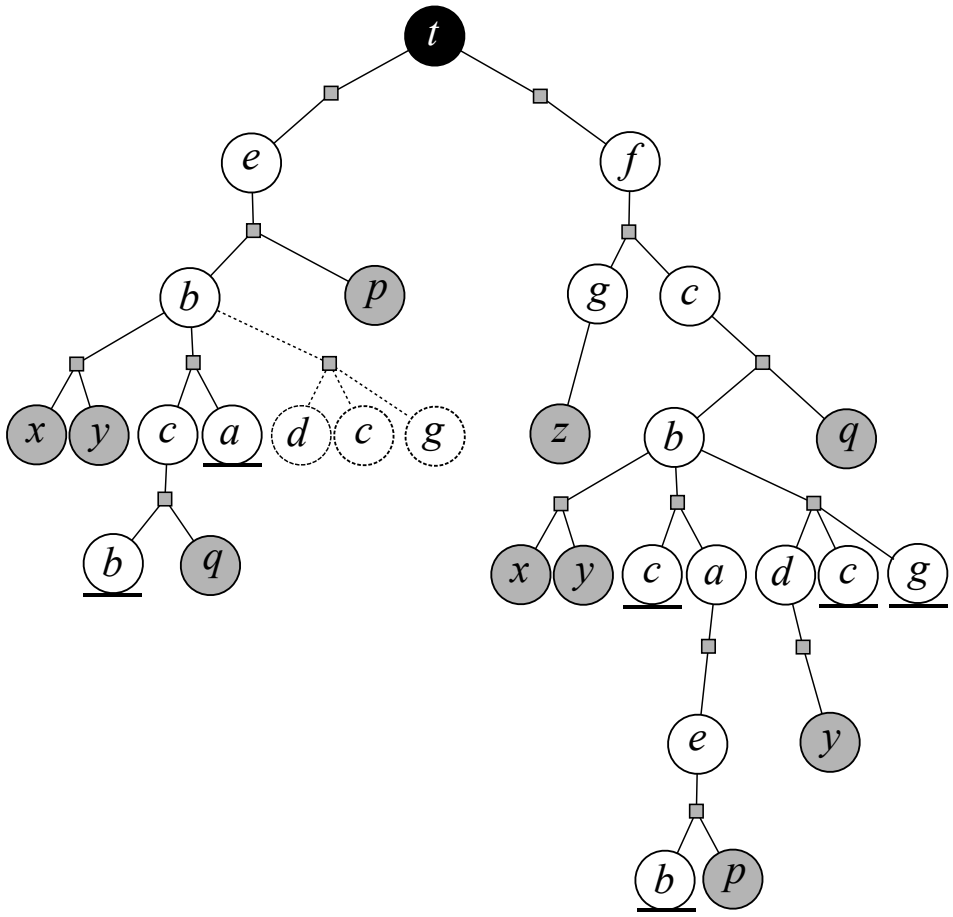

Figure 6.2: Replacement tree of network presented in Figure 6.1

We show that each hyperpath from subsets of $\mathcal{S}$ to t is represented by a special subtree of R_t called *HP-tree*.

Definition 6.1. *Let R_t be the replacement tree of a metabolic network G with target $\{t\}$. Let m be metabolite node of R_t. A subtree π_m of R_t is called a* **HP-subtree** rooted in m, *if all the following conditions are satisfied:*

1. *π_m is a tree rooted at m (m is ancestor of all the other nodes);*

2. *the set of leaves of π_m is a subset of the leaves of R_t;*

3. *if a metabolite node belongs to π_m and is not a leaf, then* **only one** *of its (reaction) children in R_t also belongs to π_m.*

4. *if a reaction node belongs to π_m then* **all** *its (metabolite) children in R_t also belong to π_m.*

If the root m is the root of R_t then we say that π_m is a **HP-tree** *and we denote it simply by π.*

We introduce some notation. We denote by $L(\pi_m)$ to the set of labels of the leaves of π_m (therefore $L(\pi_m) \subseteq \mathcal{C}$). For any subgraph A of R_t, we denote $\alpha(A)$ the set of labels of metabolite nodes of A and $\rho(A)$ the set of labels of reaction nodes of A (therefore $\alpha(A) \subseteq \mathcal{C}$ and $\rho(A) \subseteq \mathcal{R}$).

Lemma 6.2. *A set $X \in \mathcal{P}(\mathcal{S})$ is a precursor set of $\{t\}$ if and only if R_t contains a HP-tree π such that $\alpha(\pi) \cap \mathcal{S} \subseteq X$.*

Proof. Let π be a HP-tree of R_t with $\alpha(\pi) \cap \mathcal{S} \subseteq X$. We show that the set of reactions $\rho(\pi)$ satisfies the first condition to be a hyperpath from X to t, that is, $T \cup Subs(\rho(\pi)) \subseteq X \cup Prod(\rho(\pi))$. In effect, any substrate a of a reaction in $\rho(\pi)$ has two possibilities: a is the label of a non-leaf metabolite node (and therefore is a product of a reaction in $\rho(\pi)$) or a is the label of a leaf of π. We show that in this last case a is in X or it is a product of a reaction in $\rho(\pi)$. In effect, suppose that a is the label of a leaf. If $a \in \mathcal{S}$, then by hypothesis $a \in X$. By contrary if a is not a source, it is the label of a repeated metabolite node m. Therefore a is a product or substrate of $\rho(u)$ with u an ancestor reaction of the grandparent of m (w.l.o.g we take the closest to the root). If a is a product of $\rho(u)$ then clearly $a \in Prod(\rho(\pi))$. If a is a substrate of $\rho(u)$, then a is the label of a child n of u that is not repeated, hence n is not a leaf. Therefore $a \in Prod(\rho(\pi))$. We conclude that $\rho(\pi)$ satisfies the first condition in the hyperpath definition. Any minimal subset of $\rho(\pi)$ that satisfies the same condition is a hyperpath from X to T. Hence, X is a precursor set of T.

