Logic programming tools for metabolic fluxes analysis and biological applications

Maxime Mahout

université PARIS-SACLAY

Logic programming tools for metabolic fluxes analysis and biological applications

Programmation logique pour l'analyse des flux métaboliques et applications à la biologie

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580 : Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et sciences du numérique.
Référent : Faculté des sciences d'Orsay

Thèse préparée au **Laboratoire Interdisciplinaire des Sciences du Numérique** (Université Paris-Saclay, CNRS), sous la direction de **Sabine PERES**

Thèse soutenue à Paris-Saclay, le 28 Novembre 2023, par

Maxime MAHOUT

Composition du jury

Membres du jury avec voix délibérative

Wolfram LIEBERMEISTER Directeur de recherche, INRAE Jouy-en-Josas	Président
Loïc PAULEVÉ Directeur de recherche, LaBRI Université de Bordeaux	Rapporteur & Examinateur
Delphine ROPERS Directrice de recherche, INRIA Grenoble – Rhône-Alpes	Rapporteur & Examinatrice
François FAGES Directeur de recherche, INRIA Saclay	Examinateur
Margit HEISKE Cadre scientifique, iMEAN, Entreprise	Examinatrice
Éric PELLETIER Directeur de recherche, CEA Genoscope Evry	Examinateur

THESE DE DOCTORAT

NNT : 2023UPASG086

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Programmation logique pour l'analyse des flux métaboliques et applications à la biologie

Mots clés: Biologie des sytèmes, Réseaux métaboliques, Programmation logique, Analyse des voies métaboliques, Problèmes combinatoires, Cibles thérapeutiques

Résumé: En biologie des systèmes, l'analyse des voies métaboliques est une méthode essentielle pour étudier le métabolisme et améliorer la compréhension du fonctionnement des systèmes vivants. Deux concepts clés sont l'analyse des modes élémentaires de flux (EFMs), qui permet de décrire les réseaux métaboliques en termes de voies minimales, et les Minimal Cut Sets (MCSs), représentant les coupures minimales de flux du réseau en termes de réactions.

Dans le cadre de cette thèse, nous avons développé une méthode de programmation logique pour le calcul des modes élémentaires de flux: *aspefm*. L'outil est une méthode de raisonnement automatique à base de Answer Set Programming (ASP), étendue par des contraintes linéaires. Cette approche permet de récupérer des voies lorsque les méthodes classiques ne le peuvent pas, d'interroger directement le réseau et d'éviter l'explosion en mémoire. La méthode peut prendre en compte des contraintes biologiques importantes de tous types, ce que nous avons illustré sur un réseau central d'*Escherichia coli*. Elle est aussi applicable aux réseaux à l'échelle du génome, et calcule plus aisément des solutions de large taille que les méthodes à base de programmation linéaire.

Notre méthode a été appliquée, à la bactérie pathogène *Pseudomonas aeruginosa* (PA) qui est présente dans 80% des plaies chroniques. PA utilise des stratégies écologiques différentes de celles des bactéries modèles comme *E. coli*. Elle est retrouvée généralement dans les plaies chroniques avec une autre bactérie infectieuse, *Staphylococcus aureus* (SA). Nous supposons que leurs deux métabolismes sont complémentaires, ce qui permet une production de biomasse plus élevée conduisant à des mauvais pronostics pour les patients.

L'extension de notre outil *aspefm* à l'analyse des MCSs sur un modèle de consortium de ces deux bactéries nous a permis de retrouver des métabolites dont l'échange entre les deux bactéries permettrait de compenser des phénotypes prédits létaux, ainsi que d'explorer des cibles thérapeutiques potentielles contre les bactéries. Par ailleurs, dans un autre cadre, nous avons appliqué notre méthode de calcul au métabolisme de la cellule cancéreuse humaine et à la formation du stroma tumoral.

Title: Logic programming tools for metabolic fluxes analysis and biological applications

Keywords: Systems Biology, Metabolic Networks, Logic Programming, Metabolic Pathways Analysis, Combinatorial Exploration, Drug Discovery

Abstract: In systems biology, metabolic pathways analysis is an essential method to study metabolism and improve the understanding of biological systems. Key concepts include Elementary Flux Modes (EFMs), describing metabolic networks in terms of minimal pathways, and Minimal Cut Sets (MCSs), representing minimal cutting sets of reactions affecting network flux.

In the scope of this thesis, we developed a logic programming method for the computation of Elementary Flux Modes: *aspefm*. The tool is an automatic reasoning method based on Answer Set Programming (ASP), extended by linear constraints. This approach allows one to get minimal pathways when classical methods are unable to, and to directly query the network, helping with memory usage considerations. Important biological constraints of many different kinds can be integrated into the program, which we illustrated on a central metabolic model of *Escherichia coli*. The method is also applicable to genome-scale metabolic models, showing better performance than linear programming-based methods on enumeration of large-size solutions.

The method was applied to the pathogenic bacterium *Pseudomonas aeruginosa* (PA) found in 80% of chronic wounds. PA uses different ecological strategies than model bacteria. PA is commonly co-isolated from wounds with another opportunistic pathogen, *Staphylococcus aureus* (SA), and it is hypothesized the metabolisms of the two bacteria are complementary enabling higher biomass production and increasing wound bioburden leading to poor patient outcomes.

We extended our tool *aspefm* to the analysis of MCSs on a consortium model of these two bacteria, permitting us to retrieve exchanged metabolites involved in the recovery of growth after several intervention strategies, and leading to insights about potential therapeutic targets against the two bacteria. Furthermore, in an other context, we applied our computation method to cancer cell metabolism and tumoural stroma formation.

Je dédie cette thèse à mon père Claude Mahout,

Remerciements

C’est avec grande satisfaction que je vous présente mes travaux de thèse, intitulés: "Programmation logique pour l’analyse des flux métaboliques et applications à la biologie", comprenant maintenant bien trois à quatre années complètes de travail de recherche.

Je porte beaucoup d’attachement à ce sujet de thèse passionnant, à l’interface entre l’informatique et la biologie, qui a su me tenir en haleine pendant si longtemps, et à vrai dire je pense qu’il serait judicieux de dire que j’ai pu explorer activement les deux thématiques pourtant si différentes, comme cela était mon souhait.

Cette thèse n’aurait pas pu être possible sans l’affection considérable que me porte ma famille, et en particulier ma mère Lina Zakharia. Son soutien est toujours infaillible et je suis ravi de lui offrir par le biais de cet ouvrage symbole de finalisation des études un des plus beaux cadeaux que l’on puisse faire à son parent: le prestige académique.

Je voudrais spécialement remercier mes enseignantes Sandrine Vial et Hélène Débat, de l’Université Versailles Saint-Quentin, maintenant Université Paris-Saclay, pour leurs nombreuses contributions à la Double Licence de Biologie et d’Informatique dont je suis titulaire. Avoir étudié dans cette formation bi-disciplinaire, à laquelle j’ai été toujours très attaché, est une de mes fiertés personnelles. Également, je souhaite remercier mes enseignants de Master 2 Bio-Informatique à Orsay, pour leur considérable soutien vis-à-vis de ma candidature en deuxième année de master, puis en doctorat, et enfin pour la qualité de leur formation.

Je tiens par ailleurs à remercier l’ensemble des personnes contribuant à la relecture de ma thèse: mes rapporteurs Loïc Paulevé et Delphine Ropers, ainsi que mes examinateurs, et autres membres du jury: Wolfram Liebermeister, Margit Heiske, Éric Pelletier et François Fages. Je vous remercie très sincèrement d’avoir porté votre attention à ma thèse et d’avoir volontiers accepté d’assister à ma soutenance de thèse.

J’aimerais aussi remercier vivement Marie-France Sagot et son équipe pour leur merveilleux accueil au sein de leur laboratoire à l’Université Claude Bernard Lyon 1. Au cours de ces trois années riches en déplacement j’ai pu nouer de nombreuses relations dans le cadre du travail. Je souhaite donc également remercier toutes mes connaissances et tous mes collègues avec qui j’ai pu discuter au cours de ces années de doctorat. L’écoute et l’affection que mes collègues ont portées à mon égard m’ont toujours profondément touché et je leur en suis très reconnaissant. Le travail de thèse est un travail isolant, certes, mais c’est également un travail collaboratif, récompensant et fructueux, que je suis ravi d’avoir accompli.

Dans un autre temps, je tiens à exprimer ma gratitude à l'ensemble de nos collaborateurs dans la réalisation de mes travaux de thèse, notamment: Romain Attal, Laurent Schwartz, Ashraf Bakkar, et bien d'autres. Particulièrement, je tiens à remercier de tout coeur Laurent Simon. La qualité et la pertinence de son conseil m'ont permis, à de nombreuses reprises, de faire diverses avancées, que ce soit dans le domaine de la programmation, dans mes compétences en communication de mes résultats, et de manière générale par rapport à mes travaux de thèse. D'autre part, je remercie Ross Carlson, pour sa grande disponibilité, ses nombreuses connaissances, ses idées remarquables, et enfin sa grande contribution dans la rédaction en langue anglaise de mes articles, malgré les huit mille kilomètres qui nous séparent.

Enfin, j'aimerais infiniment remercier ma directrice de thèse, Sabine Peres, pour son accompagnement et sa supervision, durant ces trois années de thèse, ainsi que mes six mois de stage de Master 2, ainsi que mes deux mois de stage de Master 1. Son support inconsidérable tout au long de ces années a été une aide précieuse pour cette entreprise hasardeuse que consitue un doctorat. Je lui suis profondément reconnaissant de m'avoir engagé dans ce sujet ambitieux et innovant que l'on m'a proposé il y a trois ans, en collaboration directe avec Laurent Simon et Ross Carlson, de renommées internationales. En particulier, je voudrais saluer sa tenacité et son ambition qui m'ont permis aujourd'hui d'avoir une expertise si large, un goût du voyage, un réseau professionel fantastique et plus profondément une envie personnelle de développement académique sans limites.

Et si c'est avec le plus grand plaisir que je peux annoncer aujourd'hui que oui, notre méthode présente des cas d'utilisation où elle est bien plus performante que sa concurrence directe, ce résultat, qui fait finalement prendre tout son sens à ma recherche, est au fond le fruit de ma persévérance et de la persévérance de ma directrice, en particulier dans son soutien. Je veux donc remercier immensément Sabine de m'avoir apporté toute sa confiance dans ces domaines si vastes et si impressionants que constituent la biologie des systèmes et la modélisation informatique des systèmes vivants, confiance sans laquelle rien n'aurait été possible.

Août 2023,
Maxime Mahout

Résumé

En biologie des systèmes, l'analyse des voies métaboliques est une méthode essentielle pour étudier le métabolisme et pour améliorer la compréhension du fonctionnement des systèmes vivants. La modélisation à base de contraintes [Palsson, 2015] a énormément contribué à la connaissance des réseaux métaboliques et est devenue l'une des approches de modélisation la plus réussie dans ce domaine. Un concept clé est l'analyse des modes élémentaires de flux (EFMs) qui permet de décrire les réseaux métaboliques en termes de voies minimales [Schuster, Dandekar and Fell, 1999]. Leur application était jusqu'à présent limitée aux petits modèles métaboliques en raison de l'explosion combinatoire du nombre d'EFMs dans les grands réseaux, et ne pouvait donc pas passer à l'échelle du génome.

Par ailleurs, il existe une méthode complémentaire des EFMs, les Minimal Cut Sets (MCSs), dont le calcul peut être ramené à l'énumération d'EFMs particuliers sur un réseau métabolique dual [Ballerstein *et al*, 2012]. Cette méthode représente les coupures minimales de flux du réseau, et peut être appliquée afin de retrouver des réactions dites 'essentielles' à la croissance cellulaire, et d'évaluer la robustesse du réseau.

L'efficacité des algorithmes de calcul pour déterminer les EFMs a progressé régulièrement depuis l'introduction de la théorie EFMs. Cependant, l'énumération de tous les EFMs n'est pas nécessaire pour analyser les réseaux car dans la plupart des cas on s'intéresse seulement à l'existence (ou la non-existence) de quelques voies. Par conséquent, des méthodes à base de logique SMT (SAT modulo theories) et de MILP (Mixed-Integer Linear Programing) [Peres, Morterol and Simon, 2014; Morterol *et al*, 2016; de Figueiredo *et al*., 2009] ont été développées, permettant d'interroger les réseaux métaboliques et de trouver des EFMs sans devoir énumérer toutes les solutions.

Dans le cadre de cette thèse, nous avons développé une méthode de programmation logique pour le calcul des EFMs: *aspefm* [Mahout, Carlson and Peres, 2020], à base de Answer Set Programming (ASP), étendue par des contraintes linéaires [Janhunen *et al*, 2017]. Cette méthode permet de récupérer des voies lorsque les méthodes classiques ne le peuvent pas, d'interroger directement le réseau et d'éviter l'explosion en mémoire. En particulier, la méthode se révèle efficace pour le calcul de voies minimales de grande taille, en comparaison avec les méthodes MILP. La méthode peut prendre en compte des contraintes biologiques importantes de tous types, ce que nous avons illustré sur un réseau central d'*Escherichia coli* [Mahout, Carlson and Peres, 2020]. Elle est aussi applicable aux réseaux à l'échelle du génome.

Notre méthode a été appliquée, en collaboration avec Ross Carlson (Montana State University), à la bactérie pathogène *Pseudomonas aeruginosa* (PA) qui est présente dans ~80% des plaies chroniques. PA utilise des stratégies écologiques différentes de celles des bactéries modèles comme E. coli [McGill *et al*, 2021]. Elle est retrouvée généralement dans les plaies avec une autre bactérie pathogène, *Staphylococcus aureus* (SA). Nous supposons que leurs deux métabolismes sont complémentaires, ce qui permet une production de biomasse plus élevée conduisant à des mauvais pronostics pour les patients.

L'extension de notre outil *aspefm* à l'analyse des MCSs sur un modèle de consortium de ces deux bactéries nous a permis de retrouver des métabolites dont l'échange entre les deux bactéries permettrait de compenser des phénotypes prédits léthaux, ainsi que d'explorer des cibles thérapeutiques potentielles contre les bactéries [Mahout *et al*, 2023a]. Par ailleurs, dans un autre cadre, nous avons appliqué notre méthode de calcul des EFMs au métabolisme de la cellule cancéreuse humaine et à la formation du stroma tumoral [Mahout *et al*, 2023b].

Le manuscrit de thèse comporte cinq chapitres et deux annexes. Le premier chapitre est une introduction à la modélisation des systèmes vivants ainsi qu'une description du domaine de la biologie des systèmes, complétée par la remise en contexte de la modélisation à base de contraintes dans son champ de recherche plus global en bio-informatique. Le deuxième chapitre décrit le domaine de la modélisation à base de contraintes ainsi que les méthodes abordées dans la thèse: dont EFMs, MCSs et Flux Balance Analysis (FBA). Le troisième chapitre décrit l'utilisation d'ASP pour calculer les EFMs, complétée par l'ajout de contraintes de différents types. Ce chapitre inclut le développement de notre méthode *aspefm*, ainsi que son application à un modèle métabolique central de *Escherichia coli* [Mahout, Carlson and Peres, 2020], et à une cellule tumorale humaine [Mahout *et al*, 2023b].

Le quatrième chapitre décrit l'utilisation d'ASP pour calculer les MCSs, ainsi que son application aux métabolismes de *Pseudomonas aeruginosa* et *Staphylococcus aureus* [Mahout *et al*, 2023a], révélant les différents types de métabolites permettant aux bactéries de récupérer une croissance normale malgré des délétions de gènes, lors de leur développement dans les biofilms associés aux plaies chroniques. Enfin, le dernier chapitre élabore les perspectives et conclusions de ces travaux de thèse. Les annexes développent quant à elles différents éléments manquant au document principal, dont l'intégration de la compilation de connaissances à Answer Set Programming, élément qui a été un des points de départ du projet de thèse.

[Palsson, 2015] B. Ø. Palsson, *Systems biology: Constraint-based reconstruction and analysis*. Cambridge University Press, 2015. [Online]. Available: https://books.google.fr/books?id=QNBpBgAAQBAJ.

[Schuster, Dandekar and Fell, 1999] S. Schuster, T. Dandekar, and D. A. Fell, "Detection of elementary modes in biochemical networks : A promising tool for pathway analysis and metabolic engineering," *Trends Biotechnol.*, vol. 17, pp. 53–60, 1999, doi: 10.1016/S0167-7799(98)01290-6.

[Ballerstein *et al*, 2012] K. Ballerstein, A. von Kamp, S. Klamt, and U.-U. Haus, "Minimal cut sets in a metabolic network are elementary modes in a dual network," *Bioinformatics*, vol. 28, no. 3, pp. 381–387, Feb. 2012, doi: 10.1093/bioinformatics/btr674.

[Peres, Morterol and Simon, 2014] S. Peres, M. Morterol, and L. Simon, "SAT-Based Metabolics Pathways Analysis without Compilation," *Lecture Note in Bioinformatics*, vol. 8859, doi: 10.1007/978-3-319-12982-2_2.

[Morterol *et al*, 2016] M. Morterol, P. Dague, S. Peres, and L. Simon, "Minimality of metabolic flux modes under boolean regulation constraints," in *Workshop on constraint-based methods for bioinformatics (WCB)*, 2016.

[de Figueiredo *et al*., 2009] L. F. de Figueiredo *et al*., "Computing the shortest elementary flux modes in genome-scale metabolic networks," *Bioinformatics*, vol. 25, no. 23, pp. 3158–3165, 2009, doi: 10.1093/bioinformatics/btp564.

[Mahout, Carlson and Peres, 2020] M. Mahout, R. P. Carlson, and S. Peres, "Answer Set Programming for Computing Constraints-Based Elementary Flux Modes: Application to *Escherichia coli* Core Metabolism," *Processes*, vol. 8, no. 12, p. 1649, Dec. 2020, doi: 10.3390/pr8121649.

[Janhunen *et al*, 2017] T. Janhunen, R. Kaminski, M. Ostrowski, T. Schaub, S. Schellhorn, and P. Wanko, "Clingo goes linear constraints over reals and integers," *CoRR*, vol. abs/1707.04053, 2017, [Online]. Available: https://doi.org/10.1017/S1471068417000242.

[McGill *et al*, 2021] S. L. McGill, Y. Yung, K. A. Hunt, M. A. Henson, L. Hanley, and R. P. Carlson, "*Pseudomonas aeruginosa* reverse diauxie is a multidimensional, optimized, resource utilization strategy," *Sci Rep*, vol. 11, no. 1, Art. no. 1, Jan. 2021, doi: 10.1038/s41598-020-80522-8.

[Mahout *et al*, 2023a] M. Mahout, R. P. Carlson, L. Simon, and S. Peres, "Logic programming-based Minimal Cut Sets reveal consortium-level therapeutic targets for chronic wound infections", *Publication in submission process.*

[Mahout *et al*, 2023b] M. Mahout, L. Schwartz, R. Attal, A. Bakkar, and S. Peres, "Metabolic modelling links Warburg effect to collagen formation, angiogenesis and inflammation in the tumoural stroma", *Publication in submission process.*

Contents

List of Figures

List of Tables

List of Definitions

Chapter 1

Introduction

1.1 Introduction to life sciences

Life sciences, or biology, is the study of all living organic material, all living organisms, and their observable mechanisms. An organic object is usually said to be living if it is self-contained, can manage its resources on its own, and has the greater purpose of reproducing itself. A living cell could be the smallest unit considered an organism. For such a cellular organism, the act of reproduction consists of sharing its vessel into two, by processes such as binary fission or mitosis, and sharing material such as DNA, RNA, lipids, proteins along the two subcomponents.

This is not the only definition of living unit possible, as viruses and mobile genetic elements replicate themselves but do not adhere to the reproduction scheme described above [1, 2]. As well, some argued nucleic acids themselves could be the smallest self-replicating units characterizing life [1]. This would be supported by the hypothesis that RNA ribozymes are the earliest enzymes on Earth [3]. Scientists argue the origin of life should be differentiated from the origin of replication, and from the origin of metabolism [4, 1].

There are four major building blocks of life: lipids, sugars, amino acids, and nucleic acids [5]. Nucleic acids are mainly used to store genetic information, lipids to construct membranes, amino acids to construct proteins. Life sciences studies every intricate part of life, from the sharing of genetic material, to production of RNA, proteins, lipids, lipoproteins and to which eye color is most prevalent in the *Homo sapiens* species.

In particular, life is divided in many realms, phyla, genus and species. The three arguably defined realms are bacteria, archea and eucaryotes. Prokaryotes include bacteria and archaea which lack a nucleus, while eukaryotes are distinguishable by possessing a nucleus, and sometimes organelles such as chloroplasts and mitochondria, which share similarities to bacteria [5].

Organisms share a great deal of similarities with each other, inherited from their common DNA. They display common characteristics from the essential parts of their metabolism in common, but they might largely differ in the way they interact with other organisms and chemical compounds.

1.2 Metabolism and interactions between organisms

At the center of every observable biological mechanism, metabolism plays a role. Metabolism is the set of all chemical reactions in a cellular organism. Almost all essential biochemical reactions can be catalyzed by proteic enzyme complexes, which are often synthesized by the organism itself. In fact, translation of DNA into RNA, and traduction of RNA into proteins, are also done by enzymatic complexes of an organism's core metabolism.

Some theorists argue that metabolism should be at the center of the origin of life debates, as evidenced by the presence of ATP synthase and proton-motive forces in all known non-viral living organisms [6]. ATP, or adenosine triphosphate, is unanimously considered the main source of energy of all living cells.

The chemical reactions of metabolism are organized into metabolic pathways, in which one metabolic species are transformed through a series of steps into other chemicals, each step being facilitated by a specific enzyme. The metabolic species are called metabolites.

The major biochemical pathways are described in well-known, well-curated metabolic pathway maps [7, 8]. Meanwhile, metabolic pathways such as aromatic amino acids biosynthesis, that do not belong to the core metabolism: the part of metabolism shared by all organisms, would be called secondary metabolic pathways [9]. Mammals who cannot synthesize aromatic amino acids must obtain them from other organisms.

Metabolism is usually linked to cell survival; catabolism is the destruction of elementary bricks to produce energy, while anabolism is the synthesis of elementary bricks from available energy. According to the 'the origin of life is the origin of replication' theory, a cell's goal is usually said to be to replicate itself, that is, to produce biomass – mass of cells – a quantifiable amount of growth.

Such biomass can be measured in grams of Dry Weight (gDW). Grams of dry weight are usually defined for bacterial colonies, but the term can also apply to eukaryotes, such as for yeasts. Dry weight is quantified *in vitro* after centrifugation of colonies, either by accordingly weighting the dry cells, or by colorimetry - *i. e.* optical density (OD) - though the linear relation of OD to dry weight is not strong [S1].

Inside a cell, or a bacterial community, and in our body, many metabolic interactions exist. Metabolites are being exchanged at every turn. For example, human cells have mitochondria, in which metabolites are exchanged for ATP production in a fermentation process, generating energy. Cooperation between organisms is called symbiosis [5].

Strong evidence exists to suggest that most diseases are the result of collaboration between pathogenous organisms, with a poor response from the human host, leading to worse patient outcomes [10, 11]. An example of this would be microbes cooperatively forming biofilms, such as the bacteria *Staphylococcus aureus* and *Pseudomonas aeruginosa* [11]. These collaborations often involve points of junction in the metabolism of the organisms.

Organisms generate considerable amount of wastes, to which by law of entropy, if the wastes are not recycled properly, accumulation of wastes happen, which is toxic to neighbouring cells and organisms. An example of cells with such a chaotic metabolism: high anabolism and low catabolism, would be tumoural cells [12].

1.3 Enzymatic reactions

Biochemical reactions are largely known to be thermodynamically unfavorable, unless they are catalyzed by enzymes [13]. Enzymes come largely in proteic forms, and they are most often associated with cofactors, including ions and vitamins. Purich defines an enzyme as 'a biological catalyst for making and/or breaking chemical bonds', and estimates their number of unique catalyzed reactions at over 10 000 [14]. Enzymes have catalytic activity for many different chemical processes, and thus an attempt to describe them all exists in the form of Enzyme Commission (E.C.) numbers [15].

A ligand of an enzyme is a metabolite that can bind to its active site, usually a cavity inside the enzyme where the species can form hydrogen bonds with the protein. A metabolite that is metabolized through the biochemical reaction process catalyzed by the enzyme is called a substrate. The resulting metabolite after the biochemical reaction happened is called a product. Both substrates and products are enzyme ligands, and the reaction happens in the active site.

Enzymatic complexes designates complexes of several subunits: whether they would be proteins, peptides, nucleic acids, cofactors, ligands, other metabolites, all with possibly multiple purposes. A protein might be the subunit of several different enzymes with different metabolic functions, and an enzyme might catalyze a very wide variety of different biological processes.

Many substrates of biochemical reactions are coenzymes, such as NAD (Nicotinamide adenine dinucleotide), ATP (Adenosine Triphosphate), CoA (Coenzyme A), and most vitamins [13]. Through the process of the biochemical reactions, these coenzymes get reduced or oxidized, in the case of NAD, phosphoryled or dephosphoryled, in the case of ATP, and acylated or deacylated, in the case of CoA [13].

Other coenzymes taking part in oxidoreducing reactions include NADP/NADPH, FAD/$FADH_2$, ubiquinol/ubiquinone, cytochromes a/b/c/d, and other phospate sources include GTP.

Coenzymes play a central role in major biochemical pathways such as glycolysis, Krebs cycle, photosynthesis, etc. As an example, in human cells, aerobic glycolysis, that is, glycolysis in presence of oxygen, followed by transport of pyruvate into mitochondria, pyruvate dehydrogenase, Krebs cycle, and respiration through the mitochodrial electron transport chain, is said to be producing 31 molecules of ATP, helped by oxidation of 10 NADH and 2 $FADH_2$ [7].

1.3.1 Enzyme kinetics

The study of whether chemical reactions can proceed or not is called thermodynamics [16]. The study of rates at which chemical reactions occur is called kinetics [13, 17]. Kinetics and in particular enzyme kinetics are fields of study involving the discovery of kinetics parameters: a set of variables describing reaction rates.

In biological life, most enzymes catalyze bisubstrate reactions: reactions with two substrates and one or more products. Monosubstrate reactions are infrequent and trisubstrate reactions are less common than the bisubstrate reactions [18, 15]. Chemical reactions are considered in aqueous solutions since life is mainly composed of water [5].

A classical model for the description of biochemical reactions is described in this section. Three types of reactions are usually considered:

$$\begin{aligned} A \rightarrow P \quad & \text{unimolecular} \quad \text{order 1} \\ A + B \rightarrow P \quad & \text{bimolecular} \quad \text{order 2} \\ A + B + C \rightarrow P \quad & \text{termolecular} \quad \text{order ?} \end{aligned} \tag{1.1}$$

The molecularity of a chemical reaction refers to the theoritical mechanism in play, while its order refers to the mathematical model of the kinetics of the reaction. The order of reaction defines how many concentration terms must be multiplied together to get its rate law [18].

Kinetics is another word for dynamics, meaning that essentially the study of reaction rates should be understood as the study of rates of metabolite consumption over time. Throughout this thesis, $[X]$ denotes the concentration of a metabolite X, while $[X]_0$ denotes the initial concentration of a metabolite X, at time $t = 0$.

For a first-order reaction $A \xrightarrow{k} P$, with rate constant k in s^{-1} , the rate law gives reaction velocity, as:

$$v = \frac{d[P]}{dt} = -\frac{d[A]}{dt} = k[A] = k([A]_0 - [P]) \tag{1.2}$$

Integration of equation 1.2 gives equation 1.3. Unstable substances such as radioactive nuclei undergo decomposition through unimolecular reactions, of first-order.

$$\begin{aligned} [P](t) &= [A]_0(1 - e^{-kt}) \\ [A](t) &= [A]_0 e^{-kt} \end{aligned} \tag{1.3}$$

For a second-order reaction $A + B \xrightarrow{k} P$, with k in $mol.L^{-1}.s^{-1}$ the rate law gives reaction velocity, as:

$$v = \frac{d[P]}{dt} = -\frac{d[A]}{dt} = -\frac{d[B]}{dt} = k[A][B] \tag{1.4}$$

Termolecular reactions are unusual, as the simultaneous collision of three molecules is a very rare event [13]. Usually, these are rather the combination of two elementary steps of second order, such as $A + B \rightarrow X$ and $C + X \rightarrow P$ [18]. The order of any reaction can be determined experimentally by observing the rate v, measuring the consumption of substrate or product over time, and checking if the above models fit [13, 17].

1.3.2 Enzyme thermodynamics

On the other end, reaction thermodynamics is concerned with how far a reaction can proceed. Regardless of how fast an enzymatic reaction is, it cannot continue beyond the point of thermodynamical equilibrium [16]. As well,

reaction thermodynamics involves studying reversibility of reactions using measurable differences of energies. Note that only difference of energies can be measured, not energies themselves.

A reaction $A + B \rightleftharpoons Y + Z$ is simply said to be reversible if from Y and Z, A and B can also be produced. The direction $A + B \rightarrow Y + Z$ is called the forwards direction of the reaction and the direction $Y + Z \rightarrow A + B$ the backwards direction. Let us define the reaction in equation 1.5.

$$aA + bB \rightleftharpoons yY + zZ \tag{1.5}$$

The partial Gibbs molar free energy of A through reaction equation 1.5 is approximately given by the relation [13]:

$$\bar{G}_A = RT \ln [A] + \bar{G}^{\circ}_A \tag{1.6}$$

Where $\bar{G}^{\circ}_A$ denotes the Gibbs partial molar free energy of A in its standard state. The standard state is under the following chemical conditions: $[A] = 1\ M = 1\ mol.L^{-1}$, temperature $T = 25°$C; pressure $P = 1$ atm. R denotes the gas molar constant, and T the temperature of the system, which energy computations are highly dependant on.

In biological reaction systems, the most important thermodynamic parameter is ΔG the variation in Gibbs free energy [13]. A non null ΔG is indicative of a biological process that is spontaneous, that can proceed, regardless of the rate at which it can proceed. A null ΔG is indicative of a biological process that has reached equilibrium.

The variation of Gibbs free energy of reaction equation 1.5 is given by :

$$\Delta G = (y\bar{G}_y + z\bar{G}_z) - (a\bar{G}_a + b\bar{G}_b) \tag{1.7}$$

From equation 1.7 and equation 1.6, we derive the following:

$$\Delta G = \Delta G^{\circ} + RT \ln \left(\frac{[C]^c [D]^d}{[A]^a [B]^b} \right) \tag{1.8}$$

The property of ΔG is determining the direction of a reaction, *i. e.* if $\Delta G < 0$, then the forwards direction of the reversible reaction is the direction that can proceed, and conversely, if $\Delta G > 0$, then the backwards direction of the reaction can proceed. As seen in equation 1.8, ΔG varies greatly on the concentration of reactants and products.

This chapter is illustrated on the example of reaction equation 1.5 but more generally, we have:

$$\Delta G = \Delta G^{\circ} + RT \ln \left(\frac{\prod_{Pr \in Products} [Pr]^{stoch_{Pr}}}{\prod_{Rc \in Reactants} [Rc]^{stoch_{Rc}}} \right) \tag{1.9}$$

At equilibrium, the free energy variation is null, $\Delta G = 0$, yielding:

$$\Delta G^\circ = RT \ln \left(\frac{[C]_{eq}^c [D]_{eq}^d}{[A]_{eq}^a [B]_{eq}^b} \right) = RT \ln K_{eq} \tag{1.10}$$

Where $[X]_{eq}$ denotes the concentration of X such that the reaction is at equilibrium. K_{eq} is the familiar equilibrium constant of a reaction, its value depending on concentrations of products and reactants. As can be seen by the relation in equation 1.10, K_{eq} can also be numerically defined directly from the variation of free energy in standard state ΔG°, $K_{eq} = e^{-\Delta G^\circ / RT}$.

The variation of free energy in standard state ΔG° can be further defined in terms of free energy of *formation* $\Delta G_\mathrm{f}^\circ$ of every metabolite in its standard state. This describes the change of energy accompanying the formation of 1 mol of a substance in its standard state, from energy of its molecular composition in their standard states [13].

$$\Delta G^\circ = \sum_{Pr \in Products} \Delta G_\mathrm{f}^\circ(Pr) - \sum_{Rc \in Reactants} \Delta G_\mathrm{f}^\circ(Rc) \tag{1.11}$$

In practice, in order to determine experimentally observed reversibilities of reactions, depending on concentration in metabolites external to the system, temperature, ionic force, and pH, the free energies of formation $\Delta G_\mathrm{f}^\circ$ of every metabolite can be used. They are reported in websites such as Equilibrator [19], which offers a programmatic interface [20] to compute ΔG° and K_{eq} values ourselves.

It should also be noted that inter-compartment transport reactions and redox processes have their own energy mechanisms that should be described further than simply ΔG free energy. Ion gradients are at the basis of many transport processes including oxidative phosphorylation in mitochondria [S2].

To conclude, since processes at equilibrium can only occur at an infinitesimal rate, a reaction will either proceed in its forwards or backwards direction, depending on metabolite concentrations. Thermodynamics can be used to help determining reaction reversibilities in general cases [13].

1.3.3 Michaelis-Menten's model

In light of enzyme kinetics and thermodynamics, enzymes are catalysts that impact reaction rates, but without affecting distance of the reaction to an equilibrium state. This is usually understood in terms of enzymes decreasing the activation energy of the reactions they catalyze. This is referred to as the transition-state theory: a lower activation energy means a faster reaction rate [14, 13, 21].

Let us take a simple irreversible reaction of rate k transforming a single substrate into a product:

$$S \xrightarrow{k} P \tag{1.12}$$

The enzymatic reaction describing the mechanism of enzyme catalysis of reaction equation 1.12 might be written:

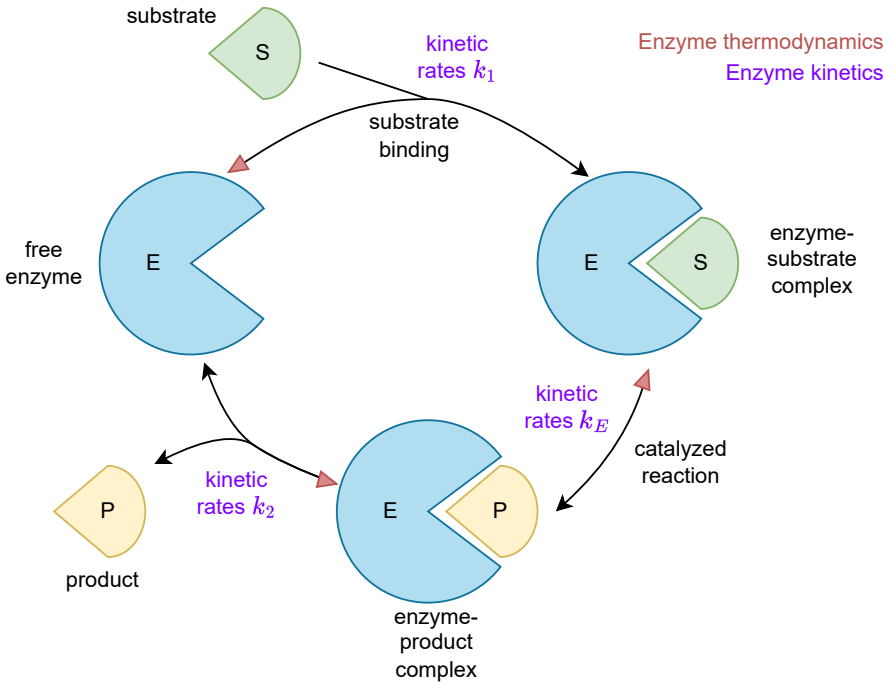


Figure 1.1: Schematic view of an enzymatic reaction with a single active site, with E representing the enzyme, S the substrate, P the product. Enzyme kinetics is the field of studying kinetic rates of reactions (in *violet*). Enzyme thermodynamics is the field of studying directionalities of reactions (*in dark red*).

$$E + S \rightleftharpoons ES \rightleftharpoons EP \rightleftharpoons E + P \tag{1.13}$$

where E, S, and P respectively represent the enzyme, substrate, and product; ES and EP are transition complexes of the enzyme with the substrate and with the product. This information is more visually summarized in Figure 1.1. We redefine the previous equation 1.13 by attributing reaction rates k_i to each direction in equation 1.15:

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \underset{k_{-E}}{\overset{k_E}{\rightleftharpoons}} EP \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} E + P \tag{1.14}$$

Note that this is a simplification of enzyme catalysis involving one active site where both substrates and products would bind, in other words the enzyme could only have a single ligand at a time. Michealis-Menten's model, which is described in this chapter, follows that assumption too, and thus can theoritically only model a single enzymatic reaction with one substrate and one product at a time.

Michaelis-Menten's model provides two abstractions to this model, first the transition reaction $ES \rightleftharpoons EP$, admittedly the moment where the actual reaction $S \overset{k}{\rightarrow} P$ occurs, is rendered 'spontaneous' and abstracted into a single metabolite EX. This is arguably due to difficulty observing this state *in vitro*, compared to simple measurements of substrate concentration $[S]$ and product concentration $[P]$. The model becomes:

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} EX \underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} E + P \tag{1.15}$$

Then, the second abstraction is rendering the overall reaction irreversible, by supposing the reaction $EX \rightarrow E+P$ to be thermodynamically favorable, and its inverse backwards reaction $E + P \rightarrow EX$ thermodynamically unfavorable.

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_2} E + P \tag{1.16}$$

Michaelis-Menten originally assumed the first reaction step $E + S \rightleftharpoons ES$ reached equilibrium fast enough to be represented by a single thermodynamic equilibrium constant $K_s = [E][S]/[ES]$, with k_2 being the limiting rate, however this is not the case when in limiting concentrations of substrate $[S]$. Briggs and Haldane introduced the kinetic rates k_1 and k_{-1}, leading to the following formulation:

$$\frac{d[ES]}{dt} = k_1([E]_0 - [ES])[S] - k_{-1}[ES] - k_2[ES] \tag{1.17}$$

Where equation 1.17 represents evolution of the enzymatic complex ES (or EX) concentration over time. Eventually, this quantity is said to reach a steady-state and at a fast rate, so for simplicity this evolution over time is assumed to already be at steady-state [14, 17]. This is also Briggs and Haldane's contribution.

$$\frac{d[ES]}{dt} = 0 \Leftrightarrow k_1([E]_0 - [ES])[S] = k_{-1}[ES] + k_2[ES] \tag{1.18}$$

The steady-state assumption stipulates that the initial rate of reaction reflects a steady state in which $[ES]$ is constant over time, that is, the rate of formation of the ES complex is equal to the rate of its breakdown [21]. From equation 1.18, we can derive the value of concentration $[ES]$ of the complex ES, with as parameters the concentration in substrate $[S]$, the initial enzyme concentration in the medium $[E]_0$, and the kinetic rates.

$$[ES] = \frac{k_1[E]_0[S]}{k_1[S] + k_{-1} + k_2} = \frac{[E]_0[S]}{[S] + (k_{-1} + k_2)/k_1} = \frac{[E]_0[S]}{K_m + [S]} \tag{1.19}$$

Where K_m in equation 1.19 designates the Michaelis constant:

$$K_m = \frac{k_{-1} + k_2}{k_1} \tag{1.20}$$

Now, let us defines the velocities of interest V_0 and V_{max}. V_0 is the velocity of the reaction $ES \xrightarrow{k_2} E + P$, thus $V_0 = k_2[ES]$. V_{max} is the the maximum velocity value taken by V_0, which would only be reached at a point where $[ES] = [E]_0$. The initial enzyme concentration $[E]_0$ (also called $[E]_{tot}$) represents the total enzyme concentration, bound with substrate or not, so $[ES] = [E]_0$ would mean all enzymes are bound to substrates, hence reaching a

theoritical maximum velocity value when all of them are catalyzing the reaction. Thus $V_{max} = k_2[E]_0$. Multiplying equation 1.19 by k_2, Michaelis-Menten's model can thus be summarized to this single equation:

$$V_0 = \frac{V_{max}[S]}{K_m + [S]} \tag{1.21}$$

V_0 is also called the initial velocity. At the beginning of the experiment, as $[S]$ is provided in low concentrations, and slowly consumed by the reaction, the velocity V_0 is linearly dependant on $[S]$, with approximate value $V_0 \approx V_{max}[S]/K_m$. On the other end, at the end of the experiment, the K_m term becomes insignificant and the velocity is approximately $V_0 \approx V_{max}$. Around the middle of the experiment, $[S] \approx K_m$ and $V_0 \approx V_{max}/2$. The well-known curve of evolution of initial velocity V_0 in function of concentration $[S]$ is presented in Figure 1.2.

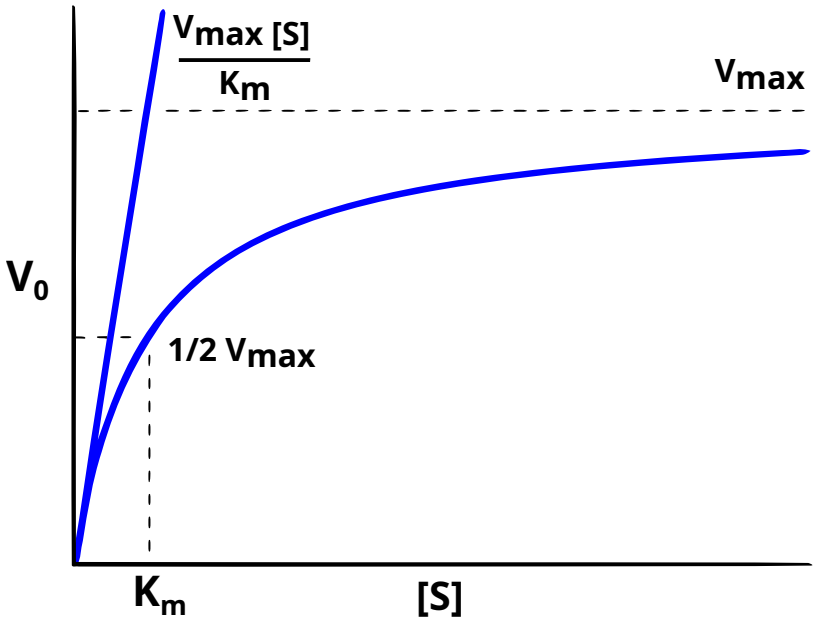


Figure 1.2: Dependence of initial velocity on substrate concentration in Michaelis-Menten's model. The graph shows the relations between the value of $[S]$ and kinetic parameters, defining the limits of the curve [21].

For each biological enzyme, there is a need to define what is called the turnover rate, or limiting rate of any enzyme-catalyzed reaction at saturation. If a reaction has several steps with one rate constant being limiting, then the turnover rate k_{cat} is that limiting rate. In more complex enzymatic models, the abstractions presented in equation 1.15 and equation 1.16. might not apply, and thus k_{cat} might be a function of several parameters. But in the simple case of standard Michaelis-Menten's models as described, $k_{cat} = k_2$, and also $k_{cat} = V_{max}/[E]_0$.

Michaelis-Menten's model, while adequate and simple to use, applies to a single substrate and product. Thankfully, there is no issue dealing with multiple ligands *i.e.* multisubstrate reactions, which is the case of many enzymes. It is possible to derive a Michaelis-Menten equation taking into account the multiple substrates and their kinetic parameters (see chapter 8. "Reactions of More than One Substrate" of [17]).

However, Michaelis-Menten's model is unable to take into account regulation by the substrate of the enzyme catalysis mechanism, also known as allosteric regulation, used by regulatory enzymes (see chapter 12. "Regulation of Enzyme Activity" of [17]). In that case, higher order models, described by the Hill equation, should be used. The higher order models are a lot more complex to use mathematically.

A last important measure of Michaelis-Menten model kinetics is enzyme specificity [S3]. Let us consider an enzyme E with two competing substrates, *e.g.* $S \xrightarrow{E} P$ and $S' \xrightarrow{E} P'$. This is not to be confused with a bisubstrate reaction $S_1 + S_2 \xrightarrow{E} P$. Note that to avoid the case of bisubstrate reactions experimentally, and always compute specificity towards a single metabolite, the other substrate of bisubstrate reactions is usually always provided in saturating quantity, while the metabolite of interest is limiting [17].

For a same enzyme E catalyzing reactions with two competing substrates S_1 and S'_1, the specificity is defined as :

$$\begin{aligned} \text{specificity to } S\text{:} \quad & k_{spec} = k_{cat}/K_m \\ \text{specificity to } S'\text{:} \quad & k'_{spec} = k'_{cat}/K'_m \\ E \text{ more specific to } S \text{ than } S'\text{:} \quad & \frac{v}{v'} > 1 \Leftrightarrow k_{spec} > k'_{spec} \end{aligned} \tag{1.22}$$

When dealing with kinetics of more than one enzyme and one substrate, computer modelling becomes more appropriate than manual calculations. The field of systems biology, aiming to describe biology and in particular metabolism with methods from the engineering of systems, offer computational solutions to help modelling.

A large field of study in systems biology and computational biology is drug discovery: searching for inhibitory metabolites to essential enzymes in virulent pathogens and diseases of interest [22]. Yet, non-productive substrate binding such as described in equation 1.23 cannot strictly follow the standard Michaelis-Menten's model [14]:

$$E + I \rightleftharpoons EI \quad (EI \text{ cannot be catalyzed into } EP \text{ and dissociated into } E + P) \tag{1.23}$$

Inhibitors that do not yield products can be incorporated in an extended Michaelis-Menten model if their existence is known and their corresponding kinetic constants such as the inhibition constant K_i can be derived. Inhibition mechanisms are classified in many multiple types (see chapters 6 and 7 of [17] and chapter 8 of [14] for details).

While essential enzymes can be targeted at the gene level, and RNA level with RNA interference techniques, a more common method used for human therapeutic applications is finding inhibitory metabolites that may be ligands to the enzyme of interest. A good inhibitory metabolite acting as a drug to the enzyme of interest may need to have good ligand specificity to the enzyme, and a significant reduction of the original kinetic rate should be observed in presence of the inhibitor. Most inhibitors usually trap the enzyme into one or more catalytically inactive forms, allowing a reduction in the concentration of the active reaction catalyst [14].

In systems biology analyzes, kinetic parameters k_{cat}, K_m, V_{max} – and K_i for inhibitors – are always sought for, and are known to be difficult to measure. They are usually reported in kinetic parameters databases [23, 24].

1.4 Microbial growth

Bacteria and unicellular organisms such as yeasts can be cultivated in experimental conditions. These organisms can be used and altered for many purposes, including testing for antimicrobial resistance and engineering of new bioproducts. Their use in bioengineering is often referred by the term "cell factories". In this section, we take a closer look at the mathematical models describing microbial growth.

1.4.1 Growth phases and growth rates

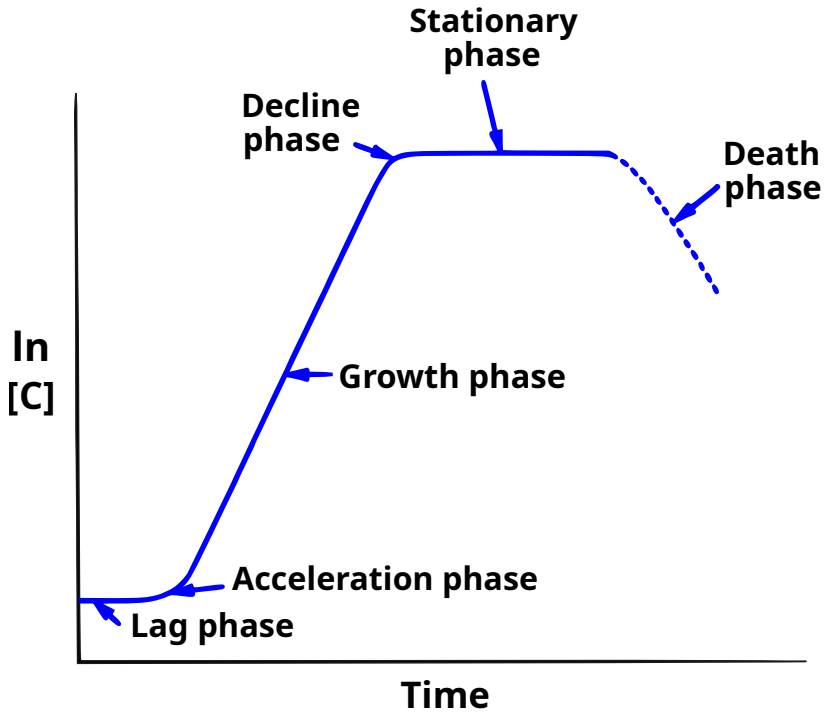


Figure 1.3: Typical batch culture microbial growth curve after adding substrate. $[C]$ denotes cell concentration [16].