For the reverse direction, if X is a precursor set of t, then there exists a hyperpath H from X to t in G. Consider the set $R_t|_H$ obtained from R_t by eliminating from R_t all the reaction nodes labelled with reactions not belonging to H and taking the connected component that contains the root of R_t. Each reaction node in $R_t|_H$ has the same children as in R_t. Since $Subs(H) \setminus X \subseteq Prod(H)$, each metabolite node in $R_t|_H$ that is not a leaf of R_t, has at least one reaction child in $R_t|_H$. Then, the leaves of $R_t|_H$ are a subset of the leaves of R_t. Moreover, a source leaf of $R_t|_H$ is not produced by any reaction, it thus has to be in X. Clearly any HP-tree π of the tree $R_t|_H$ is an HP-tree of R_t such that its sources belong to X. □

Given a reaction node u of R_t, we call $Chnr(u)$ the set of all (metabolite) children of u that are not repeated, that is, the children not in the cover set of the parent of u.

Lemma 6.3. *Given a metabolic network G with R_t its replacement tree rooted at t, let m be a metabolite node of R_t. Let u and u' be two reaction nodes children of m such that $Chnr(u) \subseteq Chnr(u')$. Then, for any HP-tree π' containing u', there exists an HP-tree π containing u, such that $\alpha(\pi) \cap \mathcal{S} \subseteq \alpha(\pi') \cap \mathcal{S}$.*

Proof. Let π' be a HP-tree containing u'. First we show that $Subs(\rho(u)) \cap \mathcal{S} \subseteq \alpha(\pi') \cap \mathcal{S}$. Let $a \in Subs(\rho(u)) \cap \mathcal{S}$. There is a metabolite node n child of u with $\alpha(n) = a$. If $n \in Chnr(u)$, then there is a metabolite node n' child of u' with $\alpha(n') = a$. Therefore $a \in \alpha(\pi') \cap \mathcal{S}$. If $n \notin Chnr(u)$, then $a = \alpha(n)$ is the label of a child of an ancestor of u and therefore ancestor of u'. Therefore $a \in \alpha(\pi') \cap \mathcal{S}$.

Now, if we have a HP-tree π containing u such that $\rho(\pi) \subseteq \rho(\pi') \cup \rho(u)$, we can conclude

$$\alpha(\pi) \cap \mathcal{S} = Subs(\rho(\pi)) \cap \mathcal{S} \subseteq (Subs(\rho(\pi')) \cap \mathcal{S}) \cup (Subs(\rho(u)) \cap \mathcal{S}) = \alpha(\pi') \cap \mathcal{S}.$$

We build therefore a HP-tree π containing u such that $\rho(\pi) \subseteq \rho(\pi') \cup \rho(u)$. Clearly π contains all the reaction nodes ancestor of m (that are also in π'). Let n be a metabolite node of $Chnr(u)$. By hypothesis, there exists $n' \in Chnr(u')$ such that $\alpha(n) = \alpha(n')$. Therefore, π' contains n' and also one reaction node v' child of n'. Therefore, for each metabolite node $n \in Chnr(u)$ we include in π the reaction node v child of n such that $\rho(v) = \rho(v')$.

Note that, since $\rho(u) \neq \rho(u')$, the cover set of n is not necessarily equal to the cover set of n'. This makes more tricky the selection of the reactions in the next levels. Let p be a metabolite node of $Chnr(v)$. There exists p' children of v' such that $\alpha(p) = \alpha(p')$. If $p' \in Chnr(v')$, we can include in π the reaction node child of p with the same label of the child of p' that belong to π'. On the other hand, if $p' \notin Chnr(v')$ then necessarily $\alpha(p') \in Prod(u') \cup Subs(u')$. We analyse the different cases. If $\alpha(p')$ is in $Prod(u')$, we choose the reaction labelled $\rho(u)$ child of p. If $\alpha(p')$ is in $Subs(u')$, there exists a metabolite $q \in Chnr(u')$ with $\alpha(q) = \alpha(p)$. We choose the reaction child of p labelled as the child of q that belongs to π'. Repeating this process we obtain a π that contains, except for u, a subset of the reactions of π'. □

The result above offers the possibility of pruning branches of the replacement tree without losing solutions. Actually, since the replacement tree provides in fact a representation of the order in which the reactions are analysed and we can locally test the conditions required, such pruning allows us to avoid traversing the network by paths that do not lead to any new solutions.

6.3 Enumerating precursor sets by searching for HP-subtrees

We present our first algorithm to enumerate all precursor sets of t. We define the collection $\mathbb{S}_m$ of subsets of $\mathcal{S}$ as

$$\mathbb{S}_m = \{\alpha(\pi_m) \cap \mathcal{S} \mid \pi_m \text{ is a HP-subtree rooted at } m\}.$$

In other words, $\mathbb{S}_m$ is the collection of all sets of sources of HP-subtrees rooted at m. Clearly, any HP-subtree π_m rooted at m (with m not a leaf) can be decomposed as:

- m itself,
- a reaction node child of m, say u, and
- some HP-subtrees $\pi_{n_1}, \ldots, \pi_{n_k}$ where $n_1, \ldots, n_k$ are the children of u.

With this decomposition in mind, we propose an algorithm that searches for the collection $\mathbb{S} = \mathbb{S}_{root}$ of all sets of sources of HP-trees of R_t. We know, by Lemma 6.2, that this collection contains all the minimal precursor sets of t. The main idea is to compute the collection $\mathbb{S}_m$ by computing, for each child u of m, the collection $\mathbb{S}_m^u$ corresponding to all sets of sources of HP-subtrees rooted at m *that contain* u. Clearly

$$\mathbb{S}_m = \bigcup_{u \text{ child of } m} \mathbb{S}_m^u.$$

We can compute each set $X \in \mathbb{S}_m^u$ by taking all possible unions of sets $X_1, \ldots, X_k$ belonging respectively to the collections $\mathbb{S}_{n_1}, \ldots, \mathbb{S}_{n_k}$, where $n_1, \ldots, n_k$ are the children of u.

Before explaining the details of the algorithm, note that we are interested only in the minimal sets of $\mathbb{S}$ and therefore, in order of improving the running time and the space used, we can avoid computing, in each $\mathbb{S}_m$, the sets that are already not minimal.

Moreover, we have the following property: Given m a metabolite node of R_t (which is not a leaf) and a subtree π_m rooted at m, if u is the child node of m in π_m, then any HP-tree π that contains π_m has as sources at least all the sources children of reaction nodes ancestors of m. We are therefore interested in finding the minimal sets considering only the *new* sources of HP-subtrees rooted at m, that is, those sources that do not belong to the cover set of m.

Considering these remarks, we can define the collection $\bar{\mathbb{S}}_m$ of subsets of new sources as

$$\bar{\mathbb{S}}_m = \{(\alpha(\pi_m) \cap \mathcal{S}) \setminus Cover(m) \mid \pi_m \text{ is a HP-subtree rooted at } m\}$$

Redefining the collection $\mathbb{S}_m$ by taking the minimal sets:

$$\mathbb{S}_m = \{X \in \bar{\mathbb{S}}_m \mid \text{ there are no } Y \in \bar{\mathbb{S}}_m \text{ such that } Y \subset X\}$$

In this way, we decrease the collection of sets of sources to compute, improving the running time of the algorithm.

Although the algorithm searches for HP-subtrees in the replacement tree R_t, this structure is not maintained in memory. Instead, the algorithm works directly on the network G, saving only the necessary information to identify the leaves of R_t (i.e. the stop conditions in the construction of R_t). Indeed, each branch of the tree is built and the information about its leaves recovered *at the same time*, avoiding to maintain in memory more than the branch being analysed.

We present a recursive version of the algorithm. It requires two inputs: the label a of a metabolite node m of R_t being analysed and the cover set of m. It returns a collection of minimal sets of sources, each set being the sources not in $Cover(m)$ of an HP-subtree rooted at m. Therefore, the collection of all precursor sets of t is obtained by calling *MinSourcesHPsubtrees*$(t, \{t\})$.

MinSourcesHPsubtrees(a [label of m], $Cover$ [cover set of m]);
$\mathbb{S}_m := \{\}$;
For each reaction r producing a, with minimal $Subs(r) \setminus Cover$ do
    $NewCover := Cover \cup Subs(r) \cup Prod(r)$;
    $\mathbb{S}^u_m := \{\{\}\}$;
    For each metabolite $b \in Subs(r) \setminus Cover$ do
        If b is in *Sources* then
            $\mathbb{S}_n := \{\{b\}\}$;
        else
            $\mathbb{S}_n :=$*MinSourcesHPsubtrees*(b, $NewCover$);
        $\mathbb{S}^u_m :=$ *CartesianUnions*($\mathbb{S}^u_m$, $\mathbb{S}_n$);
    $\mathbb{S}_m := \mathbb{S}_m \cup \mathbb{S}^u_m$;
return *MinimalSets*($\mathbb{S}_m$);

The subroutines used are formally defined as: *CartesianUnions* that gives, for two given collections of sets, the collection of all the unions of pairs of sets, one for each collection, that is, $\mathit{CartesianUnions}(\mathbb{S}_1, \mathbb{S}_2) = \{X_1 \cup X_2 \mid X_1 \in \mathbb{S}_1,\ X_2 \in \mathbb{S}_2\}$, and *MinimalSets*($\mathbb{S}$) that removes from a collection $\mathbb{S}$ the sets that are not minimal, that is $\mathit{MinimalSets}(\mathbb{S}) = \{X \in \mathbb{S} \mid \text{there are no } Y \in \mathbb{S} \text{ such that } Y \subset X\}$.

We describe in detail the algorithm proposed. Let m be a metabolite node of R_t with label a that is not a leaf, and *Cover* the cover set of m. We want to obtain the collection $\mathbb{S}_m$ of minimal sets of new sources (i.e. that are not in *Cover*) of all HP-subtrees rooted at m. For each reaction node u child of m not pruned (i.e. discarding, by Lemma 6.3, the reactions with $Chnr(u)$ not minimal), we compute $\mathbb{S}_m^u$ the collection of sets of new sources of HP-subtrees rooted at m that contains u.

The set $\mathbb{S}_m^u$ will contain unions of *valid combinations* of these sets of sources, where by valid combination is meant one and only one set for each metabolite child of u. These particular unions are called the *cartesian unions* of the collections of sources given by each metabolite child of u. Computing, for each reaction node u child of m, the collection $\mathbb{S}_m^u$ of source sets, we return the total collection $\mathbb{S}_m$ that contains the union of all the collections, discarding those sets that are not minimal.

To obtain the collection $\mathbb{S}_m^u$ of a particular reaction node u child of m, we consider each metabolite node n child of u that is not in *Cover*. For each n, we compute the new sources of all the HP-subtrees rooted at n. Let b be the label of n. Two cases are possible: (a) if b is itself a source, then the only set of new sources is the singleton $\{b\}$ and therefore b will be included in all the sets of sources of $\mathbb{S}_m^u$, (b) if b is not a source, we compute, for all HP-subtrees rooted at n, the set of sources not in $Cover(m)$ but also not in $Subs(u)$, since these sources were already included in case (a). Therefore, we call recursively the algorithm for the metabolite node b and the cover set actualised.