In Figure 1.3, we describe the standard microbial growth phases as described in common organisms, such as *E. coli* or *S. cerevisiae*. Lab experiments are able to retrieve this measurements graph, either by measuring dry weight or using colorimetry, as mentioned previously. The phase of most interest when studying how fast a strain is growing is the *exponential growth phase*. In particular, the *specific* growth rate μ of a bacteria is defined, in h^{-1}, by:

$$\mu \text{ such that } \frac{d[C]}{dt} = \mu[C] \tag{1.24}$$

Note that the *growth phase* is one where the growth rate nears its theoritical maximum, *i.e.* $\mu \approx \mu_{max}$ [16].

1.4.2 Bioreactors and mass-balance of biochemical processes

Biochemists define biochemical processes in a broad sense which includes both enzymatic reactions and microbial colony growth that is designed to perform that reaction. Bacteria and yeasts are now grown in bioreactors, a broad term which, according to IUPAC, includes all "apparati used to carry out any kind of bioprocess; examples include fermenter or enzyme reactor" [25]. Fermenters are used to grow products from bacteria and yeasts at large-scale.

Bioreactors might operate in *open* or *closed* systems. A system is said to be closed if its boundary does not allow mass to pass to the surroundings. It is said to be open if it is able to exchange mass with its surroundings. To study a dynamical chemical *process* such as bacterial growth, one would have to open the bioreactor to add a substrate to the system, resulting in generation of more bacterial mass, *i.e.* biomass [16].

In open systems, where mass flows in and out, and undergoes production and consumption by chemical reactions, chemical species are subject to mass conservation laws, which can be summarized by the following equation:

$$\text{mass accumulated} = \text{mass in} - \text{mass out} + \text{mass generated} - \text{mass consumed} \tag{1.25}$$

In the case of open systems at steady-state, no mass is accumulated over time, thus we have the following equation:

$$\text{mass in} + \text{mass generated} = \text{mass out} + \text{mass consumed} \tag{1.26}$$

Open system at steady-state here refers to a system where mass is unchanging with time, meaning that mass flowing in the system is equal to mass flowing out of the system for the whole duration of the process.

An example of such a process would be for bacterial growth on glucose, despite glucose still flowing in the system, the number of biomass generated and consumed has become constant over time. More concisely:

$$\text{mass in} = \text{mass out} \tag{1.27}$$

Considering the system as a whole, detail of continuous mass generation and consumption by reactions would be abstracted, thus equation 1.27 can apply. However, one might be also interested into the behaviour of the system at unsteady-state, to know which amount of mass is generated and consumed by reactions over time.

Mass balance is useful for checking conservation of mass into biochemical pathways from substrate utilization and product output of lab experiments. In addition, the same thermodynamical background and mathematical concepts used here can be applied to energy balance [16]. In particular, species charge (affected by pH, oxidoreduction, etc.), species phase changes, temperature, all might affect energy of a system. Common biological processes such as glycolysis and respiration all imply consideration of bioenergetics. Thus it is important to also balance electrons when performing mass balance of reactions or pathways. Energy balance can be used to help determine ATP and energy requirements for the growth of microorganisms [16].

1.4.3 Monod's model

A chemostat is defined as "a bioreactor in which constant growth conditions for microorsanisms are maintained over prolonged periods of time by supplying the reactor with a continuous input of nutrients and continuous removal of medium." [25]. A chemostat is an example of bioreactor performing *continuous cell culture*, and which can define an open system as steady-state.

In microbial growth experiments, typically the specific growth rate μ depends on the concentration $[S]$ of a limiting substrate S. We can assume that a low amount of substrate gives a low growth rate and that if the substrate concentration increases the growth rate increases. For sufficiently high substrate quantites, the growth rate becomes saturated, similarly to enzyme saturation by substrate described in Figure 1.2. Thus, Jacques Monod showed that the biochemical process of microbial growth on substrates could be described similarly to Michaelis-Menten kinetics:

$$\mu = \frac{\mu_{max}[S]}{K_s + [S]} \tag{1.28}$$

Supposedly, Monod's model should be applied to chemostat, as typically this type of continuous cell culture should eventually reach a biochemical steady-state. Indeed, formation of new biomass is balanced by loss of cells inside the bioreactor [26]. Monod's model can also be applied to non-continuous cultures, *e.g.* batch cell culture, as described in Figure 1.3 [26, 16]. The stationary phase of batch cell growth, where $\mu = 0$, should not be confused with a steady-state. Batch cell cultures typically do not allow for the existence of steady states [26, 16].

1.4.4 Stoichiometry of cell growth

It is well-known that aerobic respiration on glucose is said to respect the following mass-balance equation [7] :

$$C_6H_{12}O_6 + 6\ O_2 \rightarrow 6\ CO_2 + 6\ H_2O \tag{1.29}$$

Then, let us represent biomass, *i.e.* cell growth on a carbon source such as glucose and a nitrogen source, whether in aerobic or anaerobic conditions, by the following equation:

$$\text{carbon source} + \text{oxygen} + \text{nitrogen source} \rightarrow \text{biomass} + \text{carbon dioxide} + \text{water} \tag{1.30}$$

Doran thus provides the following master equation for stoichiometry of cell growth, using the atoms carbon, hydrogen, oxyen, and nitrogen, the main sources of biological matter [16]:

$$C_wH_xO_yN_z + a\ O_2 + b\ H_gO_hN_i \rightarrow c\ CH_\alpha O_\beta N_\gamma + d\ CO_2 + e\ H_2O \tag{1.31}$$

Where the order of operands in equation 1.31 respectively matches the order of operands in equation 1.30.

To retrieve the stoichiometric coefficients of equation 1.31, one can solve the system of mass-balance equations presented in equation 1.32, provided that the respiration quotient d/a is known from experimental data. Note that coefficients w, x, y, z and g, h, i are known, and α, β, γ can be retrieved from elemental compositions of microbial cells dry weight in literature. For instance, *E. coli*'s elemental composition[1] is $CH_{1.75}O_{0.43}N_{0.22}$. [16].

$$\begin{aligned} &\text{C balance: } w = c + d \\ &\text{H balance: } x + bg = c\alpha + 2e \\ &\text{O balance: } y + 2a + bh = c\beta + 2d + e \\ &\text{N balance: } z + bi = c\gamma \end{aligned} \tag{1.32}$$

In such cell growth, the carbon source is considered the substrate of the reaction (nitrogen source being secondary). The yield of biomass production, or biomass yield, Y_{XS}, is defined as :

$$Y_{XS} = \frac{\text{g cells produced}}{\text{g substrate consumed}} \tag{1.33}$$

In strain engineering applications, *i.e.* industrial bioreactor fabrication of bioproduct, products are added to the master equation 1.30. A product yield Y_{PS} can then be computed: the mass of product over the mass of substrate, in addition to Y_{XS}. In this classical notation, X denotes biomass, P denotes product, and S denotes substrate.

1.4.5 Measuring reaction fluxes at steady-state

Contributions in the field of microbial growth mathematical modelling eventually led to the development of flux balance analysis (subsection 1.6.7, section 2.6), a method relying on mass-balance at steady-state to study whether biomass growth through a biomass synthesis reaction is possible [27]. This is also commonly referred to as metabolic fluxes analysis by bioengineers, who work at integrating experimental flux data into metabolic flux models.

Note that in experimental conditions, not all substrate goes into biomass, thus the master equation cannot be used to represent processes other than growth. In particular, substrate utilization that is unrelated to growth is often said to go into processes called maintenance [16]. In metabolic models, inclusion of maintenance into the biomass reaction is often questioned, for instance with ATP maintenance [28].

The stoichiometry of cell growth is used to make the biomass synthesis reaction of metabolic models. More complex reaction models of cell growth typically include utilization of coenzymes: ATP hydrolysis and NADH reduction. They might also include DNA, RNA, and protein macromolecular resource considerations for growth [28].

The idea of metabolic fluxes analysis is to study fluxes going through the cell during its growth. Reaction fluxes, fluxes of metabolite consumption or production, may be reported in units of $\mathrm{mol}\cdot(\mathrm{g\ dry\ weight\ of\ cells})^{-1}\cdot\mathrm{h}^{-1}$ or, alternately, as $\mathrm{C\text{-}mol}\cdot(\mathrm{g\ biomass})^{-1}\cdot\mathrm{h}^{-1}$ or $\mathrm{C\text{-}mol}\cdot(\mathrm{C\text{-}mol\ biomass})^{-1}\cdot\mathrm{h}^{-1}$. One $\mathrm{C\text{-}mol}$ of biomass is represented by the chemical 'formula' for dry cells normalized to 1 atom of carbon, $CH_\alpha O_\beta N_\gamma$, as described in equation 1.31 [16].

[1] Also of note when performing mass-balance for organisms such as *E. coli* are degrees of reductions, used in electron balances [16].

For instance, in modern metabolic models, one would find that flux data should be normalized for 1 gram of dry weight of biomass, and with biomass reaction flux actually attempting to predict experimental growth rate in h^{-1}. No longer substrate-dependent, the newest large-scale biomass reactions might look like the following:

$$\text{protein} + \text{RNA} + \text{DNA} + \text{other molecular composition} \rightarrow \text{1g dry biomass} \tag{1.34}$$

Although steady-state flux models have now been extended to whole genomes and human cells, they rely on several large assumptions: first, that a cell produces and optimizes biomass, as in microbial growth described here; second, that all data and results are in steady-state conditions. For example, experimental data of microbial cultures used to constrain the analysis could be measured in continuous chemostat bioreactors, as these guarantee steady-state conditions [16]. In practice, chemostats make for very good fermenters [26]. The successful industrial production of compounds combined with metabolic fluxes analysis wasn't possible until the rise of high-throughput technologies.

1.5 From classical genetics to high-throughput technologies

In 1865, Mendel first described on species of peas the notion of *dominant* and *recessive* characteristics, that is, characteristics that might be either preferentially conserved, or preferentially lost in a hybrid [S4]. With his work, Mendel described what would later become known as alleles, genotypes and phenotypes. A genotype, set of all alleles – versions of genes of an individual – is responsible for the phenotype – a set of all displayed and observable characteristics of the individual [S5].

Mendel's legacy is still carried today, including through the concept of mendelian diseases in human [S6]. Today, the entire coding human genome is sequenced [S7, S8], thus diagnostic of human diseases-related alleles can easily be done by mapping a sequence of to a genome of reference, or reference genome. This makes heavy use of high-throughput sequencing technologies and computational biology. Nucleotide polymorphisms, in other words, natural deviation in sequence contents, as well as chrosomal structural variants, are now used as *alleles* of human genes [S9], to predict resulting human phenotypes *e.g.* genetic conditions, congenital conditions.

1.5.1 Genotypes and phenotypes

From classical genetics, a genotype is defined as the set of alleles of genes an individual has, while a phenotype is defined as the observable characteristics the individual porting this alleles display. An allele being a version of a gene. For instance, as genes are defined with the DNA sequence code, any letter mutation or codon mutation might result in the creation of a new "allele". The average rate of errors in DNA polymerases is estimated to 1 in 10^5 [S10].

The two notions of genotype and phenotype are related, and are meant to be observed together for two or more individuals of the same species [29]. Quanitification of genes and phenotypes has evolved a lot over the years, and so have biotechnologies for characterization and storage of biological data, such as genome sequences of human patients. The speed of development of new biotechnologies is actually thought to be rising faster than the speed of standard information technology, as usually described by Moore's law [S11, S12].

Today almost all organisms of interest have reference sequences, meaning sequencing does not have to be done 'de novo'. Fast sequencing can now be done by mapping short or long reads, the output of high-throughput sequencers, onto a reference genome [S13, S14]. In human, mapping to a reference genome might be for example done for paternity tests and prediction of illnesses. Thus deriving genotypes is becoming less and less of a problem.

Deriving phenotypes can be more complicated though. If the phenotype is not something simple such as 'healthy' and 'ill', or 'alive' and 'dead', then one might be interested into what specific protein or metabolite is produced by the studied organism. For example, when a protein of interest is known beforehand, one might use specific antibodies to detect its presence or absence in cells [S15].

However, while high-throughput technologies for deriving genotypes are efficient, high-throughput technologies for detection of proteins and metabolites on a large-scale are still lacking accuracy even today [S16].

1.5.2 High-throughput technologies and omics

High-throughput biotechnological methods are defined as "methods that perform thousands of simultaneous measurements of biological molecules", and are well-known for generating great amounts of data [S17]. As a result of the rise of new biotechnologies, it is now believed biology databases and medical databases constitute the most massive amount of data that computer scientists and data scientists have to deal with, surpassing other science fields [S18, S11].

Omics: including genomics, transcriptomics, proteomics, are defined as the study with high-throughput technologies of biological data; with genomes, transcriptomes, proteomes being the sets of all data relating to the corresponding category, genes, transcription, proteins [30, 31]. While high-throughput biotechnologies permit spectacular characterization of cells, they of course do so with a loss in precision and are always in need of improvements [S16, S19].

Genomics, defining genomes, the set of all coding genes of an organism, can be obtained today by short-read and long-read sequencing [S13, S14]. Transcriptomics, set of all messenger RNAs, can be obtained with RNA-Seq, however a downside of the method is that its main application necessitates comparison to another dataset, usually two conditions, one 'healthy' and one 'ill', giving only differential expression data [S20].

Proteomics, the set of all expressed proteins of an organism, is a field carried by mass spectrometry [S21, S22]: however the methods: MS-MS sequencing: tandem mass stoichiometry, LC-MS sequencing – mass stoichiometry coupled with liquid chromatography, etc. are subject to many possible imprecisions. In particular, absolute quantification of protein — or enzyme — levels, *i.e.* absolute by opposition to relative levels which can be obtained with differential analyses, is still a challenge today [S23, S22].

Metabolomics, the study of metabolites, can be done by mass spectrometry as well [S24, S25]. RMN has also been reported to be used, such as in [S24] and [31]. Exometabolomics is a term used when only the metabolites from the extracellular medium are seeked for. Indeed, these makes for easier experiments to calibrate, as the medium composition can be known by the experimenter [32]. By contrast, internal metabolomics are subject to more imprecisions.

Fluxomics, the study of metabolite fluxes, can be done by radioactive isotope labelling. The presence of an isotope element, *e.g.* C^{13}, makes tracking and following what becomes of an extracellular medium carbon source possible. Fluxomics can be correlated to internal metabolomics data done by mass spectrometry for better accuracy [33, 32].

As a summary, we devised a non-exhaustive table describing existing types of omics and some of their associated high-throughput technologies, presented in Table 1.1.

1.5.3 Representation of knowledge in biology

All the afflux of newly-identified biological data now has to be represented in online biological databases. Biological databases might be international web portals like NCBI (US national center) [W1], ExPASY (Swiss equivalent) [S26] and EBI (European equivalent) [S27], or self-maintained projects from labs all over the world (*e.g.* [S28, S29, S30]). For their knowledge representation method: these might be standard SQL relational databases, knowledge ontology databases, etc.

Gene Ontology repertories logic relationships on discoveries on genes and on proteins, for example information about the function of proteins, and classification of those functions in the enzymatic world [S31, S32]. Using queries on ontologies, one can derive logic links between genes and their associated proteins and conferred phenotypes. Another way to derive logic relationships between genotypes and phenotypes would be by using medical databases [S6], such as the one from the NCBI portal [W1], or well-curated organism-dependant databases [S33, S34].

Bioinformatics data, especially the ones obtained from high-throughput technologies are now massively repertoried in biological databases. Genome sequences, RNA transcript sequences, protein sequences, gene definitions, medical observations related to genes, protein structures, biological pathways, all of those are reported in databases.

In particular, several paradigms exist for the storing of genes and proteins. UniProt adopts a "one gene = one protein" policy, and stores together genes, and their protein products. UniProt consists of mainly manually reviewed and annotated proteins, making it a reference database for quality annotations of proteins [34]. However, UniProt's policy forgets the existence of messenger RNA splicing, leading to several protein products with completely different functions from the same DNA sequence, and the existence of non-coding genes. NCBI instead stores each gene, each transcript and each protein in its own entry [W1].

For metabolites, we recommend PubChem [S35] and CheBI [S36]. For metabolic pathway maps, we recommend KEGG [35]. And for complete metabolic networks of reactions and metabolites, aside from KEGG, the Rhea database [S37], the MetaNetX database [S38], the ModelSeed database [36], the BiGG database [37], the MetaCyc database [S39] would consistute a good non-exhaustive list.

Metabolic networks are distinguished from metabolic models, *i.e.* models ready for use in systems biology, often found in a XML specific specification called SBML [38]. Those models can be found on the BiGG database [37], and also on the BioModels database among others [39, 40]. Metabolic models are however absent from informative databases such as MetaCyc and KEGG. Note that BioModels contains a collection of systems biology models, also including, among differential equation models and others, signaling networks in the SBGN specification, another

XML-like format [41].

Protein structures as well, are reported on biological databases, such as PDB [42], for the complete crystallized structure, and InterPro [S40, S41], for protein domains. For protein structure, the method of choice is crystallography, which is a quite slow method to perform and thus not easily scalable to whole genomes. Thankfully, AlphaFold – an AI from Google, applied to Uniprot entries – now permits to have structure predictions, such as we can cover the whole genome and proteome of an organism, although these are not the real structures but predictions [43].

Other examples of databases include DrugBank, for repertoring commercially available drug ligands to enzymes [44]. Also, STRING, which repertories Protein-Protein Interactions (PPI) networks [S42]. PPI networks are very large networks descriptive of protein complexes, including enzymatic complexes, which are in part predicted through interactomics high-throughput methods [S43].

Classical genetics	*Omics-associated name*	*High-throughput technologies*
Genotype	*Genomics*: sequencing a whole genome	Short-read, long-read sequencing, alignment to a reference genome for detection of variants
	Exomics: sequencing only the exons in eukaryotes by reverse transcription of messenger RNA	Short-read, long-read sequencing, alignment to a reference genome for detection of variants
	Transcriptomics: counting the RNA expression of all genes in a cell	RNA-Seq, which uses short-read and long-read sequencing, and tools to determine RNA counts and variants
	Epigenomics: studying interactions of genome elements with proteins	ChIP–chip [S44]
	Metagenomics: sequencing whole genomes of several cell species, such as microbial communities	Short-read, long-read sequencing, reference genome mapping methods
Phenotype	*Proteomics*: quantifying a whole proteome	Mass spectrometry, MS-MS, LC-MS, isotope labelling
	Metabolomics: quantifying the set of metabolites displayed by a cell	Mass spectrometry, MS-MS, LC-MS, RMN
	Fluxomics: quantifying the fluxes of each reaction through the cell	Isotope labelling
	Interactomics: determining protein interactions on a proteome level, determine protein-protein-interaction networks	Co-Immunoprecipitation [S43]
	Phenomics: quantifying certain phenotypes of several cells *e.g.* of a microbial colony all at once	Bioreactor culture, ELISA test, Antibodies

Table 1.1: From classical genetics to high-throughput technologies

In conclusion, as new biotechnologies arised, biologists have been invited to move on beyond classical experiments such as classical genetics and to welcome bioinformatics and computational biology. As a result, many biological data have been listed online. However, as with any data stored in the information technology field, materials must be submitted to thorough quality control checks.

1.6 Systems biology

Systems biology is an idea introduced as long ago as the 1950s [45]. Briefly, it consists of conducting a systems-level analysis of biological processes. Mihajlo Mesarovic, often considered a father of systems biology, defines systems theory as "the theory of formal (mathematical) models of real life (or conceptual) systems", with two fundamental premises: *1)* "a theory of any real life phenomena (biological or otherwise) is always based on an image, termed a model", and *2)* "without introducing any constraints whatsoever the formal, invariant, aspects of that model can be represented as a mathematical relation. This relation will be termed a system" [46]. It follows that deductions on the formal model should be interpreted with their implications on real life processes.

Since biological systems have a high dimensional complexity, including micromolecules: metabolites; macromolecules: proteins, DNA, RNA; micromolecular processes: enzymatic reactions; and macromolecular processes: DNA replication, transcription of DNA into RNA, translation into proteins; it follows that one would benefit from using the formal theory used in the engineering of complex systems [47]. Often in systems biology one ends up defining multiple modelling levels: a genetic and a ribonucleic level, a proteic level; a metabolism level, with enzymatic reactions and metabolites. By understanding and combining modelling at these different levels, and using biological data to constrain and further specify the model, one can fully predict the effect of a genotype and its corresponding phenotype *in silico* [48].

Systems biology took off in the early 2000s, thanks to the rise of high-throughput biotechnologies and computational biology [49]. However, the field is not new, and has been around for a while, including, among others, the study of (regulatory) genetic Boolean circuits [50, 51, 52, 53]. In addition, engineering theory and mathematical models for explaining experimentally observed biological phenomena have been used for a long time, including models such as Michaelis-Menten's and Monod's. And indeed, reaction kinetic models are often of interest in the field [53, 54]. A key defining factor of systems biology might then be that the study is done at a systemic level, *i.e.* considering small to large-scale biological systems such as a cell, a cellular compartment, a multicellular organism, etc. [54].

Early examples of modelling of biosynthetic reaction networks, with dynamic modelling using Ordinary Differential Equation systems (ODEs), include the works of Rosen and Casti on "Metabolism-Repair" systems in the 1980s [55]. Another instance includes the famous *E. coli* substrate uptake shift, which could be observed by Varma and Palsson with the so-called flux balance analysis method in 1994: an example of steady-state modelling, which is of most relevance to this thesis [27, 51]. With systems biology, scientists are able to to reproduce phenotypes *in silico* from modulating genotypes on their model, providing traceable explanations to biological phenomena. When systems biology modelling methods focus primarily on metabolic reactions, one should speak of metabolic modelling [56].

Douglas Kell suggests that systems biology should be contrasted to classical molecular biology, which focuses on experimentally validating predefined hypotheses with a reductionist view [57]. Indeed, systems theory allows for the study of a complete system using a holistic approach, without complete knowledge of its functioning [57, 58]. From high-throughput data, models are able to generate many new scientific hypotheses, emergent properties of the biological system.

Jens Nielsen distinguishes top-down systems biology from bottom-up systems biology: the former is a data-driven process, where new biological information are extracted from the model and omics datasets, possibly automatically using online databases; while the latter is based on curated knowledge and experimentally-proven hypotheses, providing a model with gaps to be filled [59, 60]. Both approaches can naturally apply to the same model, *e.g.* in the case of genome-scale metabolic models [60, 61]. Intuitively, the two notions should be understood in terms of reverse-engineering (top-down) and engineering (bottom-up) of a biological system [62].

1.6.1 Scope of systems biology

Hiroaki Kitano defines four key areas of study in systems biology: the system's structure, its dynamics, its control mechanisms, and its ability to be engineered [49, 63]. These are general terms for every subdomain of systems biology, and their potential theoritical and industrial applications. For example, one could study the dynamics of a metabolic pathway, both computationally and experimentally, trying to study the kinetic parameters of enzymes.

Static metabolic networks can be reconstructed from information found in online databases, and the network's properties and structure can be further analyzed. The network's intrinsic control mechanisms could reveal which enzymes we would benefit the most from targeting *in vitro* and *in vivo*. And reverse-engineering the system would allow us to perform that genetic modification, to design new strains.

The notion of control is central to the engineering of systems [51, 64]. In his 1997 book: "Understanding the Control of Metabolism", David Fell summarizes the principles of Metabolic Control Analysis (MCA) [S45, S46], a method defining control coefficients, quantifying how much a metabolic flux varies, in a metabolic pathway [64]. This replaces the old experimental biologist concept of "rate-limiting reactions". The method is now often used by enzymologists in conjunction with experimentally-retrieved kinetic parameters, and ODEs systems modelling reaction kinetics.

In particular, the team of Hans Westerhoff has advocated for the adoption of this systems biology extension of enzyme kinetics studies [54, 65]. Their methods were applied on mitochondrial respiration [S47], to glycolysis of *Trypanosoma brucei* [S48, S49, S50], and to cancer metabolism [S51]. By determining which enzymes have the highest control coefficients, meaning the ones with most impact on metabolism, researchers are able to predict drug targets [S52]. Barbara Bakker applied this method to *Trypanosoma brucei*, proposing that enzymes with high control coefficients in the parasite but low control coefficients in the host are the best possible targets [S48, S53].

The rapid rise of systems biology in the 2000s has led some scientists to raise some criticisms, particularly towards its supposed holistic approach [66, 67, 57]. A thorough study of around 400 papers from 2011 shows that ODEs modelling dominated most systems biology studies from the 2000s [68]. Accordingly, the tool of choice from back then was MATLAB ©. The authors also reported that only a minority of the studies were reproducible [68]. The issue of reproducibility was tackled on at the time by the implementation of the SBML standard, and the construction of the BioModels library [69, 39], and it is still worked on today by the community of dynamic modellers [70].

In the 2010s onwards though, trends have changed in favor of FBA (flux balance analysis), *i.e.* steady-state modelling, stoichiometric modelling, genome-scale metabolic modelling. To get an idea of this new community bias, the 2010 paper: "What is flux balance analysis?" has now be cited no less than 3800 times [71]. Meanwhile, the 2013 COBRAPy paper will soon reach 1000 citations, indicating a shift towards the Python programming language [72]. Flux balance analysis is a true game changer in holistic systems biology, in that it can scale to whole genomes.

Genome-scale metabolic models are metabolic models computationally generated from a reference genome [28, 73]. The strength of genome-scale metabolic modelling is precisely that many different kinds of omics data can be integrated into models [54, 61]. Although, the models - and FBA - are based on bacterial and microbial growth as defined in section 1.4 and thus not theoritically applicable to human cells [74], bacteria and yeasts make for excellent organisms of study and cell factories [59]. We illustrate the links between systems biology, high-throughput technologies, omics data and computational software, and its applications to microbes in Figure 1.4 and Figure 1.6.

In addition to the previously mentioned therapeutic applications, systems biology is now at the genome-scale tackling areas of research as wide as microbial ecology, gut microbiota and metabolic engineering [59, 75, 76]. The latter, using metabolic modelling to help engineer new strains, is of particular interest to industries.

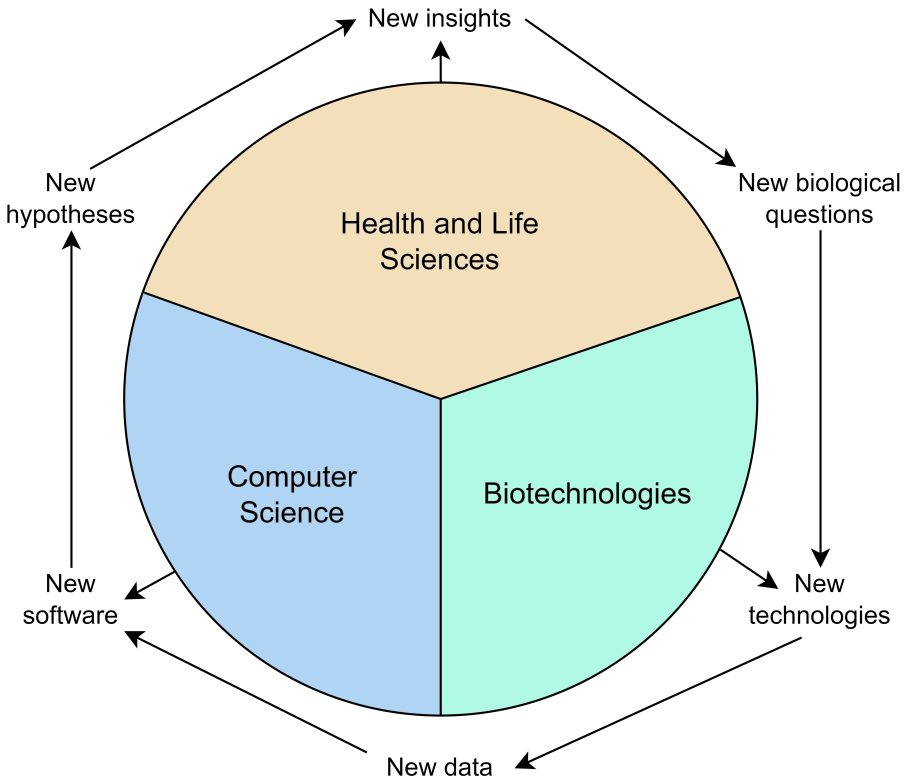


Figure 1.4: Trinity of systems biology: advances in health and life sciences, computer sciences and biotechnologies lead systems biology research

1.6.2 From systems biology to synthetic biology

The studies of Jacob and Monod on the *lac* operon and on diauxic growth of *E. coli* sparked a large interest among biologists in the design of synthetic biological circuits [53, 62, 77]. Efforts into making accurate models of gene regulation using circuit engineering – with Boolean circuits and logic gates – sparked in the years 2000s, and are still prevalent today [62, 51, 52].

MIT, one of the pioneers of synthetic biology, defined the field as : *(a)* "the design and fabrication of biological components and systems that do not already exist in the natural world" and *(b)* "the re-design and fabrication of existing biological systems" [W2]. Meanwhile, iGEM, the International Genetically Engineered Machine competition, defines synthetic biology as "an approach that uses engineering principles to design and build biological systems" [W3].

A huge prospect of synthetic biology is being able to engineer biology processes. For instance, protein engineering, enhancing a protein structure to have certain properties, can be achieved by gene modification, and expression of that gene by microbial cells [78, 79]. The field of metabolic engineering was defined – back in the 1990s – as the "manipulation of cellular enzymatic, regulatory and transport processes using recombinant-DNA technology for the purpose of enhancing specific [...] desirable properties" [80, 81].

Whether it is through circuit engineering, or metabolic engineering and strain optimization, it is fair to say the field of synthetic biology has always been linked to advances in systems biology. Learning information from metabolic fluxes analysis, or metabolic control analysis, allows one to design strain with deactivated enzymes for therapeutic applications, or for enhanced fermentation production, for instance. Or, alternately, kinetic modelling of enzymes obtained from DNA mutations allows us to determine which recombinant enzyme is the most efficient [78, 81].

In industry, bacteria and yeasts are often called cell factories for their ability to be engineered and manufactured on a large-scale [82, 83]. Recently, cell factories have been in the center of engineering of new bio-based chemical products [84], including innovative solutions such as biofuels, pharamaceutical compounds, cosmetics products [S54, S55]. These industrial applications made use of synthetic biology and metabolic engineering [85, 79, 83].

When aiming to construct new protein designs, and thus mutant strains for bioproduction, one can perform *directed* or *undirected* mutagenesis, acquiring new DNA mutations throughout genetic engineering or evolution of strains, eventually leading to possible new enzymes with the desired properties. Directed mutagenesis include, for example, CRISPR-Cas9. Meanwhile, an example of undirected mutagenesis process for bioproduct manufacturing that is industrialized on a large-scale is *adaptive laboratory evolution*. It is used in particular for consolidating fitness of manually-designed recombinant strains [S56, S57].

The classic engineering framework used in synthetic biology is called Design-Build-Test-Learn cycle. In the context of metabolic engineering, metabolic flux analysis methods, which serve as a prediction tool to check if our recombinant strain can display the desired phenotype, are said to fall into the Design part of the cycle [86, 83]. Adaptive laboratory evolution, as a mutagenesis process towards a desired fitness for the reconstructed mutant strain, would be part of the Build phase [S57]. As well, when information about the strain is learned back from the Learn phase, going back to the Design phase of the cycle, the metabolic model should be changed accordingly.

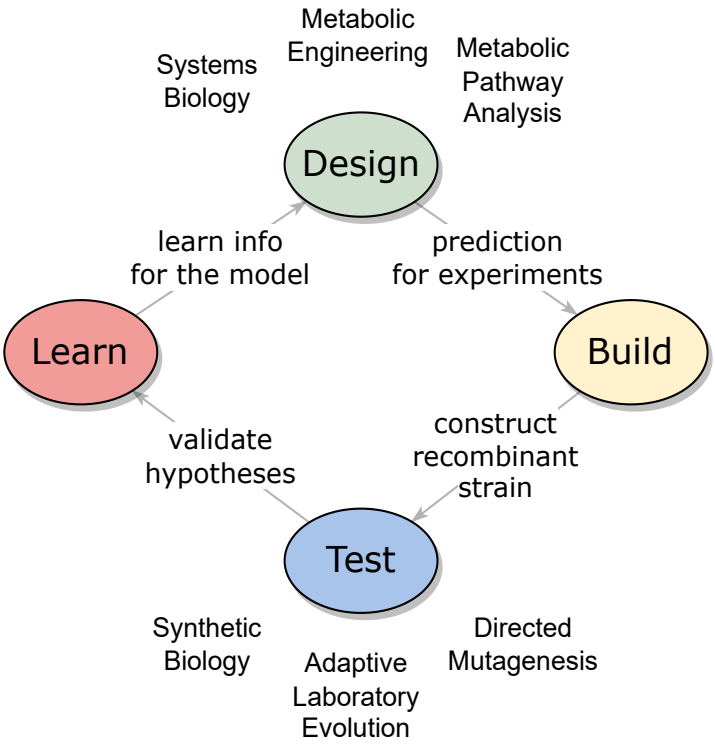


Figure 1.5: The Design-Build-Test-Learn cycle in metabolic engineering and synthetic biology

While one does not have to go as far as writing a complete Design-Build-Test-Learn cycle (Figure 1.5) when doing systems or synthetic biology, it is important to remember than the model can only work with knowledge about biological reality that would be provided by lab experiments, and thus they work in collaboration: the model brings new information for the experiments, and the experiments bring new information for the model. Since a model could predict anything, experimental verification is required, and reciprocally experiments are not error-free. The interdisciplinary systems biology field opens the door for a high technical complexity, both formal and experimental.

1.6.3 Overview of systems biology methods

Systems biology methods include any methods concerned with the modelling of a biological system. Systems biology methods are usually separated into two categories: *discrete* modelling methods and *continuous* modelling methods. The former includes Boolean modelling methods, Gillespie's SSA algorithm, and more, while the latter includes dynamic simulation by ODE systems, and steady-state modelling.

In particular, genetic regulatory systems have gotten a lot of attention. In a review paper, *Modeling and Simulation of Genetic Regulatory Systems: A Literature Review*, dating from 2000, Hidde de Jong characterizes at least eleven distinct modelling methods applied to genetic regulatory systems: subgraphs analysis in their graphs representation, Bayesian networks modelling, nonlinear ODEs, piecewise-linear differential equations, stochastic differential equations and stochastic simulation, partial differential equations, Boolean networks, generalized logic formalisms, qualitative differential equations, and rule-based simulation formalisms [87].

Partial differential equations might be used to represent a different variable than time for the evolution of chemical reactions, such as space, or diffusion. These are the so-called reaction-diffusion equations [87].

Generalized logic formalisms cover extension of Boolean formalisms from René Thomas to sets of integers [87], which in this thesis, we cover by the discussion of Constraint Satisfaction Problems (section 3.1). As well, I would like to note that rule-based simulation formalisms have common syntaxic elements with logic programming. Examples of rule-based simulation methods used today are: [S58, S59].

For the sake of completeness, we will introduce Boolean modelling and Gillespie's SSA algorithm, although they are oftentimes outside of the scope of metabolic modelling. The method of most relevance to this thesis is steady-state modelling. Its existence cannot be dissociated from dynamic modelling, or ODEs, since the formalism describes an ODE steady-state, and it compensates for ODEs inability to scale at the genome-scale level.

Definition 1.6.1 – Ordinary Differential Equations

An Ordinary Differential Equation, or ODE, is a differential equation involving ordinary derivatives, meaning derivatives dependant on a single variable.

Often, the goal is to solve an ODE: determining which set of functions satisfy the equation.

For example, for any $k \in \mathbb{R}$, the ODE:

$$\frac{dx(t)}{dt} = kx(t) \text{ or more simply } \frac{dx}{dt} = kx$$

admits as solutions the functions of the form $x(t) = Ce^{kt}$, where $C \in \mathbb{R}$ is a constant.

Systems of ODEs might be used to model evolution of dynamic systems over time.

Definition 1.6.2 – Boolean functions and Boolean formulas

A Boolean function is a function $f : \mathbb{B}^k \to \mathbb{B}$ over k Boolean variables $v \in \mathbb{B}^k$ with $\mathbb{B} = \{0, 1\}$, and giving the output of a Boolean variable $b \in \mathbb{B}$, such that $b = f(v)$.

A Boolean formula is a string of symbols which most often describes a Boolean function. It might be composed of logic operator symbols $\wedge$ (AND), $\vee$ (OR) and $\neg$ (NOT), and of names of Boolean variables.

Note that the Boolean formula describing a Boolean function might not always be known.

An additional symbol often used in biology is the implication: $x \Rightarrow z$ (Z IF X).

An example of logical equivalence (IFF, $\Leftrightarrow$) between Boolean formulas is the following: $(a \Rightarrow b) \Leftrightarrow (\neg a \vee b)$.

Additionally, the search for satisfying solutions s^* such that $f(s^*) = b^*$ might be of interest.

Boolean networks, systems of several Boolean functions, might be used to model dynamic systems over time.

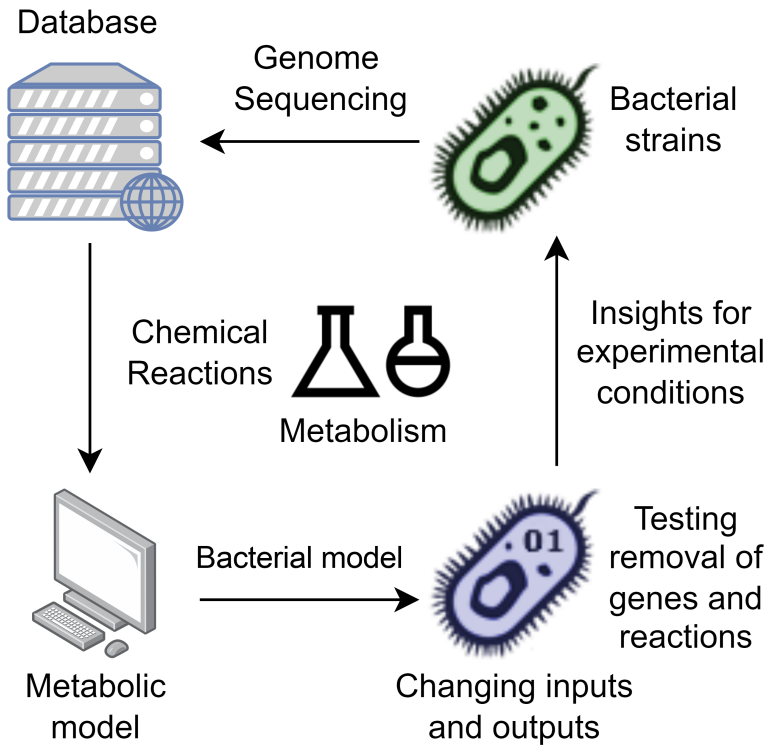


Figure 1.6: Classical systems biology workflow: example of metabolic modelling application to bacterial strains

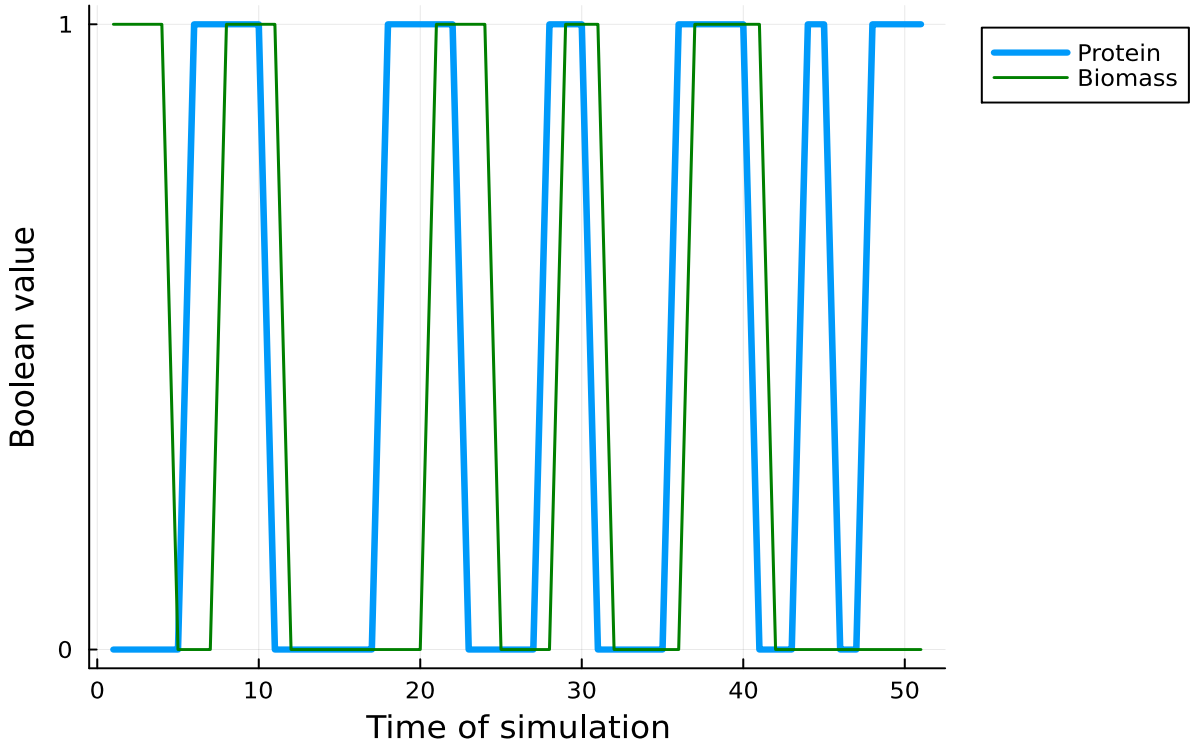


Figure 1.7: Boolean network asynchronous simulation of variables representing protein and biomass production

1.6.4 Boolean modelling methods

In systems biology, problems might be expressed through Boolean circuits [51, 52], or Boolean networks [88]. Boolean networks are appropriate for dynamic modelling of biological systems.

A Boolean network is defined as a pair $B = (N, F)$ where $N = \{v_1, \cdots, v_n\}$ is a finite set of nodes *ie.* variables representing biological agents and $F = \{f_1, \cdots, f_n\}$ is a set of Boolean functions $f_i : \mathbb{B}^k \to \mathbb{B}$, with $\mathbb{B} = \{0, 1\}$, describing evolution of states of v_i over time [88].

A vector or state $x(t) = (x_1, \cdots, x_n)$ describes the values of all nodes N at a time step t, where x_i represents the value of the node, either 1 or 0, which might also be written as $\{True\}$ or $\{False\}$, respectively [88].

An example of such a Boolean network would be the following:

$$\begin{aligned} \mathrm{Protein} &= \text{IF } \mathrm{mRNA} \text{ AND } \mathrm{TranslationPromoter} \text{ AND NOT } \mathrm{ProteinDegradation} \\ \mathrm{Biomass} &= \text{IF } \mathrm{Protein} \text{ AND } (\mathrm{WasteTransporter} \text{ OR } \mathrm{ProteinDegradation}) \\ \mathrm{ProteinDegradation} &= \text{IF NOT } \mathrm{ProteinDegradation} \end{aligned} \tag{1.35}$$

There are three described Boolean functions: for nodes $\mathrm{Protein}$, $\mathrm{Biomass}$ and $\mathrm{ProteinDegradation}$. These functions describes how the state $x(t+1)$ evolves compared to the state $x(t)$. Understandably, this means in our example, nodes mRNA and $\mathrm{TranslationPromoter}$ and $\mathrm{WasteTransporter}$ are static over time. This example is a way of expressing dynamics of protein production and degradation, another way will be as described in equation 1.38.

We can assume a *synchronous mode* of evolution, meaning all nodes update at the same time, or an *asynchronous mode* of evolution, where the first node updates first, then the second, then the third, etc., *ie.* where an order has to be defined. It goes without mentioning that Boolean networks, called Boolean circuits in synthetic biology, obey the rules of formal logic and logic resolution.

For our example, we assumed an *asynchronous mode*. At time $t = 0$, we set Boolean variables mRNA, $\mathrm{TranslationPromoter}$ and $\mathrm{WasteTransporter}$ to $\{True\}$, essentially simplifying the Boolean formulas for the other nodes. For the other nodes, we set initial values of $\mathrm{Biomass}$ and $\mathrm{ProteinDegradation}$ to 1, while $\mathrm{Protein}$ is at 0. At each time it is evaluated, $\mathrm{ProteinDegradation}$ oscillates between present and absent. We represented the evolution of $\mathrm{Protein}$ and $\mathrm{Biomass}$ in Figure 1.7 [W4]. Note that they are closely related: once $\mathrm{Protein}$ is evaluated to $\{True\}$ or $\{False\}$, the Boolean relationship between the two of them is so that $\mathrm{Biomass}$ eventually follows with that same truth value.

In particular, Boolean Networks are used to study dynamics of gene expression and protein expression [51, 88], and mechanisms such as transcriptional regulation, protein-level regulation, feedback loops, cell cycle, etc. [S60, S61]. An important object of study are also *attractors* of the Boolean dynamic systems [S60, S61]. We refer to the work of René Thomas for further details in the field of Boolean modelling methods in biology [50].

Note that logic modelling is important in biology as it conceptually matches with the intuitive reasoning from biologists [S62, S63]. Rather than discrete and continuous data, biologists would record data in qualitative ways, linked by Boolean relations to each other [S62, S64]. Examples of these include Gene Ontologies and other ontology

databases, which also notably include protein functions and metabolism functions [S65, S32, S66]; and signaling pathways describing activation and inhibition of proteins, which may be reported in BioModels in SBGN (Systems Biology Graphical Notation) format [39, 41].

Thus, it is important to incorporate Boolean data into models, even in fully discrete and continuous systems [51]. However, a too strict discrete Boolean encoding of transcriptional regulation, among other examples, might not be a reasonable hypothesis. This was directly touched upon in the studies I undertook for my thesis [89].

Nevertheless, the sigmoidal shape, observed in kinetic models, bacterial growth, gene regulation, and many other biological processes, is one that can be accurately reproduced by Boolean models, as a Boolean *ON*/*OFF* switch [S62, S67].

1.6.5 Dynamic modelling with ODEs

As well, dynamic modelling problems can be expressed by Ordinary Differential Equation systems (ODEs for short). Markus Covert proposes the following general template for modelling ODEs of concentrations of entities x [51], presented in equation 1.36:

$$\frac{dx}{dt} = \sum Rates_{production} - \sum Rates_{consumption} \tag{1.36}$$

Where the sum $\sum Rates_{production} - \sum Rates_{consumption}$ must be a function over time t, possibly depending on $x(t)$ or on other functions that are variables of the ODE system.

For instance, considering concentrations of three entities, DNA, $\mathrm{Protein}$ and $\mathrm{Biomass}$, here is an ODE system:

$$\begin{aligned} \frac{d[DNA]}{dt} &= -k_{trl}[DNA] \\ \frac{d[Protein]}{dt} &= k_{trl}[DNA] - k_{deg}[Protein] \\ \frac{d[Biomass]}{dt} &= -k_{deg}[Protein] \end{aligned} \tag{1.37}$$

This example (equation 1.37) was simulated and represented in Figure 1.9 using Julia [S68, S69].