Corollary 6.4. *Given the set of sources $\mathcal{S}$, the collection of subsets of $\mathcal{S}$ given by* $\mathbb{S} =$ MinSourcesHPsubtrees$(t, \{t\})$ *is exactly the collection of all minimal precursor sets of t.*

6.4 Enumerating precursor sets by merging reactions

The replacement tree used to search hyperpaths (by enumerating HP-subtrees) in the previous algorithm, reduces the computing time when some hyperpaths share reactions "near" the target. However, the work spent to find minimal sources along one branch of the replacement tree is not considered in any other branch, even when they share

almost all reactions (except for those near to the root). In some sense, one branch cannot "see" what was already done for a previously analysed branch.

In order to take advantage of the HP-subtrees already found, we can include in the hypergraph the HP-subtrees already covered in the form of new pseudo-reactions. In this way, instead of looking again the reactions in a HP-subtree already visited, we look at the pseudo-reactions added in a previous step.

These ideas motivated some modifications to the previous algorithm to enumerate precursor sets of t, that correspond to adding *in* the network G the information about the HP-subtrees already found. This is done by replacing some reactions of G in such a way that the collection of precursor sets is not modified.

Let H be a hyperpath from a set $X \subseteq \mathcal{S}$ to $\{t\}$. Let r_b be a reaction of H and let a be a substrate of r_b which is not a source ($a \in Subs(r) \setminus \mathcal{S}$). We want to replace r_b in H by a new reaction that does not use a as substrate. Since a is not a source, there exists $r_a \in H$ producing a. We can therefore create a new reaction r_{ab} that produces the same than r_b but consuming $(Subs(r_a) \cup Subs(r_b)) \setminus (Prod(r_a) \cup Prod(r_b))$. In this way we can replace r_b by r_{ab}, avoiding the consumption of a by one reaction of the new hyperpath H. Repeating this process for all reactions in H having substrates that are not sources, we obtain a a final hyperpath composed by a single reaction r_H producing t and consuming only metabolites of $\mathcal{S}$. Therefore, its substrates are a precursor set of $\{t\}$.

Of course, this process seems not useful if we already know H. In fact, we make this procedure without having computed H but only a branch of the replacement tree R_t. Moreover, it is done in such a way that r_b is replaced in all hyperpath that contains it. In effect r_b is replaced in the network, hence in each reaction node of R_t having r_b as label.

6.4.1 Reaction replacement

Consider a reaction r and a metabolite a substrate of r not belonging to the sources (i.e. $a \in Subs(r) \setminus \mathcal{S}$). We present below a procedure *Replace*(r,a) that replaces r by a set of reactions that do not change the precursor sets of t.

Replace($r \in \mathcal{R}$**,** $a \in Subs(r) \setminus \mathcal{S}$**)**

Compute $\mathcal{R}_a$ the set of reactions r_a such that $a \in Prod(r_a)$;

For each reaction r_a of $\mathcal{R}_a$

$NewSubs := (Subs(r_a) \setminus Prod(r)) \cup (Subs(r) \setminus Prod(r_a))$;

$NewProd := Prod(r) \cup Prod(r_a)$;

Add new reaction $(NewSubs, NewProd)$ to $\mathcal{R}$.
Remove r from $\mathcal{R}$;

Lemma 6.5. *Let $r \in \mathcal{R}$ be a reaction of G and let $a \in Subs(r) \setminus \mathcal{S}$ be a substrate of r that is not a source. Let G' be the network that results from applying the procedure $Replace(r, a)$. Then, X is a precursor set of t in G if and only if X is precursor set of t in G'.*

Proof. Let X be a precursor set of t in G. Therefore, there exists a hyperpath H from X to t in G. Consider two cases: If H does not contain r, then clearly H is also a hyperpath from X to t in G'. If H contains r, then it has to contain at least one reaction r_a of $\mathcal{R}_a$ (the set of all reactions in $\mathcal{R}$ that produce a). Let r_{new} be the reaction that results from merging r and r_a. Therefore, in G', the set $H' = (H \setminus \{r\}) \cup \{r_{new}\}$ clearly satisfies the first condition of the definition of hyperpath, and therefore contains a hyperpath from X to t.

For the reverse direction, let H' be a hyperpath from X to T in G'. Let R_{new} be the set of reactions of H' that are not in G, and R_{old} the set of reactions of $\mathcal{R}_a$ that were merged to produce R_{new}. Therefore, in G, the set of reactions $H = (H' \setminus R_{new}) \cup R_{old} \cup \{r\}$ contains a hyperpath from X to t.

□

Note that the replacement as defined, replaces a hyperpath H containing r by a new hyperpath of size at most $|H|$.

6.4.2 Algorithm compacting hyperpaths

Note that, although this *safe* replacement of reactions increases the number of reactions in the network, it mantains or reduces the size of the hyperpaths producing t. Therefore, we can do successive reaction replacements in G, until the network contains only reactions consuming only metabolites in $\mathcal{S}$. For example, we can replace for a metabolite a not in the sources, all the reactions that have a as substrate. Clearly the network obtained does not contain any reaction consuming a. Repeating this for all the metabolites not belonging to $\mathcal{S}$, we obtain the aimed network. In this modified network we can easily find the minimal precursor sets of t.

However, many of those replacements could be unnecessary. Indeed, some of the reactions could not be part of the replacement tree, because they are *below* the stop conditions or because they are pruned by Lemma 6.3 or simply because the reaction

replacement disconnected them of the target. Considering this, we do the replacement of reactions in G following the structure of the replacement tree.