For simplicity, we'll consider a system of a single equation and variable [51]. The rate constants used in the abstraction for protein production presented in equation 1.38 are k_{trl} rate of gene translation and k_{deg} rate of protein degradation.

$$\frac{d[Protein]}{dt} = k_{trl} - k_{deg}[Protein] \tag{1.38}$$

Covert then decides to further simplify the system by setting $k_{trl} = k_{deg} = 1$ in equation 1.39 :

$$\frac{d[Protein]}{dt} = 1 - [Protein] \tag{1.39}$$

The analytical solution to that simple ODE system, with initial conditions of $[Protein](t = 0) = 0$ is the following function over time:

$$[Protein](t) = 1 - e^{-t} \tag{1.40}$$

Here the case is simple enough that the solution to the ODE can be analytically retrieved. However, that is not always the case. A technique to approximate solutions to an ODE would be the Euler method. The idea is approximating the ODE system, with real steps of time Δt rather than infinitesimal steps, *i.e.* similar to computing tangents to the curve of the solution function.

The derivative notation in equation 1.38 can be rewritten with infinitesimal steps as the following:

$$[Protein](t + \Delta t) = [Protein](t) + (1 - [Protein](t)) \cdot \Delta t \tag{1.41}$$

An application of Euler method to equation 1.38, for $t = 0.01$, with a time step of $\Delta t = 0.01$ is presented in equation 1.42. We approximately retrieve the result that we would obtain by applying equation 1.40 to that time value of $t = 0.01$.

$$[Protein](t = 0.01) = [Protein](t = 0) + (1 - [Protein](t = 0)) \cdot 0.01 = 0.01 \approx 1 - e^{-0.01} \tag{1.42}$$

Of course, this is a simple example of a system of only one ODE and of first-order. In practice, biological systems models would incorporate complex kinetic parameters such as described in subsection 1.3.3, and are thus much harder to analytically solve, and approximate, if even possible. Further techniques to help integrating and simulating solutions to ODEs can be obtained thanks to Taylor series approximations [51].

ODE solvers and approximation methods are implemented in classical mathematical libraries such as MATLAB [W5], Julia [S69], SciPy [S70]. Alternately, one might approximate ODEs with Gillespie's stochastic simulation algorithm, though the parameters they operate on are different.

1.6.6 Gillespie's stochastic simulation algorithm

Gillespie's stochastic simulation algorithm (SSA) is another tool used in systems biology to approximate consumption of metabolites by reactions [S71, S72]. While at first glance it has similarities to ODEs, it takes a very different approach. It is based on the probablity of two molecules interacting at any given time, according to physics. Gillespie's SSA operates on discrete metabolite quantities rather than continuous concentrations [51].

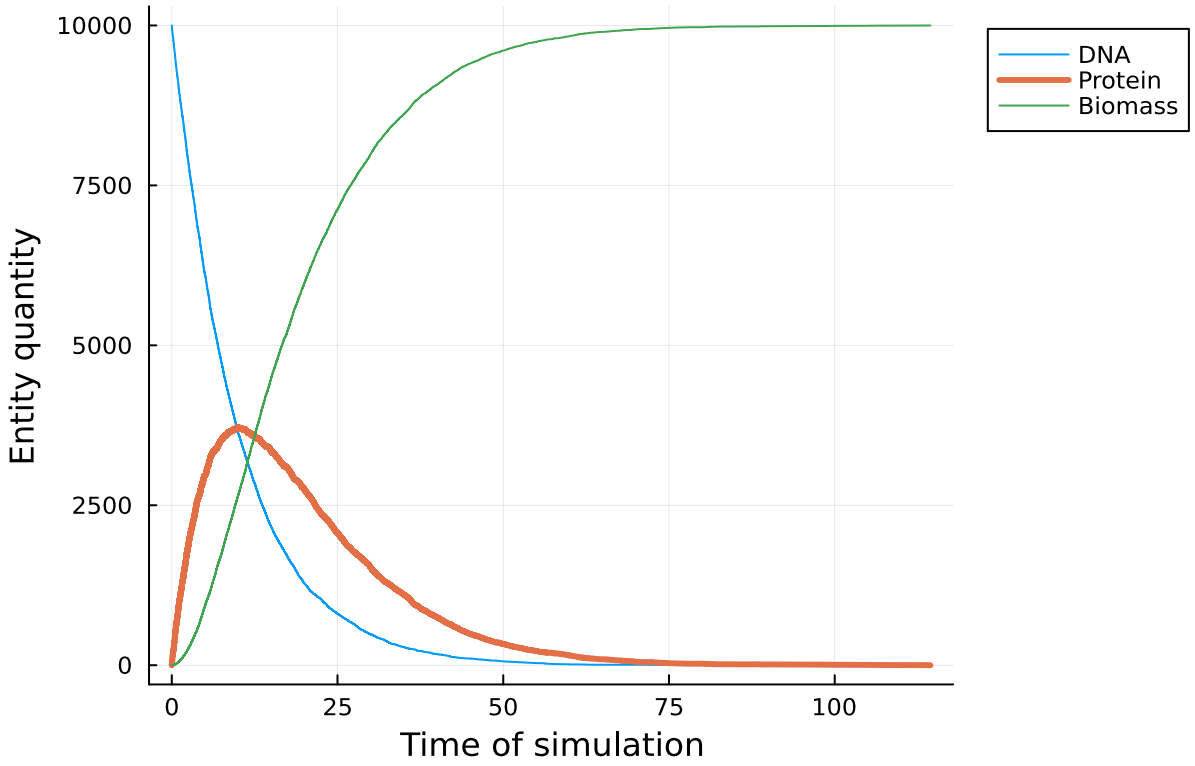


Figure 1.8: Gillespie SSA simulation of reactions representing evolution of DNA, Protein and Biomass variables

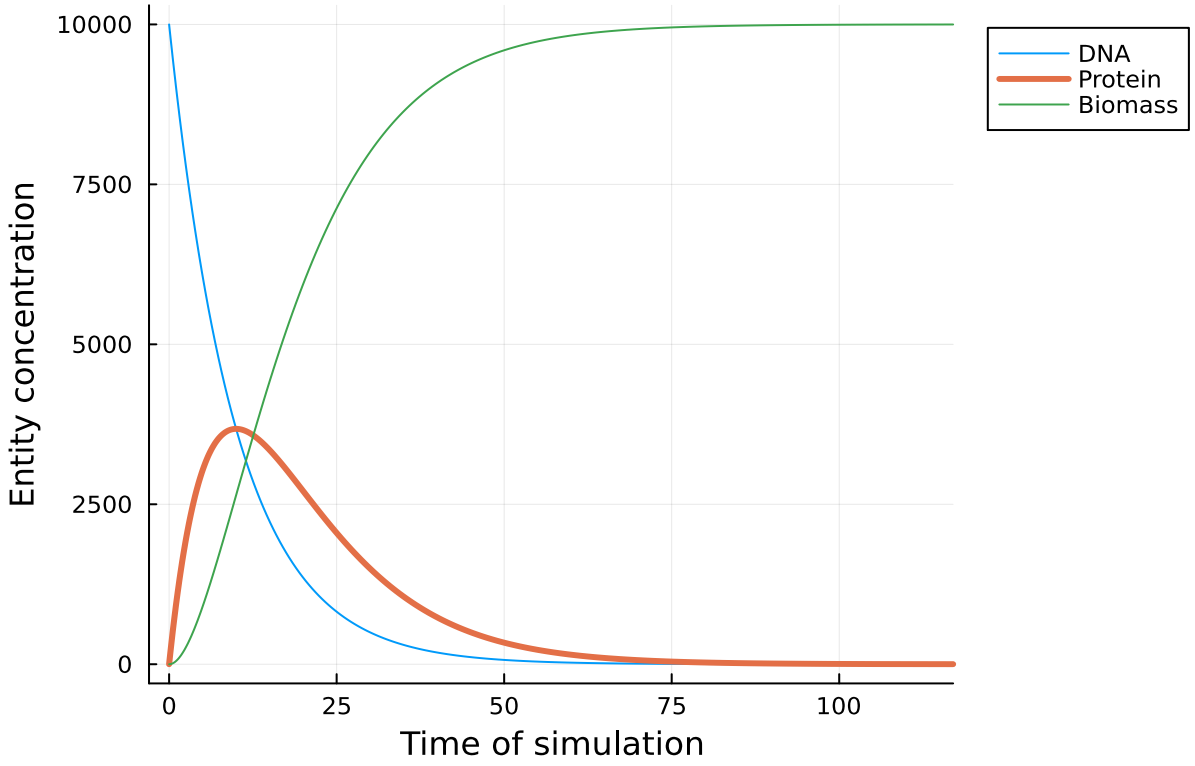


Figure 1.9: ODE simulation of reactions representing evolution of DNA, Protein and Biomass variables

For comparison's sake, we provide the same example as for ODEs (equation 1.37), which we simulated and represented in Figure 1.8 using Julia [W6].

Let us take the following system of reactions, characterized by their reactions constants c_j:

$$\begin{aligned} DNA &\xrightarrow{c_{trl}} Protein \\ Protein &\xrightarrow{c_{deg}} Biomass \end{aligned} \tag{1.43}$$

Similar to the initial condition in ODEs solving, Gillespie's simulation require definition of numbers of molecules for every molecule in the simulation. Rates of protein synthesis and degradation in organisms can be found reported on databases such as BioNumbers, which have many use cases in systems biology [S73].

In Gillespie's SSA, the reaction constants c_j are then used to compute reaction propensities a_j. Constants are not to be confused with kinetic rate constants, as they operate on number of molecules rather than volumic concentrations. The number of molecules x_i is independant of volume. The volume parameter is set elsewhere in the algorithm.

The first step in an iteration of Gillespie's SSA is computing the sum of all reaction propensities a_{total} :

$$a_{total} = \sum_{j=1}^{M} a_j(\boldsymbol{x}) = \sum_{j=1}^{N} c_j h_j(\boldsymbol{x}) \tag{1.44}$$

Where h_j is a function of the number of molecules vector $\boldsymbol{x}$ at time t, depending on reaction stoichiometry and the order of reaction j. Then the interval τ between the current time and the time of the next reaction is calculated, using a randomly generated number $U_1 \sim [0, 1]$.

$$\tau = \frac{1}{a_{total}} \ln\left(\frac{1}{U_1}\right) \tag{1.45}$$

This Monte-Carlo simulation method gives time steps τ into a familiar exponential probability distribution. Then the reaction q occuring at that time is chosen, according to a second randomly generated number $U_2 \sim [0, 1]$.

$$q \text{ is such that } \sum_{j=1}^{q-1} a_j \leq a_{total} \cdot U_2 < \sum_{j=1}^{q} a_j \tag{1.46}$$

Such an algorithm describes the probability that the next reaction q occurs during the next time interval τ, considering the number of molecules $\boldsymbol{x}$ at time t. This is expressed as:

$$P(\tau, q \mid \boldsymbol{x}, t) = a_q(\boldsymbol{x}) \ \exp(-\tau \sum_{j=1}^{M} a_j(\boldsymbol{x})) \tag{1.47}$$

Further insights into the stochastic simulation algorithm as presented by Gillespie are given in [S71, S72]. A great advantage of the SSA method is providing an actual integer number of molecules by time for each molecule, more

reflective of biological reality than concentration values in floating numbers. It also might provide more accurate insights regarding limiting metabolites in the solution, as the algorithm simply stops when one of the reactants quantity reaches 0, while an ODE system might tend towards zero indefinitely.

However, just like ODEs, SSA suffers scaling issues when the number of variables increases. Indeed, the time steps τ are inherently inversely dependant on the number of reactions and molecules in the system. Covert suggests compensating for these problems by making hybrid models [51]. Examples of hybrid models include: [90], [S74].

In conclusion, systems biology presents a vast panel of methods. The method with the most scalability which will be presented in this thesis is steady-state modelling. While steady-state makes the variation over time abstract, it could be coupled in hybrid frameworks with Boolean formalisms, ODEs solving, and stochastic simulation algorithm for further description of the evolution of the system over time.

1.6.7 Steady-state modelling

To conclude this introduction, we present a brief introduction to how dynamic modelling methods relate to steady-state modelling, and how systems biology modellers have come to consider steady-state modelling from a dynamic modelling background.

While Michaelis-Menten can be used to modelize enzyme kinetics in a variety of biological substrates, determining the kinetic parameters for even a single enzyme can be a lot of experimental work, so determining kinetic parameters for whole metabolic networks is still out of reach today.

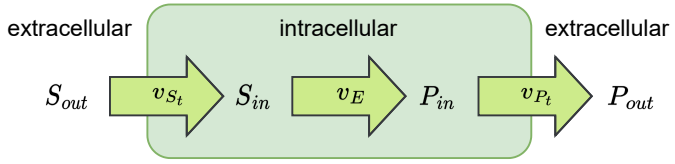


Figure 1.10: Abstraction of a metabolic model for illustrating steady-state modelling. Fig. proposed by Covert [51].

Let us consider a simple abstraction of a metabolic network and metabolic fluxes *i.e.* reaction rates, as presented in Figure 1.10. There are two compartments: extracellular and intracellular. The network possesses two enzymatic Michaelis-Menten transporter [2] reactions $\{S_t, P_t\}$, and one standard intracellular Michaelis-Menten enzymatic reaction E. It involves metabolites $Mets = \{S_{out}, S_{in}, P_{in}, P_{out}\}$. It is defined as the following:

$$\begin{aligned} S_{out} &\xrightarrow{v_{S_t}} S_{in} \\ S_{in} &\xrightarrow{v_E} P_{in} \\ P_{in} &\xrightarrow{v_{P_t}} P_{out} \end{aligned} \tag{1.48}$$

[2] Note that transport reactions might be non-enzymatic and that using Michaelis-Menten's model in that context, ignoring membranal ionic strength, is strongly discouraged. Handbooks of kinetics provide stronger formalisms for these non-standard enzymes [14, 17].

To model this metabolic network using ODEs, we rewrite it into a system of production and loss rates [51]:

$$\forall M \in Mets, \qquad \frac{d[M]}{dt} = \sum v_{production} - \sum v_{consumption} \tag{1.49}$$

Applying equation 1.49, we get the following system [51]:

$$\begin{aligned} \frac{d[S_{out}]}{dt} &= -v_{S_t} \\ \frac{d[S_{in}]}{dt} &= v_{S_t} - v_E \\ \frac{d[P_{in}]}{dt} &= v_E - v_{P_t} \\ \frac{d[P_{out}]}{dt} &= v_{P_t} \end{aligned} \tag{1.50}$$

Taking the second equation and applying Michaelis-Menten kinetics gives the parameter-dependant ODE equation below, showing that computing evolution of concentrations over time requires the knowledge of kinetic parameters :

$$\frac{d[S_{in}]}{dt} = \frac{V_{max(S_t)} \cdot [S_{out}]}{[S_{out}] + K_{m(S_t)}} - \frac{V_{max(E)} \cdot [S_{in}]}{[S_{in}] + K_{m(E)}} \tag{1.51}$$

Since metabolite concentrations and Michaelis-Menten kinetic parameters tend to be hard to determine, accurately analyzing such an ODE might be difficult [51]. Let us consider another approach.

The steady-state assumption, or steady-state modelling hypothesis, states: metabolites internal to the system are directly consumed and produced, without any time consideration. The variation in time of concentration is considered null. Here all metabolites including extracellular ones are internal. Setting a steady-state assumption gives:

$$\forall M \in Mets, \qquad \frac{d[M]}{dt} = 0 = \sum v_{production} - \sum v_{consumption} \tag{1.52}$$

Which in turn is rewritten as the following system:

$$\begin{aligned} -v_{S_t} &= 0 \\ v_{S_t} - v_E &= 0 \\ v_E - v_{P_t} &= 0 \\ v_{P_t} &= 0 \end{aligned} \tag{1.53}$$

This allows us an abstraction of Michaelis-Menten kinetic parameters and internal metabolite concentrations, meaning those do not have to be experimentally determined or computationally estimated.

These equations will need to be solved to get values of metabolite fluxes estimated over time *i.e.* reaction rates, not to be confused with metabolite quantities or concentrations. However it is now a linear system, instead of an ODEs system. This is in fact a lot easier to solve computationally, and can scale to large systems with over 1000 reactions instead of the 4 reactions presented here.

While what happens intracellularly is determined by the linear system, there is still modularity at the extracellular world, the boundaries of the system, *i.e.* what we give as input to the system, and want as output. For example, were we to consider a concentration of $[S_{out}] = C_S$ and a product yield of $[P_{out}]/[S_{out}] = Y_{PS}$, such bounds could be applied to the first and last equation of the system without harming the modelling formalisms, as those constraints would not affect the topology of the intracellular metabolic network, which is supposed invariant.

Let us this time assume an open system, where mass flows in and out, *e.g.* with mass-balance dynamics according to equation 1.25. Since we will be using the steady-state assumption, we should model inputs and outputs quantities as constant flows over time. Usually, one can assume the time of a steady-state model to correspond to initial concentrations, and final concentrations of an experiment. We rewrite:

$$\begin{aligned} \frac{d[S_{out}]}{dt} &= -v_{S_t} + C_S \\ \frac{d[P_{out}]}{dt} &= v_{P_t} - Y_{PS} \times C_S \end{aligned} \tag{1.54}$$

Assuming the steady-state hypothesis for metabolites in the system; and initial substrate concentration and final product yield as $[S_{out}]_0 = C_S = Y_{PS} = 1$ *arbitrary flux unit*, we get the following trivial linear system:

$$\begin{aligned} -v_{S_t} &= 1 \\ v_{S_t} &= v_E \\ v_E &= v_{P_t} \\ v_{P_t} &= 1 \end{aligned} \tag{1.55}$$

This time the solution is interpretable, if 1 unit of S goes through the metabolic network in Figure 1.10, then 1 unit of P will be produced. And in the process, fluxes of every reaction in the network were estimated. We now have a simple illustration of how microbial growth data as detailed in section 1.4 can be used as constraints in steady-state modelling to obtain reaction fluxes; and thus validate – or invalidate – growth predictions, product yields and potential byproducts.

For a better modelling capacity, a second hypothesis, after steady-state, is formulated: that of having pre-determined reaction reversibilities. Modelling reaction fluxes at steady-state as a linear system, one gets reaction fluxes as real-valued linear variables, and thus irreversible-only reactions should correspond to positive-only linear variables, while reversible reactions could correspond to positive or negative variables.

Note that, as underlined in subsection 1.4.5, working on a model with the steady-state assumption means that any experimental data should be collected under steady-state conditions.

Hypothesis 1.6.1 – Modelling biological systems at steady-state

By modelling biological systems at steady-state, the consumption and production of metabolites by reactions over time is considered constant. This is the steady-state assumption.

For metabolites $Mets$ internal to the biological system, the steady-state hypothesis simply states:

$$\forall M \in Mets, \qquad \frac{d[M]}{dt} = \sum v_{production} - \sum v_{consumption} = 0 \tag{1.56}$$

In other words, there is no accumulation over time of metabolites in the system.

Metabolites external to the system go directly from consumption in the system to production outside the system.

Hypothesis 1.6.2 – Reaction reversibilities are pre-determined

For the reaction fluxes v_R of a reaction R, we have:

- $v_R > 0$ if R operates forwards,
- $v_R < 0$ if R operates backwards,
- $v_R = 0$ if R does not operate.

Due to the complexity of modelling thermodynamics and reaction reversibilities, fluxes bounds are predetermined according to biological knowledge and experimental data.

Thus, a systems biology modeller must determine which reactions are reversible and which are not.

These reversibilities impose constraints on the domain of the solution space defined by the linear system derived from steady-state assumption.

Seeing the benefits of this abstraction, the interest from dynamic modellers has shifted to steady-state models, modelling the intracellular world as static, and keeping only the extracellular world dynamic. Methods to analyze steady-state models are defined in the area of constraint-based modelling, which we develop in the next chapter.

Chapter 2

Constraint-based modelling

As a complement to the dynamic modelling formalism, steady-state modelling was born. Steady-state modelling is also sometimes interchangeably termed constraint-based modelling (CBM). Here, we will use the term constraint-based modelling to refer to the construction of metabolic models specifically adapted for steady-state modelling.

The term constraint-based modelling likely comes from Bernhard O. Palsson, co-inventor of Flux Balance Analysis (FBA) [27], and author of an influential reference book named "Systems biology: Constraint-based reconstruction and analysis" [91]. The name constraint-based modelling is also present in the name of the widely used COBRA toolbox, standing for COnstraint-Based Reconstruction and Analysis methods [92], and early instances include works with Markus Covert [93, 94]. Palsson introduces the idea in "The challenges of in silico biology": imposing a successive series of biological constraints, one can limit likely cellular behavior to a formulated solution space [95].

Although our main methods of interest (EFMs, MCSs) differ from the main method of interest of Palsson (FBA), we very much abide by the same ideas. In particular, the addition of constraints into the computation of EFMs and MCSs is a major challenge that should be adressed for two reasons: first, it makes the computation of these solutions on large metabolic models achievable, and second, there is a need for methods that adress the finer details of biological reality as experimental biologists observe it, and we believe such a level of accuracy can only be permitted by the addition of constraints. The major advantage of EFMs in particular is removing the bias from the objective function from FBA, which causes a significant loss in biological accuracy.

Throughout this chapter we will detail the basis of mathematical programming and metabolic modelling, as well as every relevant method in constraint-based modelling. The methods of most importance that will be studied in this thesis are in fact Elementary Flux Modes (EFMs) and Minimal Cut Sets (MCSs). But to understand their interest and their functioning we must also introduce Flux Balance Analysis (FBA) and Linear Programming (LP).

Elementary Flux Modes (EFMs) and Minimal Cut Sets (MCSs) are the two methods that our tool, *aspefm*, proposes to do with a better computation algorithm than the competition. While this chapter introduces all relevant methods for our thesis, *aspefm* and our subsequent results will be presented from chapter 3 onwards.

2.1 Linear programming

Linear Programming (LP) consists of solving a linear program, or linear problem. To define such problems, a first look should be given at the area of optimization and mathematical programming. A mathematical program is simply a mathematical optimization problem under constraints, as defined in Definition 2.1.1 [96]. This general definition aims to englobe every common optimization problem.

Some examples of mathematical optimization subclasses include: first, linear programming, for the solving of linear constraints over linear variables, and popularized in 1947 by Dantzig with his simplex method [97]. Secondly, convex programming consists in minimizing a convex function over convex constraints; minimizing the unconstrained convex function can be done with algorithms such as gradient descent [S75]. If several objective functions must be optimized at the same time (or consecutively), solving the problem is called multi-objective optimization (or multi-level programming) [S76]. And finally, if some of the constraints or the objective function are non-linear, solving such a problem might be called non-linear programming [S77, S75].

In the absence of an objective function, a mathematical program becomes a feasibility problem, where constraints just need to be satisfied. Lustig and Puget thus argue that linear programs and mathematical programs join the area of constraint satisfaction problems, *ie.* combinatorial problems [98]. However, this chapter mostly deals with linear programming problems, which are resolved mathematically, rather than exhaustively. Combinatorial problems and constraint satisfaction problems, solved with different solvers, will be detailed in chapter 3, through the so-called constraint programming formalism.

Definition 2.1.1 – Mathematical Programming

A mathematical program is a program of the form:

$$\begin{aligned} &\text{Minimize}_x\ f(x) \\ &\text{subject to:} \\ &g_i(x) \leq 0 \quad i = 1, \cdots, m \\ &x \in Sols \subset \mathbb{R}^n \end{aligned} \tag{2.1}$$

Where $x_1, x_2, \cdots, x_n$ are the variables of the problem.
$f(x)$ is called the objective function (minimization or maximization)
$Sols$ is called the domain of solutions.
And $(g_i(x) \leq 0)\ \ i = 1, \cdots, m$ are called the constraints.
Note that the formalism englobes maximization as such a problem is equivalent to maximizing $-f(x)$.

To understand the simplicity and efficacity of linear programming solving, we first need to adress the definition of convexity. Convexity is defined in Definition 2.1.2. A special property of convex mathematical programs – programs over real variables, with convex constraints and a convex objective function – is that any local optimum they admit is also a global optimum. This is explained in Theorem 2.1.1 and illustrated in Figure 2.1.

Linear functions (or affine functions) are convex, meaning that linear programming: consisting in minimization of a linear function over linear constraints, admits a single global optimal value. Note that this is only true of values, *ie.* result of $min\ f(x)$, and not of solutions, *ie.* assignment of variables x. In fact, linear programs often admit several optimal solutions (see Figure 2.2). Linear programming is properly defined in Definition 2.1.3. A graphical example of the resolution of a linear program in dimension $n = 2$ variables is shown in Figure 2.3.

Definition 2.1.2 – Convexity

A set K of $\mathbb{R}^n$ is convex if, for each pair of distinct points $a = (x_1, \cdots, x_n)$, $a' = (x'_1, \cdots, x'_n)$ in K, the closed segment s with endpoints a and a' is contained within K.

From a set of points $A = (a_1, a_2, \cdots, a_n)$, its convex hull $conv(A)$ is defined as the smallest convex set $K \subset \mathbb{R}^n$ containing all of A. In dimension 2, the convex hull of any given polygon or set of points is a convex polygon. In dimension n, the convex hull of any given finite set is a convex polytope [W7].

A function $f(x) : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain is a convex set and for all x, x' in its domain, and all $\lambda \in [0, 1]$:

$$f(\lambda x + (1 - \lambda)x') \leq \lambda f(x) + (1 - \lambda)f(x') \tag{2.2}$$

All real-valued affine functions, or linear functions, of the form $f(x) = a^T x + b,\ x \in \mathbb{R}^n$, are convex.

A mathematical program is said to be convex if it consists in minimizing a convex function under convex constraints and over a closed convex domain [96].

Theorem 2.1.1 – Optimality and convexity

In optimization, finding local optima to functions is often achievable. However, there might not be a guarantee that a local optimum solution is a global optimum. Also, exhaustive testing of all possible solutions is not a computationally adequate maneuver. As a result, approximation methods or heuristics can be used to give the best possible solutions in reasonable times, depending on the problem.

On the other hand, convex mathematical programs follow a fundamental special property: in a convex program, any local optimum is a global optimum. This means convex programs are relatively easier to solve.

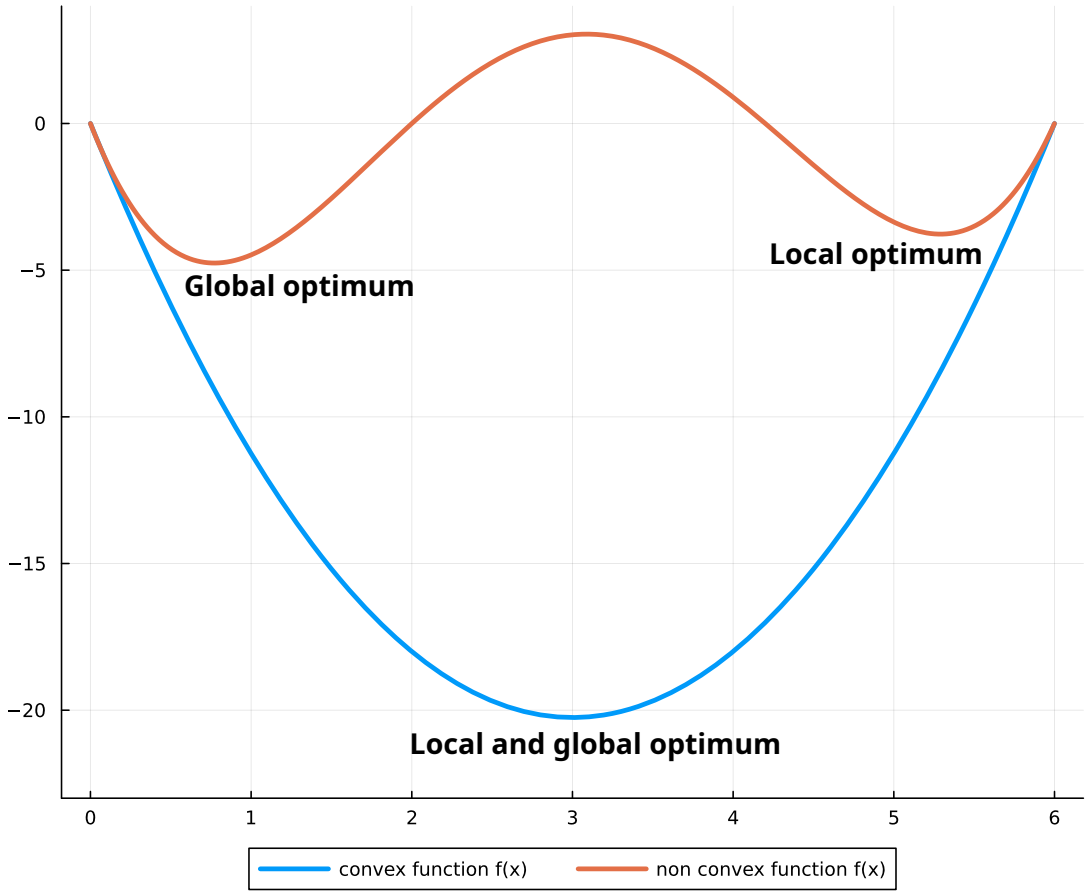


Figure 2.1: Local and global optimum of functions to be optimized

We further develop the convexity of linear programming in Definition 2.1.5. The theory surrounding linear programming and its resolution cannot be dissociated from the theory of polytopes and polyhedra [96, 99, 100]. A linear program, or linear problem (LP) is formally defined by: m constraints over n real-valued variables, forming a polyhedron $P = \{x \mid Ax \leq b, x \in \mathbb{R}^n\}$, and an objective function min or max $f(x) = c^T\ x$. Where A is a $m \times n$ real-valued matrix, and $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ are real-valued vectors.

Following this definition, a canonical linear programming problem is simply defined as Definition 2.1.4 [96]. This is the definition used by common linear programming solvers. The solver of choice for linear programs is IBM© *cplex* [W8], although other alternatives exist, including gurobi [W9], cvxpy [S78], GLPK [W10].

Algorithms for solving linear programming include the aforementioned simplex method [97], and the now widely used and very efficient interior-point methods [101]. The simplex method is supposedly exponential in worst-case, though a probablistic complexity analysis places the algorithm as polynomial in only the number of variables n. Interior point methods are polynomial in worst-case in the dimension n but also in the binary length of the data L, and have a polynomial probablistic complexity of $O(\sqrt{n} \ln n)$ [101]. Polynomial degree may range up to 4 [101].

Definition 2.1.3 – Linear Programming

A generic linear program is a program defined by the form:

$$\begin{aligned}
&\text{Minimize}_x\ f(x)\\
&\text{subject to:}\\
&g_i(x) = 0 \quad i \in I_0 \qquad \text{(equality constraints)}\\
&\left.\begin{aligned} g_i(x) \leq 0 \quad i \in I_-\\ g_i(x) \geq 0 \quad i \in I_+ \end{aligned}\right\} \text{(inequality constraints)}\\
&x = (x_1\ x_2 \cdots x_n) \in \mathbb{R}^n
\end{aligned} \tag{2.3}$$

Where f and $g_i\ \forall i$ are all affine or linear functions over real variables x.

Any linear program is a mathematical program. Indeed, (I_0) constraints can be rewritten as (I_+) and (I_-) constraints, and any (I_+) constraint can be rewritten as an (I_-) constraint by multiplying the inequality by -1.

Secondly, any linear program can be rewritten in a canonical form, operating on positive variables.

Definition 2.1.4 – Canonical form of Linear Programming

A linear program in canonical form is a program defined by the following:

$$\begin{aligned}
&\text{Max}_x\ f(x) = c^T x\\
&\text{subject to:}\\
&Ax \leq b\\
&x \geq 0
\end{aligned} \tag{2.4}$$

Where linear variables are $x \in \mathbb{R}^n$, A is a $m \times n$ matrix, b is a vector $b \in \mathbb{R}^m$, c is a vector $c \in \mathbb{R}^n$, n is the number of variables and m is the number of constraints, f is the function to be maximized or objective function.

The canonical form is the one used in linear solvers. Usually, the solver asks that constraints and variables conform to that canonical form. In addition, strict equality $(=)$ and strict inequalities $(<,\ >)$ can only be modelled as normal inequalities over a very small tolerance ϵ, for example $Ax < b$ becomes $Ax - \epsilon \leq b$.

This is implemented in solvers as an ϵ tolerance parameter.

Definition 2.1.5 – Convexity of Linear Programming

Linear Programming is convex. The solution space, the definition domain of linear programming, is the set of all reals under linear constraints, which is a convex polyhedron. The minimized (or maximized) function is also linear. Thus linear programming is convex and admits only global optima.

More precisely, a subset $P \subset \mathbb{R}^n$ is a (convex) polyhedron if there exists a positive integer m, an $m \times n$ matrix A, and a vector $b \in \mathbb{R}^m$ such that P is of the form:

$$P = \{x \mid Ax \leq b, x \in \mathbb{R}^n\} \tag{2.5}$$

When $b = 0$, the polyhedron contains the origin, and is said to be a polyhedral cone [99, 100].

Note that using this definition from the linear programming field, a polyhedron is not necessarily bounded. Optimization procedures require polyhedra to be bounded. A bounded polyhedron is then called a polytope. In practice, linear programming is performed with polytopes, which are bounded, finite, and their own convex hulls. Arbitrary maximum variables bounds are automatically set, avoiding unboundedness. Note that definitions of polytopes and polyhedra may vary [W11].

Definition 2.1.6 – Common issues in Linear Programming

In linear programming, problems might arise. A first common problem is multiple optimal solutions: there might be many optimal solutions, which might result from the choice of the objective function, or the problem being unconstrained. Solvers have to order solutions and return the first solution found. Secondly, when the problem is too constrained, constraints might be conflicting, and no solutions might exist, in that case the problem is said to be infeasible. And thirdly, if objective functions, bounds and constraints are not set correctly, a problem might be unbounded, and cannot admit optimal solutions.

Common issues encountered in Linear Programming are summarized in Definition 2.1.6. Multiple optimal solutions is illustrated in Figure 2.2. Infeasible problems are illustrated in Figure 2.5 and unboundedness is illustrated in Figure 2.6.

2.2 Mixed-Integer Linear Programming

There are multiple optimization methods derived from linear programming, including quadratic programming and geometric programming. These will not be detailed here but they are relevant in many constraint-based modelling approaches. These variants all fall into the area of mathematical programming, but we might also refer to these as 'linear programming-related methods' in the following sections.

The other method of interest that we will present is Mixed-Integer-Linear Programming (MILP), a notable subdomain of mathematical programming which allows restriction of variables to integers. This method is of particular interest to this thesis, as it is the basis for the concurrent method to ours for computation of EFMs and MCSs, as we will detail later.

In Integer Linear Programming (ILP), while the objective function and constraints are linear like for Linear Programming (LP), variables are now integers instead of reals [99]. The solution space is thus no longer convex, and is rather an union of convex solution spaces, for each fixed integer point.

For instance, comparison between LPs and ILPs is shown in Definition 2.2.1 and equation 2.1.4, as well as for their graphical resolution in Figure 2.3 and Figure 2.4. ILPs bridges the gap between linear programming and combinatorial problems, but it does so with a radical loss in computation performance. As expected, exhaustively enumerating integers is terrible in practice, thus most often integer variables end up restricted to small sets of values, such as Booleans.

A more generalist definition than ILPs are MILPs (Mixed Integer Linear Programs), which may contain both linear variables and integer variables. A definition of MILPs is proposed in Definition 2.2.2. In practice, MILPs are most often used with Boolean variables, which may act as indicators of positivity or nullity for real variables.

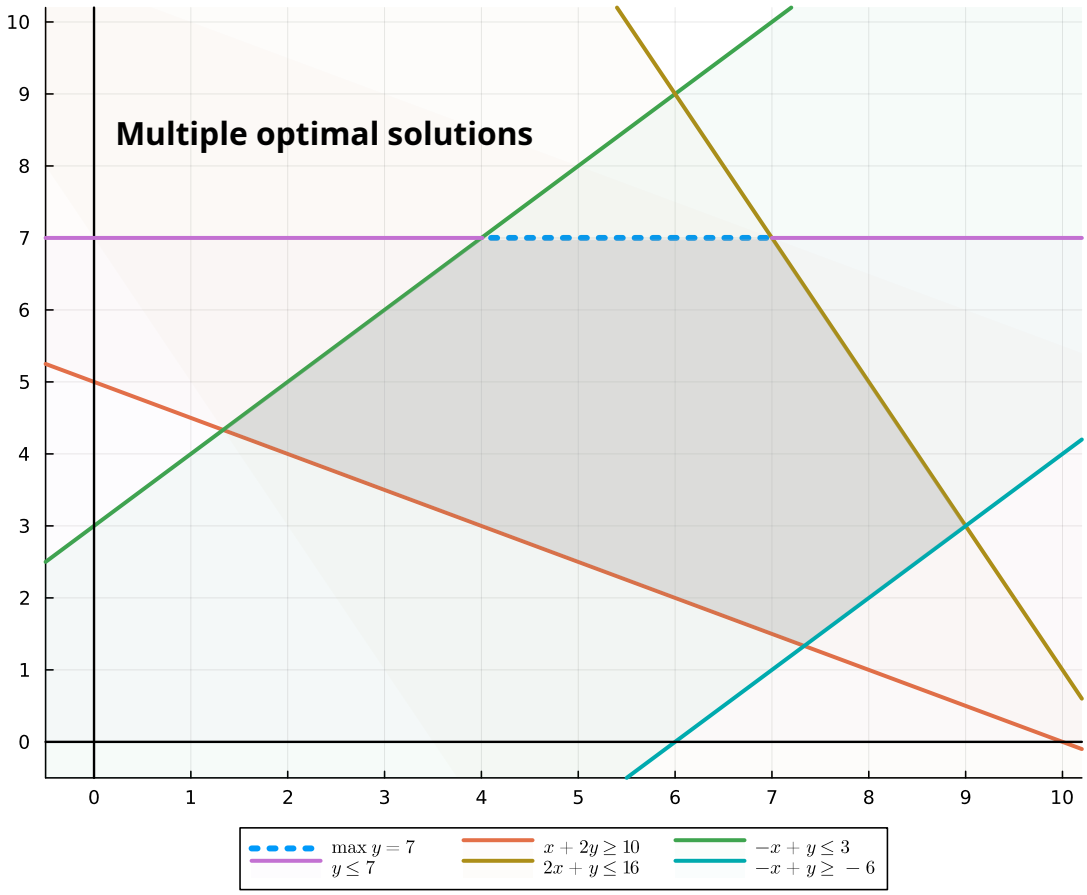


Figure 2.2: Common issues in Linear Programming: multiple optimal solutions

Definition 2.2.1 – Integer Linear Programming

Let us take the canonical linear program form from equation 2.4, but express it as a mathematical program over Boolean variables.

$$\begin{aligned} \text{Max}_x\ & f(x) = c^T x \\ & \text{subject to:} \\ & Ax \leq b \\ & x \in \{0,1\} \end{aligned} \tag{2.6}$$

This is an integer linear program, or integer program, over integer variables $\{0,1\} \subset \mathbb{Z}$. In particular, this program corresponds perfectly to the 'Knapsack Problem', which is one of the Karp combinatorial problems [102]. As a result, complexity of integer linear programming, and of the related mixed integer linear programming, is known to be NP-hard [99].

Such a program can be solved by linear relaxation of the problem, that is, by solving the linear program in equation 2.4 with $0 \leq x \leq 1$ as additional constraint, and then exhaustively testing if solutions after setting $x = 0$ or $x = 1$ belongs to the solution space. This procedure is called Branch-and-Bound [103].

Definition 2.2.2 – Mixed Integer Linear Programming

A Mixed Integer Linear Program, or MILP, is a linear program that might contain a 'mix' of linear variables, integer variables, Boolean variables, under standard linear constraints. For instance:

$$\begin{aligned} \text{Max}_x\ & f(x) = c^T x \\ & \text{subject to:} \\ & A_l x_l \leq b_l \\ & A_z x_z \leq b_z \\ & x_l \in L \subset \mathbb{R} \quad \text{(linear variables)} \\ & x_z \in I \subset \mathbb{Z} \quad \text{(integer variables)} \end{aligned} \tag{2.7}$$

Where $x = (x_l\ x_z)$, and A_l, A_z and b_l, b_z denotes the submatrices and subvectors corresponding to either linear or integer variables. For example, common MILPs presented in this thesis are programs where we have two classes of variables: linear variables over positive reals, *ie.* $L = \mathbb{R}^+$, and Boolean variables, or logical variables, defined in $I = \mathbb{B}$. Of course, more than two classes of variables can be considered.

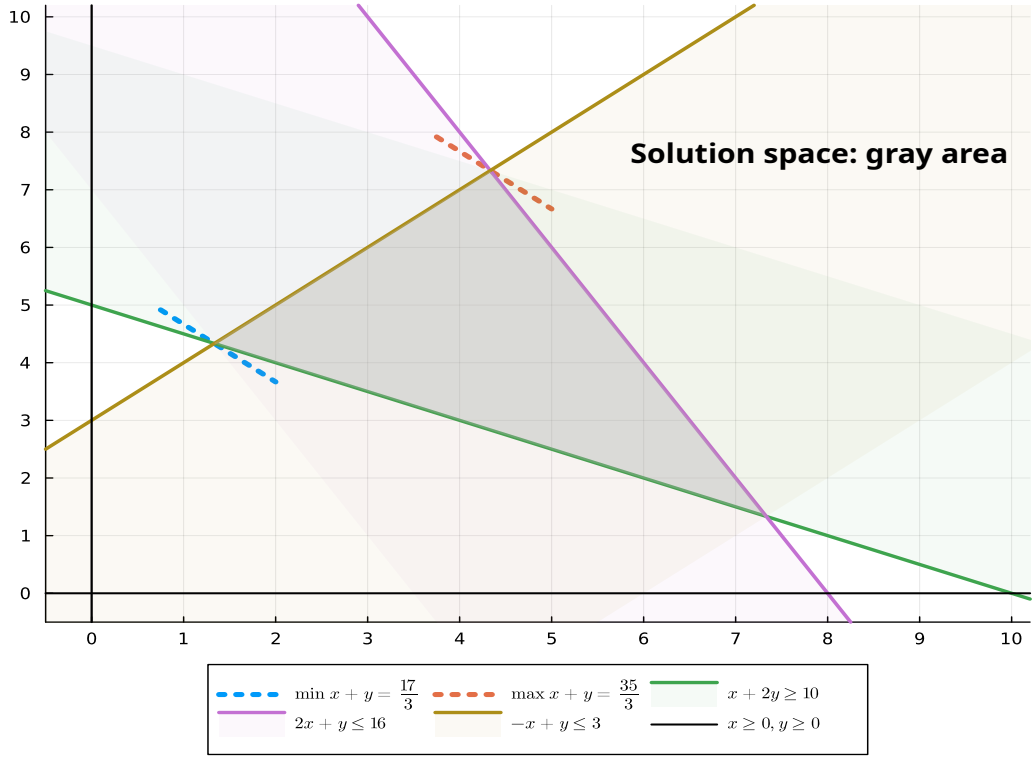


Figure 2.3: Resolution of a simple linear program

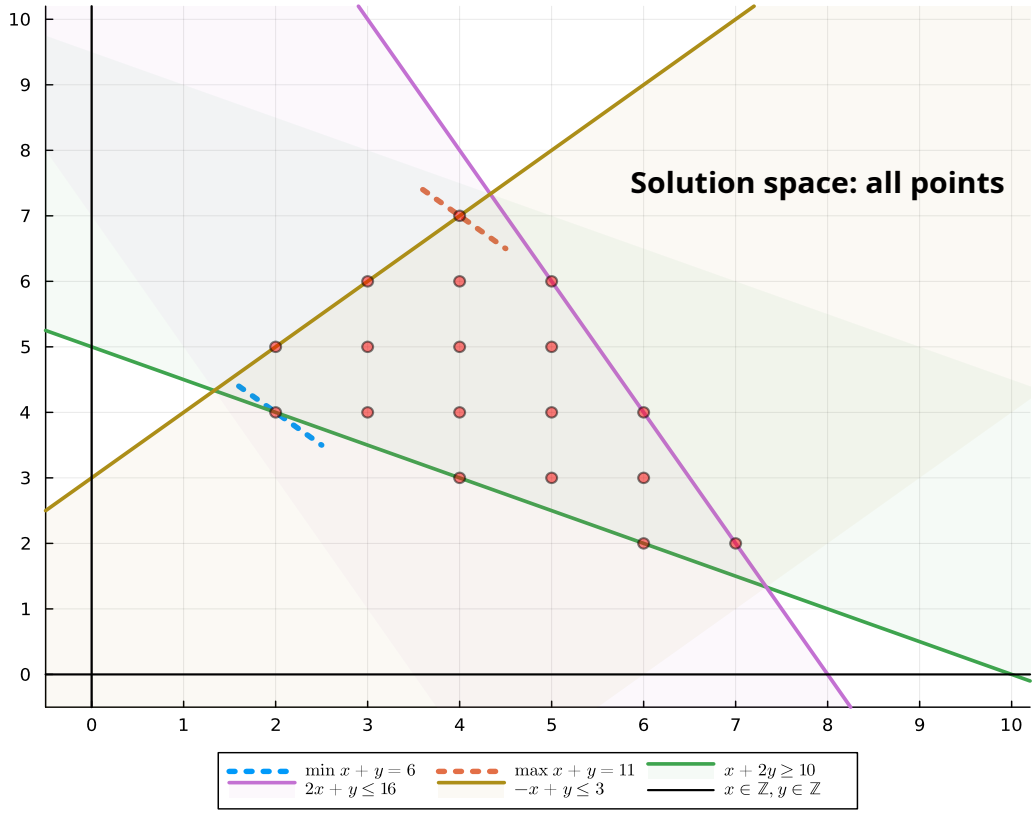


Figure 2.4: Resolution of a simple integer linear program.
Linear relaxation of this integer program gives Figure 2.3

From there on, the expression *indicator variables* refers to Boolean variables in a MILP that indicate nullity or positivity of linear variables in the MILP. For example, let us define $(z_1, z_2, \cdots, z_n)$ Boolean variables and $(x_1, x_2, \cdots, x_n)$ linear variables. The program must be such that the *indicator constraints* $(z_i = 0) \Leftrightarrow (x_i = 0)$ and $(z_i = 1) \Leftrightarrow (x_i > 0)$ apply. Additionally, these must be the single constraints mixing both of these linear and Boolean variables; if we have such a program, then Boolean variables might be called *indicator variables*.

As you could see here with logical equivalence, logic operations can be incorporated in MILPs. These are internally encoded differently within the solver, either translated to linear constraints, or with a Boolean resolution backend [104]. Of course, solvers for LPs generally include resolution of MILPs, and of other 'linear-programming-related methods' – especially in the case of *cplex*.

A special property of MILPs with indicator variables is that *integer cut* constraints can be added, as is described in Definition 2.2.3. Although not thoroughly documented online, integer cut constraints have been around for a while [105] and were notably used in a metabolic modelling context as early as 2005 [106]. Integer cut constraints allow exhaustive enumeration of all MILP solutions. Now that integer cut constraints were discussed, we have every necessary element to understand computation of EFMs and MCSs with MILPs. But to start, we will introduce and formalize constraint-based metabolic modelling, and the classical FBA optimization procedure.

Definition 2.2.3 – Integer cut constraints

Let us suppose a MILP problem (MP) where the goal is to minimize the number of active Boolean indicator variables k_i, *i.e.* corresponding to linear variables v_i with non-zero values.

An integer cut constraint is then an additional constraint to the MILP, of the form:

$$\sum_{i=i_1}^{i_P} k_i \leq P - 1 \tag{2.8}$$

where $k_i, \forall\, i = i_1, \cdots, i_P$ are the P active variables of a previously found solution.

The constraint ensures that in next calls to the solver, a solution with these active variables will not be found, meaning that solution and all supersets of that solution will be excluded.