Moreover, the order in which we choose the reactions to be replaced is relevant: we start by merging the reactions near the leaves of the replacement tree. Indeed, given any metabolite node m whose metabolite grandchildren are leaves of the tree, we replace the reaction node u parent of m by merging it with each reaction node child of m (that is, we apply $Replace(\rho(u), \alpha(m))$). In this way, if u is one of the new reaction nodes added to the replacement tree, the set $Chnr(u)$ should be small (because the cover set of the parent of u is the greatest possible), and will contain only metabolites in the sources $\mathcal{S}$. We can thus easily discard, by Lemma 6.3, those new reactions nodes whose $Chnr$ set of children is not minimal. This allows having some control on the amount of new reaction nodes added to the replacement tree.

Again, the algorithm does not build the replacement tree, but works directly on the network. We present a recursive version of the algorithm, which is basically the same as *MinSourcesHPsubtrees* but doing the reaction replacement instead of the cartesian union. The procedure *CompactHyperpaths* takes as input the label a of the metabolite m been analysed and the cover set of m. We suppose that m is not a leaf. It makes the necessary replacements until m has only reaction children whose own children are leaves. Calling $CompactHyperpaths(t, \{t\})$ and taking the source leaves of the reaction children of the root gives the sources of all hyperpaths to t. Therefore, those sets that are minimal will correspond to the minimal precursor sets of t.

The algorithm takes each reaction r producing a except those that, because of Lemma 6.3, do not need to be considered. Since new reactions producing a can be added during the course of the iterations, this selection is done dynamically by marking the reactions already analysed (we start, of course, with all reactions unmarked).

If reaction r has a child s which is not a leaf, then s is recursively transformed into a metabolite node with only leaves as grandchildren. Then, the set of reaction children of s are merged with r and the latter is removed. Each new (and therefore unmarked) reaction added produces a, and will thus be considered in the next iterations of the cycle until all its children are leaves.

CompactHyperpaths(a [label of m], $Cover$ [cover set of m]);
While there exists an unmarked reaction producing a do
    Choose r producing a with minimal $Subs(r) \setminus Cover$;
    Mark any reaction $\bar{r}$ producing a with $Subs(\bar{r}) \setminus Cover \supseteq Subs(r) \setminus Cover$;
    $NewCover := Cover \cup Subs(r) \cup Prod(r)$;

If exists *some* metabolite s substrate of r not in $Cover \cup \mathcal{S}$ do
CompactHyperpaths(s, $NewCover$);
Replace(r,s).

Corollary 6.6. *Applying the procedure* CompactHyperpaths$(t, \{t\})$ *over the network* G *gives a new network* G' *such that a set* X *is a precursor set of* $\{t\}$ *if and only if there exists a reaction* r *in* G' *with* $X \subseteq Subs(r)$.

6.5 Performance analysis

Some benchmarks have been done in order to measure the performance of five different algorithms over some real metabolic networks. These algorithms were compared for several different singleton target sets (p.e, aminoacids, compounds related to the synthesis of cell wall, DNA, RNA, membranes, etc.) in four networks of different size and topology. Table 6.1 presents an extract of these results for those targets that spent more time to obtain the minimal precursor sets. The table shows for each network the size of the set of compounds and reactions and for each target the number of the precursor sets found and the time in seconds that each algorithm spent. The algorithms analysed are:

- The original PITUFO VERSION, originally published in the WABI 2008 paper Cottret *et al.* (2008), which builds the replacement tree in order to find the minimal precursor sets (refered as Tree in the table).
- Two different implementations of the algorithm *MinSourcesHPsubtrees* that finds the HP-subtrees directly on the network without building the tree (refered as HP-subtree in the table). The first one analyses all possible hyperpaths from the targets to the sources and a second version that applies a test of minimality between alternative paths in order to avoid going further on directions that can not produce a minimal precursor set (according to Lemma 6.3).
- Two versions of the *CompactHyperpaths* algorithm (refered as Merge in the table) with the same variation described earlier to avoid go through all reactions.

All algorithms have been implemented in Java 6 and the running times were collected under a machine with a Intel 2.66 GHz processor and 4GB RAM memory from which 1GB was dedicated to the programs execution.

As expected, the algorithm that has the worst behaviour in almost all cases is the one that builds the replacement tree. This is the only version that has memory issues even for networks not specially large like the one of *Buchnera aphidicola* (418 compounds and 304 reactions). In the original work Cottret *et al.* (2008) this method was applied only after a compression of the network performed for a forward propagation step using as seeds glucose and some other nutrients, a pre-process not applied in our benchmark. The newer versions that works directly on the network does not have memory issues for the networks analysed

However, as can be observed in the *Escheria coli* results, it makes a huge difference to avoid going through all reactions. In effect, the HP-subtree version without this test did not finished processing in 24 hours of execution for none of the targets tested. The merge version finished but taking very long time compared to the version that checks the minimality. For instance, for the *E. coli* network and the compound GLY as target the variation was from 200 minutes to less than 1 second.

Comparing only the minimal reaction version of HP-subtree and Merge algorithms and the Replacement Tree version it is possible to make the following analysis. For the smaller network (*S. muelleri*) all implementations finished in less than 1 second and despite of some variations they are equivalent. For the network of *B. cicadellinicola*, which is almost four times bigger, the replacement tree version took more than one second for some cases (HEME-O and CPD-9454 target sets) but the time was still very fast. Note that for these instances the number of solutions is not greater than 2 precursor sets.

For the first three networks, both HP-subtree and Merge obtained the best performances for different target sets. However, for the bigger network *E. coli*, the Merge algorithm starts to have a considerably better performance for all targets presented. This result can be explained by the fact that this last algorithm has a cost in doing the merge of reactions, which is more advantageous when there are more hyperpaths from sources to the target sharing more reactions (since the merge of reactions avoid repeating part of the work already done). Of course, this have more chance to happen when the size of the network is greater, which could explain the better performance of this algorithm. It is difficult, tough, to detect exactly what are the variations that make one or the other version be better since it depends not only on the size of the network but also on the target set chosen and on the topology of the network.

Network	$\lvert\mathcal{C}\rvert/\lvert\mathcal{R}\rvert$	Target	# Prec. Sets	Tree	HP-subtree		Merge	
					All	Minimal	All	Minimal
S. muelleri	78 / 64	TRP	1	0.005	0.002	0.002	0.005	0.007
		LYS	1	0.010	0.003	0.002	0.003	0.003
		ARG	1	0.014	0.002	0.003	0.001	0.002
B. cicadellinicola	249 / 230	HEME-O	1	1.677	0.148	0.112	0.018	0.006
		CPD-9454	1	1.300	0.064	0.079	0.015	0.003
		ERYTHROSE-4P	1	0.006	0.002	0.002	0.082	0.006
		6-PYRUVOYL...	2	0.032	0.011	0.009	0.006	0.068
B. aphidicola	418 / 304	NAcMur-Peptide...	8	*	0.559	0.072	0.249	21.169
		GTP	5	*	1.100	0.007	0.006	1.013
		DGTP	1	*	1.135	0.011	1.622	0.001
		MET	4	*	1.217	0.002	10.183	0.004
E. coli	887 / 861	GLY	5	*	-	7.151	12,018.173	0.024
		NAcMur-Peptide..	16	*	-	78.170	0.493	0.104
		L-ALPHA-ALANINE	6	*	-	50.031	442.246	0.019
		KDO	1	*	-	26.432	17.717	0.003

Table 6.1: Runtime for computing precursor sets for some targets in 4 different networks, using a Pentium 4 2.66 GHz processor and 1GB RAM available for the JVM. In cases marked '-' the algorithm did not finish within 24 hours while in cases marked '*' it ran out of memory. Runtime is rounded up to seconds. The targets with their names abbreviated are *6-PYRUVOYL-5678-TETRAHYDROPTERIN* and *NAcMur-Peptide-NAcGlc-Undecaprenols*.

6.6 Some extensions

6.6.1 Enumerating hyperpaths from sources to the target set

In some cases we are interested not only in finding which set of sources are able to produce the target, but *how* the target set is produced from them. Given a minimal set of sources, it is easy to find *one* hyperpath just by computing the maximal target of the sources and successively removing reactions of the network that do not eliminate the target from the new maximal target.

We can enumerate *all* the hyperpaths from each minimal set of sources to the target by introducing some modifications to the algorithms proposed. Indeed, the information about the reactions contained in each HP-subtree or new reaction can be easily saved and additional tests of minimality of the set of reactions in the HP-subtrees can be applied (since minimality of reactions is required in the Definition 5.7 of hyperpaths). Also, if we want to enumerate all sets of sources (not necessarily of *minimal*) such that there exists a *precise* hyperpath from it to the target, we just need to remove the test of minimality of the sources, and discard repeated solutions (since the same solution can be obtained by different ways).

If we consider any method that enumerates the precursor sets of T, then we can introduce some modifications to the network in order to enumerate all the hyperpaths

from sets of sources to T. This requires just adding, for each reaction r, a fake substrate s_r and including s_r to the set of sources $\mathcal{S}$. Each set of sources given as solution will contain the real sources in the precursor set plus the fake sources indicating the reactions used. The minimality of the sources implies that the reactions used are minimal although the precursor set is not necessarily minimal.

6.6.2 Discarding trivial cycles

The model proposed implies that some precursor sets found cannot produce the target when stoichiometry information is added to the reactions. Indeed, the amounts of metabolites consumed and produced in some cycles involved might not be possible to maintain. In particular the trivial cycles composed by the two directions of a same reversible reaction are obviously not real cycles, however their metabolites are considered as available in the model. Clearly, these cycles are not realistic and should be discarded as a possible way of producing the target.

We present a way to discard any solution that, in order to produce the target, requires using both directions of a same reversible reaction. One method is, like before, to save the direction of the reversible reactions that participate in the HP-subtrees or new reactions that are computed, discarding those that use the same reaction in opposite direction.

For a general method that enumerates the precursor sets of T, we can modify the network in a similar way as before. For each direction $r+$ and $r-$ of any reversible reaction, we introduce a fake substrate for each direction s_{r+} and s_{r-}. We then add a new fake reaction p_r with substrates s_{r+} and s_{r-} and products the metabolites of the target T. In other words, we force the set $\{s_{r+}, s_{r-}\}$ to be a precursor set. Therefore, the method gives those sets as solutions as well as all precursor sets that do not contain both compounds s_{r+} and s_{r-}. Discarding the fake solutions and removing the fake compounds of the rest of solution, we obtain a collection of precursors that are not necessarily minimal. We need just to discard the non minimal sets to obtain the collection of all minimal precursor sets using the reactions in only one direction.

Conclusion and Perspectives

In this PhD, we presented some algorithms and complexity results for two general problems that arise in the analysis of a metabolic network: the search for elementary modes of a network and the search for minimal precursors sets.

Searching for elementary modes of a network

Elementary modes and minimal reaction cuts are common tools in the study of the cellular characteristic of a metabolic network. An elementary mode can be seen as a minimal set of reactions that can work in steady state independently of the rest of the network. It has therefore served as a mathematical model for the possible metabolic pathways of a cell. Their computation is not trivial and poses computational challenges. Some algorithms have been proposed based on variations of the double description method. Despite the algorithms proposed, no systematic complexity analysis had been carried out.

We showed that some problems, like checking consistency of a network, finding one elementary mode or checking that a set of reactions constitutes a cut are easy problems, giving polynomial algorithms based on LP formulations. In this way, we showed that they can be solved using existing LP software.

We also proved the hardness of central problems like finding a minimum size elementary mode, finding an elementary mode containing two given reactions, counting the number of elementary modes or finding a minimum reaction cut.