Integer cuts are added for each previous solution. The program (MP) becomes:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{i=1}^{n} k_i \\
\text{Subject to} \quad & \text{Constraints on linear variables } v_i \\
& k_i \text{ indicator variables of } v_i \\
\text{And integer cuts} \quad & \sum_{i=i_1}^{i_P} k_i \leq P - 1 \;\; \forall \{i_1, \cdots, i_P\} \in PreviousSols \\
\text{Where} \quad & k_i \in \{0, 1\} : \forall\, i,\, 1 \leq i \leq n
\end{aligned} \tag{2.9}$$

In practice, the (MP) program can enumerate distinct solutions for which active variables are subset-minimal.

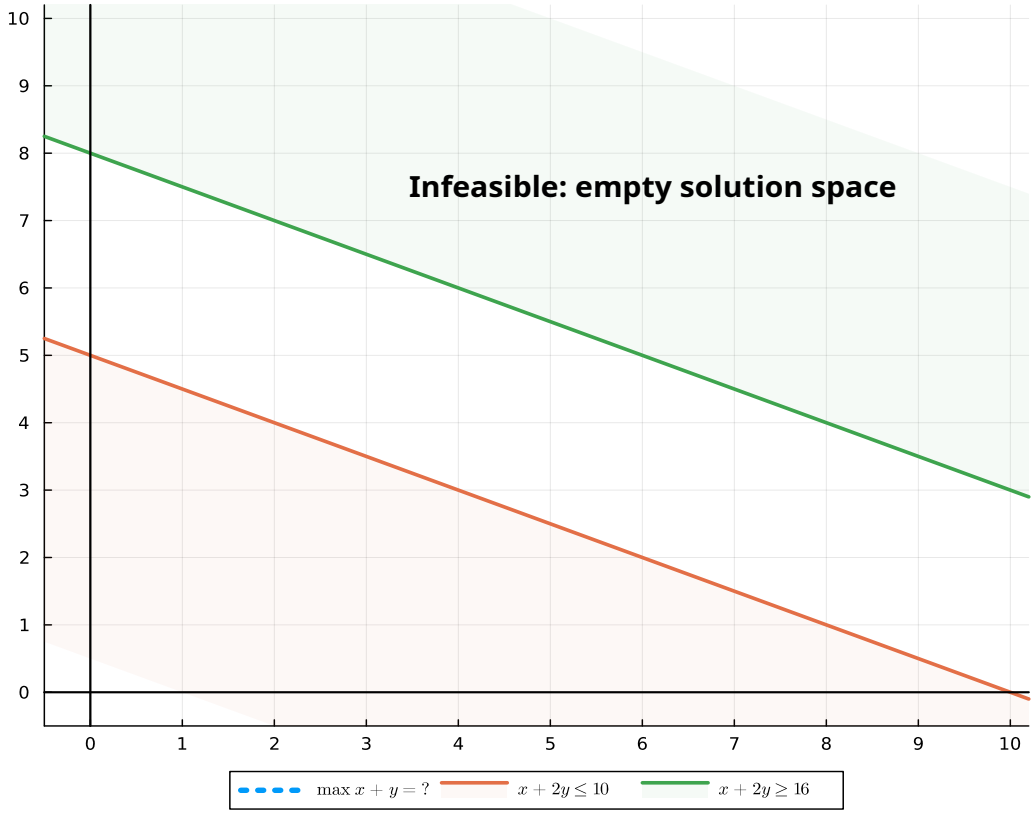


Figure 2.5: Common issues in Linear Programming: infeasible problems

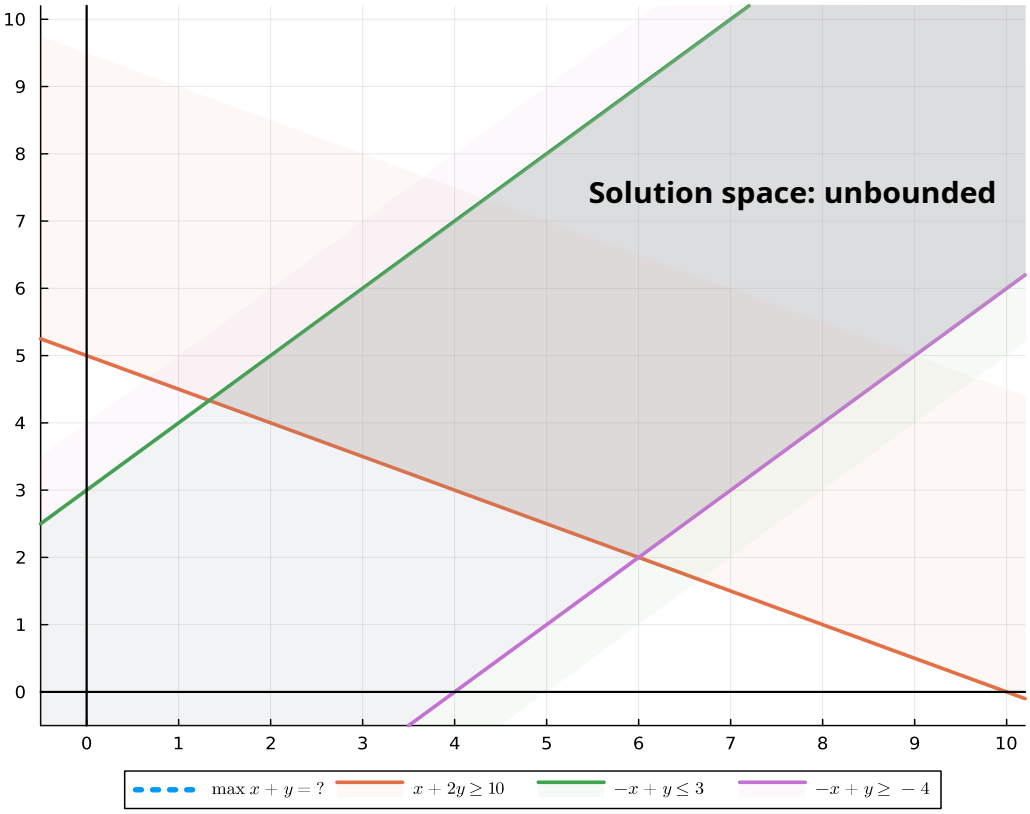


Figure 2.6: Common issues in Linear Programming: unboundedness

2.3 Metabolic modelling

There are multiple ways of representing metabolic networks. For example, one may represent metabolic networks as a bipartite graph with metabolites and reactions as the two constitutive disjoint sets. Others may represent metabolic networks as Petri nets [S79]. Our analyses will deal with metabolic models, which are directed hypergraphs linking metabolites and reactions through hyperedges which are weighted by stoichiometry [107]. We will use the term metabolic networks and metabolic models interchangeably; it refers to a metabolic model as defined here.

Definition 2.3.1 – Metabolic model

A metabolic model, or metabolic network, representing a set of metabolic reactions, is a directed hypergraph $H = (Met, Reac, Stoch)$ of nodes Met, of hyperedges $Reac$, and of integer-valued or real-valued stoichiometry weights on hyperedges $Stoch$. The latter which are usually better represented by a stoichiometry matrix.

Additionally, one might want to distinguish subsets of Met: the set of internal metabolites Int from Ext: the set of external metabolites. Ext is generally defined as the set of nodes m for which either the metabolite m is never consumed or the metabolite m is never produced.

In other words, $Met = Int \cup Ext$, with $Ext = \{m \in Met \mid (d^+(m) = 0) \vee (d^-(m) = 0)\}$ and $Int = Met \setminus Ext$, where $d^+(m)$ and $d^-(m)$ respectively represent *in degree* and *out degree* in graph theory terms.

The field of dealing with analysis and construction of metabolic models is called metabolic modelling. In metabolic modelling, the system that is most commonly being modelled is a cell: internal metabolites are intracellular, and external metabolites are extracellular. These are called compartments of the model.

Metabolic networks are greatly interconnected. In particular, in usual cellular networks, hydrogen protons and coenzymes such as NAD and ATP could be considered hub metabolites, as they are nodes of many hyperedges of the networks. Measures of the connectedness of the networks are often used in systems biology and can be used to derive interesting properties [S80]. In particular, since metabolic networks are hypergraphs [107], hypergraph partition techniques might be applied to parallelize problems [S81, S82].

Metabolites are separated into two categories: internal, and external, external metabolites being the hypergraph nodes that are *sources* (*i.e.* inputs, nodes with null in degrees) and *sinks* (*i.e.* outputs, null out degrees). Usually, metabolic models represent cellular systems, thus an easy distinction between internal metabolites and external metabolites would be being intracellular and extracellular.

With these distinctions, a metabolic model would be defined by a set of extracellular inputs and outputs, while the intracellular world would be constant, similar to how a biologist would make expriments in their lab. However, due to the high interconnection of hydrogen protons and coenzymes ADP and NAD as mentioned before, sometimes, intracellular metabolites are set as external, as if they were transported in and out of the cell, in order to abstract their mechanisms (*e.g* see ATP in [108]). This is not an absurd abstraction, as limiting the confines of our model to simply inside/outside the cell, while more comprehensive, is arbitrary.

Hydrogen protons are often even removed from models due to the complexity of modelling them (*e.g* again see [108]). Hydrogen protons quantity can be represented by the familiar chemical notion of pH in aqueous conditions.

Definition 2.3.2 – Stoichiometry matrix

The stoichiometric matrix, also called S-matrix for short, is an alternate way of representing the metabolic network, by its sparse weighted adjacency matrix. It is such that every column corresponds to a reaction and every row corresponds to a metabolite, and it is most-often real-valued.

Let us define the stoichiometry matrix $S \in \mathbb{R}^{m \times r}$, m number of metabolites, r number of reactions, by:

$$\forall i \in Met, \quad \forall j \in Reac, \quad S_{ij} = \begin{cases} k \text{ if reaction } j \text{ produces } k \text{ units of metabolite } i, \\ -k \text{ if reaction } j \text{ consumes } k \text{ units of metabolite } i \\ 0 \text{ otherwise.} \end{cases} \tag{2.10}$$

Or, in other words, the reaction stoichiometry coefficients are negative for the *reactants* of the reactions, and positive for the *products* of the reaction.

In *steady-state* modelling, including *external* metabolites in the S-matrix is not a recommended practice.

Indeed, the steady-state assumption only applies to metabolites *internal* to the system.

Another way to understand the stoichiometry matrix is in terms of mass-balancing. All chemical compounds are assumed to be mass-balanced, therefore this should be reflected in the stoichiometry matrix [91].

Definition 2.3.3 – Dealing with reversibilities

To incorporate predefined reversibilities, we can extend our hypergraph formalism into a metabolic model $M = (Met, Reac, Stoch, Rev)$, where Rev is the subset of hyperedges $Reac$ defining reversible reactions.

Now, instead of having two separate reactions defining a reversible one, we can incorporate bi-directionality into a single edge. However, this has the downside of requiring more information to be stored.

If dealing with the stoichiometric matrix formalism, we can add an additional informative 0-1 vector $rev \in \mathbb{B}^r$, and the metabolic network becomes described by the pair (S, rev).

Metabolic models are also called stoichiometric models: since they can be described solely by a stoichiometric matrix S, incorporating information for the mass-balance of every chemical reaction (see subsection 1.4.2 for mass-balance). This stoichiometry matrix can be seen as a linear transformation of the flux vector to a vector of time derivatives of the concentration vector [91]. This allows us to study dynamics of the system, and in particular we will be interested in the system's steady-state (see Hypothesis 1.6.1). In addition, our second assumption is that the reversibility of reactions are pre-defined (see Hypothesis 1.6.2). We define the stoichiometric matrix and explain the incorporation of reversibilities in Definition 2.3.2 and Definition 2.3.3.

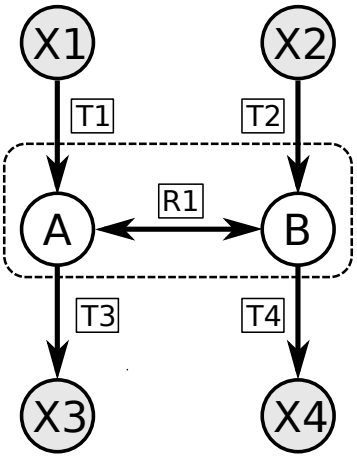


Figure 2.7: Toy model network of 5 reactions

A toy model metabolic network of 5 reactions (1 reversible, 4 irreversible) is presented in Figure 2.7. The network possesses 4 external metabolites and 2 internal metabolites. However, for helping with the steady-state assumption, external metabolites are assumed to be excluded from the stoichiometry matrix.

Therefore, its associated stoichiometric matrix S is:

$$S = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cccccc} T1 & T2 & T3 & T4 & R1 & R1_{backwards} \end{array} \\ \begin{pmatrix} 1 & 0 & -1 & 0 & -1 & 1 \\ 0 & 1 & 0 & -1 & 1 & -1 \end{pmatrix} \end{array} \tag{2.11}$$

This matrix includes the split of the reversible reaction $R1$ into two irreversible reactions for the forwards and backwards direction. Representing the model's $R1$ reaction by a single bi-directional hyperedge, we would instead get:

$$S = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{ccccc} T1 & T2 & T3 & T4 & R1 \end{array} \\ \begin{pmatrix} 1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & -1 & 1 \end{pmatrix} \end{array} \tag{2.12}$$

$$rev = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

A particularly convenient format to store metabolic networks is the METATOOL format [109]. All information required for metabolic modelling is stored on that format – except lower flux bounds and upper flux bounds: if those are needed they should be inferred from reversibilities. The format includes internal and external metabolites, reaction stoichiometry and reaction reversibility in an easily readable way, as seen in Listing A.1.

Indeed, when analyzing reaction fluxes, if dealing with irreversible reactions, then only the forwards direction is authorized, meaning flux bounds are $[0, \infty[$, while with reversible reactions, the backwards direction is also authorized, meaning flux bounds are $]-\infty, \infty[$.

When dealing with the definition of an extracellular medium however, or with a specific flux value of ATP maintenance one wants to constraint a model to [71], not having the possibility to specify flux bounds is an inconvenience. Often the lower flux bounds on external metabolites from the medium correspond somewhat to the quantity of metabolite present in the experimental growth medium. To incorporate lower flux bounds and upper flux bounds, the further notion of constraint-based model is used, as well as the more complete SBML modelling format [69].

2.4 Stoichiometric flux cone

Let us consider a stoichiometric matrix $S \in \mathbb{R}^{m \times r}$ of m internal metabolites (lines) and r reactions (columns). For each reaction, consumed or produced metabolite quantities are reported on the matrix coefficients, negative for consumption and positive for production. The notion of reaction flux relates to the evolution of metabolite quantity over time, similarly consumption of metabolites over time will be negative and production will be positive. We denote by v a reaction flux vector of rates dependent on metabolite concentrations over time, and consistent with pre-determined reaction reversibilities.

The evolution of internal metabolite concentrations $x(t)$ over time is written by an ODE system:

$$\forall i \in Int, \quad \frac{dx_i(t)}{dt} = S.v(x(t))$$

Where, for each metabolite, its concentration $[M]$ or x_i is given by a linear combination of all fluxes of reactions consuming or producing that reaction, computed by multiplying the S-matrix defining the reaction stoichiometry by the flux vector v, with the fluxes themselves being functions of metabolite concentrations over time.

At steady-state, the evolution of internal metabolite concentrations over time is null, therefore $dx(t)/dt = 0$ and $S.v(x(t)) = 0$, which we may simplify into $S.v = 0$, with fluxes becoming constants rather than functions. In addition, note that from linear algebra, we know that the set $\{v \mid S.v = 0\}$ defines the null space of S.

A flux distribution is a steady-state vector $v \in \mathbb{R}^r$ giving constant reaction consumption or production rates for each reaction. Its support is the set of active reactions: $Supp(v) = \{j \mid v_j \neq 0\}$. Irreversible reactions, for which flux is a non-negative real, are distinguished from reversible reactions, for which flux may be negative.

Therefore, any flux distribution v is included in the null space of S. In particular, we could compute the stoichiometric kernel, kernel of the stoichiometric matrix $Ker(S)$, which would provide a basis of the null space. This could be calculated with Gauss-Jordan elimination methods, or approximation methods for large matrices. However, the issue with the null space alone is that it does unfortunately not respect reaction reversibilities.

We denote here by C the set of stoichiometric null space vectors which respects reaction directionalities. This solution space is a polyhedral convex cone [96, 100].

$$C = \{v \in \mathbb{R}^r \mid Sv = 0 \text{ and } \forall j \text{ irreversible } v_j \geq 0\}$$

The solution space C is called the stoichiometric flux cone. polyhedral convex cones are particular polyhedra useful in the theory of Linear Programming. And indeed, exploring this set for solutions could be done by setting $Sv = 0$ as linear constraints and using Linear Programming.

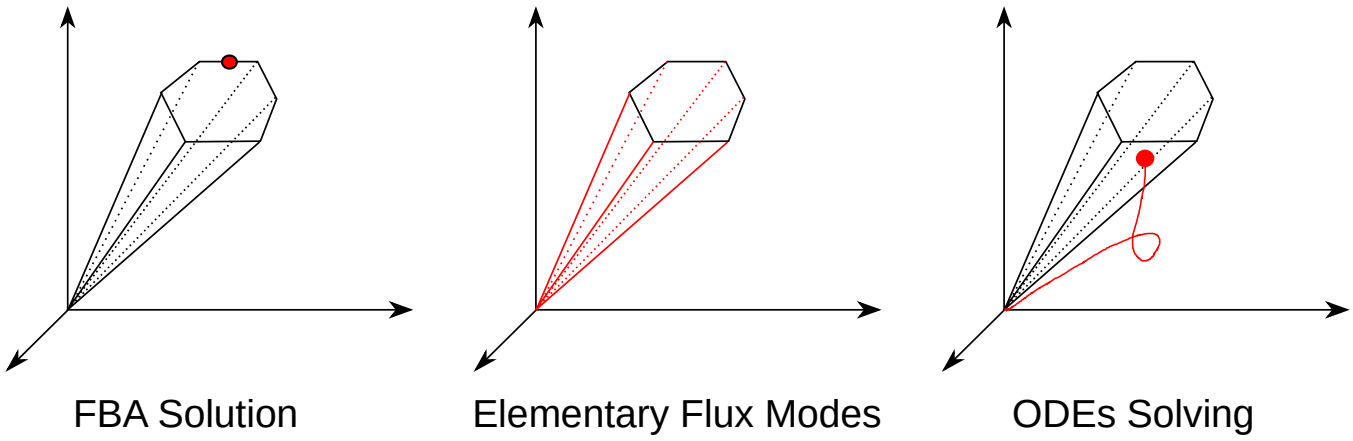


Figure 2.8: Constraint-based modelling in a nutshell: Steady-state modelling: FBA, EFMs vs. Dynamic Modelling

In fact, the stoichiometric flux cone is the LP solution space of our methods of interest, Flux Balance Analysis (FBA) (section 2.6) and Elementary Flux Modes (EFMs) (section 2.7). We represent the different applications of steady-state modelling: FBA, EFMs and for comparison, ODE dynamic modelling in Figure 2.8. While the flux vector in ODEs modelling is dynamic and might eventually reach – or not – a steady-state, FBA and EFMs help us characterize well-defined steady-state flux distributions [51].

2.5 Constraint-based modelling

In the introduction to this chapter, we remarked that the term constraint-based modelling, somewhat synonymous to steady-state modelling, likely comes from Bernhard Palsson and his team. Now, we properly define constraint-based modelling and a constraint-based model in Definition 2.5.1. Also, we will detail particular forms of constraint-based models: the ones where all reactions are irreversibles, and the ones where exchange reactions are added for every single extracellular metabolite. The latter is specifically an idea from Palsson's team, as noted in Markus Covert's book [51]. A short history of Palsson's team and their toolbox named COBRA is given in the next section: section 2.6.

An introduction to constraint-based modelling might be found in [95, 91], and two instructive complete reviews of the constraint-based modelling field and its concepts include [110, 92]. Other reviews, which include mentions to the use of – in particular – omics data to constrain a metabolic model, include [54, 59, 61, 31]. And an application of the constraint-based approach to our subject of interest, EFMs, which includes transcriptional regulation and which we will come back a lot to later, is [93]. The figure Figure 2.9 illustrating addition of constraints is inspired from [92].

Models with only irreversible reactions, which, if we have a constraint-based model cbm, we may simply note $Irrev(cbm)$, deserve a mention as they correspond to the proper encoding of these models in modern LP solvers. Indeed, since reversible reactions are represented by a single real-valued variable, one might think solvers prefer this to having two non-negative-only variables, but the reverse is actually true: having two non-negative-only variables for each reversible reaction speeds up the computation. To convert any constraint-based model into a constraint-based model with only irreversible reactions, one simply splits each reversible reaction into two forwards-only. A stoichiometric matrix such as in equation 2.11 is obtained instead of equation 2.12. Ideally, this is done by a backend and is hidden from the end user, but for more complex tools such as ours, we prefer to have it exposed.

In fact, the encoding of reversible reactions should no longer be done by a single 0-1 vector in constraint-based models (CBM). Since flux bounds are of major importance in FBA applications, we chose to define constraint-based models as the kinds of models were complete description of lower and upper flux bounds for each reaction are incorporated. This corresponds to the SBML format with Flux Balance Constraints (FBC) specification [69].

Definition 2.5.1 – Constraint-based modelling

Constraint-based modelling (CBM) is the subset of the metabolic modelling field which deals with the analysis of reaction fluxes at the steady-state with the stoichiometric flux cone.

A constraint-based metabolic model is a quadruplet $CB = (Met, Reac, Stoch, Bounds)$ with metabolites as hypergraph nodes Met, reactions as hyperedges $Reac$, weights on hyperedges $Stoch$, and additional lower and upper bounds $Bounds = \{(LB_i,\ UB_i) \in \mathbb{R} \mid i \in Reac\}$ for the flux of reactions $Reac$.

A constraint-based model brings as additional information to the metabolic model $(Met, Reac, Stoch)$ the lower bounds and upper bounds of the fluxes to be considered for linear programming computations.

As a result, a constraint-based model defines the following cone as solution space for linear programming:

$$C = \{v \mid Sv = 0,\ LB_i \leq v_i \leq UB_i\} \tag{2.13}$$

Where S is the stoichiometric matrix of (CB), defined from the stoichiometry $Stoch$, and v_i are the fluxes.

For simplicity, we can represent the model (CB) by its S matrix of m lines and n columns, m the number of metabolites, n the number of reactions, and its reaction bounds $(LB_i,\ UB_i)\ \ \forall i = 1, \cdots, n$.

Note that changes to these lower and upper bounds impact greatly the result of CBM analysis methods such as Flux Balance Analysis (FBA) and Elementary Flux Modes (EFM).

Palsson and collaborators devised the notion of exchange reactions in constraint-based metabolic models [51]. These are factice transport reactions that are added on top of the metabolic model's intracellular to extracellular transporters. Adding these new external transporters to every extracellular metabolite allows for a more proper definition of the external metabolites of the stoichiometry matrix. Setting the exchange bounds of these exchange reactions also allows one to simulate an extracellular growth medium, which is of very high convenience.

The idea is to have an easy predefined way to constrain the linear systems with elements from the extracellular world, as we did in equation 1.55. As well, this avoids the stoichiometry consistency issue caused by cells having several biologically-determined transporters such as $H^+ \rightarrow \emptyset$ and $X + H^+ \rightarrow \emptyset$. Since the SBML format also provides a flexible definition for model compartments, exchange reactions as modelled by the COBRA team fit perfectly within SBML. We represent adding exchange reactions in Figure 2.10.

A subtility should be noted however, since we are at the boundary of the system, fluxes represent rates of mass going in or going out, therefore $H^+ \rightarrow \emptyset$ corresponds to mass going out, or secretion rates, while $\emptyset \rightarrow H^+$ corresponds to mass going in, or uptake rates. Exchange reactions are encoded as reversible reactions with the forwards direction being $H^+ \rightarrow \emptyset$ and the backwards direction being $\emptyset \rightarrow H^+$, allowing experimentally determined rates to correspond to consumption and production by the cell.

Adding exchange reactions is a very useful concept, but it should be kept in mind that it artifically increases metabolic network size, and thus those reactions must often be compressed together with the actual biologically-determined metabolite transporters when performing exhaustive metabolic pathways analysis such as EFMs (see section 2.7).

Finally, CBM models as defined by COBRA specify default flux bounds of $[0, 1000]$ instead of $[0, \infty[$ for every irreversible reaction and $[-1000, 1000]$ instead of $]-\infty, \infty[$ for every reversible reaction. This is because unbounded variables in linear programs cause unboundedness (see Figure 2.6) and numerical instability, and by bounding every variable's domain, this kind of errors are avoided. These flux bounds are directly encoded in the SBML file.

2.6 Flux Balance Analysis and variants

Briefly, Flux Balance Analysis (FBA) can be explained as a method allowing one to obtain an optimal point, or optimal flux distribution, on the stoichiometric flux cone, through the usage of linear programming, with an appropriate objective function. This is illustrated on Figure 2.8. Additionally, FBA applications should respect flux bounds as defined in constraint-based models (CBM).

Flux Balance Analysis was introduced in the mid-1990s by the laboratory of Bernhard Palsson [111], who is now at University of California, San Diego (UCSD). The major regroupment of CBM and FBA methods, which was started by Palsson, is now called the COBRA toolbox, for Constraint-Based Reconstruction and Analysis methods toolbox, and it is open-source. It is implemented in mainly MATLAB [112], then Python [72] and Julia [S83], and it is currently maintained by Ronan Fleming from National University of Ireland, Galway, Nikolaus Sonnenschein from the Technical University of Denmark, and Nathan Lewis from UCSD [W12].

Due to its implementation in MATLAB© [W5], the COBRA MATLAB toolbox [112], as well as additional wrappers around it, such as the RAVEN toolbox from Chalmers University of Technology, Göteburg [S84], are hard to use for truly open projects and outside of collaborations with MathWorks©.

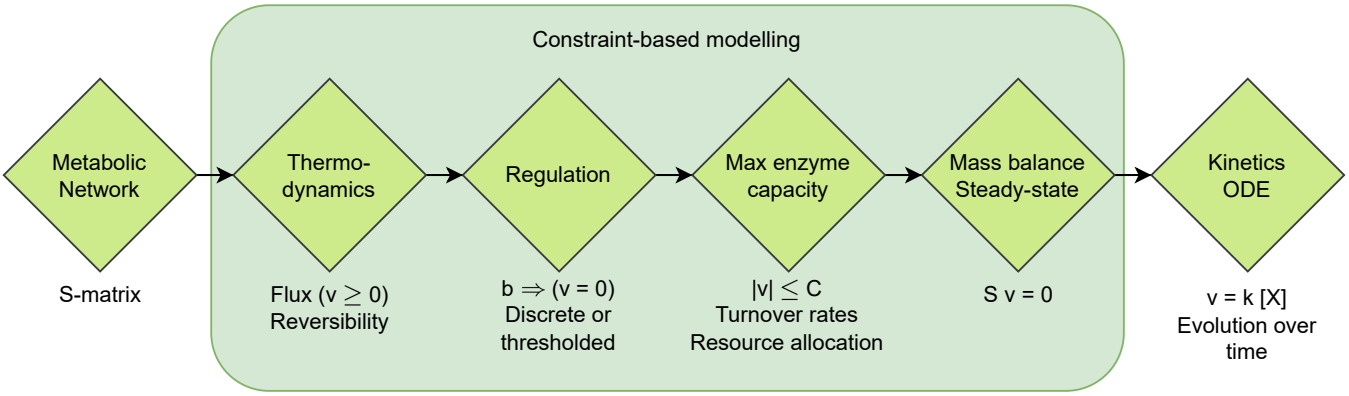


Figure 2.9: Constraint-based modelling in a nutshell: Addition of constraints

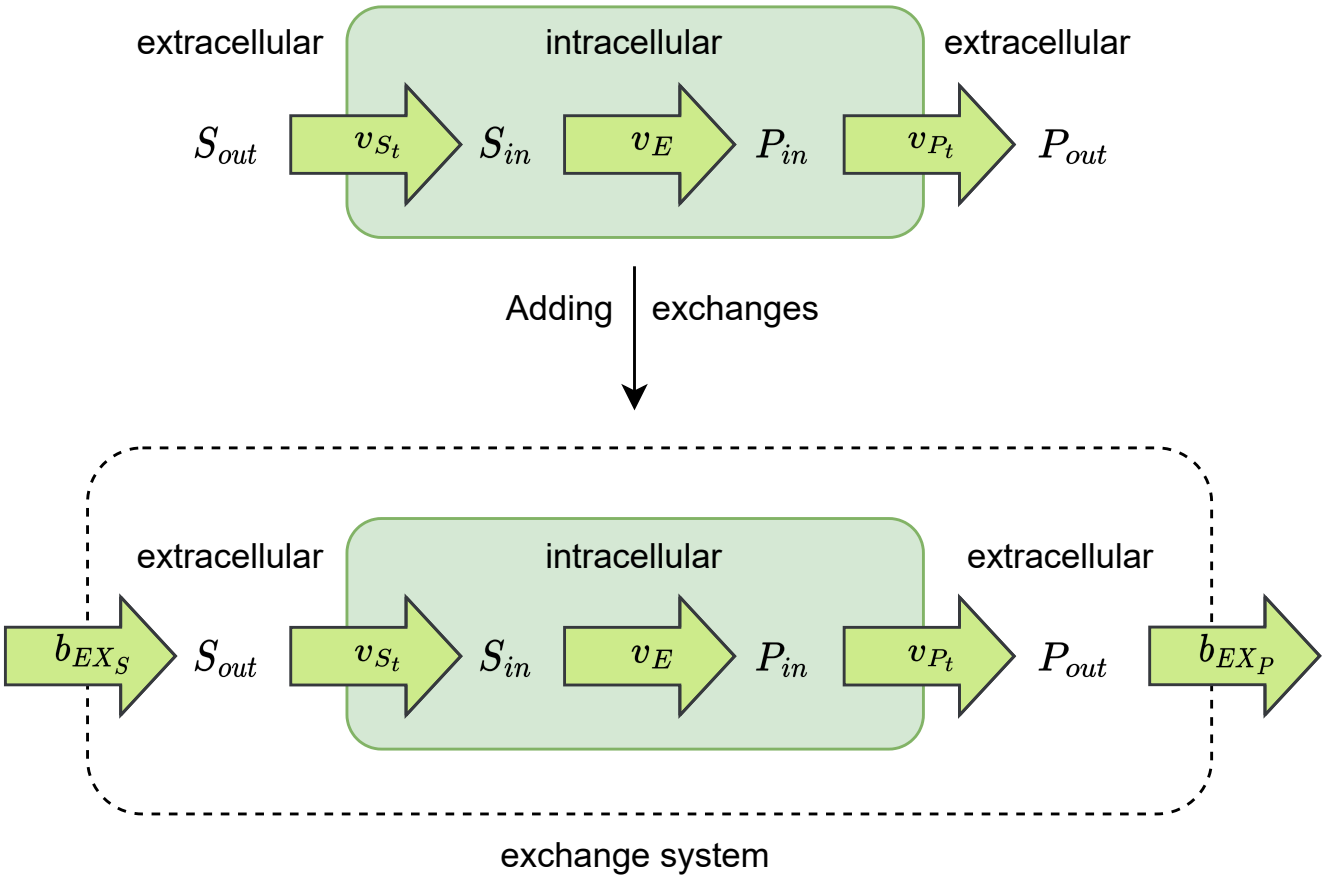


Figure 2.10: The COBRA approach: adding exchange reaction to fluxes

A great advantage of the COBRA toolbox in Python [72] is that depending on the problem chosen, it selects the most appropriate solver according to their computations, between cplex, gurobi, scipy and glpk. To do so, it uses a wrapper library called optlang, of great utility [S85]. However, depending on the applications, additional solvers not in optlang might be used. An example of this is performing geometric programming to get metabolite concentrations of an FBA or EFM solution, this might require the use of CVXPy [S78, S86].

The COBRA toolbox in Julia is the most promising compromise between the efficiency of MATLAB and the openness of Python, but it still lacking features [S83]. Other concurrent toolboxes attempts exist, such as merlin [S87] and mewPy [S88] developed by the Centre of Biological Engineering, University of Minho, Braga. The unique standard for the encoding of constraint-based models is the SBML format [38].

Thanks to its ease of accessibility, COBRAPy is usually the tool of choice for FBA in constrained-based modelling and related methods. COBRAPy makes loading and rewriting of SBML models easy, using libSBML, and in accordance with the version 2 of the SBML Flux Balance Constraints (FBC) plugin [69], specifically made for constraint-based modelling models. The standard tool for editing such SBML FBC models is CBMPy [S89], which is a wrapper around libSBML [113] allowing for performing FBA and more. COBRAPy however is a lot more widely used, and quite a bit more convenient. COBRAPy performs its own quality checks, which are different from the ones done by libSBML [113], and does automatic changes to the model, signalled with warnings, something that libSBML on its own doesn't.

However, COBRAPy is not without bugs. On some occurences, reversibilities and bounds on metabolic models we retrieved have not been correctly defined. COBRA would either recorrect them, or prioritizing the bounds defined in the SBML over the reversibility Boolean toggle, instead of the reverse, although better behaviour would have been to return an error to the user inviting them to fix the inconsistency in the model. Conversely, libSBML conforms to a very well-defined SBML XML specifiation, as well as the SBML FBC plugin specification [69], providing more programatically correct and user-friendlier ways to signal clear modelling errors.

As well, one should be in fact careful as to specify the COBRAPy version one utilizes, as between versions there are different behaviours of loading SBML models, for example, there might be different handling on flux bounds, and forcing every flux bound to 1 000, or surprisingly, the bound might be automatically set to 100 000, when there are coefficients too small in the biomass. Since this is a community tool, issues such as this are most often documented online [W13].

In our case, for most of our FBA analyzes, COBRAPy was used [72], and for the linear programming solver, we almost always used IBM© cplex, which provides a free academic license with Python bindings [W8]. GurobiPy also provided us a free academic license, but we found its performance on LPs and especially MILPs to be lacking [W9]. As well, since the SBML specification is solidly well-defined, we made sure to store all modifications done with COBRAPy in the SBML model files, and tried to prevent COBRAPy from any automatic changes to the model on loading as much as possible.

We define Flux Balance Analysis (FBA) as the problem of maximizing the biomass production flux on a constraint-based model (Definition 2.6.1). The biomass synthesis reaction is a standard reaction in constraint-based models [114, 115, 116]; it is defined as detailed in subsection 1.4.4. FBA is usually performed with one or more carbon sources as substrates, in order to be in accordance with well-known microbial growth data, but a complete growth medium might also be considered. FBA knows many applications, whether it is metabolic engineering, strain design, research of therapeutic targets, or simply theoritical biology research.

In particular, the first and most well-known application of Flux Balance Analysis was to predict the diauxic growth shift in *E. coli* in 1994 – by Varma and Palsson [27]. This was not solely FBA, in fact there was a dynamic simulation element to it, today the method is called dynamic FBA, or dFBA [117]. At each time iteration, the extracellular metabolite concentrations at the boundaries are recalculated, according to an ODE system. Variations of this algorithm were made by Markus Covert by adding transcriptional regulation (rFBA: regulated dynamic FBA) [118].

Nowadays the dFBA algorithm is implemented in COBRAPy [72], however the implementation in that code is very slow, and susceptible to errors, due to failing to find correct initial value conditions [S90]. A significantly faster Python implementation based on a more recent algorithm [S90] was published by Tourigny in 2020 [S91].

Definition 2.6.1 – Flux Balance Analysis

Flux Balance Analysis is the name given to the standard application of Linear Programming to Constraint-Based Modeling. It is a method originally developed for bacterial strains. The standard objective function is biomass growth, and the standard flux unit should be $mmol \cdot l^{-1} \cdot h^{-1} \cdot cDW^{-1}$. The classic biomass-optimizing linear program is given below, considering a constraint-based model (CB). The method finds optimal values to fluxes of all reactions such that biomass production is maximized.

$$\begin{aligned} &\text{Maximize } v_{biomass} \\ &\text{Subject to: } Sv = 0 \\ &LB_i \leq v_i \leq UB_i \\ &\forall i = 1, \cdots, n \end{aligned} \tag{2.14}$$

Biomass production, or maximizing product output for strain optimization, are hardly the only applications of FBA. We provide here a review of many different kinds of alternate FBA objective functions by Robert Schuetz [119].

In general, Flux Balance Analysis comes with its lots of variants. First, Flux Variability Analysis (FVA) deserves a mention. FVA consists in testing flux bounds permitting the production of at least $\mu\%$ of biomass for every reaction [120]. This is a very useful method as we know reactions which are detected as having a zero lower bound and upper bound flux for production of biomass could be for example removed from the model. Next, we should mention flux sampling [121], randomly generating solutions from the LP solution space. Several algorithms for flux sampling exist, and COBRAPy in particular implements two of them: OptGP [122] and ACHR [W14].

Finally, we should mention parsimonious FBA [123]. Parsimonious FBA (pFBA) is a bilevel linear programming procedure that is used to counter the problem of finding multiple optimal solutions [124]. Indeed, on top of finding the maximal biomass-producing solution, pFBA asks the solver to find the solution with that optimal value with the smallest sum of reaction fluxes. This returns FBA solutions with a small number of non-zero reaction fluxes, *i.e.* a small support. Pseudo-code algorithms for FBA and four of its variants are proposed in appendix[1].

Note that unfortunately, most often in FBA the LP system is underdetermined, meaning that it has more variables than constraints. This is due to the solution space being underconstrained. However, as par with "constraint-based modelling", by adding constraints the system's underdetermination can be overcome [95, 91]. Additionally, the system being underconstrained also means problems with multiple optimal solutions as shown in Figure 2.2 are very common. This is actually very problematic in FBA, since often that very optimal solution is the one used as the end result. This is the main reason why we consider FBA analysis to yield unsatisfactory results, and why we turn to EFMs analysis. Fortunately, the flux distributions retrieved with EFMs are unique.

2.7 Elementary Flux Modes

Contrarily to Flux Balance Analysis (FBA), Elementary Flux Modes (EFMs) define a finite set of many possible extremal solutions. These solutions are unique and of minimal support, meaning they correspond to minimal flux pathways instead of an optimal linear combination of several minimal flux pathways for FBA.

Definition 2.7.1 – Elementary Flux Modes

As explained in section 2.4, with the steady-state assumption, the set of flux distributions v is included in the null space of $S \in \mathbb{R}^{m \times r}$ the stoichiometry matrix. However, we cannot simply compute the basis of the null space, as we also have to consider reaction reversibilities.

We denote by C the set of vectors from the stoichiometric matrix which respects directionalities of reversible and irreversible reactions. C is a polyhedral convex cone.

$$C = \{v \in \mathbb{R}^r \mid Sv = 0 \text{ and } \forall j \text{ irreversible } v_j \geq 0\} \tag{2.15}$$

Elementary Flux Modes are then the solutions of C of subset-minimal support:

$$E = \{e \in C \mid \nexists e' \in C \; Supp(e') \subset Supp(e)\} \tag{2.16}$$

These solutions are all unique: they can be expressed by their support, which defines a Boolean relationship: non-zero flux or zero flux. The stoichiometric coefficients of an elementary mode can be computed solely from its support's binary pattern and the stoichiometric matrix [125, 126].

[1] See the following: Algorithm A.1, Algorithm A.3, Algorithm A.2 and Algorithm A.4.

Let us re-define the example of metabolic network in Figure 2.7. Let us present its Elementary Flux Modes derived from its stoichiometry matrix in Figure 2.11 and Figure 2.12. Reactions in red in Figure 2.12 are those with a non-zero flux, indicating the support of the EFMs.

$$S = \begin{matrix} T1 & T2 & T3 & T4 & R1 \end{matrix} \begin{pmatrix} 1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & -1 & 1 \end{pmatrix}$$

$$E = \begin{matrix} T1 \\ T2 \\ T3 \\ T4 \\ R1 \end{matrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Figure 2.11: Modes élémentaires de flux d'un réseau métabolique simple

Since EFMs are solutions of minimal support, there exists no two EFMs with the same support. Thus, we can denote EFMs by their support, in the following way: $M1 = \{T1, T3\}$, $M2 = \{T1, R1, T4\}$, $M3 = \{T2, -R1, T3\}$, $M4 = \{T2, T4\}$.

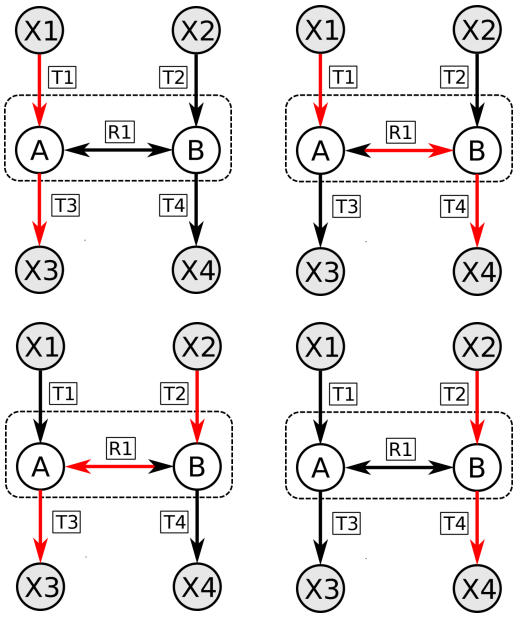


Figure 2.12: EFMs of a toy model network of 5 reactions

In the case where all reactions are irreversible, the cone C is zero-pointed and Elementary Flux Modes E are the extreme rays of the cone (Figure 2.8). For computation performance, splitting reversible reactions into two irreversible reactions is therefore recommended. By definition of the set of EFMs E, any support, set of reactions s, that is a strict subset of an EFM $Supp(s) \subset Supp(e)$ with $e \in E$, cannot define a flux distribution belonging in the flux cone C. Trying to find such a flux distribution with respect to the solution space's LP constraints will thus result in an infeasible program (Figure 2.5). This is insightful for the development of computational methods to calculate EFMs.

Elementary Flux Modes are not the only mathematical objects of interest on the stoichiometry flux cone. We compiled these supplementary notions in Definition 2.7.2. As a note, since reversible reactions are split during the computation of EFMs, one must ensure no flux goes into both the resulting irreversible reactions at the same time. As well, post-processing treatment is required to convert flux in split reactions into the flux of a single reaction.

Definition 2.7.2 – Other mathematical objects of interest

In his thesis, Marco Terzer, developer of EFMTOOL, separately defines EFMs from minimal generators and extreme pathways [127]:

— First, minimal generators, convex bases of C, correspond to our definition of EFMs when reversible reactions are not split into irreversible ones: these are extreme rays of the unpointed cone. On the other hand, he only defines EFMs as extreme rays of the pointed cone, when reversible reactions are split into irreversible ones. Throughout this thesis, we only study EFMs in the case when reversible reactions are split into two, thus staying true to Terzer's definition, on which property Theorem 2.7.1 is said to apply.

— Secondly, extreme pathways are extreme rays of a particular cone where only exchange reactions are allowed to be reversible, while internal reversible reactions are split into irreversible ones, where a non-negativity constraint for the variables applies.

The latter, extreme pathways, is notably used by the Palsson team, such as on the *E. coli core* network [91]. EFMs are in fact a superset of extreme pathways [127].

Related concepts to EFMs also include Generating Flux Modes (GFMs) [128], Elementary Flux Vectors (EFVs) [129], and the recent Elementary Conversion Modes (ECMs) [130].

Elementary Flux Modes present a particular property: any FBA solution can be linearly decomposed into EFMs (see Theorem 2.7.1). This is not always the case with the other mathematical objects detailed in Definition 2.7.2: *extreme pathways* and *minimal generators* [127]. This property is one of the major reasons why one should still be interested in EFMs, despite the rise of faster methods in computation times such as Elementary Conversion Modes [130]. Another major property is its relationship to Minimal Cut Sets, which we detail further in section 2.12.

Theorem 2.7.1 – Decomposability of FBA solutions into EFMs

Let v be any flux vector of C – that respects directions of reactions. Then, there exists a set of elementary flux modes $\{e_1, \cdots, e_k\}$ such that v can be linearly decomposed solely from EFMs:

$$v = \sum_{i=1}^{k} \lambda_i e_i \quad \{\lambda_1, \cdots, \lambda_k\} \geq 0 \tag{2.17}$$

In particular, Jean-Marc Schwartz proposed an algorithm to decompose flux solutions such as the ones obtained with FBA into linear combinations of EFMs [131, 132].

Finally, since EFMs are vectors of the nullspace of the stoichiometry matrix S, a special property can be used to check whether a flux vector is truly an elementary mode or not. This is called a kernel test or rank check, and it is performed by verifying the S-matrix's rank as presented in Theorem 2.7.2. Rank checks can easily be performed using Numpy's SVD-based rank calculations [S70], or the slower but more exact Gaussian elimination method.

Theorem 2.7.2 – Rank check for EFMs

In order to know if a flux vector v of support $Supp(v)$ is indeed an EFM, one can check if the kernel of the submatrix $S^{Supp(v)}$ — the stoichiometric matrix indexed by its support, that is, the S-matrix where columns j for which $v[j] = 0$ were removed — is of dimension 1 [125].

Then, by the rank-nullity theorem, one can instead check whether the rank of that submatrix is equal to $|Supp(v)| - 1$. If it is equal, then it is an EFM, if it is not, then it isn't. This is called the rank check.

2.8 Methods for EFM Computation

The computation of EFMs presents many challenges. Traditionally, the algorithm used to compute Elementary Flux Modes is Double Description (DD) [133]. It is an efficient algorithm based on matrix computations. However, the number of EFMs increases with the size of the metabolic networks, when networks are large-scale – over 100 reactions – the more there are EFMs and it becomes less practicable to enumerate all solutions – there is a combinatorial explosion in the number of solutions. Indeed, counting the number of EFMs has been proved to be #P-Hard [134, 135]. Therefore it is at the moment impossible to compute EFMs on the so-called genome-scale networks which may have – in the most extreme cases – up to 10 000 reactions [37].

Another related problem to the computation of EFMs is finding the EFMs of interest. Indeed, on metabolic networks, among the many EFMs retrieved, only a small part is truly observed in lab experiments. The majority of elementary modes found are not credible when relevant biological constraints are integrated into the computation. Therefore scientists proposed to directly integrate constraints during the calculations, rather than filtering the biologically-infeasible EFMs after enumeration was finished. This is a lot less costly in both computation time and memory requirements.

To directly integrate constraints, new methods arised, in particular using linear programming, we refer to those as LP-based tools. These methods take advantage of the fact that any first EFM can be found in polynomial time with a Linear Program [134]. As a result, these methods can enumerate biologically relevant subsets of EFMs on the fly on very large-scale networks – even with over 1 000 reactions, which was previously not thought to be possible. Whereas Double Description struggles to deal with networks with over 100-300 reactions – complete enumeration even with constraints is too long and needs to be interrupted.

We illustrate the dichotomy between complete enumeration and constrained enumeration on a toy network example in Figure 2.13. Here the biologically-determined constraint is asking for transport reaction $T2$ to be present. Using a complete enumeration algorithm such as DD would necessitate one to enumerate all 4 EFMs of the network, then retrieve the 2 most at right ones, which contain $T2$. In contrast, using LP-based algorithms, one could get directly the EFMs containing reaction $T2$, as part of the full computation procedure, without ever enumerating the other two.

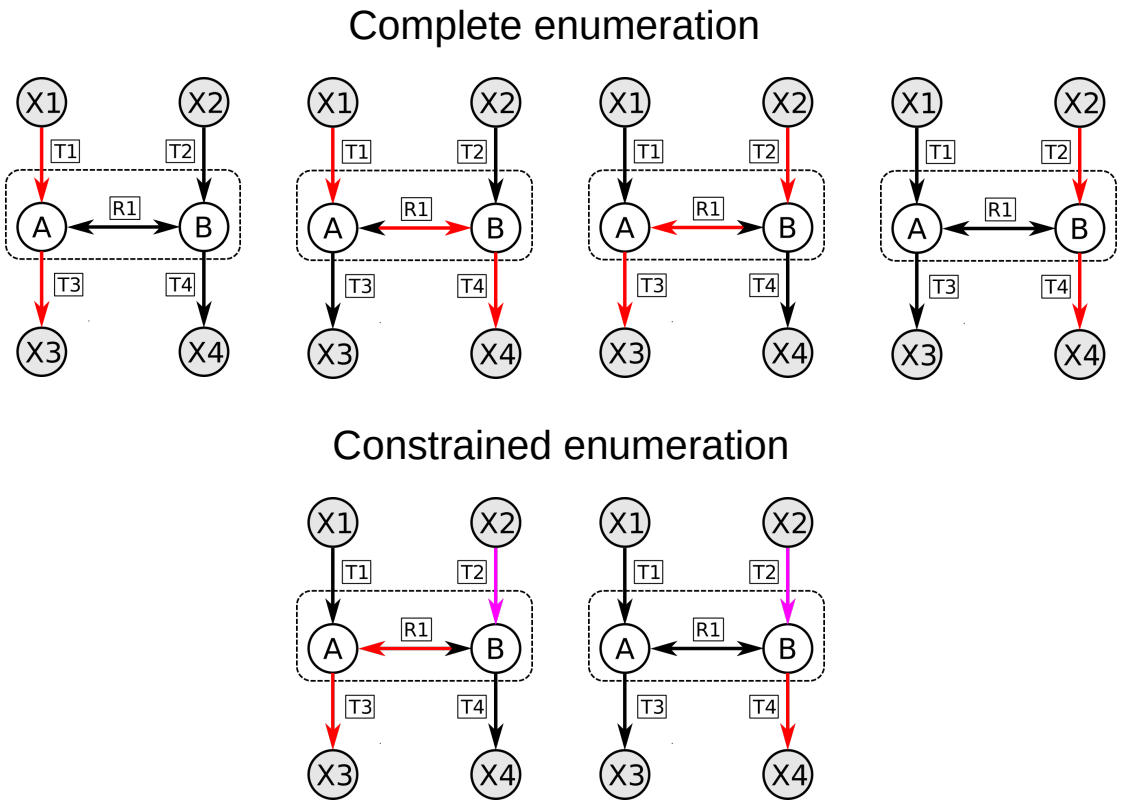


Figure 2.13: Complete enumeration approach vs. Constrained enumeration approach

EFMs computation is implemented in the well-known tools METATOOL [109, 136], ScrumPy [137], CellNetAnalyzer [138], COPASI [S92] and most importantly EFMTool [139], published in 2008, which performs Double Description. METATOOL, published in 1999, performs a method utilizing Gauss-Jordan elimination [109]. In 2004, Gagneur and Klamt stated that all known enumeration algorithms back then were variants of the Double Description [126]. No matter the algorithm, the EFM tools listed here perform *complete enumeration* with limited possibility of filtering. By contrast, linear programming-based tools and therefore *constrained enumeration* tools were most popularized by de Figueireido's paper in 2009 [140].

2.8.1 Double Description

Double Description (DD) is an algorithm permitting enumeration of extreme rays. It was proposed by Motzkin in 1953 [141], and revisited in 1994 by Fukuda [133]. It is a widely used algorithm, implemented in EFMTool developed by Marco Terzer [139]. In his thesis, Terzer describes the Double Description algorithm as is presented in Algorithm A.5.

It is an incremental algorithm – and the later iterations are the ones where the combinatorial exploration is the most problematic. We refer to Terzer's thesis and Morterol's thesis for details [127, 142].

The main drawback of the DD method is that enumerating all solutions and filtering the ones of interest afterwards is not convenient. For example, Jungreuthmayer, developer of regEFMTool, an extension of EFMTool, computed all EFMs of the *E. coli core* network [143]. These were said to take 259 GB and 31h to compute. They added regulations, reducing the computation time in pre-processing and during the algorithm for negative constraints that could be included, and reducing the number of EFMs post-processing. Despite this, computation still took 7.1 hours and the researchers obtained over 2 million EFMs to be analyzed further.

Another method based on a network splitting algorithm allowed the computation of $\sim$ 2 billion EFMs from a large metabolic model of microalga *Phaeodactylum tricornutum* consisting of 318 reactions [144]. But in comparison, many genome-scale networks have on the order of thousands of reactions. Thus, it is thought to not be currently possible to enumerate all EFMs from genome-scale metabolic models.

Using DD is a major inconvenience, as in practice one would be interested by EFMs regardless of their number in the network. For example, one might be interested in the specific EFMs that decompose a certain FBA solution. Or, alternatively, one would want the EFMs corresponding to optimal yield rates.

So, an inherent problem is finding EFMs of interest from the large solution set. Among the many EFMs computed by DD, only a small fraction are thought to be active in cells. To save computational time and memory and to focus on biologically relevant phenotypes, it becomes necessary to integrate biological constraints directly during the computation of EFMs.

More often than not biological constraints cannot be added to DD. While Jungreuthmayer succesfully added transcriptional regulation to DD [143], and while Peres *et al* proved that thermodynamic equilibrium constraints could be added to DD [145], flux yield constraints for example, are not to our knowledge known to be integrable into DD.

Other emerging extreme ray enumeration tools applied to EFMs include lexicographic reverse search [146]. In general, advances in the field of enumeration of polyhedron vertices, polyhedron facet enumeration, polyhedron convex hull computation [147, 148] might be related to advances in cone extreme ray computation, as extreme ray enumeration is simply a particular case[2] of vertex enumeration [133].

2.8.2 Linear Programming-based tools

In 2009, de Figueireido *et al* [140] proposed a method enumerating the shortest elementary flux modes with Mixed Integer Linear Programming (MILP). The method is named k-shortest EFMs, and it enumerates the smallest EFMs until iteration k, where k is the number of reactions of the EFM.

Let us consider a metabolic model $M = (Mets, Reac, Stoch, Rev)$, where $Mets$ denotes m *internal* metabolites, $Stoch$ denotes the stoichiometry matrix $S \in \mathbb{R}^{m \times r}$ for these internal metabolites, with s_{mr} denoting stoichiometry

[2] Vertex enumeration applies on the polyhedron of constraints $Ax = b$ while extreme ray applies on the cone of constraints $Ax = 0$.

coefficients, $Reac$ is the set of r irreversible reactions, obtained after the split of reversible reactions of the constraint-based model into two irreversible ones, Rev a set of pairs associating together irreversible reactions issued from the split of reversible reactions.

The 'k-shortest EFMs' MILP [140], providing linear variables for reaction fluxes v and Boolean indicator variables z, is given by the following equations:

$$
\begin{aligned}
&\text{Minimize} && \sum_{r \in Reac} z_r && \text{EFM size} \\
&\text{Subject to} && z_r \leftrightarrow v_r > 0 \quad \forall r \in Reac && \text{Indicator constraints} \\
& && z_{r_{fwd}} + z_{r_{bwd}} \leq 1 \quad \forall (r_{fwd}, r_{bwd}) \in Rev && \text{Handle split reversible reactions} \\
& && \sum_{r \in Reac} z_r \geq 1 && \text{Eliminate trivial solutions} \\
& && \sum_{r \in Reac} s_{mr} \times v_r = 0 \quad \forall m \in Mets && \text{Steady-state constraint} \\
&\text{Where} && z \in \mathbb{B}^r, v \in \mathbb{R}^r && \text{Domain of flux and indicator variables}
\end{aligned}
\tag{2.18}
$$

Each flux vector v retrieved with this formulation is an Elementary Flux Mode, minimizing support size.
And for each EFM solution retrieved with the MILP, the following exclusion constraint to the program is added:

$$
\text{Integer cut} \quad \sum_{i=j_1}^{j_P} z_j \leq P - 1 \;\; \forall \{j_1, \cdots, j_P\} = Supp(e) \qquad \forall e \in PreviousEFMs \tag{2.19}
$$

As we can see in equation 2.18 and equation 2.19, the 'k-shortest EFMs' MILP enumerates the minimal EFMs in terms of size, then for each EFM found it adds an integer cut as was defined in Definition 2.2.3. However, a major drawback can be found in that in its default setting, this MILP only iteratively enumerates the smallest EFMs first, while EFMs, subset-minimal solutions but not minimum in terms of size, can come in any size. The program would enumerate first solutions of size 1, then of size 2, then 3, etc. This problem of the method was however fixed by the Klamt team in CellNetAnalyzer [149], and its recent Python implementation CNAPy [138].

The method was revisited several times, including [150, 151, 138], and notably for other applications such as Generating Flux Modes (GFMs, subsets of EFMs) [152] and Minimal Cut Sets (MCSs) [153, 154]. Other methods based on Linear Programming also include Alternate Integer Linear Programming, a method that alternately computes EFMs and MCSs [155].

Furthermore, in 2014 and 2016, the problem of Boolean Satisfiability (SAT) and of Satisfiability Modulo Theories (SMT) were utilized for the constrained enumeration of EFMs [156, 157, 142]. These methods are also partly linear programming-based. The linear programming solving is taking place this time inside the Linear Real Arithmetic theory of the SMT solver. To distinguish this type of constrained enumeration methods from MILP-based methods, we decided to devise the term Logic Programming with Linear Constraints, abbreviated to LoPLC.

The method we developed during the course of this thesis, *aspefm*, to compute EFMs and MCSs, is an example of such a LoPLC method. It is based on logic programming with linear constraints, especially, Answer Set Programming logic programming [89] (see chapter 3).

2.9 Genome-scale modelling

Early on, one of Palsson's ambitions was bringing metabolic models to genome-scale [95, 94, 54]. What this means is having metabolic models reconstructed from a complete genome, reference genome sequences as one can find on the RefSeq database [158] from NCBI [W1]. From a reference genome, one might find annotated coding sequences (also known as CDS), which might be translated into proteins, which might then correspond to enzymes. The idea is to scale the metabolic models to all enzymes predictable from the genome.

In order to find the corresponding enzymes from the reference genome, one might map the genomic functions to enzymes from encyclopedias such as KEGG [35, 71], which contain many metabolic maps. In addition, RefSeq sometimes contain mentions of E.C. numbers, easily linked to KEGG entries [15]. As previously mentioned, there are now many database portals referencing enzymes which contain a large coverage of many organisms, helping the analysis of reference genomes (such as [36] and [S37, S38]). For the remaining proteins derived from coding sequences (hypothetical or unannotated), one might look at the process of gap-filling [159, 160], based on sequence homology with phylogenetically close organisms: seeing what enzymes are present or absent in the metabolic model, which gaps are retrieved in otherwise linear pathways, fill the gaps with an enzyme that is present in a very close homologue taxon, for example. This complete process is called metabolic network reconstruction [71].

In this section, we will be focusing on the major advances made by the Palsson group, the SBML format, and what this means in terms of integration of biological constraints in constraint-based models. A major application of Flux Balance Analysis is being able to provide phenotypic predictions [71, 161]. In a constraint-based metabolic model, only external metabolite concentrations and flux bounds can vary, therefore if we want predictions close to reality, integration of biological constraints is needed [95].

A first approach at integrating biological constraints by Palsson team was realized by Covert, Schilling and Palsson in 2001 on a toy metabolic model [118]. This approach integrates a Transcriptional Regulation Network (TRN), a set of regulation rules representing interations between genes and transcription factors, usually of the type activation or inhibition. Transcriptional Regulation Networks are Boolean Networks (see subsection 1.6.4) that can be used in dynamic analyzes, in particular, dynamic FBA [111], and dynamic regulated FBA (rFBA), described in their paper [118]. We briefly described dynamic FBA and its algorithm in section 2.6. Moreover, Covert also used the TRN as constraints for the EFMs computation, and their study was a major motivator for studying TRNs in our analyzes with our tool *aspefm* [93, 89] . We will develop the TRNs and their utility further in section 3.6.

The dynamic rFBA method is then applied to *E. coli* at the genome-scale, considering genomic / gene-protein data and transcriptional regulation data from literature of many papers, in a very extensive work, in 2002 [94]. Here is presented an example of three transcriptional regulation rules from the metabolic model of Covert and Palsson:

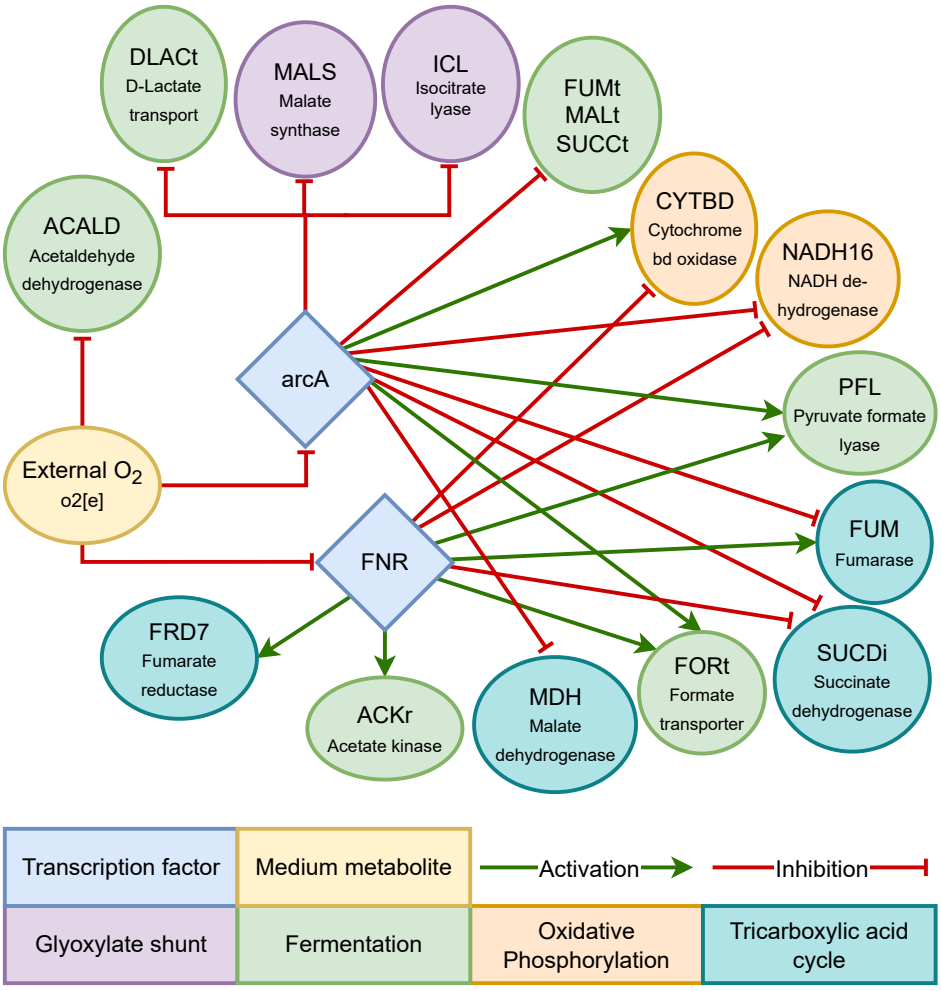


Figure 2.14: Example TRN rules of *E. coli* core. We refer to Orth *et al* for a visualization of the complete TRN

$$FUMAR \implies \neg\,(ArcA \vee FNR)$$
$$FUMBR \implies FNR$$
$$ADHER \implies \neg\,(O2xt \wedge FruR)$$

With the reactions $FUMAR$ and $FUMBR$ being two isozymes, homologue proteins coded at different loci in the genome, catalyzing the fumarase enzymatic function. Here, each reaction can be regulated ($\Rightarrow$) by transcriptional regulators or external metabolites according to Boolean formulas composed of NOT ($\neg$), AND ($\wedge$) and OR ($\vee$) [3]

The authors then construct a complete metabolic network of over 1 000 reactions at the genome-scale of *E. coli*, integrating transcriptional regulation networks, and transcriptomic data obtained with microarray, which they use to predict new phenotypes. Covert's article is published in Nature in 2004 [73].

[3] Color code: Reactions, Environment metabolite, Regulator protein, Genes

A nice review of genome-scale models published back then is presented in [110]. According to the paper, the first instances of genome-scale models reaching around 1 000 reactions include *E. coli* str. K12 in 2003 [S93] and *S. cerevisiae* in 2003 and 2004 [S94, S95]. Covert then publishes an extended model where kinetics modelling is included on top of transcriptional regulation in 2008 [90].

In 2010, Jeffrey Orth, Ronan Fleming and Bernhard Palsson published a guide for the reconstruction of genome-scale models, including getting enzymes from references genomes, but also transcriptional data. The example utilized was a model of central carbon metabolism of *Escherichia Coli*, which we refer to as *E. coli* core [71].

Orth also dedicates a part of a guide about reconstruction of transcription regulation networks, based on ChIP-chip data [S44]. Unfortunately, despite the guide, as of today many models still lack TRNs. This can perhaps be explained by scientists realizing the inherent complexity of transcriptional regulation thanks to the rise of transcriptomics data with the new RNA-Seq method in the 2010s.

On the bright side, this means that the *E. coli* core model (Figure 2.15) is one of the few that possess a transcriptional regulation network (Figure 2.14) [71]. Instead of containing a reaction for each isozyme like the model from 2002, this model defines a Boolean relationship between genes and reactions, and transcriptional regulation constraints are applied later, but only on genes. This allows one to considerably decrease the number of reactions of the model.

$$FRD7 \implies (frdA \wedge frdB \wedge frdC \wedge frdD)$$
$$FUM \implies (fumA \vee fumB \vee fumC)$$

Here we can see that *fumarase A* and *fumarase B* are this time encoded by a OR. While OR represent isozymes, AND, in contrast, represents different subunits of the same enzyme complex. Note that in such associations between genes and reactions, the NOT ($\neg$) is forbidden, only AND ($\wedge$) and OR ($\vee$) are allowed. These Boolean associations between genes and reactions went on to be called Gene-Protein-Reaction associations; or rules, denoted GPRs [162]. Something to be noted is that this notation might either represent a gene or a protein, depending on the context, similarly to how Uniprot's 'one gene = one protein' paradigm defines entries encompassing both possible types. Sometimes, the appelation gene-product is used to refer to these elements representing both the genes and the proteins. Although similar-looking, GPRs rules are not to be confused with TRNs rules.

The rise of such constraint-based models allowed a standard measure of genome-scale models quality to develop: gene essentiality prediction[4]. Using GPRs, a model is considered good quality if it is able to accurately predict *in silico* the majority of lethal knock-outs retrieved *in vitro*, assuming lab experimenters have performed knock-outs of each known gene of an organism. An example of study combined with experiments which uses such a metric is [163].

[4] An extension of the computation of essential reactions is the computation of Synthetic Lethals (SLs) presented later in section 2.11.

Nowadays there exists many genome-scale models. They are repertioried in part on the database BiGG Models [37] or on the website BioModels [39] in SBML format (Systems Biology Markup Language) [38]. Note that BioModels models are not all manually curated as some of them are straight out of the automation pipeline. The advantage from using BiGG models is also that they require to conform to a specific way of doing reconstruction, and the models there all have GPRs rules, which are now essential for modelling, especially for integration of omics data.

Various studies linked before describe how integration of omics data is possible in genome-scale models thanks to GPRs [59, 61], and indeed, since these Boolean relationships allow access to a level representing both genes and proteins, one might be interested into integrating genomic, proteomic, transcriptomic data, and many more [31].

All models on the BiGG database are provided in SBML format, and as it turns out in its latest SBML version with the FBC plugin also used for flux bounds, SBML defines a native encoding for GPRs relationships [69], as XML Boolean formulae. The corresponding SBML tags attached to each reaction are **<fbc:and>**, **<fbc:or>** et **<fbc:geneProduct>**. No standard encoding for TRNs was developed however.

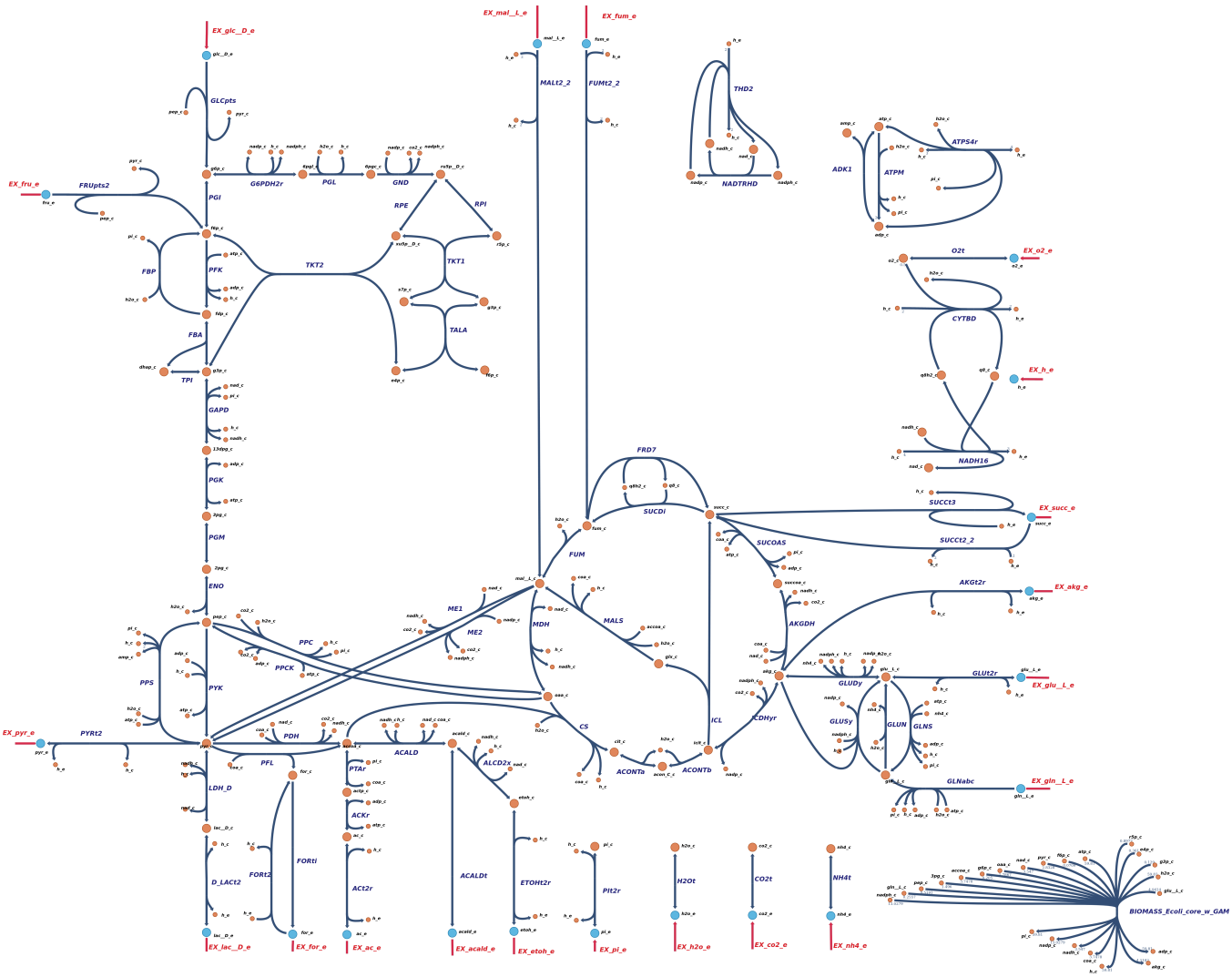


Figure 2.15: Escher view of the *E. coli* core metabolic model (Orth, Fleming, Palsson, 2010)

Definition 2.9.1 – Gene-Protein-Reaction association rules

Gene-Protein-Reaction association rules, or GPRs, are *positive monotone* Boolean functions, *i.e.* they do not contain any NOT operator [S96], and they define the relationship between an enzymatic function and its associated enzymes.

$$\text{ex:} \quad (b_1 \wedge b_2) \vee (b_1 \wedge b_3) \qquad \text{two isozymes which share a subunit in common} \tag{2.20}$$

An OR relationship symbolizes isozymes while an AND relationship symbolizes an enzymatic complex. These are said to be at the Gene-Protein level, or gene product level, since although they represent enzymes which are proteins, if one were to disable genes, then one would have to deactivate the Boolean function – or in other words – find an assignment such that the Boolean function becomes $\{False\}$.

A little known fact about BiGG models is that the uppercase letters in the model's abbreviated names correspond to the modeller's initials, while the numbers correspond to the number of genes in the model, since these are models reconstructed from annotated reference genomes. For example, iSB619, a *Staphylococcus aureus* model from 2005, is named this way since it was developed by Scott Becker, and it contains 619 genes [164]. As well, *iYS854*, a *Staphylococcus aureus* model from 2019, was made by Yara Seif, and contains 854 genes [163]. The number of genes in a genome-scale model simply refers to the number of gene products, associated to reactions via GPRs.

Note that GPRs as defined in Definition 2.9.1 are not the most accurate representation of protein subunits as it lacks for many properties commonly found in biology, such as, most notably, homodimers, and other complexes made of the same subunit several times. For examples, in the GPR for fumarase FUM, there is no mention that the gene $fumA$ should be encoded twice and form a homodimer complex to perform the fumarase enzymatic function, although this is experimentally observed [S97].

Therefore, people have proposed to further complete the GPRs formalization by including subunit quantity for each gene, and also to include ions which often serve as cofactors in enzymatic complexes. Although, from just a gene perspective when considering knock-outs, and not a protein perspective, maybe that information might not be needed. This is discussed in an issue on the COBRAPy toolbox GitHub [W15], and we might hopefully see changes in later specifications of the SBML FBC plugin.

In the meantime, there exists more complex models of the proteins involved in GPRs, and more complex frameworks retrieving proteic data from reference genomes, to perform extensions of the FBA formalism, but taking into account proteomic data. An example of such a method is Resource Balance Analysis (RBA), which considers the enzyme resource usage of every enzyme in its FBA-derived problem, and to do so, it requires knowledge of enzyme complexes composition for every gene associated to the genome-scale metabolic model [165, 166, 167]. Their Python API, RBAPy, automatically downloads data from UniProt and computationally generates an extended genome-scale model where information about cofactors and subunit composition is present [168].

Omics	
Genomics	constraints on GPRs, regulation, knock-outs through GPRs
Transcriptomics	constraints on GPRs, modulating enzyme quantity
Proteomics	constraints on GPRs, modulating enzyme quantity, enzyme quantity constrained by total enzyme quantity
Metabolomics	constraints on external metabolite concentrations, exchange fluxes, internal fluxes, allows thermodynamics checks
Fluxomics	constraints on flux bounds, possibly metabolite concentrations
Non-omics	
Transcriptional regulation	knock-out of GPRs if actively inhibited by regulation
Thermodynamics	constraints on flux bounds, metabolite concentrations
Yields *Growth rates*	constraints on flux bounds, biomass reaction modifications GPRs may be used for resource allocation at different growth rates
Kinetics	constraints on flux bounds, GPRs through modulating enzyme quantity

Table 2.1: Integration of omics and non-omics data in genome-scale models.
Highlights the importance of GPRs if they are to be developed further in the future.

The equivalent Palsson team iteration of RBA is called Metabolism-Expression models or ME-models, a combination of metabolic modelling and constraining the enzyme quantities by a total enzyme pool, modelling protein expression[5], and therefore requiring protein concentrations for every enzyme, in order to integrate omics data [S98, S99]. Once enzyme concentrations and metabolite concentration data are known, further calibrations of the genome-scale models might be done, such as by performing thermodynamic and kinetic calculations.

In conclusion GPRs describe which enzyme and which coding gene are involved in an enzymatic reaction. Whether or not GPRs are to be extended to resource allocation problems in the future, we illustrate how they could be used to impose constraints for most if not all omics kinds of data in Table 2.1. Ultimately, constraints imposed on GPRs imply the application of constraints on flux bounds, and therefore by integrating omics data onto GPRs we would obtain flux distributions that are closer to biological reality.

Finally, a nice review of resource allocation with ME-like and RBA-like models, together with transcriptional regulation models including dynamic rFBA-like methods is presented in [S100]. Genome-scale models know many applications, and we refer the reader to review papers for further information.

2.10 Model curation and compression

As the size of metabolic models grew, models could not be curated purely through manual work, and thus methods for model curation and model compression were needed. Notably, since most genome-scale models are not manually curated after their automatic generation, they might contain a large number of inconsistencies.

[5] In RBA-models, transfer RNAs are also needed for protein translations. This is part of the encoding of the translation process. Acyl-transferase reactions charging amino acids with tRNAs are now provided in modern genome-scale models.

In recent years, efforts of the community have been made towards tools correcting inconsistencies, most notably in the domain of gap-filling as mentioned before, but also towards possible stoichiometric coefficient inconsistencies, inaccurate definition of biomass, blocked reactions and dead-end metabolites – meaning they cannot bring flux, lack of GPR associations, and lack of proper SBML annotations. Such a tool encompassing all of these possibilities of error is MEMOTE, the community tool for assessing the quality of metabolic models [169].

2.10.1 Curating annotation errors with MEMOTE

MEMOTE only takes a SBML model in input, and rates its quality through a percentage score. In practice, models rarely get positive scores, unless they are developed by people with expert knowledge of the COBRA toolbox, that is [170]. SBML in COBRA specification should contain annotations (in RDF format) of all GPRs with their reference genome from which it was reconstructed from. They also should have reaction, metabolite, gene identifiers that are repertoried on the BiGG databases [37]. BiGG identifiers are the new standard, providing comprehensible standard names for all reactions and metabolites, regardless of the organism of interest [37]. When BiGG standard is not applied, one might be confronted with models with nonsensical numerical identifiers such as SEED compound identifiers [36], these models are – in my opinion – not possible to work with.

A large percentage of MEMOTE's reports are related to annotation issues and format specifications. Models which do not adhere to SBML FBC version 2 would have their bounds defined in annotations, as was the previous standard, or in other ways, such as obsolete FBA formats, or in the worst situations, no bounds are defined, and even no reversibility attribute are defined on reactions. Models such as these would get a poor MEMOTE score from this alone. Again this is also understandable, as the COBRA toolbox has trouble dealing with those models, and most often one needs to read the model and simply re-export the model into a by COBRA's SBML writing functions. We therefore recommend usage of SBML FBC version 2 for constraint-based metabolic models.

GPRs annotations also present some issues. In the legacy versions of COBRA SBMLs – before FBC v2 – GPRs could not be read programatically from the XML with libSBML, the submodule COBRA uses for parsing. This means libSBML alone is not enough to read old COBRA models: the COBRA toolbox should be used. As an example, the version we retrieved of HMR, the human model reconstuction with thousands of reactions across different cell lines, available on the Virtual Metabolic Human database, does not adhere to these standards [171]. A BiGG version is available, but it lacks reactions of some tissue-specific models. This is a shame, as again BiGG and SBML FBC v2 define very well suitable standards for the constraint-based modelling community.

Aside from identifiers and format specification problems, the issues of most interest to us in model curation are stoichiometric consistency errors, and errors which result from the definition of flux bounds. We will be covering the latter when discussing metabolic model compression – the process of reducing the number of reactions of the model, in subsection 2.10.4.

2.10.2 Stoichiometric consistency errors

A stoichiometric consistency error might be an error of the following form:

$$\begin{aligned} R1: &\quad A + B \rightarrow C \\ R2: &\quad C \rightarrow 2\ B + A \\ R3: &\quad C \rightarrow D \end{aligned} \tag{2.21}$$

This system of three reactions is inconsistent since in that case the mass conservation law between metabolites cannot be properly respected. Indeed, mass conservation law dictates that no matter is created from nowhere, but in that case, C is said to be stoichiometrically equivalent to both $A + B$ and $A + 2B$ at the same time, which cannot be correct. Thus, the system could be used to generate infinite amounts of B matter if this would be a dynamic model.

In our case, with the steady-state assumption $Sv = 0$, this will simply lead to a wrong computation of the reaction fluxes v. With this inconsistency, the system of equations $Sv = 0$ is described by:

$$\begin{aligned} [A]: &\quad 0 = -v_1 + v_2 \\ [B]: &\quad 0 = -v_1 + 2v_2 \\ [C]: &\quad 0 = v_1 - v_2 - v_3 \\ [D]: &\quad 0 = v_3 \end{aligned} \tag{2.22}$$

The solution space of that system is $S = \{v \mid v_1 = v_2 = 0\}$, essentially forcing fluxes of the two inconsistent reactions R_1 and R_2 to be null.

On the other hand, fixing reaction $R2$ to the backwards direction of $R1$, ie. $R2: \quad C \rightarrow B + A$, gives the following equations system:

$$\begin{aligned} [A]: &\quad 0 = -v_1 + v_2 \\ [B]: &\quad 0 = -v_1 + v_2 \\ [C]: &\quad 0 = v_1 - v_2 - v_3 \\ [D]: &\quad 0 = v_3 \end{aligned} \tag{2.23}$$

The solution space of that system is $S = \{v \mid v_1 = v_2\}$, which is a more permissive solution space allowing both reactions R_1 and R_2 to have non-zero flux.

Notice that in the first solution space, we have reactions which flux is always zero, *i.e.* blocked reations, and in the second solution space, we have reactions with always the same fluxes, linearly dependant reaction fluxes. We will come back to these characteristics later.

2.10.3 Detection of stoichiometric inconsistencies

Now, let us consider such a consistency error and a model defined by its stoichiometry matrix S. A model is defined to be stoichiometrically consistent if it respects the property defined in Definition 2.10.1.

Definition 2.10.1 – Stoichiometric consistency of a model

A metabolic model is said to be stoichiometrically consistent if and only if the following linear program admits a solution:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{m} m_i \\ \text{Subject to} \quad & S^T m = 0 \\ \text{Where} \quad & m_i \geq 1 : \forall\, i,\ 1 \leq i \leq m \end{aligned} \tag{2.24}$$

Where S^T is the transpose of the stoichiometry matrix S, extended with external metabolites.

In a consistent metabolic model, all metabolites are said to be conserved. An inconsistent metabolic model admits unconserved metabolites, which can be retrieved by solving the following MILP.

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{m} k_i \\ \text{Subject to} \quad & S^T m = 0 \\ \text{Where} \quad & 0 \leq k_i \leq m_i,\ k_i \in \{0,1\} : \forall\, i,\ 1 \leq i \leq m \end{aligned} \tag{2.25}$$

All unconserved metabolites are metabolites m_i for which k_i is equal to 1 in the MILP of equation 2.25.

These definitions were introduced by Gevorgyan, Poolman and Fell in their paper from 2008: *Detection of stoichiometric inconsistencies in biomolecular models* [172]. Gevorgyan *et al* also introduced more notions: the concepts of minimal net stoichiometries, and elementary leakage modes. These concepts are described in Definition 2.10.2. The methods were implemented into the versatile ScrumPY metabolic modelling software [137].

Definition 2.10.2 – Minimal net stoichiometries, elementary leakage modes

Considering $K = Ker(S)$ the kernel of the stoichiometric matrix, a minimal net stoichiometry associated to an unconserved metabolite u is a vector y such that y is a solution to:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{m} k_i \\ \text{Subject to} \quad & y^T K = 0, \;\; y_u > 0 \\ \text{Where} \quad & 0 \leq y_i \leq k_i, \; k_i \in \{0,1\} : \forall \; i, \; 1 \leq i \leq m \end{aligned} \tag{2.26}$$

Adding integer cut constraints for each solution found helps one to find all minimal net stoichiometry for a single unconserved metabolite.

Then, the computation of elementary leakage modes is recommended. Elementary leakage modes should be retrieved on an augmented matrix $(S|y)$, the stoichiometry matrix S where the minimal net stoichiometry y is added as a new reaction. Finding the EFMs, extreme rays of the cone defined by $(S|y)v = 0$, and removing the net stoichiometry reaction from the EFMs afterwards gives us the elementary leakage modes, indicating which set of reactions are conflicting, along with vector coefficients.

Note that the computation of minimal net stoichiometries for each unconserved metabolite is extremely costly, and elementary leakage modes can only be computed once a minimal net stoichiometry is obtained, which is a very troublesome disadvantage when one is trying to simply get the set of conflicting reactions. Thus, if multiple inconsistencies are present in a model, multiple net stoichiometries might be obtained for each unconserved metabolite, and leakage modes are supposed to be computed for each one of them. It should also be mentioned that once a set of conflicting reactions is obtained, depending on the size of the reactions set, once might still not be able to detect the inconsistency. Thus, this computation is poorly applicable to genome-scale models in practice.

The check for model stoichiometric consistency computation is incorporated in the MEMOTE toolbox [169], along with computation of unconserved metabolites and minimal net stoichiometries, though these latter two are skipped by default. Elementary leakage modes are omitted from MEMOTE. Another way to detect stoichiometric inconsistencies, which is quite more efficient, is the algorithm GAMES by Shin and Hellerstein [173]. It also computes conflicting reaction sets in the form of reaction isolation sets and metabolite isolation sets, and is parametered by default to only output a few solutions. A process one might take to detect the inconsistency is computing the mass balance equation from the sum of all inconsistent reactions and take guess as to what metabolite imbalance is inconsistent and where it comes from. For example, the output for the equation in equation 2.21 provided by GAMES is presented in Listing A.10. GAMES is applicable to SBMLs of any specification.

2.10.4 Three principles of curation of compression of EFMTOOL

In 2004, an article from Gagneur and Klamt underlined the important steps in compressing and removing errors of a metabolic model [126]. This is solidified by Appendix B of Urbanczik and Wagner in 2005 [174]. Later, Terzer further defined these steps, separating into the categories of *feasibility analysis*, *nullspace analysis* and *graph consistency analysis* [127].

Errors which result from the definition of flux bounds can be filtered with *feasibility analysis*. Feasibility analysis can be summarized roughly as FVA but performed without the μ parameter, or without maximization of biomass. Terzer defines several types of reactions detected by feasibility analysis, which consists of performing LPs to minimize and maximize the flux value of each reaction, and links reaction removal to lethality. He seems to suggest lumping essential reactions with biomass synthesis reaction, which is a fair point potentially helping to decrease the number of variables in computations.

Terzer calls reactions that have their minimal and maximal flux values at zero *zero flux reaction* – we will be using the term *blocked reactions* – these reactions should definitely be removed. Next, by not maximizing biomass in our model and performing a FVA that is not linked to optimal growth, we might be able to retrieve the minimal and maximal flux values of all reactions, and constraining our CBM to simply these flux bounds. This is a major step in correcting models, which sometimes has bounds that it simply can never reach from its stoichiometry and topology.

On the other hand, *nullspace analysis*, depends purely on the stoichiometry of the network, by the use of analysis of the kernel of stoichiometric matrix $Ker(S)$, or nullspace. Terzer notices zero-rows in the kernel matrix corresponds to reactions that are always blocked reactions, regardless of flux bounds.

The second concept is the one of enzyme subsets, or reaction subsets, as introduced back in 1999 in METATOOL [109, 126]. The idea is to notice linearly dependant lines in the stoichiometric kernel $K = Ker(S)$, lines only differing by a constant factor α, and lump them together. These are reactions that are by definition always flux-carrying at the same time, the flux through one reaction is always a multiple of the flux through the other reaction. Reaction subsets allows for a *reaction compression* process. Obtained reaction subsets can be of very large size, in particular for all reactions essential to the synthesis of biomass.

Using the information gained from reaction subsets, one can detect inconsistent reversibilities. For example, two reactions that are together in the same enzyme subset with a strictly positive α might have been considered as backwards-only for one and forwards-only for the other. Terzer does not detail what he does in that case. We assume that it simply conforms with the reversibilities defined in the model, and if no flux can be carried in the direction specified in the model, then that reaction has to be removed.

Finally, Terzer defines three types of structures we could identify through *graph consistency analysis*: *dead-end metabolites*, *linear pathways*, and *unique flows* [127]. For dead-end metabolites, that is, metabolites for which there is never consumption or production, they should be removed along with their producing or consuming reactions. For *linear pathways*, internal metabolites are removed recursively until only a single reaction remains. Finally, uniquely produced and uniquely consumed metabolites, which Terzer calls *unique flow*, can be removed by incorporating

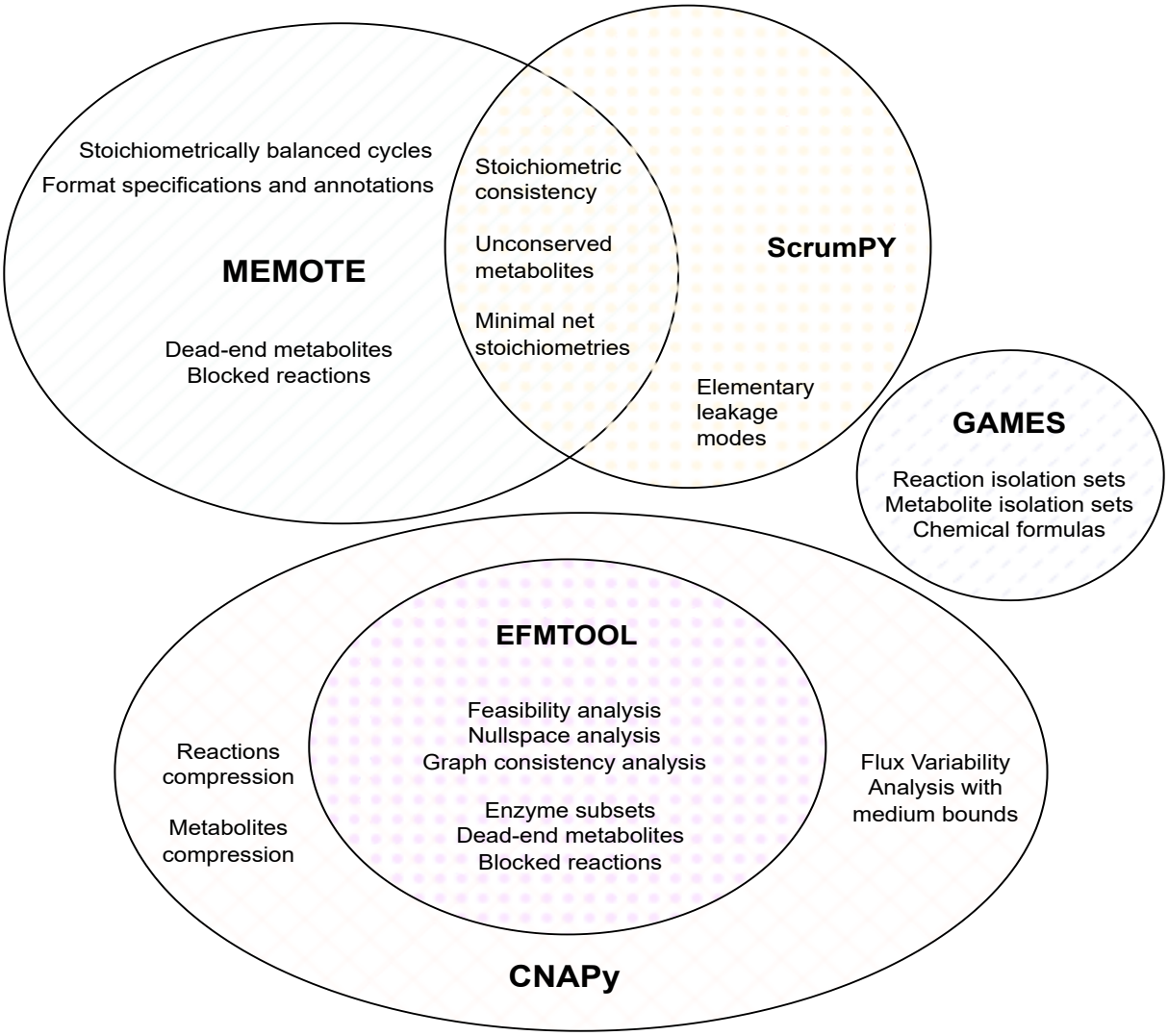


Figure 2.16: Overview of the different model curation and compression techniques. Venn diagram relationships are simplified for reading convenience and should not be considered exact.

the functionally unique reaction into all the other reactions interacting with the metabolite. For example, if one X is uniquely produced by $R1$ through the use of one Y, and X is consumed by $R2$ and $R3$, then delete X and $R1$ and incorportate consumption of one Y in $R2$ and $R3$ [126]. Together, these three ideas define a solid *metabolite compression* process.

Following from the ideas by Terzer, the team from CNAPy developed an interface tool with EFMTool ([W16] and [138]). Their interface tool goes a step further and flips the reaction reversibilities for flux-carrying reactions which are backwards-only, to make them forwards-only for better convenience. It also performs a FVA with the flux bounds given when the model is loaded through the COBRA toolbox, which is of great utility to perform a compression that is dependant on the model's defined growth medium. We note here that, in particular, blocked reactions, and more generally min and max reaction bounds, are intrisically dependant from flux bounds defined by the growth medium. The compression from CNAPy should incorporate all elements discussed above.

In conclusion, for comparison, the standard tool MEMOTE [169] provides elements that might lead to similar insights than the ones provided by Terzer. In particular, it calls *orphan metabolites* metabolites that are only consumed and *dead-end metabolites* metabolites that are only produced. It also provides universally blocked reactions (ie. kernel matrix zero-rows) and provides reactions in stoichiometrically balanced cycles. Extending MEMOTE and proposing a standard tool for performing both curation and compression of constraint-based models might be of major interest in the coming years. We provide Figure 2.16 as a summary of this section.

2.11 Computation of synthetic lethals

Synthetic lethals (SLs) refers to combinations of gene-deletions or enzyme interference targets which prevent growth [175]. While the term initially refered to pairs of genes, it is now used to describe n-tuples of reaction targets. The synthetic lethals may explictly consider both the metabolic potential of the organism and the role of the nutritional environment provided by the extracellular medium [176]. Synthetic lethals apply on metabolic modelling contexts and is done with an exhaustive procedure based on Flux Balance Analysis.

An exhaustive search for synthetic lethal pairs is implemented in COBRAPy [72]. Exploration is purely exhaustive, meaning all combinations are tested for knock-out using FBA. The computation of n-tuples of synthetic lethals can be achieved too with a combinatorial exploration of every possible n-tuple of reactions. Improved algorithms for computing synthetic lethal strategies have been proposed to speed up the calculation process, such as Fast-SL [177] and Rapid-SL [178]. On large networks of over a thousand reactions, computation runs slower as the size of n-tuples increases, and we argue it becomes impracticable if n-tuples of size over 4 are requested – see [177].

We present in appendix the standard algorithms for computing synthetic lethals, illustrating their link to FBA. Algorithm A.6 illustrates the knock-out of a metabolic model reaction. Algorithm A.7, Algorithm A.8, and Algorithm A.9 respectively illustrate the computation of essential reactions, synthetic lethal pairs, synthetic lethal triplets with a large tolerance for biomass synthesis of 10^{-5}. FBA is assumed to be performed with the standard objective function of maximizing biomass synthesis.

Knocking-out a reaction is done by setting the reaction's flux lower bound and upper bound to zero. Although, a less usually done approach but just as much important in our point of view is differentiating the knock-out approach of reversible reactions to be done on either the backwards or forwards direction. This is important as while the production flux of a product P from a substrate S might be essential, the backwards direction of producing S from the substrate P might not. When splitting reversible reactions into irreversible ones, this approach is done naturally.

As one can see, with the proposed exhaustive search algorithms, for computation of SLs of size s on metabolic models of size n reactions, the time complexity would be estimated around $O(n^s)$. Among the lines of codes performed n^s times in the presented procedures, the only costly call is FBA, which is itself polynomial in n but below $O(n)$ in the mean case [101]. Thus, we argue that the exhaustive search of SLs on GSMMs starts becoming impracticable – *i.e.* in the order of magnitude of seconds, minutes, hours – when n-tuples size becomes above 4.

The reasoning for this is that for $s \geq 4$ and $n \geq 1000$ – GSMMs have thousands of reactions – performing above 1000^4 operations is required, and the average personal computer needs a second to perform a billion operations [W17].

Instead of an exhaustive search, the algorithms presented in [175], [177] and [178] propose methods based on bi-level linear programming. Bi-level programming saves on time complexity since the linear program is now no longer run n^s times, however complexity is not necessarily polynomial [179]. Complexity of the SLs exhaustive search and bi-level optimization procedures is reported in Table 2.2 and its comparison to models size in Table 2.3.

Another method proposed for identifying synthetic lethals, whether n-tuple size is under or over 4, is the computation of Minimal Cut Sets (MCSs), with the biomass synthesis reaction as a target reaction.