On the enumeration problem, we showed that enumerating all elementary modes containing one given reaction cannot be done in polynomial total time unless P=NP. This result provides some idea about the complexity of enumerating all the elementary modes. Although we did not solve the question whether the complexity of this enumeration problem can be done in polynomial total time, the result given in this

work can provide some insights about which strategies could be useful to answer this question. Indeed, this is a major open issue, which corresponds to a particular case of the enumeration of vertices of a bounded polyhedron, whose complexity is, in its turn, one of the major open questions in computational geometry.

Beyond that, Schwartz et Kanehisa (2006) have shown that all elementary modes are not equal contributors to physiological cellular states. It remains an open biological question how to identify elementary modes that are physiologically relevant to which Schwartz and Kanehisa (2006) provide some pointers. Once this is fully answered, it will then become an open question whether such subsets of all elementary modes can be more efficiently enumerated.

Searching for minimal precursors sets

The search for precursor sets is motivated by discovering which external metabolites are sufficient to allow the production of a given set of target metabolites. In contrast with previous proposals, we presented a new approach which is the first to formally consider the use of cycles in the way to produce the target.

In our definition, we introduce the concept of internal supplies as compounds that can be used in the forward propagation subject to be regenerated by the same process. Such compounds can also have a biological importance in the sense of representing necessary supplies to start a process but that can be discarded once the process is ongoing.

We presented also an alternative characterisation of precursor sets which does not require the previous search of internal supplies and which allows to compute the maximal target of a set of sources. With this maximal target, we present a polynomial algorithm to decide whether a set is a precursor set of a given target. We also show that, given a target set, finding a minimal precursor set is easy but finding a precursor set of minimum size is NP-hard. We further show that finding a solution with minimum size internal supply is NP-hard.

We give a simple characterisation of precursors sets by the existence of hyperpaths between the solutions and the target. The definition of hyperpaths could be a useful concept not only to model precursor sets but also for other processes occurring in the network. Indeed, hyperpaths could help in giving a more formal definition to the biological concept of metabolic pathway.

If we consider the enumeration of all the minimal precursor sets of a given target,

we found that this problem cannot be solved in polynomial total time unless P=NP. Despite this negative theoretical result, we presented two algorithms that have good performance for medium-size networks. Both algorithms enumerate all the minimal hyperpaths between the sources to the target by traversing, starting from the target, the network in reverse order. By a comparative test, some hyperpaths that are not part of the solution can be discarded in early iterations. A future work on this algorithm should improve or add new tests that allow avoiding traversing reactions that cannot lead to minimal solutions.

The second algorithm works in a similar way but modifies in each step the network in order to "remember" the solutions already found. This extra process can increase the time by iteration, but can decrease considerably the total time of the process when minimal hyperpaths have many reactions in common. It could be interesting to have an algorithm combining the two algorithms presented, that evaluates applying this modification to the network only if a reaction has a high probability of belonging to many hyperpaths. Improvements of both algorithms should also go on the direction of including stoichiometry that could be checked by some simple LP formulations in each iteration.

We also defined the concept of precursor cut set, that is a useful way to avoid producing a given compound of the network. We showed for this problem similar results of complexity: it is easy to find a minimal cut but it is NP-hard to find a cut of minimum size. We also show that this problem is hard to enumerate in polynomial total time. In a more theoretical perspective, we show that enumerating all minimal precursor sets of a target is dual to the problem of enumerating all the minimal cut sets of the same target. Moreover, using a result of Gurvich et Khachiyan (1999), we showed that enumeration of both at the same time can be done in quasi-polynomial total time.

Bibliography

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A. et Protasi, M. (1999). *Complexity and approximation – Combinatorial optimization problems and their approximability properties*. Springer-Verlag, Berlin.

Ausiello, G., Franciosa, P. G. et Frigioni, D. (2001). Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. *In* Restivo, A., Rocca, S. R. D. et Roversi, L., éditeurs : *ICTCS*, volume 2202 de *Lecture Notes in Computer Science*, pages 312–327. Springer.

Bang-Jensen, J. et Gutin, G. (2008). *Digraphs: Theory, Algorithms and Applications,Ää*. Springer-Verlag, London.

Berge, C. (1989). *Hypergraphs, Combinatorics of Finite Sets*. North-Holland.

Caspi, R. *et al.* (2006). MetaCyc: a multiorganism database of metabolic pathways and enzymes. *Nucleic Acids Res*, 34(Database issue):D511–D516.

Clarke, B. L. (1981). Complete set of steady states for the general stoichiometric dynamical system. *J. Chem. Phys.*, 75:4970–4979.

Cook, S. A. (1971). The complexity of theorem-proving procedures. *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158.

Cottret, L., Milreu, P., Acuña, V., Marchetti-Spaccamela, A., Martinez, F. V., Sagot, M.-F. et Stougie, L. (2008). Enumerating precursor sets of target metabolites in a metabolic network. *In Workshop on Algorithms in Bioinformatics (WABI)*, volume 5251 de *Lecture Notes in Computer Science*, pages 233–244. Springer.

Covert, M. W. et Palsson, B. O. (2003). Constraints-based models: Regulation of gene expression reduces the steady-state solution space. *J. Theor. Biol.*, 221:309–325.

Diestel, R. (2006). *Graph theory*. Springer-Verlag, New York.

Dyer, M. (1983). The complexity of vertex enumeration methods. *Mathematics of Operations Research*, 8(3):381–402.

Dyer, M. et Proll, L. (1977). An algorithm for determining all extreme points of a convex polytope. *Mathematical Programming*, 12:81–96.

Edwards, J. S., Ibarra, R. U. et Palsson, B. O. (2001). In silico predictions of Escherichia coli metabolic capabilities are consistent with experimental data. *Nat. Biotechnol.*, 19(2):125–130.

Edwards, J. S. et Palsson, B. O. (2000). Robustness analysis of the escherichia coli metabolic network. *Biotechnol Prog*, 16(6):927–939.