2.12 Minimal Cut Sets

Minimal Cut Sets (MCSs) are cuts in a metabolic network such that a target reaction or function is disabled. A proper mathematical definition is given in equation 2.12.1.

Definition 2.12.1 – Minimal Cut Sets

Minimal Cut Sets are minimal cuts in a metabolic network disabling flux into certain reaction targets.

Let us denote by $T \subset Reac$ the set of target reactions to be disabled, typically this would be biomass production. From the set of EFMs $E = \{e \in C \mid \nexists e' \in C, Supp(e') \subset Supp(e)\}$ (equation 2.16), we derive the subset of target-containing EFMs E_T according to the following:

$$E_T = \{e \in E \mid e_T \neq 0 \textit{ i. e. } T \subset Supp(e)\} \tag{2.27}$$

We then define the set of MCSs disabling targets as M_T the 'Hitting Sets' of E_T. A Minimal Cut Set in M_T is a subset-minimal set of reactions that forbids function of every EFM in E_T.

$$CS_T = \{cs \subset Reac \mid \forall e \in E_T, cs \subseteq Supp(e)\} \tag{2.28}$$

$$M_T = \{cs \in CS_T \mid \nexists cs' \in CS_T, Supp(cs') \subset Supp(cs)\} \tag{2.29}$$

Thus, Minimal Cut Sets with biomass synthesis reaction as target reaction can be applied to the computation of Synthetic Lethals, whether they are of size under 4 or over 4. For clarity, we could refer to MCSs of size 3 or less as 'small MCSs' and MCSs of size 4 or more as 'large MCSs'.

For example, we might define M_S the MCSs of small size and M_L the MCSs of large size as:

$$M_S = \left\{cs \in M_T \;\middle|\; |cs| \leq 3\right\} \text{ and } M_L = M_T - M_S \tag{2.30}$$

We will detail an application of this arbitrary definition of small/large size MCSs in subsection 4.3.2.

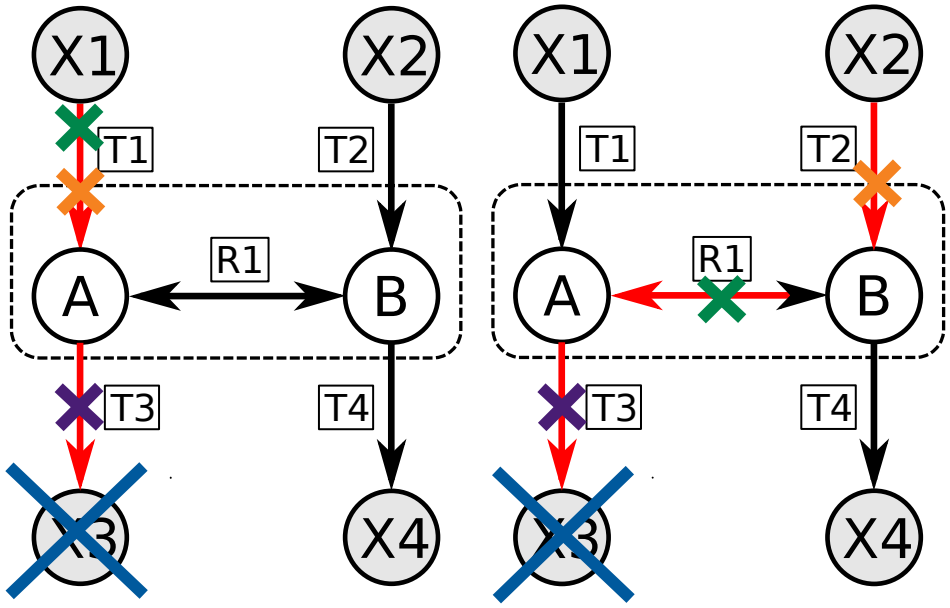


Figure 2.17: MCSs of the toy model network of 5 reactions, cutting target reaction $T3$

As an example of MCSs, let us take Figure 2.17 with target reaction $T3$. Cutting the flux into $c_1 = \{T3\}$ obviously disables $T3$, however, out of the 4 EFMs of the toy network, there are only two involving reaction $T3$. The EFMs with their support in red are represented in Figure 2.17. From the EFMs, we can see there is two other alternate possible ways to cut flux into reaction $T3$, but this time it has to cut one reaction from each EFM, respecting equation 2.28. The intervention strategies are: $c_2 = \{T1, R1\}$ and $c_3 = \{T1, T2\}$. Indeed, we can see that cutting $T1$ or $R1$ alone is not enough to disable $T3$. We therefore have our three subset-minimal MCSs solutions: c_1, c_2, c_3.

Minimal Cut Sets were first introduced by Steffen Klamt and Ernst Gilles in 2004 [180]. MCSs are traditionally defined as the 'Hitting Sets' of Elementary Flux Modes (EFMs) [134, 181, 182], and are an exhaustive way of exploring robustness of a network. Setting a certain reaction as target for inactivation, MCSs define all sets of reactions capable of preventing flux through the target reaction [183]. MCSs have demonstrated remarkable performance identifying synthetic lethals in cancer cells [184].

MCSs suffer the same computational time hindrances as EFMs. The number of possible MCSs grows exponential with the number of reactions [185]. Interestingly, it has been proven that MCSs can be enumerated as the EFMs of a so-called dual metabolic network [186, 187]. As a result, similarly to how Mixed-Integer Linear Programming (MILP) methods were developed for computing the shortest EFMs of a metabolic network, [140, 150], MILP methods for computing the shortest MCSs have been developed [153, 154].

With biomass synthesis reaction as the target reaction, Minimal Cut Sets can be applied to the research of synthetic lethals. It is necessary to convert the obtained MCSs into sets of target genes or proteins for biological interpretation. Methods have been developed to incorporate multilevel data, namely the Gene-Protein-Reaction association rules (GPRs) from GSMMs, into the stoichiometric matrix [162]. These solutions have been repurposed for the MCSs computation [184, 188, 189]. One method in particular can be found in COBRA MATLAB © [188, 112].

Finally, Minimal Cut Sets have practical application in the design and optimization of biotechnological processes. Methods such as OptKnock can be used to study minimal intervention strategies for overproduction of target biochemicals in microbial strains [190, 182]. In that case, the computed cut sets are reactions to be knocked-out that disable certain target flux modes instead of certain target reactions, and such that a certain desired product yield is improved after knocking-out [182].

In conclusion, while EFMs enumerate all possible pathways in a metabolic network, MCSs enumerate all possible cuts in a metabolic network, that disable a certain set of target EFMs. Therefore the two notions are highly related.

2.12.1 Ballerstein's duality property between EFMs and MCSs

In 2012, Kathrin Ballerstein and colleagues demonstrated that MCSs could be computed as special EFMs of a so-called dual network, constructed from the original network [186, 187]. We roughly summarize the related information in Theorem 2.12.1.

Theorem 2.12.1 – Duality property of Minimal Cut Sets

Let $M = (Met, Reac, Stoch, Rev)$ be a metabolic network and $T \subset Reac$ targets.

Then there exists a dual network $D = (f(M,T), g(M,T), h(M,T), \phi(M,T))$ where a particular subset of the EFMs of D corresponds to the MCSs of M cutting target T [186, 187].

Where f, g, h, ϕ are transformations of elements of the original metabolic network M and of the target-s T.

For easier reading, we might write: $D = (Met_{dual}, Reac_{dual}, Stoch_{dual}, Rev_{dual})$.

A special property of the dual network proposed by Ballerstein and colleagues is that $Met_{dual} = Reac$.

This means *metabolites* of the *dual* network correspond to *reactions* in the *original* network.

We will illustrate this network conversion process with the von Kamp formulation in subsection 4.1.1.

In particular, Ballerstein's dual network formulation, based on the stoichiometry matrix $S \in \mathbb{R}^{m \times n}$, is the following:

$$
\begin{aligned}
S_{dual} v_{dual} = \begin{pmatrix} S^T & \mathbf{I} & -\mathbf{I}_{Irrev} & -\mathcal{T} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ \mathbf{t} \end{pmatrix} = 0 \\
u \in \mathbb{R}^m, v \in \mathbb{R}^n, w \in \mathbb{R}^{|Irrev|}, \mathbf{t} \in \mathbb{R}^{|T|} \\
w \geq 0, \mathbf{t} > 0
\end{aligned}
\tag{2.31}
$$

Where $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix and $\mathbf{I}_{Irrev} \in \mathbb{R}^{n \times |Irrev|}$ is the identity matrix indexed at the position of irrversible reactions. As well, $\mathcal{T} \in \mathbb{R}^{n \times |T|}$ is an identity matrix indexed at the position of target reactions $T \subset Reac$. Remember that reactions of the original network become metabolites, hence they become row positions.

Note here that, for the steady-state assumption, instead of the single variables v, we now have four sets of variables: u, v, w, and $\mathbf{t}$. This is significatively important. In fact, unlike the name of their article might lead you to believe, the MCSs of the primal network cannot simply be enumerated as EFMs of the dual network [186]. Indeed, only the variables called v are the ones where subset-minimality should apply. This means that if one were to use traditional EFMs computation algorithms to compute MCSs, EFMs non-subset minimal in v might appear for each MCS, and post-processing filtering would have to be performed to remove all EFMs composed only of the other three kinds of variables together. Additionally, all EFMs not containing the target reaction variable $\mathbf{t}$ are not of interest.

The advantage of this method is being able to enumerate all MCSs without having knowledge of all or even any EFM. This was not the case before, as people used to compute the MCSs as 'Hitting Sets' of EFMs [134, 182]. We can therefore use tools such as the Double Description to enumerate all MCSs in a metabolic network. However, due to the points we just raised, the Double Description algorithm would have to be modified accordingly [187].

Incidentally, notice here that we force the flux of the target reaction-s $\mathbf{t}$ to be non-zero. This might have a detrimental impact on the addition of additional constraints in the computation of MCSs (later discussed in subsection 3.5.1). But most importantly, the interest of having such a property where MCSs of the primal network are EFMs of the dual network is that we can check for the validity of a MCSs by performing the rank test (Theorem 2.7.2) on the dual matrix. However, as we'll see, von Kamp's formulation is condensed and yields faster computation times.

In conclusion, we represent how the dual metabolic network as encoded by Ballerstein's formulation looks like in Figure 2.18. Reactions annotated `'M_'` represent u variables, reactions annotated `'R_'` represents v variables, `'irr_'` represents w variables, `'tgt_'` represents the $\mathbf{t}$ variable. This represents the dual network of the network in Figure 2.17 for target reaction $T3$.

2.12.2 Von Kamp's Mixed-Integer Linear Programming formulation

As seen before, Double Description methods require enumeration of all solutions, which is as inconvenient in the case of MCSs than it is for EFMs. MILP (Mixed-Integer Linear Programming) methods have therefore been developed to compute MCSs. Like EFMs, these methods can easily be adapted to other methods, including Logic Programming with Linear Contraints (LoPLC). However, for the sake of presentation, we are going to refer to the original MILPs as they were formulated.

Two equivalent formulation of MILPs exist, the one from Ballerstein's, and the one from von Kamp's. The advantage of Ballerstein's formulation is that MCSs can correspond to a subset of all EFMs on its network, meaning MCSs can be verified by using the rank test. On the other hand, on von Kamp's network, MCSs might not correspond to special EFMs. This is not an issue, as there are other ways to check validity of MCSs solutions in polynomial time, such as simply using FBA and optimizing target reactions.

The strength of von Kamp's method is that it reduces significantly the number of variables and of constraints compared to Ballerstein's method, thanks to removing the w variables from Ballertein's method. Here is von Kamp's formulation for the MILP constraints:

$$\begin{pmatrix} S_{Rev}^T & \mathbf{I}_{Rev} & -\mathbf{I}_{Rev} & -\mathcal{T}_{Rev} \\ S_{Irrev}^T & \mathbf{I}_{Irrev} & -\mathbf{I}_{Irrev} & -\mathcal{T}_{Irrev} \end{pmatrix} \begin{pmatrix} u \\ v \\ \mathbf{t} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \geq \tag{2.32}$$

$$u \in \mathbb{R}^m, v \in \mathbb{R}^n, \mathbf{t} \in \mathbb{R}^{|T|}$$

$$v_{Irrev} \geq 0, \mathbf{t} > 0$$

Where Rev and $Irrev$ indices designates indexes for the reversible and irreversible reactions, as was done before for the Ballerstein formulation with identity matrices, with $Rev \subset Reac$ and $Reac \setminus Rev = Irrev$. Notice that reactions u corresponding to metabolites are reversible, reactions v where subset-minimality apply are this time only reversible if the original reaction at that index is reversible, while t is always irreversible.

This formulation should be read, for the first row of the dual stoichiometry matrix, impose constraint equals zero, and for the second row, impose constraint greater than zero. In both cases, matrix multiplication with the variables vector occurs. The original network reactions' reversibility determines the kind of steady-state constraint the dual metabolite imposes: the first row for reversible reactions, and the second row for irreversible row.

We further detail the conversion of a metabolic network into a dual network, as well as the treatment of reversible reactions, which are split into two irreversible reactions for these computations, in subsection 4.1.1.

We represented how a dual metabolic network in von Kamp's formulation looks like in Figure 2.19. Annotations are the same as for Figure 2.18, and we can notice two differences between the two: the absence of `'irr_'` reactions or w variables, and the change in reversibility of reactions `'R_'` or v variables.

For further detailing of the example, here are the target-containing EFMs we would find with Ballerstein's formulation of the dual network of the toy model: $e_1 = \{R\_T3, tgt\_T3\}$, $e_2 = \{-M\_A, -R\_R1, R\_T1, tgt\_T3\}$, $e_3 = \{-M\_A, -M\_B, R\_T1, R\_T2, -R\_T4, tgt\_T3\}$, $e_4 = \{-M\_A, -M\_B, R\_T1, R\_T2, irr\_T4, tgt\_T3\}$. We can see that EFMs e_1, e_2, e_4 correspond to all three MCSs c_1, c_2, c_3 of Figure 2.17, while e_3 includes $\{R\_T4\}$ on top of $\{R\_T1, R\_T2\}$, and is thus not subset-minimal in regards to variables v.

Meanwhile here are the target-containing EFMs we would find on von Kamp's formulation: $e'_1 = \{R\_T3, tgt\_T3\}$ and $e'_2 = \{-M\_A, -R\_R1, R\_T1, tgt\_T3\}$. We can see that in von Kamp's formulation EFMs do not correspond to all MCSs, and inversely. Of course though, algorithms computing MCSs could check validity of MCSs by elementary of EFMs by having both dual networks computed, using von Kamp's network for a faster computation and Ballerstein's for verification. For *aspefm*, our logic programming with linear constraints implementation of MCSs, we preferred using von Kamp's implemetation with simple FBA tests with the target (see section 5.2).

EFMs for this example were computed using COPASI [S92] and figures were visualized with CellDesigner [S101]. Unfortunately, to use CellDesigner, we have to convert SBML models which are written in version 3 with FBC version 2 into the old SBML version 2 models. But thankfully, CBMPy is a good and quick tool to perform such a conversion [S89]. Other tools for the visualization of models include ESCHER [191], or simply modifying a SVG file, like is performed by FAME [S102] and FluxVisualizer [S103].

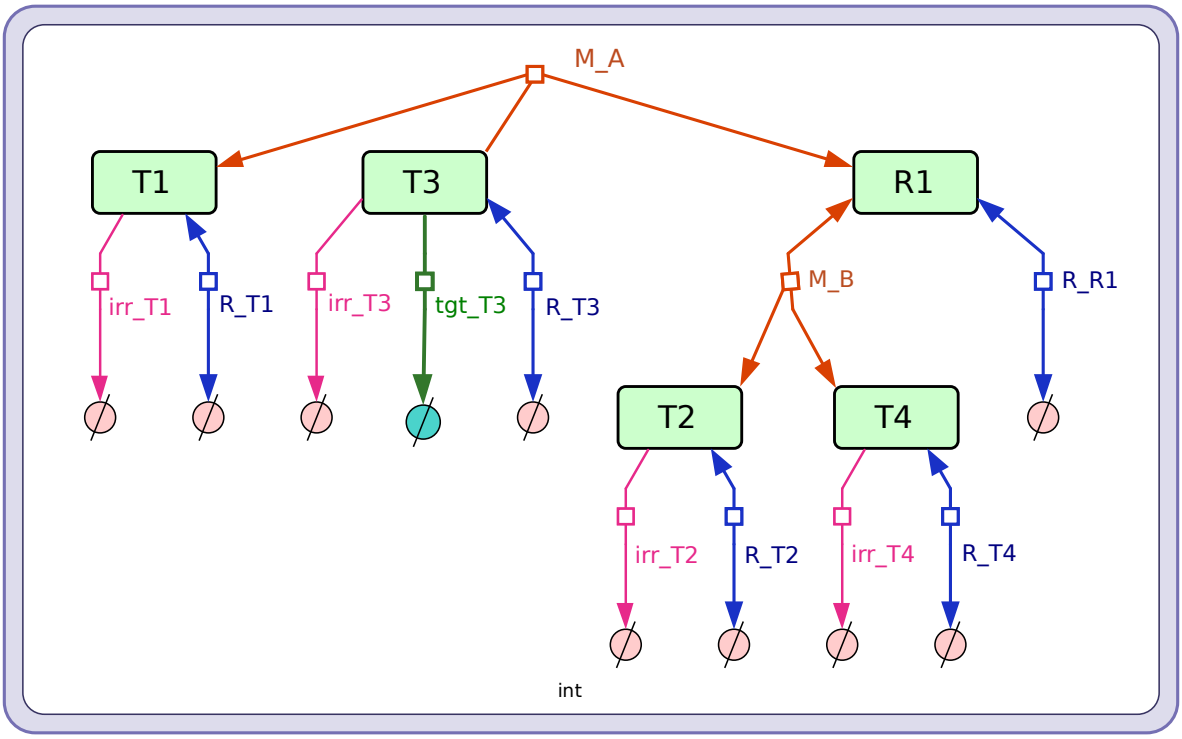


Figure 2.18: MCSs dual network of the toy model, using Ballerstein's formulation

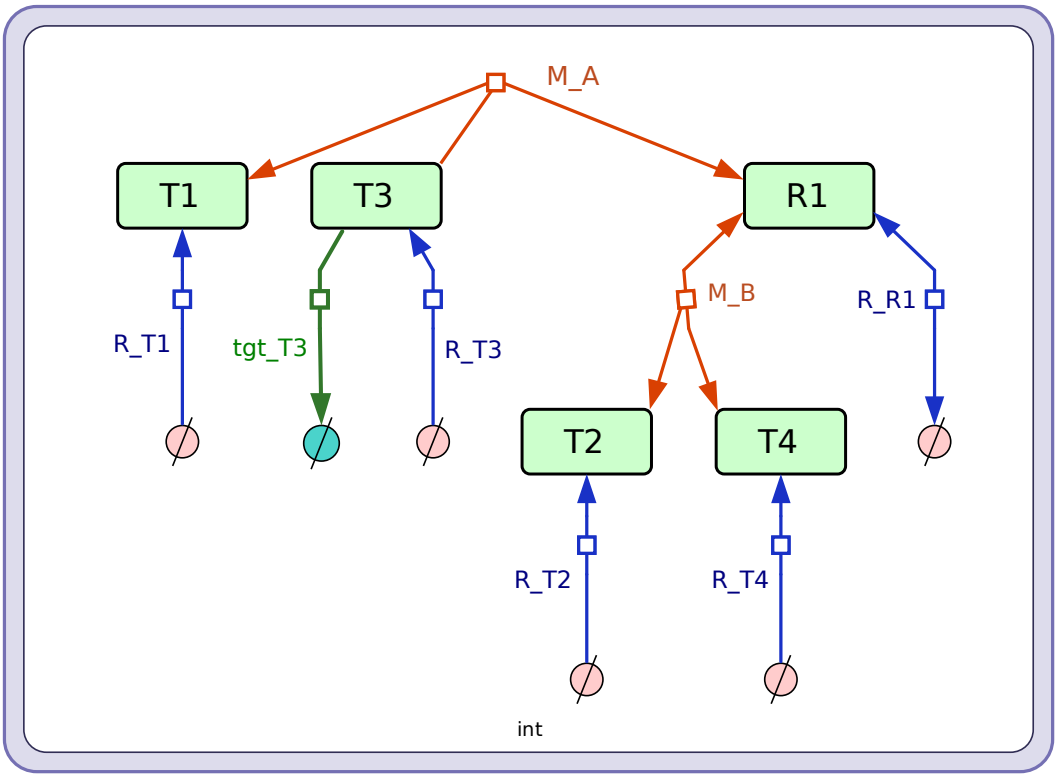


Figure 2.19: MCSs dual network of the toy model, using von Kamp's formulation

2.13 Current applications of metabolic modelling

To conclude this chapter, let us discuss the current applications of metabolic modelling. With genome-scale models now being a widely adopted standard, recent applications have shifted towards larger and larger models. And therefore, people are looking to reduce the computation times of FBA methods. For this reason, we believe a major hurdle that will be necessary to overcome in the future is determining the exact complexity of the methods, in particular for FBA-derived methods.

As an example, a phylogeny of many FBA and other constraint-based modelling methods published in 2012 [161], which is way outdated in the current year; seems to indicate that much effort is given into making new methods, while less effort is given into maintaining those methods and studying their exact time complexity for them to run faster on larger models.

Therefore, to that end, we present in Table 2.2 a rough overview of the computational complexity of the methods presented in this thesis. And in Table 2.3 we display an estimated overview of how methods scale to larger models, including the verifiable performance of our LoPLC tool *aspefm*, represented as *(LP)*. We were indeed able to apply the method to the wide human reconstruction model [192] – although constrained to a specific tissue.

The metabolic model sizes are given as indicative milestones. For instance, the AGORA gut microbiota model consists of single-species bacterial models, which we did not perform computations on, and the models are of size ranging from 1 thousand to 3 thousands, but one could build a larger model if these were combined [75]. The Whole-Body models are said to contain a total of 81,094 reactions [171], which is the basis for us asserting that FBA up to the hundred thousands of reactions is possible. In terms of linear programming, this should not pose an issue if one possesses sufficiently performant machines.

The interest of the metabolic modelling community has now shifted towards multicellular models, such as the AGORA bacterial community models [75], and the whole human body models [171]. These models contain a lot more reactions, due to encompassing several cellular compartments. Other examples of modelling of microbial communities include [193] and [S104]. However, a higher number of reactions does not correlate to a better quality of the model, which is why the principles for metabolic network curation and verification as described in section 2.10 become increasingly important, especially as people keep pushing for larger models.

On the other hand, a fantastic application of the integration of omics data in our opinion is the creation of context-specific genome-scale models – see [163, 194] and [S105, S106, S107]. This is something that has also been done for a while on the human reconstruction model HMR – also called Recon [192]. One can select from which tissue the metabolic model should come from, according to tissue expression data. This is a great advance as constraints greatly reduce the complexity of models and we therefore go back to smaller-scale models, while staying faithful to constraint-based modelling ideas. Tools are also now able to automatically construct microbial genome-scale models constrained to a growth medium, such as KBase [S108]. While a lot of curation is necessary to use these models, we believe the addition of transcriptomics and metabolomics data will lead us to many advances in the understanding of metabolism and its phenotypical expression in different tissues, in human in particular.

Methods	Algorithms	Estimated time complexity
Flux Balance Analysis (FBA) *Flux Variability Analysis (FVA)*	Linear Programming (LP) Constant number of calls to LP	Polytime in n, n number of reactions, *i.e.* linear variables [101]
Dynamic Flux Balance Analysis (dFBA) *Dynamic regulated FBA (rFBA)*	Constant number of calls to LP Constant calls to ODE solving	At least polytime in n, n number of reactions, *i.e.* linear variables, complexity most dependant on ODE solving
Resource Balance Analysis (RBA) *ME-models with enzyme considerations*	Feasibility Linear Problem Constant number of calls to LP	Polytime in $a(n)$, where $a(n)$ denotes the number of linear variables after resource model construction
Synthetic Lethals (SLs)	Exhaustive search : polytime calls to Linear Programming	Around $O(n^s)$, n number of linear variables and s size of reaction set
	Bilevel Linear Programming	Polytime in n best case, exponential in worst case [179]
Parsimonious FBA (pFBA)	Bilevel Linear Programming	Polytime in n best case, exponential in worst case
Elementary Flux Modes (EFMs)	Double Description (DD)	Exponential in the number of reactions n No solutions yielded until the end of enumeration
	Mixed-Integer-Linear-Progamming (MILP)	Exponential in the number of reactions n Best case, single solution in polytime [134]
	Logic Programming with Linear Constraints (LoPLC)	Exponential in the number of reactions n Best case, single solution in polytime
Minimal Cut Sets (MCSs)	Double Description (DD) Hitting Sets (HSs) computation	Exponential in the number of reactions n No solutions yielded until the end of enumeration
	Mixed-Integer-Linear-Progamming (MILP)	Exponential in number of reactions $d(n)$ $d(n)$ denotes number of reactions of the constructed dual network Best case, single solution in polytime
	Logic Programming with Linear Constraints (LoPLC)	Exponential in number of reactions $d(n)$ $d(n)$ denotes number of reactions of the constructed dual network Best case, single solution in polytime

Table 2.2: Constraint-based metabolic modelling methods mentioned in this thesis and their estimated complexity

Finally, it is worth mentioning that the metabolic modelling community is now very interested into the integration of resource allocation constraints [S99, S109]. For example, one might cite the birth of the GECKO toolbox [S110], also linked with mewPy [S88], or formalisms such as Elementary Growth Modes to explore beyond the steady-state [S111, S112]. People are shifting from single objective optimization with FBA to multi-objective optimization *i.e.* Pareto optimization [195], or simply to feasibility problems as is done in RBA [166]. And a great remaining issue to solve in FBA and EFMs analyzes is the integration of experimentally-retrieved kinetic parameters [196, 197].

We will present *aspefm* and our results with the method in the two subsequent chapters (chapter 3 and chapter 4). In the perspectives (chapter 5), we will discuss whether *aspefm* can be an answer to some of these questions.

Models \ *Methods*	FBA/FVA	dFBA	RBA/ME	4-SLs (exhaustive)	4-SLs (bilevel)	pFBA	EFMs (DD)	EFMs (LP)	MCSs (HSs)	MCSs (LP)
Toy model of 10 reactions	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Central E. coli model of ∼50 reactions	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Central E. coli model of ∼100 reactions	✓	✓	✓	✓	✓	✓	✓	?	✓	?
Central Human cell model of ∼150 reactions	✓	✓	✓	✓	✓	✓	✓	C	✓	C
Model of 300 reactions	✓	✓	✓	✓	✓	✓	✓	C	✓	C
Bacterial model of 1000 reactions	✓	✓	✓	✓	✓	✓	?	C	?	C
Consortium bacterial model of 3000 reactions	✓	✓	✓	?	✓	✓	✗	C	✗	C
HMR human cell model of 10000 reactions	✓	✓	?	✗	?	?	✗	C	✗	C
Combined gut microbiota model of 20000 reactions	✓	?	?	✗	?	?	✗	?	?	?
WBM whole human body model of 100000 reactions	✓	?	?	✗	?	?	✗	?	✗	?

Table 2.3: Models size and scalability of main CBM methods of interest.
✓: Computationally feasible. **?**: Unknown, not tested. **X**: Above current computation power.
C: enumeration possible of subset of solutions using biological constraints.
(4-SLs): Synthetic Lethals of size up to 4. *(DD)*: Double Description. *(HSs)*: Hitting Sets.
(LP): linear-programming based methods, including both MILP and SAT-based.

Chapter 3

aspefm: a collection of logic programming tools for exhaustive metabolic fluxes analysis

During the course of this thesis, we developed *aspefm*: a collection of logic programming tools for computing Elementary Flux Modes, Minimal Cut Sets, and more, based on logic programming with the Answer Set Programming paradigm. Briefly, Answer Set Programming is a logic programming language optimized for the resolution of combinatorial problems. To touch upon the subject of combinatorial problems, we introduce Constraint Satisfaction Problems, as defined in the Constraint Programming area of study. Our particular case of Elementary Flux Modes and Minimal Cut Sets are combinatorial problems that can be solved with Logic Programs with incorporated Linear Constraints, making use of Logic Programming, and Linear Programming as presented in the previous chapter.

Throughout this chapter, we describe the *aspefm* method for computing EFMs, as well as the research of related biological constraints, of logical and linear nature. Integration of biological constraints leads to amelioration of enumeration performance and more thorough selection of biological pathways of interest, which we illustrate on an *E. coli* core model, and a central human cell model.

3.1 Constraint Programming

Constraint Programming (CP), the domain of automated reasoning computer programs through sets of constraints and variables, is a significant area of study in artificial intelligence research [98]. Constraint Satisfaction Problems (CSPs), mathematical problems defined according to equation 3.1.1, are of interest in CP.

Following this definition, the Boolean Satisfiability problem, better known a SAT, is a particular case of CSP, with Boolean variables. Other well-known algorithmic problems such as 'Map Coloring' and 'Stable Marriage' can also be expressed as CSPs [98, 102].

CP is closely related to logic programming: CSPs can be seen as a more basic prototype of a complex declarative programming language model, in which an agent specifies constraints to the machine, and the machine deduces the solution by itself. In fact, logic programming might simply be a subset of constraint programming, and well-known solvers of CSPs programs are classic logic programming solvers, including extended SAT-solvers.

Definition 3.1.1 – Constraint Satisfaction Problem

A Constraint Satisfaction Problem (CSP) is defined by a triplet (X, D, C): [198, 98]

$$\begin{aligned} X &= \{X_1, \cdots, X_n\} \quad \text{a set of variables,} \\ D &= \{D_1, \cdots, D_n\} \quad \text{a set of variables value domains,} \\ C &= \{C_1, \cdots, C_m\} \quad \text{a set of constraints.} \end{aligned} \tag{3.1}$$

We define a constraint C_j of scope $V_j = (X_1, X_2, \cdots, X_n) = X$ as a mathematical relation: a subset R of the set $D_1 \times D_2 \times \cdots \times D_n$, such that if the assignment $(v_1, v_2, \cdots, v_n) \in R$, then the constraint is said to be satisfied.

The scope of a constraint C_j is a tuple of variables $V_j \subset X$ involved in the constraint's relation.
A solution to a CSP problem is such that every variable X_i get assigned to a value in its domain D_i, and such that every constraint C_j is satisfied. If not all variables are assigned to values, we have a partial assignment. An assignment might be said to be consistent or inconsistent with the constraints. In the case of inconsistency, the constraints and variables creating a conflict are of interest.

For instance, a CSP with two variables and one constraint is $X = \{X_1, X_2\}$, $D = \{\{A, B\}, \{A, B\}\}$, and $C = \{\langle X_1 \neq X_2 \rangle\} = \{\{(A, B), (B, A)\}\}$. The CSP admits two solutions. If we add the constraint $\{(A, B)\}$, then the CSP only now admits one solution. If we instead add the constraint $\langle X_1 = X_2 \rangle = \{(A, A), (B, B)\}$, now the program does not admit any solutions, since constraints are conflicting.

Parallelly, Linear Programming is possible to consider as declarative programming, as the formalism dictates a way to retrieve solutions from user-imposed directives. It seems unclear whether the paradigm is embraced as a Constraint Programming subdomain by the community, but LP can indeed be seen as a particular case of CSP where domains are reals, constraints are linear, and an objective function is optimized.

In fact, the *IBM* API proposes two different distinct sections of its solver, one for LP: *cplex* and one for CSPs: *CP Optimizer*, which is lesser known [W8].

To avoid confusion between our hybrid logic+linear programming method and linear+logic programming methods such as MILP, we chose to call our area of interest Logic Programming with Linear Constraints (LoPLC). Solving MILPs would fall into the area of Mathematical Programming, while solving LoPLCs and CSPs would fall into the area of Constraint Programming. As we will see, our *aspefm* application uses a CSP solver rather than a MILP solver at its base, thus we argue that our study is definitely befitting of the Constraint Programming study domain.

3.2 Logic Programming and SAT

Logic Programming (LoP) is a *declarative programming* paradigm. Declarative programming defines a class of programming languages where a descriptive program code in comprehensive terms is sent to an engine, and resolved without further user input. An example of this would be SQL for database exploration. It is in opposition to imperative programming, where the program is instructed what to do from start to finish. In *logic programming*, a user inputs a logic problem, or logic program, and a logic solver finds an answer respecting all logic constraints.

Well-known logic programming solvers and languages include ProLog [199], Inductive Logic Programming [S113], Answer Set Programming [200], *clingo* [201], DLV [S114], etc. These tools have been widely used for many applications including theorem solving [S115], robotics [S116], breast cancer diagnosis [S117], electrocardiogram pattern recognition [S113]. Such solvers are essential in the artificial intelligence field and more particularly the automated reasoning domain for the construction of human-intelligible machines, thanks to providing a solid and rational interpretability and explicatibility of results [202].

Logic programming is seen as a particular case of *constraint programming* and may be used to solve CSPs, which are problems as described in equation 3.1.1. Most logic programming solvers are thus able to deal with classical propositional logic, first-order logic, but also constraints with integer domains. At its most basic level, it can be said that solving a Boolean satisfiability problem or SAT problem is also a form of logic programming [203].

Propositional logic is the study of the properties of logical propositions, composed of propositional variables, or *literals*, statements that can be either $\{True\}$ or $\{False\}$, and logical operators. The language used for propositional logic is the language of Boolean functions, thus we refer the reader to Definition 1.6.2. The use of propositional logic can be dated as far back as Aristotles [204].

The Boolean Satisfiability problem, better known a SAT, can be seen as the following: for a set of logic propositions $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$, over propositional variables $\mathcal{V} = \{V_1, V_2, \cdots, V_m\}$, SAT is the problem of finding if there exists an assignment of the literals $\mathcal{V}$ to $\{True\}$ or $\{False\}$ such that the Boolean function $P_1 \wedge P_2 \wedge \cdots \wedge P_n$ yields $\{True\}$. We give more insights into determining satisfiability of a Boolean formula in Definition 3.2.1.

3-SAT is a particular iteration of the SAT problem, dealing with a special specification of Boolean formulas: propositions must be *clauses*, *ie* disjunctions of literals, and in its most simplified form the largest clause of the formula must have exactly three literals. 3-SAT is the first computational problem proved to be NP-Complete [205, 102, 206]. Further definitions for understanding the 3-SAT problem are presented in Definition 3.2.2 and Definition 3.2.3.

Definition 3.2.1 – Satisfiability of a Boolean formula

Given a Boolean formula, or a set of logic propositions, the SAT problem is the decision problem of determining whether or not there exists a solution satisfying the whole formula. A Boolean formula is said to be satisfiable if, for an assignment of variables, the associated Boolean function simply yields $\{True\}$.

A related problem to SAT is enumeration of all solutions satisfying the Boolean formula.
Let us rewrite any Boolean formula F over Boolean variables $\mathcal{V} = \{V_1, V_2, \cdots, V_n\}$ as a set of logic propositions $P = \{P_1, P_2, \cdots, P_m\}$, for instance such that $F = P_1 \wedge P_2 \wedge \cdots \wedge P_m$. We thus have:

$$\begin{aligned} X &= \{X_1, \cdots, X_n\} && \text{our set of variables,} \\ D &= \{\mathbb{B}, \cdots, \mathbb{B}\} && \text{variables are Boolean,} \\ C &= \{P_1, \cdots, P_m\} && \text{our set of constraints.} \end{aligned} \tag{3.2}$$

And solutions to this problem are such that every constraint in C is respected.
SAT is thus a special case of a CSP.

Satisfiability is therefore linked to computational complexity. In practice the problem of formula satisfiability can be seen solved by finding a single assignment that sets the Boolean formula to $\{True\}$, or failing to do so. This is called a *decision* problem. In contrast, throughout this thesis we are dealing with *enumeration* problems. For the SAT problem, enumeration would mean, if there exists assignments satisfying the Boolean formula, finding all of those solutions with the use of a special enumeration tool called a SAT-solver. The characterization of the complexity of an enumeration problem is also different than for decidability[1][2].

Definition 3.2.2 – Conjunctive and Disjunctive Normal Form

A Boolean formula is in Conjunctive Normal Form (CNF) if it is a conjunction of clauses, *ie.* a conjunction of disjunctions of (positive or negative) literals $(\bigwedge \bigvee x_i)$. The negation operator $(\neg)$ is only allowed inside clauses.
A Boolean formula is in Disjunctive Normal Form (DNF) if it is a disjunction of conjunctions of literals $(\bigvee \bigwedge x_i)$.
Every Boolean formula, regardless of its form, can be converted in CNF and in DNF, although the conversion might (or might not) be computationally complex.

[1] We previously mentioned that the complexity of counting EFMs is in the complexity class #NP-Hard. The complexity of counting problems differs from the complexity of decision and enumeration problems, but depending on problems these complexities might be strongly related.

[2] In their Knowledge Compilation Map [207], Darwiche and Marquis define classes of compiled languages of Boolean formula – DNF and CNF are such examples of compiled languages. Some of the compiled languages end up being queriable in polynomial time for counting or enumeration of all solutions.

Definition 3.2.3 – Boolean Satisfiability Problem 3-SAT

A Boolean formula, assumed in its most simplified form, is said to be in k-CNF if it is a conjunction of clauses of at most k literals, and it has at least one clause with k literals. For instance, $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_3)$ is a 2-CNF. While the SAT problem is trivial on 1-CNF and 2-CNF, it is proved to be NP-Complete on 3-CNF. We thus refer to the corresponding problem as 3-SAT, and say that 3-SAT is NP-Complete [205, 102].

3.2.1 Propositional logic with examples

Let us take an example in biology to understand propositional logic. Of course this is merely a simplied example.

$$\begin{aligned} &\textit{Respiration} \text{ happens in presence of } \textit{oxygen}. \\ &\textit{Fermentation} \text{ happens in absence of } \textit{oxygen}. \end{aligned} \tag{3.3}$$

To transform these statements into propositional logic, we must define literals, the elementary variables of propositional logic, assignable to $\{True\}$ or $\{False\}$. Let us define the following literals: fermentation, oxygen et respiration and the following logical formulas:

$$\begin{aligned} respiration &\Longrightarrow oxygen. \\ fermentation &\Longrightarrow \neg\, oxygen. \end{aligned} \tag{3.4}$$

Simply, the SAT problem of finding satisfiable solutions to these two logical formulae consists in assigning values to each literal, such that, if possible, the two formulae are respected.

Here is an example of resolution performed by a SAT-solver: let us assign *oxygen* to $\{True\}$, then *fermentation* is necessarily $\{False\}$. Conversely, let us assign *oxygen* to $\{False\}$, then *respiration* is necessarily $\{False\}$. A solution is found where all literals are assigned either to $\{True\}$ or to $\{False\}$. Assignments can be seen as branches of a resolution tree, while solutions, or non-solutions, are the tree leaves.

For instance, if we have *oxygen* to $\{True\}$, and *fermentation* to $\{False\}$, then there are two valid solutions, one with *respiration* to $\{True\}$ and *respiration* to $\{False\}$. For simplicity, these solutions are respectively written: $\{oxygen, respiration, \neg\, fermentation\}$ and $\{oxygen, \neg\, respiration, \neg\, fermentation\}$

The logic problem in equation 3.4 thus admits four solutions:

$$\begin{aligned} &\{oxygen, respiration, \neg\, fermentation\}, \\ &\{\neg\, oxygen, fermentation, \neg\, respiration\}, \\ &\{oxygen, \neg\, respiration, \neg\, fermentation\}, \\ &\{\neg\, oxygen, \neg\, fermentation, \neg\, respiration\}. \end{aligned} \tag{3.5}$$

Indeed, the four other possible assignments of the three variables do not satisfy the formulae.

A SAT-solver is a tool allowing to find one, several, all assignments satisfying a formula. Here, using a SAT-solver would return all the solutions in equation 3.5. An example of a SAT-solver-based tool is *clingo*, the Answer Set Programming solver used in this thesis.

3.2.2 First-order logic

An extension of propositional logic is first-order logic, which can be roughly summarized as the addition of quantifiers $\forall$ and $\exists$. As well, predicates and quantifier variables can now be defined. Predicates are relations or tuples of unlimited arity and might involve several quantifier variables.

This first-order formalism allows us to define problems on large sets of values, for example, taking the previous example, let us say we replicate it on a set T of several experiments. We get:

$$\begin{aligned} \forall t \in T,\ respiration(t) &\implies oxygen(t). \\ \forall t \in T,\ fermentation(t) &\implies \neg\, oxygen(t). \end{aligned} \tag{3.6}$$

Where $respiration(t)$ is an example of predicate, and t is a so-called quantifier variable. In this new example, the number of variables is multiplied by $|T|$. Previously, there was 3 variables, and thus 2^3 possible assignments. If we fix $|T| = 4$, we get 12 variables, and thus 2^{12} possible assignments. Notice the exponential nature of SAT problems.

Here, the number of satisfiable assignments is also taken to the power of $|T|$, in this case because each problem was independant. Dependance could have been modelled, for example $respiration(t) \implies oxygen(t+1)$.

First-order logic notations are convenient for logic modellers. In practice though, first-order problems can be converted back to simple propositional logic for solvers.

3.2.3 Satisfiability Modulo Theories

An extension of the Boolean SAT problem to more diverse varieties of CSP problems is possible with the so-called SAT Modulo Theories (SMT). SMT introduces non-Boolean variables, removing the need to convert every combinatorial problem to pure SAT Boolean formalism. SMT provides very strong specifications for its non-Boolean variables and is able to solve classic combinatorial problems such as N-queens [S118] and Sudoku [S119] with ease.

The hybrid method allows the user to add constraints and variable of numeric type into the formalism, for instance real numbers and linear constraints with the Linear Real Arithmetic theory. The theory part of the solving is a second solver called during the propagation of Boolean literals in the SAT solver. The second solver is called a theory solver.

Here is an example of what SMT logic constraints would look like:

$$\begin{aligned} respiration &\implies \left([O_2] \geq 10^{-3}\right). \\ fermentation &\implies \left([O_2] < 10^{-3}\right). \end{aligned} \tag{3.7}$$

Where $[O_2]$ is a linear real-valued variable indicating oxygen concentration.

The theories available to SMT solvers are usually reported on websites such as SMT-LIB [W18]. SMT with Linear Real Arithmetic theory was successfully applied previously to perform the computation of EFMs [157].

3.3 Answer Set Programming

Answer Set Programming (ASP) is a particular specification of logic programming. It is a widely used tool for combinatorial problems. ASP has been utilized to solve a variety of biological problems including metabolic network problems [208]. Gebser *et al*. [209] used the formalism to check the consistency of large-scale data sets and provided explanations for inconsistencies by determining minimal representations of conflicts. Razzaq *et al*. [88] combined ASP and model checking to integrate time series of phosphoproteomic data into signaling networks.

More recently, Frioux *et al*. [160] developed a hybrid ASP and linear programming approach for the network gap-filling problem using the solver *clingo*[LP] [210], an extension of the state-of-the art ASP solver *clingo* [211] for solving logic problems with linear constraints over integer and real numbers. This has been possible since *clingo* implemented a theory solver in its last release [201, 212, 213]. Note that, interestingly, *clingo*'s internal solver, *clasp*, uses a SAT-solver-like backend [211], greatly helping performance but also theory integration [201, 212].

The Answer Set Programming logic programming paradigm is oriented towards the resolution of constraint satisfaction problems (CSPs, constraint programming), combinatorial optimization applications, and NP-hard problems in general. As with other logic programming methods, it defines automated reasoning programs, declarative programs that are "solving themselves". In ASP's case, a set of solutions can be derived, and solutions are called *answer sets*. The language defines the so-called *stable models* semantics, where a model is solution if and only if it is stable, and thus answer sets, complete assignments of atoms, are also called *stable models* [214].

In her 2017 paper: "What is answer set programming to propositional satisfiability", Lierler describes how ASP and the SAT problem are related, and how they can express the same formalisms [214]. However, ASP can actually express naturally first-order logic, cardinality constraints, integer variables in its language and with its native solver, and it can integrate most of the formalisms provided by SMT and corresponding theories, thanks to the aforementioned integration of theory solvers. Thus the ASP specification can actually express a much wider variety of CSP problems than SAT. Besides, ASP is actually expressed through comprehensive human-readable variables, rather than mathematical symbols. Since ASP specification is quite complex even for SAT modellers, we recommend Lierler's paper as both an entry-level review and a thorough look at the Answer Set Programming field [214].

3.3.1 Answer Set Programming specification

Answer Set Programming (ASP) is a declarative approach oriented toward knowledge processing with a logic programming approach. Problems are formulated according to first-order propositional logic in order to facilitate the problem modeling. A logic program in ASP is a finite set of *rules* of the form:

$$a \leftarrow b_1, \ldots, b_m,\ not\ c_{m+1}, \ldots,\ not\ c_n$$

where $a, b_1, \ldots, b_m, c_{m+1}, \ldots c_n$ are atomic propositions. An atom a must either belong in a program solution or not, in which case it is denoted by $not\ a$. If an atom is known as a fact, or inferred from a rule, it is considered *true*: the atom becomes part of the answer set. If not, then it cannot be *true*: Closed-world assumption (CWA) applies, meaning that by default, unknown atoms do not belong to solutions, *i.e.* they are *false*. CWA is a principle absent from classical logic that states that everything not currently known to be $\{True\}$ is $\{False\}$ [214].

The head of a rule denotes atom a and the body denotes positive atoms $b_1, \ldots, b_m$ and negative atoms $c_{m+1}, \ldots c_n$. If all positive body atoms are present and all negative body atoms are absent then the head atom should be present. To state that an atom should be present in the solution, the body is omitted. This is called a *fact*. Alternatively, to state *integrity constraints* on body atoms, the head atom is omitted. For a more formal introduction to answer set programming, we refer the reader to [200].

A typical ASP tool is composed of two parts: the *grounder* which handles predicate variables – it converts first-order logic to classical propositional logic – and the *solver* which finds stable sets of atoms satisfying the logic program. The software *clingo* from the University of Potsdam is one such tool, it performs ASP grounding through the *gringo* interface and ASP solving through the *clasp* interface [215]. Its *clasp* solver takes advantage of high performance solving using Boolean satisfiability (SAT) resolution techniques [211]. Both interfaces can be used independantly from *clingo*, and we refer to the official *clingo* guide for a complete explanation of the language capabilities [215].

Other ASP solvers and specifications include DLV [S114] and *smodels* [S120]. However, in this section we will only describe *clingo* ASP specification, as this is the one that we will be using.

```
(1)  head :- statement_1; ... ; statement_n.
(2)  known_fact.
(3)  {unknown_fact}.
(4)  :- statement_1; ... ; statement_n.
```
(3.8)

The program above in equation 3.8 is called a *logic program*. For syntaxic elements, we can no longer use the term literals: we use the term atoms. Note that atoms might be propositions, or first-order logic predicates. Atoms might be either *known* or *unknown*, depending on the state of logic resolution, and either *true* or *false*.

The program is composed of *logic rules*, which must terminate by a point. Here, we represented the four principal ASP rules of our interest. We wil go over the four types of rules and detail their application.

— ***Rule (1)****:* the head and the body of the rule are present, the rule must then be read body $\rightarrow$ head. However, this is **not** an implication in classical logic terms. A reminder of this important distinction might be that we are inferring about knowledge state of atoms (*known/unknown*), rather than about truth values (*true/false*).

The *body* in ASP should be read as a conjonction (AND) of atoms. Thus in our example the head is part of the answer set if all of the statements from the first to the n-th are also part of the answer set.

In ASP, unknown elements are *false* by default, which is called Closed World Assumption (CWA). Thus we must specify to the program what is *known*, using logic phrases with only a head, no body: these are called *facts*.

— ***Rule (2)****:* we specify that atom `known_fact` is necessarily *true*.

In biological experimental conditions we are often confronted with elements of unknown nature, with observations assumed to be *true*, and like in CWA inferring truth from elements of unknown nature is forbidden. By opposition, classical logic do not possess any assumption for elements of unknown nature. Thus we believe that through CWA, ASP can be a tool of great importance for biology. However, it must be said that CWA is restrictive, and implies there might no longer be any lack of knowledge in the corresponding reasoning [S121].

The opposite hypothesis: Open-World Assumption (OWA), exists, allowing that a statement may be true even if it is not certainly known that it is true. This hypothesis is used in web ontologies [S122].

— ***Rule (3)****:* we specify that atom `unknown_fact` can be either *true* or *false*. It thus corresponds roughly to a literal from classical logic.

Note that when performing logic resolution of ASP programs, atoms are indeed associated with classical Boolean literals. There are thus ways to "remove" CWA from ASP logic rules, to perform classical logic reasoning.

— ***Rule (4)****:* the last rule is an *integrity constraint*. It should be read as a negation: body $\rightarrow \{False\}$, *ie.* expressing the negation of the conjonction of constraints defined by the body.

Thus here it is read as at least one of those statements are wrong, using De Morgan's law to convert the negation of the conjunction into a disjunction of negation of atoms.

Integrity constraints are the *only* reliable way to express a constraint in *classical logic*, along with the use of the "classical literals" such as `unknown_fact` in rule *(3)*. Thus integrity constraints are used to "remove" CWA [214].

The keyword allowing to express negation is `not`. Here is an example of usage with first-order logic predicates: `fermentation(T) :- not oxygen(T).` Just like implication in *rule (1)*, `not` does not express classical logic negation.

However, coupled with the use of integrity constraints, one can express classical logic constraints with positive and negative literals, *e.g* `:- not support(R1); support(R2).` becomes the constraint $R1 \vee \neg R2$.

3.3.2 Cardinality constraints and further syntax elements

The rule *(3)* in equation 3.8 : `{ unknown_fact }` is in fact a shorthand for the full notation of cardinality constraints in ASP. In actuality, such a constraint would translate into:

```
0 { unknown_fact } 1.
```
(3.9)

Which is a particular case of cardinality constraint, defined in first-order logic for a set $S = \{s \mid \texttt{subset}(s)\}$ as:

```
LB { subset(S): set(S) } UB.
```
(3.10)

Where we have $0 \leq LB \leq UB \leq |S|$ and $(LB, UB) \in \mathbb{Z}$, the lower and upper bound imposed on the number of atoms in S. By default, if (LB, UB) are omitted and thus not set, we have $(LB = 0)$ and $(UB = |S|)$, a disjunction of atoms. And this notation itself is a shorthand for this full specification:

```
LB <= #count { subset(S): set(S) } <= UB.
```
(3.11)

The equation 3.11 should be read exactly as it is written: the number of atoms `subset(S)` should be comprised between LB and UB. The colon (`:`) separator indicates membership: find all elements `subset(S)` for which atoms `set(S)` is a fact. This helps defining first-order logic relations, which are instantiated by the ASP grounder into named atoms, separated by semicolons (`;`). Semicolons generally define a conjunction of atoms, unless when embedded in curly braces where it defines a disjunction of atoms instead.

Where `#count` is the special keyword for counting and the H-tag `#` indicates special keywords in *clingo* syntax. Other special keywords include `#sum` (sums instead of counting), `#const` (constants), `#show` (when displaying answer sets, displays only the wanted atoms), and `#theory` (defines a theory for the theory solver). Special keywords for theory constraints handled by a theory solver, starting with `&`, such as `&sum`, can also be added.

As mentioned previously, cardinality constraints are not natively incorporated in SAT-solving, and finding the right encoding for them is one of the areas of research [216, 217]. With ASP though this becomes easy. ASP also possesses a natural encoding of integer constraints, useful for task planning for instance [215]. However, the use of a particular theory solver for integers: *clingcon*, might also be more convenient when dealing with integers [218].

Uppercase letters indicate predicate variables, meaning `subset(S)` refers to any atom `subset(s)` $\forall s \in S$, while lowercase letters `set(s)`, and string constants `set("element")`, indicate a specific named atom, a specific instance of the predicate variable.

The syntax defined in here and the previous subsection is specific to *clingo*. *clingo*'s installation is possible with Anaconda or pip: `conda install -c potassco clingo`, and online: `https://potassco.org/clingo/run/`.

3.3.3 Illustrating conversion between SAT and ASP

Let us express the following problem from equation 3.4 in ASP:

$$\begin{aligned} respiration &\Longrightarrow oxygen. \\ fermentation &\Longrightarrow \neg\, oxygen. \end{aligned} \tag{3.12}$$

```
{oxygen}.
{fermentation}.
{respiration}.
 :- respiration; not oxygen.
 :- fermentation; oxygen.
```
(3.13)

Using *clingo*'s full enumeration mode, launched with command line `clingo -n 0`, we can enumerate all solutions:

$$\begin{aligned} &\{oxygen, respiration, \neg\, fermentation\}, \\ &\{\neg\, oxygen, fermentation, \neg\, respiration\}, \\ &\{oxygen, \neg\, respiration, \neg\, fermentation\}, \\ &\{\neg\, oxygen, \neg\, fermentation, \neg\, respiration\}. \end{aligned} \tag{3.14}$$

Now let us illustrate the conversion of the first-order logic example.

$$\begin{aligned} \forall t \in T,\ respiration(t) &\Longrightarrow oxygen(t). \\ \forall t \in T,\ fermentation(t) &\Longrightarrow \neg\, oxygen(t). \end{aligned} \tag{3.15}$$

```
#const nb=5.
{oxygen(T)} :- T=1..nb.
{fermentation(T)} :- T=1..nb.
{respiration(T)} :- T=1..nb.
:- respiration(T); not oxygen(T); T=1..nb.
:- fermentation(T); oxygen(T); T=1..nb.
```
(3.16)

We can observe the augmentation of the number of solutions by increasing the value of constant `nb`. Here is an example of command line execution: `clingo -n 0 -c nb=5` sets `nb` to the value 5.

3.3.4 Subset-minimal ASP solutions

We believe many applications in biology could benefit from the usage of ASP. For example, a simple novel idea would be to implement diagnostic tools with ASPs. From a given phenotype, we could enumerate all possible genotypes, and from a given genotype, we could compute all possible phenotypes. And by bringing in more data, the enumeration can be redirected towards a more likely solution. Most often computational biology analyzes take statistical-related approaches, when sometimes exhaustive combinatorial enumeration could be just as informative.

In particular, the applications of our focus in this thesis is enumeration of mathematical objects relating to fluxes of metabolic networks: EFMs, and MCSs. Such applications are possible thanks to the capacities of solver *clingo* to enumerate all subset-minimal solutions of a given constraint satisfaction problem.

Reminder: ($s \in P$ is subset-minimal) $\Leftrightarrow$ ($\nexists s' \in P$ such that $s' \neq s$ and $s' \subset s$).

Let us present the following simple ASP example:

```
{a}.  {b}.
:- not a; not b.
```
(3.17)

The program admits as answer sets: {a}, {b} et {a, b}.

To enumerate all subset-minimal solutions in ASP, we should add minimization heuristics.

In particular, *clingo* must be run with the parameter `--heuristic Domain --enum-mode domRec` and for each atom to be minimized, the rule `#heuristic atom. [1, false]` should be added to the ASP program.

More precisely, `--enum-mode domRec` specifies to record every found solution, and add *nogoods*, negative literal clauses forbidding the solution to reappear again, for the next enumerations.

For example, if a solution is $\{T1, R1, T4\}$, the solver adds the following negative Boolean clause: $\neg(T1 \wedge R1 \wedge T4)$, this negative clause is a *nogood*.

As far as I understand, the minimization heuristics are particular heuristics that force the corresponding Boolean literals to the atoms to always be set to $\{False\}$ during the literal propagation, unless inferred otherwise, and if no other option is left, doing the trick and computing subset-minimal Boolean assignments. The heuristics are slightly more detailed in the *clingo* guide [215] and in the *clasp* article [211].

Here, we add the following heuristics:

```
#heuristic a. [1, false]
#heuristic b. [1, false]
```
(3.18)

With the *clingo* command, we then obtain two subset-minimal solutions to the program : {a}, {b}. Note that previously enumerated solution {a, b} was eliminated here due to not being subset-minimal.

3.4 *aspefm* for computation of subsets of EFMs

During this thesis, we developed *aspefm*, a tool to compute EFMs and MCSs with ASP logic programming [89]. To incorporate steady-state constraints, we used the extension of ASP to linear variables and constraints [210].

The motivation for using ASP came from the fact that it is an extremely efficient tool to compute subset-minimal solutions, as explained in the previous section. It turns out the problem of computing EFMs is also a problem of computing subset-minimal solutions. As well, the advantage of using ASP over SAT for example is that we can add any logical or linear constraints in human-readable format, which is a bonus for biologists. ASP is also suited for resolution of combinatorial problems and NP-hard problems, which coincides with our use case perfectly.

Elementary Flux Modes computation is a combinatorial problem for which enumeration is said to be #P-complete. With traditional methods, it is impossible to compute all EFMs of a metabolic network of large size, as detailed in chapter 2. An inherent problem of EFMs computation is computing the EFM of interest, integrating many biological constraints. Previously, methods based on SAT and SMT to compute EFMs were developed to address this issue [156, 157]. Parallelly, MILP methods were developed [138]. Therefore, our *aspefm* method is inscribed into the panel of existing methods, as seen in Figure 3.1.

For SAT, SMT and *aspefm*, we call our problem resolution area Logic Programming with Linear constraints, or LoPLC. The *aspefm* computation procedure can be summarized by one single program that will be expressed as a LoPLC problem. The problem will look similar to MILP, with linear constraints being prominent, while logic constraints are expressed as is, and are required by the solver to be satisfied.

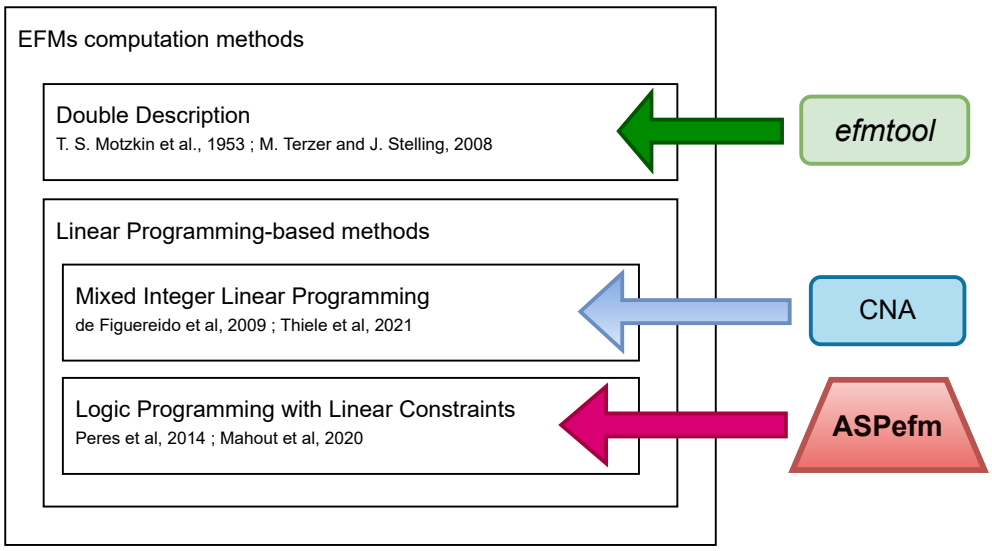
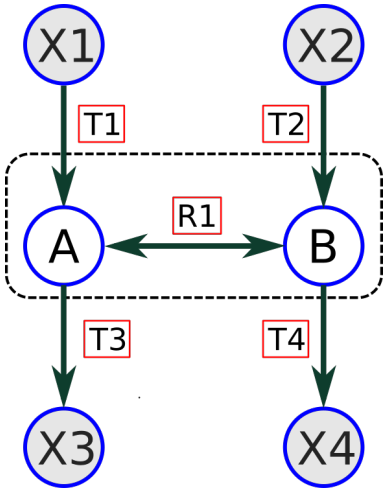


Figure 3.1: *aspefm* compared to the principal computation methods of EFMs




```
reaction(t1).
reaction(t2).
reaction(t3).
reaction(t4).
reaction(r1).
metabolite(a).
metabolite(b).
stoichiometry(a, t1, 1).
stoichiometry(b, t2, 1).
stoichiometry(a, t3, -1).
stoichiometry(b, t4, -1).
stoichiometry(a, r1, -1).
stoichiometry(b, r1, 1).
reversible(r1).
```

Figure 3.2: Expressing a metabolic model in ASP

Since ASP is its own programming language, metabolic networks given in input for EFMs computation first need to be converted into a set of ASP facts to be read by the solver.

We give an example of conversion into a metabolic network in Figure 3.2. In practice, we want the reversible reaction fluxes to be split into two different variables. To do so, we extend the stoichiometry matrix to include the backward directions of reversible reactions with opposite coefficient, as detailed in section 2.3.

Thus we must define a set of rules with duplicated facts for forwards and backwards direction of reversible reactions. To illustrate on Figure 3.2, `reversible(r1)` would be replaced by the pair `reversible(r1, r1_rev)`, and we would add the following lines to the program:

```
reaction(r1_rev).
stoichiometry(a, r1_rev, 1).
stoichiometry(b, r1_rev, -1).
```
(3.19)

The tool we developed for easy conversion of metabolic networks from their SBML format to ASP format is called MPARSER. It is a Python module, and it should be distributed as a submodule of the *aspefm* tool. The module also allows conversion of metabolic networks *from* and *to* other popular input formats such as EFMTOOL, SCRUMPY, METATOOL (see section 2.3).

3.4.1 ASP encoding of metabolic networks

In this section we develop our full formulation of metabolic network models (see section 2.3, section 2.5) in ASP.

Let us represent a metabolic network by a quintuplet $N = (M, R, S, Ext, Rev)$ with M a set of metabolites, R a set of irreversible reactions, S a stoichiometric matrix of size $|M| \times |R|$, $Ext \subseteq M$ the subset of external metabolites, and $Rev : R \times R$ the set of all pairs (r, r_{rev}) of reactions such that r and r_{rev} are issued from the splitting of a reversible reaction. We denote by s_{mr} the stoichiometric coefficient from S associated with metabolite m and reaction r.

To encode the stoichiometric matrix into answer set programming, we translate an input metabolic network $N = (M, R, S, Ext, Rev)$ into a set of the following facts:

$$\begin{aligned}\texttt{ASP}(N) = &\{\texttt{reaction}(r) \mid r \in R\} \cup \\ &\{\texttt{reversible}(r, r_{rev}) \mid (r, r_{rev}) \in Rev\} \cup \\ &\{\texttt{metabolite}(m) \mid m \in M \setminus Ext\} \cup \\ &\{\texttt{external}(m) \mid m \in Ext\} \cup \\ &\{\texttt{stoichiometry}(m, r, s_{mr}) \mid s_{mr} \in S \wedge s_{mr} \neq 0\}\end{aligned}$$

For the problem of finding EFMs of such a network in ASP, the hybrid logic programming-linear programming solver will deduce solutions composed of the following atoms:

- $\{\texttt{flux(r)} \mid r \in R\}$ representing the flux values ν_r for every reaction r. These are theory atoms valued during the solving by *clingo[LP]*. The vector ν composed of all values contained in the `flux` atoms of a solution is a flux vector.

- $\{\texttt{support(r)} \mid r \in R\}$ representing active reactions, reactions r such that Boolean indicator variable $z_r = 1$. There is no atom `support(r)` for reactions r for which $z_r = 0$. In this way, the set of all `support` atoms represents the support $Supp(\nu)$ of the solution flux vector ν.

Definition 3.4.1 – Logical and linear constraints

Throughout the rest of the document, we designate by logical constraint and linear constraint biological constraints for the computation of EFMs and MCSs that can be expressed by the following principles:

- logical constraints or Boolean constraints are Boolean functions over Boolean variables $z_r \ \forall r \in R$
- linear constraints are linear constraints over flux linear variables $v_r \ \forall r \in R$

While linear and logical variables are linked by the following relation: $\forall r \in R, \ z_j \leftrightarrow \nu_j > 0$.

Or in other words, $z_r \ \forall r \in R$ are indicator variables of flux variables $v_r \ \forall r \in R$.

A Boolean constraint is respected if the corresponding Boolean function is satisfiable.

3.4.2 *aspefm*'s input program

aspefm's main component is a declarative logic program in ASP that can be generalized to any metabolic network written in ASP using *clingo* and its grounder. Indeed, rules are written in first-order logic.

We define a set of hybrid predicate logical and linear constraints on the network to be encoded into a set of ASP *rules* in the *clingo[LP]* syntax. Given a reaction r, we represent its flux by linear variable ν_r and if it is active by the Boolean indicator variable $z_r \in \{0, 1\}$. Since all reactions are irreversible, this means all fluxes have non-negative values. In order to be a flux vector at steady-state, a solution should satisfy the following constraints on variables ν_r and z_r :

$$\nu_r \geq 0 \quad \forall r \in R \tag{3.20}$$

$$z_r \leftrightarrow \nu_r > 0 \quad \forall r \in R \tag{3.21}$$

$$\neg z_r \vee \neg z_{r_{rev}} \quad \forall (r, r_{rev}) \in Rev \tag{3.22}$$

$$\bigvee_{r \in R} z_r \tag{3.23}$$

$$\sum_{r \in R} s_{mr} \times \nu_r = 0 \quad \forall m \in M \setminus Ext \tag{3.24}$$

Notice that Equations (3.20), (3.21) and (3.24) need the likes of a linear programming solver, while the other equations are solved with propositional logic only. Equation (3.20) ensures that all fluxes are non-negative values. Equation (3.21) ensures that the Boolean indicator variables are true if and only if the flux has a strictly positive value. Equation (3.22) ensures that the resulting flux does not contain both directions of a split reversible reaction. Equation (3.23) excludes the trivial solution, and the steady state assumption is fulfilled by Equation (3.24). These program rules and the metabolic networks are expressed in ASP using the predicates presented in Section 3.4.1.

The problem formulation is reminiscent of the k-shortest EFMs method [140]. In the MILP problem, on top of these rules, the solver is given the task to minimize the sum of indicator variables, thus returning the shortest flux modes. In our method, such a minimization was not considered. Instead, clingo allows us to set heuristics to enumerate answer sets that are a subset minimal in regards to the indicator variables [215]. This gives us flux solutions with subset minimal support or elementary flux modes. In summary, we are able to enumerate the EFMs of a given input metabolic network by translating the network and the rules presented above into a *clingo[LP]* logic program and by using *clingo* heuristics.

As with COBRA constraint-based modelling (section 2.5), in practice, flux values are usually bounded by large constants. Thus, equation 3.20 is represented in the *clingo[LP]* syntax with the following rule: `&dom{0..nb} = flux(R) :- reaction(R)` where `nb` is a large constant ($\sim$ over 1 000). Note that EFMs enumeration is sensitive to this `nb` parameter, and to flux bounds in general, they might affect the number of EFMs computed.

Let us quickly summarize the main rules composing the input program for EFMs computation with ASP in textual form. *aspefm* consists of a logic program with the following rules:

- a rule defining the domain of **linear variables** — reversible reactions are split so that every flux is a non-negative real: $\forall r,\ \nu_r \geq 0$
- a rule for ensuring the reactions **reversibility** is still respected
- a rule excluding trivial solutions
- a linear constraint describing the **steady-state**: $S.\nu = 0$
- a rule defining the **support** as when a reaction is active, *ie.* the reaction has a positive flux: $\forall r,\ \ z_r \leftrightarrow \nu_r > 0$

❖ and finally **heuristics** to get answer sets with **subset-minimal** support

The LoPLC program is composed of five logical and linear rules in *clingo[LP]* syntax, describing the basis for elementary modes, namely the steady-state constraint and the description of the reversibility, as well as rules such as non-triviality and defining Boolean indicator variables for when a reaction is active.

And to these rules, we add *clingo* heuristics allowing us to compute all solutions of subset-minimal support using the Boolean indicator variables defined before: solutions correspond to minimal assignments to $\{True\}$.

Each EFM is a subset-minimal answer set of the logic program composed of the input network, the problem rules, and potential additional constraints. It is represented by the solution set of hybrid theory atoms `flux(r)` representing flux values ν_r for each reaction r in R and the solution set of propositional logic atoms `support(r)` representing Boolean indicator variables z_r for each reaction r such that indicator variable z_r equals 1.

Thus one can use *aspefm* to check if a given flux vector is indeed an EFM, or enumerate all EFMs respecting given constraints, with command `clingo -n 0 aspefm.lp4 network.lp4 --heuristic Domain --enum-mode domRec`. The full command with parameters such as *epsilon* parameter for conversion of strict inequalities to loose inequalities and float accuracy is given in subsection A.6.1. The input program `aspefm.lp4` is presented in Listing A.2.

3.4.3 Framework and computation details

We represent the framework of *aspefm*, including *clingo[LP]* computation details in Figure 3.3. As a refresher, the software *clingo* from the University of Potsdam allows for the resolution of an ASP logic program. It is composed of an ASP grounder: *gringo*, and an ASP solver: *clasp* [211], and it has theory solving capacities [201]. The extension *clingo[LP]* uses *clingo* as the ASP solver and an external LP solver, allowing us to solve LoPLCs [210]. In particular, *clingo[LP]* syntax, defined by theory atoms marked by `&`, allows one to handle linear constraints in an ASP logic program, such as the steady-state assumption in our case. For *aspefm*, we use *clingo[LP]* with strict semantics – see [210] – and the linear programming solver CPLEX ©.

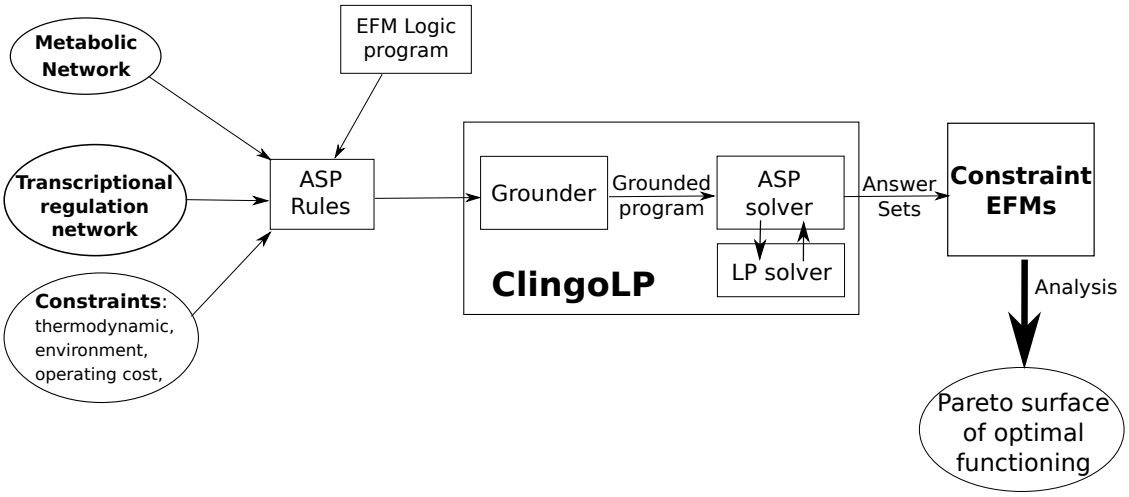


Figure 3.3: Schematic overview of the *aspefm* workflow for computing EFMs under constraints. The ASP rules representing the metabolic model and additional biological constraints are given as input into clingo[LP] along with the logic program for computing EFMs. From all of these rules, the grounder of clingo[LP] generates instantiated constraints, which are sent to the ASP/LP solver. The resulting answer sets are EFMs consistent with all the constraints. These EFMs can be analyzed in post-processing to select the optimal functioning ones.

The main advantages of the *aspefm* method are the following:

- the use of logic programming makes for user-readable constraints, as we can name our variables with a meaning that can be understood by biologists. This is an improvement over previous methods such as SAT and even MILP, for which constraints' names are usually nonsensical and hidden under APIs and GUIs
- unlike MILP, *aspefm* does not attempt to minimize the number of reactions in every EFM it returns as solutions. Both algorithms differ in that MILP is primarily linear-based, while *aspefm* is primarily logic-based. As we will see later, we believe that our *aspefm* tool, while underperforming against MILP for small solutions, achieves better performance than MILP for larger solutions, as a result of these differences
- thanks to the hybrid nature of the tool, any additional *logical* and *linear* constraints can be incorporated, and they would be handled well by the solver. Further types of constraints might be possible to incorporate thanks to the versatility of *clingo* as a tool. Logical constraints are of particular interest since they might reduce the solution space the most and since *clingo* is primarily a logic resolution tool
- since *aspefm* is based on constraint satisfaction, it can still yield a result for models where complete enumeration methods might fail, especially when adding many biological constraints. The typical example is genome-scale models: complete enumeration on genome-scale models is deemed impossible, but with sufficient enough constraints it is possible with *aspefm*

Note that while *clingo[LP]* originally allows for another LP solver, *aspefm* is intended for the use with the *cplex* solver solely. We in fact provide our own version of the *clingo*[LP] extension within *aspefm*. Indeed, multiple code ameliorations were made to *aspefm* since our first application to *E. coli* core, making it much faster. We also extended it to incorporate yet even more types of constraints.

The method was applied and tested on several networks on which complete enumeration was possible and the number of EFMs was known, from toy models to small-medium-scale networks of fifty reactions: including one from M. Covert and B. Palsson [93], and one from R. Carlson [108], which underwent several modifications by S. Peres. Finally, the method was applied to the computation of subsets of EFMs of the *E. coli* core network [28], a network of about a hundred reactions but two hundred million EFMs, as a case study.

clingo comes with various configuration parameters, including different constraint resolution methods and heuristics for *clasp*, and modifying parameters might lead to better tool performance [215]. We tried the different configuration parameters of *clasp i.e trendy, tweety, crafty, frumpy, handy, jumpy* and the parameter used by default: *tweety*, performed better on average each time. However, with a tool as complex and customizable as *clingo*, there is no telling if there might be a parameter that increases average performance that we might have missed.

3.5 Retrieving biological constraints

A major functionality of our tool is the ability of computing EFMs under a variety of constraints. This is done directly during the computation – prior to potential filtering steps in post-processing.

We characterize two different types of constraints: logical constraints and linear constraints. Any additional set of logical and linear constraints can be given as input to our encoding using *clingo[LP]*, utilizing the hybrid nature of the tool. When given to *clingo* alongside the input network and the problem rules, the solver will compute directly the EFMs under constraints (Figure 3.3). Biologically relevant constraints tested with our tool include transcriptional and environmental regulation, thermodynamic equilibrium and biomass operating cost.

We will give a brief overview of the aforementioned kinds of biological constraints, which can be natively integrated to *aspefm*, then develop their analysis and validation further, especially for transcriptional and environmental regulation (see section 3.6). Logical and linear constraints are defined according to Definition 3.4.1 and equation 3.21. For better distinction, we will color constraints *blue* for linear and *purple* for logical.

3.5.1 Examples of logical and linear constraints

As a remainder, here are the definitions for the EFMs solution space and for logical and linear variables:

- Solution space: $P = \{\nu \in \mathbb{R}^r \mid S.\nu = 0 \text{ et } \nu_{irrev} \geq 0\}$
- Reaction fluxes after splitting reversible reactions: $\forall j \in R,\ \ \nu_j \geq 0$
- Literals indicating support of reaction fluxes: $\forall j \in R,\ \ z_j \ \leftrightarrow\ \nu_j > 0$

A modeller may add the following types of constraints, expressed in formal terms:

❖ Wanted reactions, *ex:* $\nu_{biomass} > 0$ (*alt.* $z_{biomass}$)

- Unwanted reactions, *ex:* $\nu_{lac} = 0$ (*alt.* $\neg z_{lac}$)
- Flux bounds, *ex:* $\nu_{ATP} = 8.39$ (no Boolean alternative)
- Transcriptional regulation, *ex:* $z_{biomass} \rightarrow b_{regulateur}$ (*alt.* $\neg\, z_{biomass} \vee b_{regulateur}$)
- Environmental regulation, *ex:* $z_{oxy\_transport} \rightarrow b_{oxy\_env}$
- Thermodynamical equilibrium, *ex:* $k_{eq1}\, \nu_{oxygen} \; + k_{eq2}\, \nu_{biomass} > 0$
- Operational costs, *ex:* $\nu_{oxygen} < K\, \nu_{biomass}$

Where $b_i \;\; \forall i = 1, ..., n$ is part of a Boolean Network symbolizing transcriptional and environmental regulation, or Transcriptional Regulation Network (TRN).

Notice that wanted reactions and unwanted reactions constraints are highlighted in *purple*. Indeed, when alternative formulation as Boolean constraints are available, one should use the Boolean constraint rather than the linear constraint formulation with *aspefm*, since its main component is a logic solver. Linear constraints for the equivalent formulations were tested on multiple occasions and found to return slower computation times.

3.5.2 Expressing additional constraints in ASP

Coming from the modeller mathematical formalism, biologists might appreciate the user-readability of the related ASP constraints. See subsection 3.4.1 for a refresher on corresponding ASP atoms for logical and linear variables.

The additional constraints for the EFMs computation should be expressed in ASP as *integrity constraints* (see subsection 3.3.1). Therefore, negations of the classical logic phrases must be taken.

Thus, z_{r1} becomes `:- not support("R01").`
and $\neg z_{r1}$ becomes `:- support("R01").`

For the literal clause $z_{r1} \vee \neg z_{r2}$:

```
:- not support("R01"); support("R02").
```
(3.25)

And a conjunction of clauses (ex: $z_{r1} \wedge z_{r2}$) will be modelled by a different ASP rule for each clause.

```
:- not support("R01").
:- not support("R02").
```
(3.26)

Linear constraints ($\nu_{r1} - \nu_{r2} > 0$) respect a particular syntax, read by the theory solver.

```
&sum{flux("R01"); -1*flux("R02")} > 0.
```
(3.27)

3.5.3 Operating costs constraints

An operating cost, in metabolic engineering, can be defined as the ratio between the carbon source or oxygen source uptake rate and the biomass production rate. It basically corresponds to the inverse of the biomass yield defined in equation 1.33 in subsection A.6.4.

$$\frac{\nu_{r_1}}{\nu_{r_2}} < \frac{\beta}{\alpha} \quad r_1 \in R,\, r_2 \in R,\, \alpha \in \mathbb{R},\, \beta \in \mathbb{R} \tag{3.28}$$

By adding an upper bound on the operating cost, we further restrict the solution space. It is expressed as a linear constraint, see equation 3.29:

$$\alpha\, \nu_{r_1} - \beta\, \nu_{r_2} < 0 \quad r_1 \in R,\, r_2 \in R,\, \alpha \in \mathbb{R},\, \beta \in \mathbb{R} \tag{3.29}$$

These types of constraints are called operating costs constraints, or yield constraints.

Since we are working with fluxes, and flux experimental measurements such as fluxomics are not easily available nor simple to perform, often one gets experimental measurements only for uptake rates and biomass yields instead. For example, for *E. coli*, uptake rates corresponding to certain biomass compositions are well known [108].

In addition, while constraining flux bounds to experimental flux data in metabolite mass $\times$ dry weight $^{-1}\times$ time $^{-1}$ in FBA might make sense, for EFMs solutions we are dealing with vectorial elements, which could be multiplied by any factor α in linear combinations corresponding to FBA solutions.

Thus, when looking at the solution space of EFMs, *operating costs*, which only constrain ratios of fluxes rather than fluxes themselves, are more appropriate compared to *flux bound* constraints [3]. Simple flux bounds constraints might of course still be enforced, but we believe they must be put in application only if all the other fluxes are bounded significatively, such as what we will do in the analysis of subsection 3.8.4.

3.5.4 Positive and negative constraints

Logical and linear constraints can be added easily to the computation of flux modes with *aspefm*, but they do not actually guarantee that the subset-minimal flux modes returned are EFMs (Theorem 3.5.1).

We decide to separate additional Boolean and linear constraints, into two types: *positive* constraints and *negative* constraints. The former are defined as linear constraints that imply one or more fluxes cannot be inactive anymore. Adding more than one positive constraint will bound the polytope defining the solution space, resulting in a solution space with different subset-minimal flux modes. This was demonstrated and illustrated by Pey and Planes [150].

We provide Definition 3.5.1 and Definition 3.5.2 for how positive and negative constraints apply to logical and linear constraints. Note that positive and negative logical formulae inherently imply positive and negative linear constraints, respectively, thus we only need to provide the formal definitions for linear constraints.

[3] In fact, for referring to EFMs, since they are vectorial solutions, it is more accurate to talk about *flux yields* than *flux values*.

The solutions resulting from constraining the solution space, in the presence of Boolean constraints, were previously coined Minimal Constrained Flux Modes (MCFMs) [157]. In particular, for negative Boolean constraints, Jungreuthmayer *et al.* demonstrated that unsatisfiability was monotone regarding subset-minimality of EFMs supports [143]. A more general definition of MCFMs is that they are minimal flux modes of the *constrained* network, while EFMs are minimal flux modes of the *unconstrained* network. Unfortunately, MCFMs are not always a subset of EFMs.

The MCFMs formalism provides a basis for excluding false solutions, *i.e.* solutions that are MCFMs but not EFMs, found when performing enumeration of solutions. When not EFMs, these MCFMs solutions are instead linear combinations of EFMs, as illustrated in Figure 3.4 on the toy network. The most practical idea for filtering them out is simply to use the rank test, as defined in Theorem 2.7.2, for each solution found [125]. Nowadays, this procedure can be integrated directly into the *aspefm* propagation by the use of extensions (see chapter 5).

In hindsight, it is logical to assume that when only disabling reaction fluxes – with negative constraints – all MCFMs are EFMs [142, 143], thus there is no need to check and filter out potential erroneous solutions. The problems would only arise with constraints actively modifying the solution space – which we termed positive constraints. The terms positive and negative constraints are meant to coincide with positive and negative clauses of literals.

Definition 3.5.1 – Positive constraints

Let us consider the solution space (P), non-negative flux values $\nu \in \mathbb{R}^R$ and Boolean indicators $z \in \mathbb{B}^R$.
A linear constraint (LC) defined by $a^T\nu \geq b$ or $a^T\nu = b$; with $a \in \mathbb{R}^R, b \in \mathbb{R}$ is said to be positive if :

$$\exists\, j \in R \text{ such that } (P) + (LC) \implies (v_j > 0) \text{ when } (P) \not\Longrightarrow (v_j > 0) \tag{3.30}$$

From this definition of positive constraints, we can derive that any Boolean function (BF) that is a positive conjunction of literals $z_r\ \forall r \in R$ is a positive constraint. For disjunctions of positive literals, this applies as well. In fact, from the moment there is one single positive literal in (BF) it is considered a positive constraint.

Definition 3.5.2 – Negative constraints

Let us consider the solution space (P), non-negative flux values $\nu \in \mathbb{R}^R$ and Boolean indicators $z \in \mathbb{B}^R$.
Conversely, a linear constraint (LC) defined by $a^T\nu \geq b$ or $a^T\nu = b$; w. $a \in \mathbb{R}^R, b \in \mathbb{R}$ is said to be negative if :

$$\nexists\, j \in R \text{ such that } (P) + (LC) \implies (v_j > 0) \text{ when } (P) \not\Longrightarrow (v_j > 0) \tag{3.31}$$

Any Boolean function (BF) that is a negative conjunction or disjunction of literals $z_r\ \forall r \in R$ is said to be a negative constraint. In addition, these constraints are proved to be monotone regarding subset-minimality of the support, thus do indeed not impact subset-minimality of EFMs [143].

Theorem 3.5.1 – Property of positive constraints

Following the reasoning of Pey and Planes, when constraining the solution space with two or more positive constraints, two cases arise: extreme points[a] either match with EFMs of the network without these constraints, or they might match with intersections with one of the newly added hyperplanes [150].

From this property it follows that for any additional set of constraints that corresponds to more than one positive constraint, LP-based algorithms do not guarantee non-decomposability of solutions, *i.e.* we cannot guarantee that the solutions retrieved with MILP and *aspefm* are really EFMs.

[a] And by extension, any solution found by LP which is used as the oracle for EFM computation in MILP and LoPLC methods.

Unfortunately, it turns out that operating costs as mentioned in subsection 3.5.3 are positive constraints. Therefore, more than one of these at the same time cannot be added without generating solutions that are MCFMs but not EFMs. In fact, all interesting linear constraints are by definition positive constraints, including changing flux bounds. The issue of positive constraints needing a separate extension to filter out erroneous solutions is quite a problem, and as we'll see, it also greatly hinders the computation time in subsection 3.8.4.

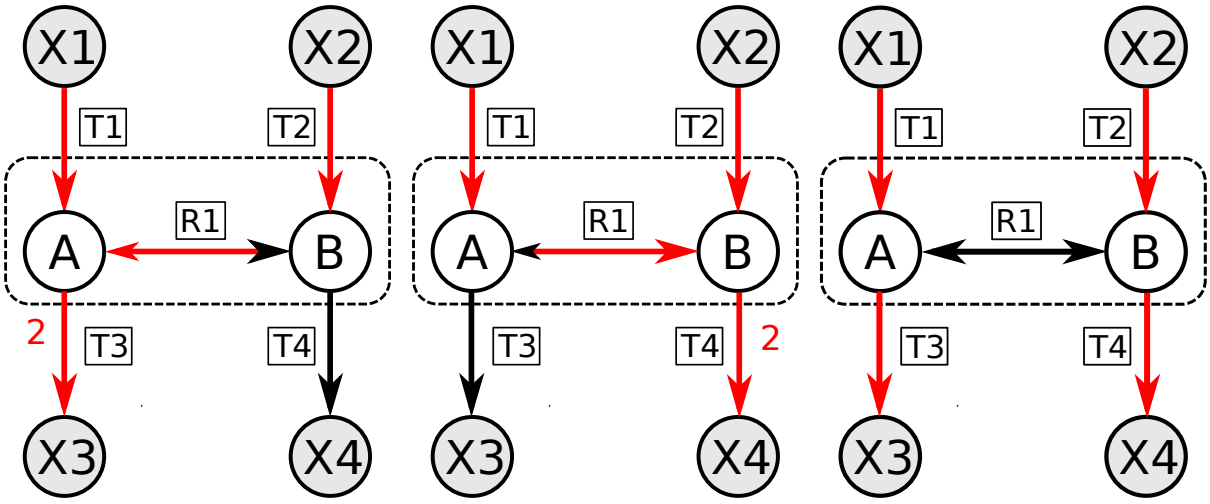


Figure 3.4: MCFMs respecting constraint $T1 \wedge T2$. No EFMs respecting this constraint exists. These MCFMs are returned by *aspefm* upon adding the constraint, unless we filter them out.

3.5.5 Thermodynamic equilibrium

The notion of thermodynamical equilibrium was introduced in subsection 1.3.2. In our case, thermodynamic equilibrium depends on external metabolite concentrations $[EX_i]\ \forall i \in Ext$, since the internal system is at steady-state.

An EFM e is consistent with the thermodynamic equilibrium if $e^{\mathsf{T}} \ln \hat{K}_{eq} > 0$ [145] with $\hat{K}_{eq}$ the vector of apparent equilibrium constants such that for each reaction j:

$$\hat{K}_{eq}^{j} = \frac{K_{eq}^{j}}{\prod_i [EX_i]^{S(i,j)}} \tag{3.32}$$

Apparent equilibrium constants are calculated from standard reaction equilibrium constants, external metabolite stoichiometry, and external metabolite concentrations. This constraint is expressed very simply in our formalism for ASP (equation 3.33, Listing A.5).

$$\sum_{r \in R} \nu_r \times \ln \hat{K}_{eq}^{r} > 0 \tag{3.33}$$

Thermodynamic equilibrium constraints have the great advantage of having been proven to be monotone in regards to subset-minimality of EFMs supports [145]. This means these should not count towards the count of positive constraints for the guarantee of non-decomposability. We tested thermodynamic equilibrium constraints with *aspefm* on the modified example from R. Carlson [108, 145], and it was able to remove a few solutions.

3.5.6 Discussing the addition of biological constraints

In light of the existence of two categories of additional logical and linear constraints: positive and negative, we believe the best possible approach to computing subsets of EFMs is integrating a maximum of negative constraints, and a minimum of positive constraints.

The best approach to get a large amount of reactions disabled has been in our experience constraining a metabolic model to a minimal growth medium. Indeed, all metabolites that are not present in the growth medium can by definition not be consumed by the cell model, leading to the transporter reactions to be deactivated. Combined with network compression, this will lead to great improvement of the performance of the computation of EFMs.

Secondly, in EFMs analysis, we are often only interested in biomass-producing EFMs. Sadly, this is a positive constraint, but adding it is necessary as focusing on only these EFMs allows us to eliminate futile cycles from the enumeration. We found that operating costs, which are constraints with an effect on biomass flux, are also very useful to remove many solutions from the solution space.

In addition, some metabolic networks, such as the Covert and Palsson model [93], and *E. coli* core [28] possess a Transcriptional Regulation Network (TRN). These networks greatly help reducing the number of EFMs as well. However, such a Boolean network should be expressed *only* in terms of negative constraints. We will discuss why in the following section 3.6.

Further versions of *aspefm* will be able to incorporate additional constraints beyond the scope of simple logical and linear constraints, thanks to the use of propagator extensions (see chapter 5).

3.6 *aspefm* encoding of Transcriptional Regulation Networks

In this section we will be interessed in enumeration of EFMs respecting constraints of transcriptional regulation. This is a type of Boolean constraints that could be incorporated in tool *SMTTool* [157], as well as integrated into Description by the method *regEFMTool* [143].

In 2003, Covert and Palsson computed *extreme pathways* under regulation rules constraints on a toy model they developed together with C. Schilling in 2001 [93, 118]. We will refer to that network as CSP2001 (Covert, Schilling, Palsson, 2001). In the case of this model, EFMs correspond exactly to *extreme pathways*, meaning we could use it to test the validity of *aspefm* computation of EFMs under regulation rules constraints. Their regulation rules take into consideration the impact of the growth medium as well: meaning we separate rules into two types: *transcriptional regulation*, and *environmental regulation*.

In comparison, the software *regEFMTool* from Jungreuthmayer *et al*. [143] was tested on Orth, Fleming and Palsson's *E. coli core* model from 2010. As far as we know, *regEFMTool* does not take into account growth medium considerations, even though they are described into the *E. coli core*'s Transcriptional Regulation Network (TRN).

For our study of EFMs under regulation constraints, we shall then look at the two following networks: CSP2001 and ECOLICORE. Informations on the size of the models are presented in Table 3.1. We will entrely describe the TRN of CSP2001 in the following section, while *E. coli core* was used for its own analysis, which we published. A small view of several Boolean nodes of the TRN of *E. coli* core is provided in Figure 2.14.

Note that in order to take into account *environmental regulation* on top of *transcriptional regulation* in the TRN, we introduce supplementary regulation rules linked to transport reactions, *e.g*: an oxygen transport reaction can only be present if external oxygen is present.

Metabolic network	**CSP2001**	**ECOLICORE**
Number of reactions	20	95 (including 20 exchange reactions)
Number of metabolites	11 internal, 8 external	72 internal, 20 external
Number of genes	No genes	137 genes, associated to reactions
Number of regulation rules	11 (7 regulated reactions, 4 regulatory proteins)	78 (56 regulated reactions, 17 regulatory proteins)
Nombre total d'EFMs	80 (59 respecting regulation)	$226.3 \cdot 10^6$ total

Table 3.1: Summary table of informations for the two metabolic networks of interest

3.6.1 Types of rules and variables in Transcriptional Regulation Networks

A regulation rule R_i on a variable b_h of the transcriptional regulation network Reg can be expressed as the following:

$$(A) \quad R_i : b_h \implies \phi_i(b_1, \ldots, b_m) \tag{3.34}$$

$$(B) \quad R_i : b_h \iff \phi_i(b_1, \ldots, b_m) \tag{3.35}$$

Where ϕ_i is the associated Boolean formula, and $b_1, \ldots, b_m$ other Boolean variables of the network.
The network Reg is therefore composed of variables $b_1, \ldots, b_n$ and of the conjonction of rules $R_1 \wedge \cdots \wedge R_p$.

Let us separate variables $b_1, \ldots, b_n$ into four diferent types of variables controlled by regulation rules in the TRN: regulatory proteins, active reactions, genes, and metabolites from the growth medium. These four types of variables are based on what is found in litterature for TRN, especially [219, 94, 71].

Most importantly, notice that we have two possible types of regulation rules, and that one of the types of variables will be the active reactions, which are therefore used and determined during the computation of EFMs from the Boolean indicator variables. As we will be detailing, for all three other types of variables, both types (A) and (B) are possible, but for active reactions, only type (A) is allowed.

Generally, we may associate TRNs used to help computation of EFMs to static Boolean networks: as opposed to those from subsection 1.6.4, they do not evolve over time, except for the active reactions part of the logic resolution with ASP. All other Boolean inputs are therefore known from the start, and Boolean rules will be simplified and compressed. If used for drFBA instead, the full TRN Boolean network would of course be dynamic this time [W19].

3.6.2 Regulatory proteins and growth medium metabolites

Regulatory proteins are intermediate products activated by the environment, the presence of active reactions, or other regulatory proteins. They correspond to known regulatory proteins of the metabolism, such as transcription factors *ArcA* and *FNR* in *Escherichia coli*, or they might be factice indicators marking a particular state of the metabolic network, such as '*surplusFDP*'. The truth value of these Boolean variables is determined automatically by inference and the user has no way of controlling them.

Growth medium metabolites are external metabolites which determine the environment of growth of a microorganism at the start of the simulation. It will determine which regulatory proteins and which reactions will be activated or deactivated. Environment metabolites are in particular implicated in transport reactions, removing a metabolite from the growth medium will automatically disable its transporter. The Boolean variables indicate the presence or absence of these metabolites in the growth medium and are controlled manually by the user, or also possibly determined by the metabolic model's exchange flux bounds.

3.6.3 Active reactions and associated genes

Active reactions are regulated by regulatory proteins or directly by external metabolites for cases that are too complex. On a biological point of view, regulatory proteins are acting as promoters or inhbitors of the transcription of the mRNA coding the enzyme catalazing the reaction.

From a given elementary mode, we can determine which reactions are active and which are not, using the support, represented by Boolean indicators in *aspefm*. Thus for each flux mode, active reactions will set the corresponding Boolean variables of the Boolean network to 1, and inactive reactions will set the corresponding Boolean variables to 0, simplifying the Boolean expressions into a formula that is either satisifiable – if the EFM is consistent with the TRN – or not. In other words, querying the regulation rules network during the computation will let us know if an EFM is consistent with the regulation.

However, if the reverse practice is authorized, *i.e.* Boolean inference on genes gives us a result that implies forced activation of reactions, there is no way to incorporate this into the ASP logic resolution without setting a positive constraint on Boolean indicators. Therefore, rules for active reactions Boolean variables should only be of type (A) and never of type (B). Respecting this different modelling for active reactions, we arrive at the same conclusions for EFMs consistent with transcriptional regulation than Covert and Palsson [93].

In general, we believe this modelling limitation is for the better, as this is consistent with biology's natural order, of reactions being impacted by genes, but genes not being impacted back by these same reactions. Genes are impacted by different enzymes of course, which is why regulatory proteins are also modelled separately. We will keep arguing that point of separating genes and reactions throughout this thesis.

In genome-scale metabolic models, reactions are associated to genes through Gene-Protein-Reaction association rules (GPRs), and transcriptional regulation is done with regulatory proteins at the gene level [28]. In practice, the truth value of Boolean variables for genes is automatically determined by inference, just like for regulatory proteins.

3.6.4 Transcriptional and environmental regulation in *aspefm*

Now let us explore how transcriptional and environmental regulation from TRNs are incoprated as additional constraints into the computation with *aspefm*. For the final equation, $S_{E \cdot tsp}$ denotes stoichiometry of an external metabolite E in its transport reaction tsp. These constraints are meant to be rewritten in ASP syntax.