Fong, S. S. et Palsson, B. O. (2004). Metabolic gene-deletion strains of *escherichia coli* evolve to computationally predicted growth phenotypes. *Nat. Genet.*, 36(10): 1056–1058.

Francke, C., Siezen, R. J. et Teusink, B. (2005). Reconstructing the metabolic network of a bacterium from its genome. *Trends Microbiol*, 13(11):550–558.

Fredman, M. et Khachiyan, L. (1996). On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*.

Fukuda, K. et Prodon, A. (1996). Double description method revisited. *In Combinatorics and Computer Science*, volume 1120 de *Lecture Notes in Computer Science*, pages 91–111. Springer.

Gagneur, J. et Klamt, S. (2004). Computation of elementary modes: a unifying framework and the new binary approach. *BMC Bioinformatics*, 5:175.

Gallo, G., Gentile, C., Pretolani, D. et Rago, G. (1998). Max Horn SAT and the minimum cut problem in directed hypergraphs. *Math. Program.*, 80:213–237.

Gallo, G., Longo, G., Nguyen, S. et Pallottino, S. (1993). Directed hypergraphs and applications. *Citeseer*.

Garey, M. R. et Johnson, D. S. (1979). *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freeman.

Gurvich, V. et Khachiyan, L. (1999). On generating the irredundant conjunctive and disjunctive normal forms of monotone boolean functions. *Discrete Applied Mathematics*.

Hall, P. (1935). On representatives of subsets. *J. London Math. Soc.*, s1–10(1):26–30.

Handorf, T., Christian, N., Ebenhöh, O. et Kahn, D. (2007). An environmental perspective on metabolism. *J Theor Biol*.

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

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

Kanehisa, M. *et al.* (2006). From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res*, 34(Database issue):D354–D357.

Khachiyan, L., Boros, E., Borys, K., Elbassioni, K. et Gurvich, V. (2008). Generating all vertices of a polyhedron is hard. *Discrete and Computational Geometry*, 39:174–190.

Khachiyan, L., Boros, E., Elbassioni, K., Gurvich, V. et Makino, K. (2005). On the complexity of some enumeration problems for matroids. *SIAM Journal on Discrete Mathematics*, 19(4):966–984.

Klamt, S. (2006). Generalized concept of minimal cut sets in biochemical networks. *Biosystems*, 83(2-3):233–247.

Klamt, S., Gagneur, J. et von Kamp, A. (2005). Algorithmic approaches for computing elementary modes in large biochemical reaction networks. *IEE Proc.-Syst. Biol.*, 152(4):249–255.

Klamt, S. et Gilles, E. D. (2004). Minimal cut sets in biochemical reaction networks. *Bioinformatics*, 20(2):226–234.

Klamt, S. et Stelling, J. (2002). Combinatorial complexity of pathway analysis in metabolic networks. *Mol. Biol. Rep.*, 29(1-2):233–236.

Klamt, S. et Stelling, J. (2003). Two approaches for metabolic pathway analysis? *Trends Biotechnol.*, 21(2):64–69.

Lacroix, V., Cottret, L., Thébault, P. et Sagot, M.-F. (2008). An introduction to metabolic networks and their structural analysis. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 5(4):594–617.

Lacroix, V., Fernandes, C. G. et Sagot, M.-F. (2006). Motif search in graphs: application to metabolic networks. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 3(4):360–368.

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

Motzkin, T., Raiffa, H., Thompson, G. et Thrall, R. (1953). The double description method. *In* Kuhn, H. et Tucker, A., éditeurs : *Contributions to the Theory of Games*, volume II, pages 51–73. Princeton University Press.

Oxley, J. G. (1992). *Matroid theory*. Oxford Science Publications. The Clarendon Press Oxford University Press, New York.

Palsson, B. O. (2000). The challenges of *in silico* biology. *Nat. Biotechnol.*, 18:1147–1150.

Papadimitriou, C. H. (1994). *Computational complexity*. Addison-Wesley.

Raz, R. et 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)*, pages 475–484.

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

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

Schrijver, A. (1986). *Theory of Linear and Integer Programming*. John Wiley & Sons.

Schrijver, A. (2003). *Combinatorial optimization: Polyhedra and Efficiency*, volume 24 de *Algorithms and Combinatorics*. Springer-Verlag, Berlin.

Schuster, S., Fell, D. A. et Dandekar, T. (2000). A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nat. Biotechnol.*, 18(3):326–332.

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

Schuster, S., Hilgetag, C., Woods, J. H. et Fell, D. A. (2002a). Reaction routes in biochemical reaction systems: algebraic properties, validated calculation procedure and example from nucleotide metabolism. *J. Math. Biol.*, 45(2):153–181.

Schuster, S., Pfeiffer, T., Moldenhauer, F., Koch, I. et Dandekar, T. (2002b). Exploring the pathway structure of metabolism: decomposition into subnetworks and application to *mycoplasma pneumoniae*. *Bioinformatics*, 18(2):351–361.

Schwartz, J. et Kanehisa, M. (2006). Quantitative elementary mode analysis of metabolic pathways: the example of yeast glycolysis. *BMC Bioinformatics*, 7:186.

Seymour, P. D. (1977). The matroids with the max-flow min-cut property. *J. Comb. Theory Ser. B*, 23(7):189–222.

Stelling, J. (2004). Mathematical models in microbial systems biology. *Current opinion in microbiology*.

Szallasi, Z., Stelling, J. et Periwal, V. (2006). *System Modeling in Cellular Biology: From Concepts to Nuts and Bolts*. The MIT Press.

Terzer, M. (2009). *Large Scale Methods to Enumerate Extreme Rays and Elementary Modes*. PhD-Thesis, ETH Zürich.

Valiant, L. G. (1979). The complexity of computing the permanent. *Theor. Comp. Sci.*, 8:189–201.