Let us denote by Reg the set of Boolean variables corresponding to transcriptional regulation constraints. A Boolean function $f(Reg)$ on these variables is any Boolean expression that may be formed from the variables and from NOT, AND, and OR logic operators. Using this formalism, we say that a reaction r is active only if its regulation rule $f_r(Reg)$ returns true (equation 3.36).

$$z_r \rightarrow f_r(Reg) \quad r \in R \tag{3.36}$$

For example, the regulation for a transport reaction $tspA$ may be $z_{tspA} \rightarrow A_{ext} \wedge reg_{tspA}$ where Boolean variable $A_{ext} \in Reg$ indicates the presence of external metabolite $A \in Ext$ and Boolean variable $reg_{tspA} \in Reg$ indicates the presence of transcriptional regulator reg_{tspA}. The truth values of Boolean variables can either be automatically inferred with other Boolean functions provided in the transcriptional regulation network or manually set before starting the computation of EFMs.

In practice, following from the formalism proposed by Covert and Palsson [93], we may introduce Boolean variables for every external metabolite and add regulation rules for each transport reaction (equation 3.37), providing us with full control of the environments and environmental regulation. This is a crucial step as restricting us to a single environment reduces drastically the number of EFMs.

$$z_{tsp} \rightarrow E_{ext} \quad \forall tsp \in R,\ \forall E \in Ext \ \text{ such that } \ S_{E \cdot tsp} < 0 \tag{3.37}$$

3.6.5 Exemple of the Covert and Palsson toy metabolic network

The following example presents regulation rules of the CSP2001 metabolic network. The metabolic network is a toy model with standard reactions from a metabolic model, *i.e.* glycolysis-like and Krebs-like reactions, as well as two carbon sources, one oxygen and one hydrogen source, and other sources and byproducts to complete the model.

Reactions

$$r2a \implies \neg rpb$$
$$r5a \implies \neg rpo2$$
$$r5b \implies rpo2$$
$$r7 \implies \neg rpb$$
$$r8a \implies \neg rph$$
$$rres \implies \neg rpo2$$
$$tc2 \implies \neg rpc1$$

Regulatory proteins

$$rpo2 \iff \neg oxygen$$
$$rpc1 \iff carbon1$$
$$rph \iff th$$
$$rpb \iff r2b$$

Metabolite inputs

$$tc2 \implies carbon2$$
$$tc1 \implies carbon1$$
$$th \implies h$$
$$tf \implies f$$
$$to2 \implies oxygen$$

Reactions : $\{r1, r2a, r2b, r3, r4, r5a, r5b, r6, r7, r8a, r8b, rres, tc2, tc1, th, tf, to2, growth, td, te\}$

Regulatory proteins : $\{rpo2, rpc1, rph, rpb\}$

Metabolite inputs : $\{carbon1, carbon2, h, f, oxygen\}$

The elementary mode $\{r2b, r3, r4, r5b, r8b, rres, th, to2, growth\}$ is not consistent with the regulation. Indeed, we have: $rres \implies \neg rpo2$ and $r5b \implies rpo2$, a contradiction. Moreover, if the growth medium did not contain oxygen, then the rule $to2 \implies oxygen$ would not be respected.

We provide the ASP code for the Covert and Palsson network as well as its transcriptional regulation network in Listing A.7, Listing A.6 and Listing A.8. Results obtained with *aspefm* can be compared with the table presented in their 2003 article [93].

In conclusion, Transcriptional Regulation Networks are a great way to help reduce the number of EFMs and only compute a biologically-relevant subset of EFMs. Indeed, these are encoded as negative Boolean constraints, which allows for disabling inactive reactions. However, these kinds of networks are in fact rarely described in the literature, aside from the one of *E. coli* core [28]. And as we will see in the following analysis, the logical constraints imposed by regulation encoded in a rigid TRN might be too strict.

3.7 Application to the *E. coli* core model

The *E. coli* core model is a model on which Double Description, the standard algorithm for computing EFMs, struggles to work with. Therefore, in our publication: "Answer Set Programming for Computing Constraints- Based Elementary Flux Modes: Application to Escherichia coli Core Metabolism", in collaboration with Ross Carlson, we attempt to show that *aspefm* greatly expands the size range of metabolic models that can be analyzed for EFMs and, thus, greatly expands the potential for using EFMs to interpret complex biological behaviors [89].

Elementary Flux Modes (EFMs) provide a rigorous basis to systematically characterize the steady state, cellular phenotypes, as well as metabolic network robustness and fragility. However, the number of EFMs typically grows exponentially with the size of the metabolic network, leading to excessive computational demands, and unfortunately, a large fraction of these EFMs are not biologically feasible due to system constraints. This combinatorial explosion often prevents the complete analysis of genome-scale metabolic models. Traditionally, EFMs are computed by the double description method [133, 139], an efficient algorithm based on matrix calculation; however, only a few constraints can be integrated into this computation. They must be monotonic with regard to the set inclusion of the supports; otherwise, they must be treated in post-processing and thus do not save computational time.

To enumerate only a subset of EFMs, de Figueireido *et al*. [140] proposed the k-shortest EFMs, a Mixed Integer Linear Programming (MILP) method that lists the shortest EFMs up to an iteration k, k which is the number of nonzero flux reactions in the EFM. This method has been revisited several times [150, 151, 220], in particular for other applications such as GFMs (Generating Flux Modes) [128], Minimal Cut Sets (MCSs) [153], an application of EFMs that allows one to identify essential reactions within a metabolic network, and to compute EFMs containing a given set of target reactions [220]. Another variation termed Alternate Integer Linear Programming (AILP) was proposed by Song *et al*. for computing EFMs and MCSs in a sequential manner [155]. Both the SMT and MILP methods can enumerate EFMs on the fly on large models (defined here as networks with $\sim$ 200+ reactions), for which the DD algorithm may not work.

We present *aspefm*, a hybrid computational tool based on Answer Set Programming (ASP) and Linear Programming (LP) that permits the computation of EFMs while implementing many different types of constraints. We apply our methodology to the *Escherichia coli* core model, which contains 226×10^6 EFMs [28, 143]. In considering transcriptional and environmental regulation, thermodynamic constraints, and resource usage considerations, the solution space is reduced to 1118 EFMs that can be computed directly with *aspefm*. The solution set, for *E. coli* growth on O_2 gradients spanning fully aerobic to anaerobic, can be further reduced to only four optimal EFMs by a Pareto front analysis in post-processing.

aspefm is a new hybrid ASP method using *clingo[LP]*, for computing EFMs under Boolean and linear constraints, and it is inspired by the works of Frioux *et al*. on gap-filling of metabolic networks [160]. As SMT and MILP, the computation of EFMs in ASP aims to enumerate EFMs upon request from large networks. However, the use of logical programming with linear constraints provides a method for enforcing numerous types of biological constraints including transcriptional and environmental regulation, thermodynamics, and resource operating costs on the computation of EFMs, all within a human-readable format.

To show its versatility, our *aspefm* tool was applied to a well-known *E. coli* core model with a significant number of EFMs. The method proved capable of computing a subset of biologically-relevant EFMs while a Pareto front optimization was performed as a final analysis step. The framework returned a small number of EFMs which could be analyzed manually and compared with experimental data. The Pareto optimality analysis complements *aspefm* by revealing the most efficient phenotypes, represented by EFMs. In summary, the constraint-based approach succesfully identified, what are deemed to be, all biologically relevant EFMs for producing biomass in a minimal glucose medium.

3.7.1 Biomass production and Pareto optimality constraints

EFMs analysis fully characterizes the metabolic capabilities of an organism since every steady state flux can be represented as a non-negative linear combination of EFMs. This property is useful in many applications such as in analyzing the stability of metabolic systems [221, 183], or in identifying gene deletions that are lethal to the network [222, 223], or in designing optimal cell factories [224, 182].

Most microbial habitats are dynamic, and the availability of resources like electron donors, electron acceptors, and anabolic forms of nitrogen can change with time. Phenotypic plasticity, where the utilized metabolic pathways change with the changing environment, permits microorganisms to remain competitive. Analyzing potential metabolic strategies in the phenotypic tradeoff space permits the identification of EFMs that are competitive for gradients of resource scarcity. EFM analysis of *E. coli* phenotypic acclimation to gradients of resource availability, including O_2 and anabolic nitrogen, have been reported using tradeoff analysis and Pareto optimization [108, 225, 226, 227]. The methodology tabulates the resource requirements to realize each EFM; these resources can be anabolic, e.g., nitrogen to assemble metabolic enzymes, which are described here as resource investment costs or catabolic, e.g., O_2 which serves as an electron acceptor, which are described here as resource operating costs. Some resources can serve both anabolic and catabolic functions like glucose which is both an energy source and carbon source for enzyme synthesis. Optimal phenotypes for acclimating to environments along gradients of resource scarcity can be identified by plotting the resource costs for each EFM in a tradeoff space where Pareto optimality identifies the most competitive phenotypes [228]. Those EFMs that minimize the resource requirements to achieve a target cellular function are considered most competitive because the phenotypes would permit the most biomass to be made based on a finite supply of a substrate. Tradeoff analysis has accurately predicted and interpreted *E. coli* acclimation to O_2, carbon, and nitrogen, scarcity based on physiological, proteomics, and fluxomics data from *E. coli* chemostat cultures [115, 229].

The optimal solution of a constraint-based enzyme allocation problem, with general kinetics, is an EFM [196]. Wortel *et al*. [195] studied growth rate vs. growth yield tradeoffs using an Enzyme-Flux Cost Minimization (EFCM) method. All biomass producing EFMs were screened and it was assumed that the growth rate depended linearly on the enzyme investment per rate of biomass production. EFMs can also be used for dynamic metabolic modeling such as macroscopic biochemical reaction models [230] or hybrid cybernetic models [231]. In these cases, the enumeration of all EFMs is not needed, but the enumeration of EFMs of interest is essential.

For this analysis of EFMs of interest on *E. coli core*, we will base ourselves on the following major hypothesis: Hypothesis 3.7.1. The analysis with *aspefm* will illustrate how the integration of biologically relevant constraints helps getting minimal pathways that are close to representatives of the biological reality. And, to that means, we supposed the hypothesis that all experimentally observed Biomass-producing flux distributions lie on a Pareto front, of biomass yield from carbon source, and of biomass yield from oxygen source, similarly to analyzes in [108, 225]. This simplification lying on two axes allows one to get all main pathways and states classically observed in *E. coli* metabolism, as we will see later. Due to the nature of EFMs, with the theorem stating all flux distributions are decomposable into linear combinations of EFMs, and the nature of Pareto optimality, and of convex hulls, our model providing the main pathways of the cell will be able to be used to determine all flux states lying on a Pareto front.

Hypothesis 3.7.1 – Experimentally observed Biomass-producing flux distributions lie on a Pareto front

An analysis of the bidimensional operating cost space was performed as described in [108] to identify the most efficient EFMs for converting substrates into biomass. The technique found Pareto optimal EFMs, specific EFMs that minimized operating costs for both substrates of interest: Glucose and O_2, and that defined in aggregate, a surface of optimal functioning.

The analysis was based on the assumption that evolution has selected phenotypes, represented by EFMs, that minimize both operating costs simultaneously. Cells expressing phenotypes that do not minimize both costs would not produce as much biomass as cells that do, limiting their fitness in the considered environment. EFMs (or linear combinations of the EFMs) found along the edge of the bidimensional substrate operating cost space represent optimal phenotypes for growth on glucose and a gradient of O_2 availability spanning sufficiency to anaerobic conditions.

The method to identify the EFMs that were on the Pareto front, with respect to both operating costs, required to calculate Pareto optimality of solutions and then compute the convex hull of the operating cost space of EFMs. Solutions that were both Pareto optimal and belonging to the convex hull of the operating cost are the ones lying on the Pareto front. Further information is given in Definition 3.7.1 and Definition 3.7.2.

Definition 3.7.1 – Pareto optimal

A solution $\nu^* \in Sols$ is said to be Pareto optimal with respect to cost functions f_i for all i if and only if:

$$\nexists \nu \in Sols \text{ such that } f_i(\nu) \leq f_i(\nu^*) \text{ for all } i \text{ and } f_i(\nu) < f_i(\nu^*) \text{ for at least one } i \tag{3.38}$$

Definition 3.7.2 – Pareto front

An EFM e of $Sols$ subset of EFMs computed with *aspefm* is said to belong on the Pareto front if:

- It is Pareto-optimal with respect to both glucose and oxygen operating costs e_G and e_O
- It is an extreme point of the convex hull of the projection of $Sols$ into axes (G, O)

We can then say that the Pareto front line composed of all EFMs belonging to the Pareto front represents all possible linear combinations of those extreme points. Any point on the line is a combination of EFMs, can be found with FBA, LP methods; and is assumed to correspond to an experimentally observed distribution flux.

In this analysis, we only have two operating cost axes, but the Pareto optimality analysis can of course be extended to several more axes, although with loss in computation time. In practice, EFMs are prealably normalized by biomass fluxes e_B so that operating costs can be read on uptake fluxes e_G and e_O and so that linearity can be preserved.

3.7.2 Short overview of the methods

The *aspefm* method makes use of a metabolic network and biological constraints translated into a set of ASP rules and integrates them into the hybrid ASP and LP solver *clingo[LP]* to compute constraint-based EFMs. Finally, the resulting EFMs can be processed with a Pareto surface analysis. An overview of the framework is presented in Figure 3.3. The necessary files to run the analysis on the *E. coli* core network are provided in Supplementary Files of the article and described in Appendices subsection A.6.1 and subsection A.6.2.

As a remainder, the biological constraints integrated into the *E. coli core* analysis are: transcriptional and environmental regulation: (equation 3.36) and (equation 3.37), thermodynamic equilibrium (equation 1.5) and biomass operating costs (equation 3.29). We manually curated the TRN of *E. coli core* to make sure all regulation rules involving the activity of reactions are implications and not equivalences, as explained in section 3.6.

Note that the *epsilon* parameter, responsible for conversion of strict inequalities into loose inequalities, and given in subsection A.6.1, is critical for the method and analysis to be successful. This has been the case for several models, also including the one where thermodynamical equilibrium constraints were tested [145]. *aspefm* and *clingo[LP]* [210] are in general very *epsilon*-sensitive.

Pathway visualizations using Escher are available in Supplementary Files of the article [191]. Note that the analysis was performed with an old version of *aspefm* where checking for MCFMs could only be performed in post-processing. Every post-processing analysis, including removal of MCFMs, biomass normalization, Pareto front computing, generation of Escher maps, and so on, was performed in Jupyter Notebooks [S123].

The operating costs bounds were choosen as arbitrary values, based on distance from minimum operating costs computed by *aspefm*. For convenience, the bounds are the same ones as the desired axes limits in the figures. When not enforced, the number of EFMs increases drastically, as observed in subsection A.6.6.

3.7.3 Computing subsets of EFMs on the *E. coli* core Model

The *aspefm* method was applied to the *E. coli* core model by Orth *et al*, 2010, which includes a full transcriptional regulation network [28]. As well, thermodynamic equilibrium data for that network is simple to obtain.

The *E. coli* core metabolic network consists of 95 reactions, 72 internal metabolites, 20 external metabolites, and 78 regulation rules. Fifty-nine reactions were reversible. The core model was found to contain 226.6×10^6 EFMs based on a previous study [223].

The ASP-based EFMs analysis tool computed a biologically relevant subset of EFMs belonging to this network by integrating thermodynamic and regulatory constraints. Additionally, the simulations considered environmental constraints based on growth in a minimal medium containing glucose, CO_2, NH_4^+, inorganic phosphate, H^+, H_2O and O_2. Accordingly, all other transport reactions were inactivated. The biomass-producing EFMs were selected to represent cellular growth. To further reduce computational burden, the solution space was limited to EFMs with a O_2 operating cost of less than 0.7 O_2 moles per biomass C mole and a glucose operating cost of less than seven glucose C moles per biomass C mole. Since the presence of O_2 had a large impact on the regulatory constraints, two separate scenarios were considered: (1) aerobic and (2) anaerobic conditions.

The ASP-based tool identified 1118 aerobic and 363 anaerobic EFMs in 542s and 232s, respectively (Table 3.2). The tool also returned 39 aerobic MCFMs that were filtered out in post-processing. Results were obtained on a commercial laptop with Intel® Core™ i5-7440HQ CPU 2.80GHz [89] [4].

Table 3.2: Number of EFMs retrieved from the *E. coli* core network depending on culturing conditions. The computation time of a single *clingo[LP]* execution given within brackets. Disabling the formate regulation returned EFMs for both aerobic and anaerobic conditions in a single execution.

		Standard Regulation	**No Formate Regulation**
Processing	Aerobic conditions	1118 EFMs [542s] [4]	11017 EFMs [5318s] [4]
	Anaerobic conditions	363 EFMs [232s] [4]	
Post-processing	Filtered out MCFMs	39 MCFMs	119 MCFMs
	Pareto optimal in biomass yield	4 EFMs	5 EFMs

The aggregate set of aerobic and anaerobic EFMs was processed using a phenotypic tradeoff analysis with Pareto optimization of biomass production relative to O_2 availability, as described previously in Carlson and Srienc, 2004 [108]. EFMs that permitted optimal acclimation to gradients of O_2 scarcity had the lowest substrate operating costs (C moles glucose consumed/C mole biomass produced and O_2 moles consumed/C mole biomass produced) defining a Pareto front. Four EFMs defined the Pareto surface with the applied constraints (Figure 3.5, Appendix A.6.3).

[4] Outdated results, in date of December 2020, date of publication of the article. Computation times on *E. coli* core today are about 5x faster.

3.7.4 Removing the strict formate regulation

The regulation network applied in Orth *et al.* [28] was examined for refinement. A modification to formate metabolism was made based on experimental data. Formate has been measured in *E. coli* cultures in the presence of O_2. The pyruvate formate lyase (PFL) enzyme, which produces formate, is O_2 sensitive, but activity is possible when dissolved O_2 concentrations are low, as occurs when cells grow rapidly or in high density cell cultures [229, 115, 232, 233]. In the regulation network of this model, the PFL enzyme is disabled in the presence of O_2 by transcriptional regulators ArcA and FNR (Figure 2.14). However, we believe this too strict of a regulation rule.

Removing this regulation rule for formate metabolism resulted in a $\sim$ 10-fold increase in the number of total EFMs (Table 3.2) and a slightly different Pareto front, which predicted formate production at low O_2 availability (Figure 3.6), consistent with experimental data and previous EFM analyses [108, 229, 115, 233]. Briefly, the Pareto front included the most efficient EFM for producing biomass from glucose, the upper left EFM, which also had a relatively high O_2 requirement. As environmental O_2 availability decreases, optimal use of the network shifts right along the Pareto front quantifying the increased requirement for glucose as metabolic byproducts are secreted. The first predicted byproduct moving down the Pareto surface was acetate, followed by a combination of acetate and formate and, finally, under anaerobic conditions acetate, formate and ethanol.

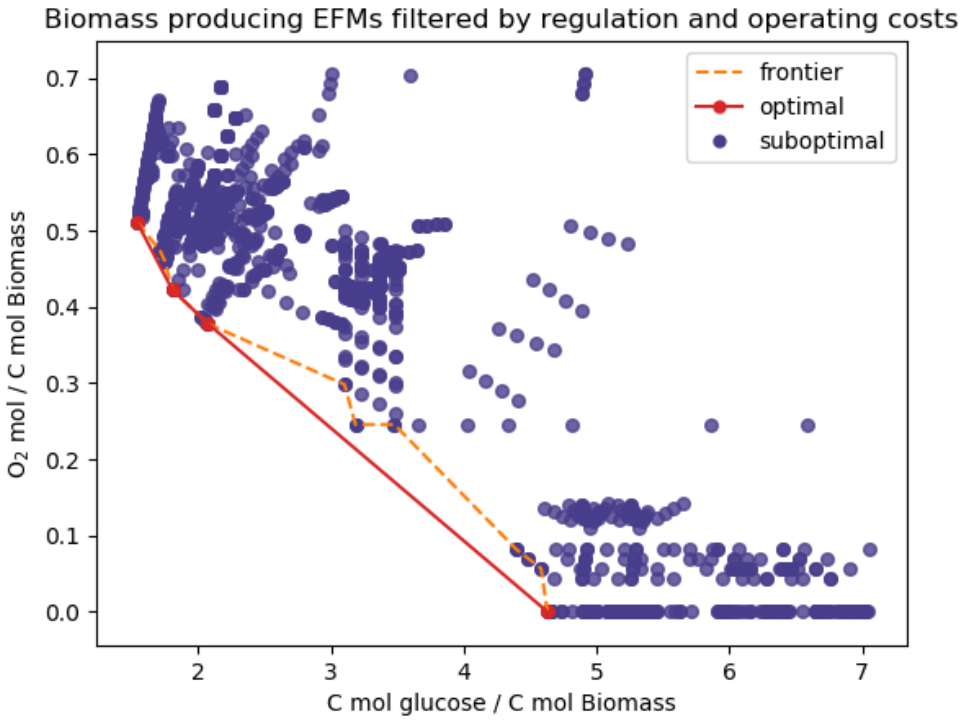


Figure 3.5: *E. coli* core EFMs sorted by carbon/biomass and oxygen/biomass operating costs. Regulation constraints are as described in Orth *et al.* 2010.

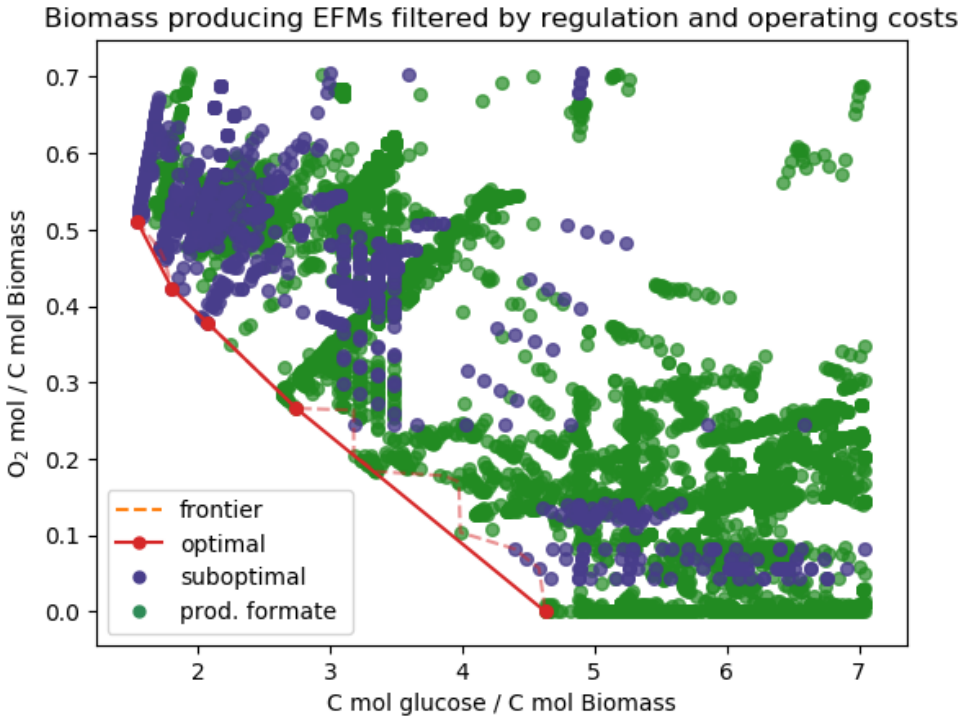


Figure 3.6: *E. coli* core EFMs sorted by carbon/biomass and oxygen/biomass operating costs. Regulation constraints allow production of formate in aerobic conditions.

3.7.5 Summarizing results on regulation and optimal pathways

Our results are based on the information that experimentally-grown *E. coli* shows four different main phenotypes, assumed optimal, in between aerobic conditions and anaerobic conditions. They are the following:

(A) In fully aerobic conditions, oxidative respiration is used, along with Krebs cycle

(B) Then, as we decrease oxygen availability, we observe acetate production

(C) Then we observe combination of acetate and formate, this is also referred to as microaerobic conditions

(D) Finally, under anaerobic conditions, we observe fermentation, *i.e.* acetate, formate and ethanol secretion

In their article from 2004, Carlson and Srienc find exactly these four states, expressed in the form of four Pareto-front EFMs. As well, they were able to retrieve these results not only for the biomass growth derived from carbon source consumption, but also for the ATP energy derived from glucose [108].

In our case, let us name the EFMs in Figure 3.6, from left to right, (1), (2), (3), (4), and (5). Assuredly, when looking at the representation maps of the pathways in Escher – see subsection A.6.3, we have: (1) – corresponds to (A); both (2), (3) – corresponds to (B); (4) – corresponds to (C), (5) – corresponds to (D). Notice that in Figure 3.5, 4 is missing, due to the stringent formate regulation. Therefore Figure 3.6 is the analysis that is the closest to reality, despite an increase in number of EFMs.

At the time when the article was published, computing aerobic formate-producing pathways was an annoying endeavour, as each execution took about 1h28 minutes (Table 3.2). High number of EFMs and high computation times greatly hinder the constraint-based approach, in which we aim to test many different constraints to get the most biologically relevant pathways; case in point: we didn't push further the analysis and implemented a logic rule to exclude formate-producing EFMs in fully aerobic conditions. Thankfully, nowadays with *aspefm* computation times are about five times faster so extending this analysis would be much easier (see commentary in subsection A.6.7).

Our study underlines that one of the key logic rules from the TRN defined in *E. coli* core is too stringent. ArcA, one of the transcriptional regulators at fault, acts differently depending of at least three states: aerobic, microaerobic, and anaerobic conditions [233]. So in conclusion, we believe that modelling oxygen consumption as a simple {0, 1} Boolean state was an oversimplification, where really there are all kinds of thresholds for activation that may exist depending on specific concentrations of external metabolites. Since the TRN was used for dynamic analyzes in the past, where transcriptional regulators might be active or not at any point of time, maybe this strictness of the regulation was, in that context, less important.

3.7.6 Integrating non-growth associated maintenance

From literature data, we know that the average aerobic biomass yield rate is is $\sim$0.4 Cmol biomass per Cmol glucose and that the average anaerobic biomass per glucose is $\sim$0.1 Cmol biomass per Cmol glucose. Currently, in Figure 3.6, optimal aerobic EFM scores at $\sim$1.6 Cmol glucose per Cmol biomass and optimal anaerobic EFM scores at $\sim$4.7, meaning aerobic yield rates of $\sim$0.625 Cmol biomass per Cmol glucose and anaerobic yield rate of $\sim$0.2 Cmol biomass per Cmol glucose[5]. In order to get closer yields to experimental data, we modify the model.

The *E. coli* core model was originally formulated for Flux Balance Analysis (FBA) [71] and the biomass synthesis reaction did not include non-growth associated ATP maintenance (subsection 1.4.5 and [28]). The biomass reaction was modified to facilitate its integration with EFM analysis by account of the maintenance energy required for a culture with a 40 minute doubling time. The biomass reaction was also updated to create elemental stoichiometry of growth, including the degree of reduction, consistent with experimental measurements (see subsection 1.4.4). A detailed explanation of the modifications and additional results are provided in subsection A.6.4 and subsection A.6.5.

We observe a significant change in operating costs, which are now closer to experimental data. Indeed, we now observe optimal aerobic EFM scores at $\sim$2 Cmol glucose per Cmol biomass and optimal anaerobic EFM scores at $\sim$8, meaning aerobic yield rates of $\sim$0.5 Cmol biomass per Cmol glucose and anaerobic yield rate of $\sim$0.125 Cmol biomass per Cmol glucose, in Figure A.1 and Figure A.2.

This indicates that maybe the separation of ATP requirements into growth and non-growth associated maintenance as it is often done in FBA metabolic models is perhaps not as reasonable as it sounds, at least in the scope of our modelling method. This is part of the modelling hypotheses that are inherent to modelling growth requirements, should ATP maintenance be separate or included?

[5] Remember that operating costs are simply the inverse of yield rates.

In FBA, modellers often optimize growth while having the ATP maintenance flux – in the case of *E. coli* core, the ATPM reaction – be bound to a single value, however, what should we do in EFMs analysis? We believed this question was of most relevance.

Although these are good results, in practice, this model modification is quite inconvenient for us as it significatively increases the number of EFMs. In addition, since the biomass stoichiometry was modified, the metabolic network and its stoichiometry matrix are no longer the same, thus the number of total EFMs differs and we can no longer compare our results to the literature and in particular to Jungreuthmayer's results [143].

3.7.7 Discussing the scope of our analysis

The presented *aspefm* method greatly improves the calculation of constraint-based EFMs analyzes. It is capable of enumerating the EFMs of interest without having to calculate and store the complete set of EFMs and it negates the requirement for secondary processing required to select the desired subset. Indeed, *E. coli* core contained 226.3×10^6 EFMs (251 GB) which were computed using EFMtool in 34.1 h [143]. When the regulation network rules were considered, using tool RegEFMTool, the number of EFMs dropped to 2.1×10^6 (2.3 GB) with a run time of 7.1 h. The substantial requirement for disk space to store the complete set of EFMs hampered further analysis.

In contrast to these DD-based methods, *aspefm* makes it possible to integrate a large number of constraints reducing the calculation of non-relevant EFMs. The ASP-based method calculates the desired EFMs relatively quickly without the need for huge storage capacity. In addition, while FBA-based problems are often easily solved, they typically only identify solutions when the constraints make the solution space convex. For example, when stoichiometric and thermodynamic constraints are considered together, the set of possible flux configurations does not generally define a convex set, and thus, it is generally difficult to solve with FBA-relevant optimization algorithms, contrary to the presented analysis. See [234] for a review that tackles the different class of problems.

It is worth noting that computing a minimal set of EFMs with constraints is fundamentally different from computing EFMs and filtering them. In our previous work, we established that the set of EFMs satisfying a constraint c does not always match with the set of flux distributions at the steady state of minimal support satisfying c, which we coined as Minimal Constrained Flux Modes (MCFMs) [157] (see subsection 3.5.4). In particular, this is the case when c is an additional linear constraint $\nu_1 + \nu_2 > 0$, or alternately, a conjunction of positive Boolean literals $z_1 \wedge z_2$. Steady-state solutions of minimal support for such a constraint c (i.e., MCFMs) may be combinations of several EFMs. These MCFMs can be easily discarded by a kernel test. A solution vector Sol is a MCFM and not an EFM if $dim(Ker(S^{Supp(Sol)})) \neq 1$ [125, 150].

In other cases, the set of MCFMs would correspond exactly to the set of EFMs satisfying the constraint. For example, disjunctions of negative literals do not impact the decomposability of solutions. When we bound the operating cost of several metabolites, we add linear constraints in the set of ASP rules which can generate MCFMs which are not EFMs. This is the case in our analysis of the *E. coli* core model, but their number is small compared to the total

number of EFMs (39 MCFMs filtered out versus $1\ 118$ EFMs with the standard regulation and 119 MCFMs filtered out versus $11\ 017$ EFMs with the revised formate regulation, see Table 3.2 and the additional results in Appendix A.6.6). Further work will be performed to integrate verification of elementarity of *aspefm* solutions during execution of the algorithm rather than post-processing (see section 5.2 for our successful integration of the elementarity check for each solution found by *aspefm*).

This work highlights the importance of integrating different types of constraints when performing EFMA on a metabolic model. First, integration of strict Boolean constraints allows the user to restrict analysis to a specific environment and to consider the effects of transcriptional regulation. However, as illustrated by the presented formate metabolism regulation of the *E. coli* core model, a transcriptional regulation network that is too stringent might lead to the omission of experimentally relevant pathways. Second, the integration of curated thermodynamic data enables the computation of EFMs consistent with the equilibrium constants. Conversely, thermodynamic data can be overly lenient, as is the case in this analysis where no EFMs were filtered from the set. Finally, when analyzing biomass production, the application of substrate operating costs bounds constrained the enumerated EFMs to biologically reasonable ranges, but the process may have generated unwanted MCFMs, which had to be removed. Biomass operating costs are convenient for performing Pareto front analyses, which, in turn, facilitate the comparison of model results with experimental data. Further work should be done by also looking at nitrogen investment costs, as was performed in [225]. All gene products were retrieved in the current analysis, and by recuperating data from Uniprot, one could get amino acid costs for every enzyme [34]. This would subsist in a third dimension of the Pareto front.

The presented results are promising as they expand substantially the range of model sizes that can be decomposed into EFMs. However, in order to be applied to large-scale models, the tool will likely require a large number of biological constraints. Otherwise, *clingo[LP]* may struggle with the number of linear problems that need to be solved. Boolean constraints work notably well since *clingo[LP]* is primarily a logic solver, and Boolean constraints mean cutting solutions early before solving any linear problems. The current standard for metabolic models is to link genes to reactions through Boolean associations [69]. *clingo[LP]* is a very efficient tool for solving these Boolean constraints while still representing the syntax in a readable format; and thus, many models found on the BiGG database [37] could be analyzed with our tool using only a reasonable number of additional constraints.

The computation time could be further improved via network reduction and using multi-thread computation routines. The ASP-based implementation with *clingo[LP]* does not currently use multi-threading, so computing EFMs on a server would have minimal benefit in terms of computing time. The method is compatible with network reduction techniques such as the 'enzyme subsets' (i.e., groups of enzymes that operate together in fixed flux ratios at steady state) computation as described in [235, 236], although in this case, only the reduced reactions and metabolites should be used as the input metabolic network. Applied constraints would need to be cast in a manner consistent with the reduced network representation. The network reduction process, including appropriate translation of regulatory constraints, will be the focus of future work (see subsection 3.8.2 for our successful application of compression).

3.8 Application to a model of the human tumoural cell

In order to further demonstrate applicability of our tool *aspefm*, we decided to apply it to a central human cancer cell model, C2M2NF (Central Carbon Metabolic Model with added Nitrogen and Folate) by Jean-Pierre Mazat and Stéphane Ransac [237, 238]. The results presented in this section will be published soon as part of the article "Metabolic modelling shows a correlation of neoangiogenesis, collagen production, and inflammation to Warburg effect in cancer" [239], a work in collaboration with Laurent Schwartz, Ashraf Bakkar and Romain Attal.

Cancer cells are surrounded by a so-called tumoural stroma, which is composed mainly of immune cells, fibroblasts, abnormal blood vessels and collagen. Cancer cells also undergo the well-known Warburg effect, or "aerobic glycolysis", which is abnormal use of the fermentation pathway from glucose to lactate. Lastly, cancer cells show abnormally high uptake of glutamine [240].

To incorporate properties of the tumoural stroma, and show its relation to Warburg effect and glutamine uptake, we devised an improved version of the C2M2NF model, which we called C2M2NFS, for Central Carbon Metabolic Model with added Nitrogen, Folate, and Stroma formation. Stroma formation is itself characterized by collagen synthesis, inflammatory response markers IL1β and TNFα, and growth factor VEGF-A, linked to neoangiogenesis.

We then attempt to demonstrate the two hypotheses presented in Hypothesis 3.8.1 and Hypothesis 3.8.2, first, that neoangiogenesis, collagen production, and inflammation are correlated to Warburg effect in cancer, and second, that glutamine is key to the formation of tumour-related collagens.

Hypothesis 3.8.1 – Neoangiogenesis, collagen production, and inflammation are correlated to Warburg effect in cancer

Throughout our study of the cancer cell metabolism on C2M2NFS, we attempt to show that neoangiogenesis, collagen production and inflammation are correlated to Warburg effect and glutamine uptake, or at least that they happen all simulateneously. Inflammation is represented by inflammatory response markers IL1β and TNFα. Neoangiogenesis is represented by growth factor VEGF-A. Collagen is represented by peptides of a hundred bricks, with its three main components, glycine, proline and hydroxyproline [241].

Hypothesis 3.8.2 – Glutamine is key to the formation of tumour-related collagens

By conversion to glutamate, which is then converted to proline, glycine and hydroxyproline, glutamine might be the key to the formation of tumour-related collagens. The synthesis of collagen can happen endogenously in cells solely from uptake of glutamine.

In order to show our prospect that production of lactate from glucose, collagen from glutamine and release of cytokines are linked together with tumoral growth in cancer cells, we computed Elementary Flux Modes (EFMs) on a modified version of a core metabolic model of central human metabolism: the C2M2NF model by Mazat and Ransac [237, 238].

C2M2NF, Central Carbon Metabolic Model with added Nitrogen and Folate, is a reduced metabolic model of central carbon metabolism comprising about a hundred reactions and metabolites total. The model possesses three compartments, external, cytoplasmic and mitochondrial. It includes an oxidative phosphorylation (OxPhos) reaction system, as well as mitochondrial transporters with pseudo-metabolites (DPH and DPSI) representing the proton gradient through the mitochondrial membrane. Another pseudo-metabolite (PMFm) derived from this gradient is used to represent the mitochondrial protomotive force.

Metabolites of note comprised in the C2M2NF biomass reaction for modelling tumoral growth include: ATP, palmitate, nucleotides, pyruvate, formylmethionine, glutathione and the following amino acids: serine, glycine, glutamine, glutamate, aspartate, arginine, methionine. No changes were made to the biomass reaction in C2M2NFS.

An important point of note is that for simplicity, metabolism of the following amino acids: (Asp and Asn), (Thr, Iso, Val), (Tyr, Phe, Leu, Lys, Trp) are conflated together. In particular, the latter two groups are combined into single metabolites: TIV and YFLKW, and their uptake (TIVUP, YFLKWUP) or catabolism (TIVDG, YFLKWDG) are defined by single group reactions. This allows for working with a smaller-scale model.

3.8.1 Construction of C2M2NFS

Production of proteins is not usually taken into account into metabolic models, as these tend to only include purely metabolic processes. It is occasionally done as a resource optimization procedure such as in RBA [166, 165, 167], meaning every enzyme catalyzing the metabolic processes must be synthesized. However, this requires a tremendous amount of experimental data to calibrate.

In the C2M2NFS model, protein production is done by simply redirecting amino acid metabolism (AAs) to the production of proteins of interest using specific production reactions. Our proteins of interest are collagen and inflammation response markers, ie. IL1β and TNFα, and growth factor VEGF-A.

In order to define the C2M2NFS model, reactions were added to the C2M2NF model to incorporate missing amino acids and transporters: proline, histidine, alanine, asparagine, as well as associated reactions and pathways: alanine aminotransferase, asparaginase, histidine degradation pathway, etc. Pathways were retrieved from Human KEGG PATHWAYS.

Collagen synthesis was defined from literature data as follows: the proportion of AAs in collagen tripeptides was found to be roughly 33% Gly, 16% Pro+Hyp (Hyp: hydroxyproline), and 50% rest [242].

We arbitrarily defined a peptide of collagen as 100 bricks of tripeptides, and thus associated collagen flux to the amount of produced 100 tripeptide collagen strands for conveniences. This is done with the following reactions[6]:

$$\text{COLLAG: } 100\ \text{CBrick} \rightarrow \text{Collagen}$$
$$\text{CBS}^{6} : 0.33\ \text{Gly} + 0.50\ \text{XYAA}^{6} + 0.085\ \text{Pro} + 0.085\ \text{Hyp} \rightarrow \text{CBrick}$$

[6] *CBS*: Collagen Brick Synthesis, *XYAA*: Amino Acids other than Gly-Hyp-Pro, *XYAAS*: Synthesis of Amino Acids except Gly-Hyp-Pro

For other AAs than Gly and Pro, excluding Met, Cys, Asn, and His, which are four of the least common amino acids found in the XY part of collagen tripeptide [242], we assumed an equiprobability distribution.

$$\text{XYAAS}^6 : \text{ALAc} + \text{ARGc} + \text{SERc} + \text{TIVc} + \text{YFLKWc} + \text{GLNc} + \text{ASPc} + \text{GLUTc} \rightarrow 14\ \text{XYAA}$$

Inflammatory markers IL1β and TNFα, and growth factor VEGF-A, were incorporated as protein synthesis reactions, based on the amino acid content of their consensus Uniprot FASTA protein sequence [34]. Corresponding Uniprot entries were `IL1B_HUMAN`, `TNFA_HUMAN` and `VEGFA_HUMAN`.

Stoichiometric coefficients in the three following protein synthesis reactions correspond to amino acid proportions of the protein sequences. Thus, the stoichiometric coefficient next to the protein is the inverse of its length.

$$\text{IL1B: } 0.045\ \text{MET} + 0.048\ \text{ALAc} + 0.078\ \text{GLUTc} + 0.13\ \text{TIV} + 0.056\ \text{PROc} + 0.275\ \text{YFLKW} + 0.078\ \text{SERc} + 0.048\ \text{GLYc} + 0.045\ \text{ASNc} + 0.074\ \text{ASPc} + 0.067\ \text{GLNc} + 0.019\ \text{CYSc} + 0.022\ \text{ARGc} + 0.015\ \text{HISc} \rightarrow 0.00371\ \text{IL-1B}$$

$$\text{TNFA: } 0.009\ \text{MET} + 0.086\ \text{SERc} + 0.163\ \text{TIV} + 0.069\ \text{GLUTc} + 0.06\ \text{ARGc} + 0.03\ \text{ASPc} + 0.245\ \text{YFLKW} + 0.082\ \text{ALAc} + 0.064\ \text{PROc} + 0.073\ \text{GLYc} + 0.056\ \text{GLNc} + 0.017\ \text{CYSc} + 0.017\ \text{HISc} + 0.03\ \text{ASNc} \rightarrow 0.00429\ \text{TNF-A}$$

$$\text{VEGFA: } 0.034\ \text{METc} + 0.03\ \text{ASNc} + 0.237\ \text{YFLKWc} + 0.065\ \text{SERc} + 0.108\ \text{TIVc} + 0.047\ \text{HISc} + 0.034\ \text{ALAc} + 0.056\ \text{GLNc} + 0.065\ \text{PROc} + 0.069\ \text{GLUTc} + 0.06\ \text{GLYc} + 0.034\ \text{ASPc} + 0.082\ \text{ARGc} + 0.078\ \text{CYSc} \rightarrow 0.00431\ \text{VEGFA}$$

If one of the inflammation markers IL1β or TNFα is present in reasonable quantity through its production flux, and growth factor VEGF-A is also being produced, then an inflammatory response with neoangiogenesis is supposed on the model.

$$\text{FACT: VEGFA} \rightarrow \text{GrowthFactor}$$

$$\text{INF1: IL1B} \rightarrow \text{Inflam}$$

$$\text{INF2: TNFA} \rightarrow \text{Inflam}$$

$$\text{STRO: Inflam} + \text{GrowthFactor} \rightarrow \text{Stroma}$$

The resulting C2M2NFS model is of size 119 metabolites and 150 reactions, including 36 exchange reactions. After network compression, the network comprises 94 reactions, 66 internal metabolites, 25 external metabolites.

3.8.2 Metabolic network curation and compression

Since C2M2NFS contains a large amount of EFMs, it is wise to apply network compression to the model. Our network compression pipeline procedure in *aspefm* was combined with network curation, since they go hand in hand. For the network compression, we retrieved code from the CNAPy and EFMTool libraries. A general framework idea is presented in Figure 3.7.

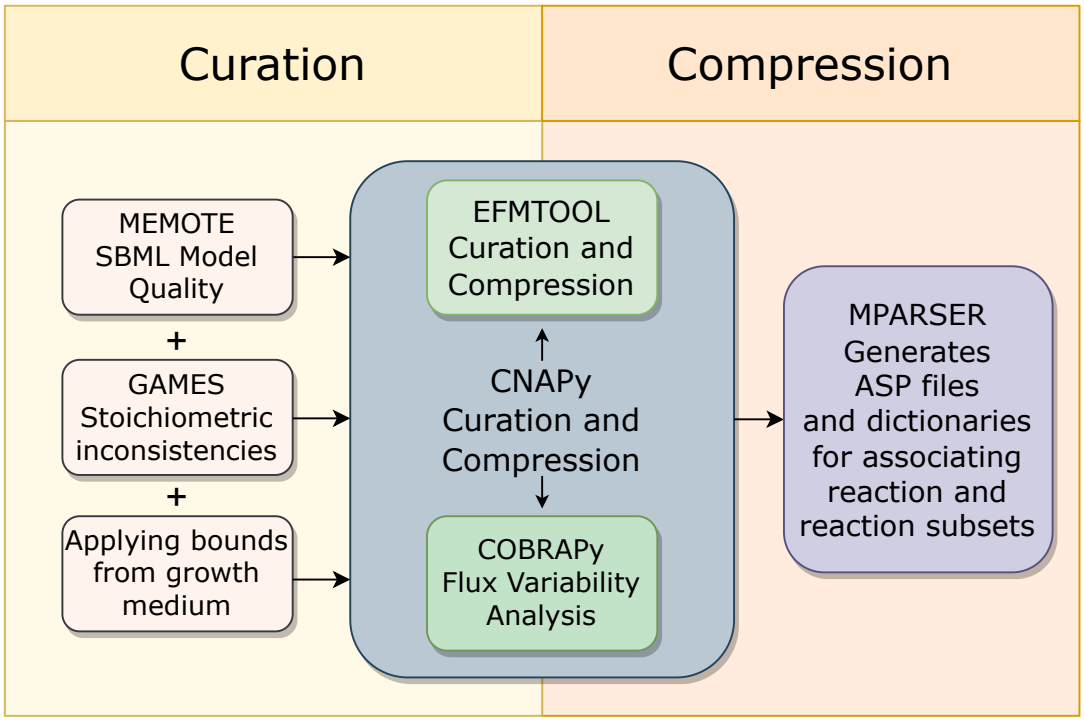


Figure 3.7: General framework for curation and compression of metabolic models in practice

The three principles of metabolic network curation and compression proposed by Terzer (subsection 2.10.4), are of major importance. They are thankfully encoded in the EFMTool compression process, thus when using EFMTool compression code, the analysis is performed right away. More specifically, we used code from the CNAPy library [138] (or, equivalently, `strainDesign` [243]). We retrieved from the helper GitHub repository `efmtool_link` [W16], which calls EFMTool's Java archive through Python interface code.

The compression of enzyme subsets performed by EFMTool in Java is very reliable. We developed our own enzyme subsets compression with a Python procedure with SymPy, a rational arithmetic library [S124]. Indeed, since the goal of *nullspace analysis* is to get linearly dependant rows of the kernel matrix, the kernel must be computed with exact methods, and expressed as rationals, so that correct α ratios are then identified. However, performance with SymPy was lacking when compared to EFMTool. We do not recommand using floats or approximation methods for kernel computation, exact Gauss-Jordan should be performed.

The code from CNAPy also performs FVA from COBRAPy [72] to exclude reactions that are blocked, *ie.* flux that never occurs, after application of the growth medium. Thus, the process of compression is also highly dependant on the growth medium, which has significant impact on the exchange reactions flux bounds. Finally, we use our MPARSER Python module to convert the compressed network into ASP logic programs. Since we work with compressed networks, constraints must apply on those.

Note that although this has to be checked further, we are unsure that the code we retrieved from the Klamt lab Python interface for our tools performs full metabolite compression as described by Terzer. This might be an element of future improvement for our methods.

We found that the consistency analysis described in subsection 2.10.3 developed by Gevorgyan was not applicable. That method was tested on the 2004 model by Ross Carlson [108] (with one intentional inconsistency we added by hand) and on a model of starch production in potato tuber (30 reactions and 31 metabolites, with 7 out of them unconserved, and 123 elementary leakage modes, all caused by a single input error) [S125, S126]. This was done by rewriting the MILP implementations in ASP, for which we present the code in Listing A.9 and Listing A.11.

However, this led to unsatisfying performance, even when using logic programming with linear constraints. Indeed, for a single error on a small network, we retrieve an amount of minimal net inconsistencies so considerable that enumeration might as well run for more than a day. And as well, once elementary leakage modes are obtained, it was not possible to link that set of reactions with inconsistencies back to single faulty metabolites or reactions. As a result, we opted for the implementation of network curation by Shin and Hellerstein, called GAMES [173].

On the C2M2NF base model, we retrieve an inconsistency presented in Listing A.12. GAMES retrieves reaction isolation sets, which are minimal sets of reactions describing a mass balance inconsistency, in the same vein as elementary leakage modes; and metabolite isolation sets, a similar notion but with minimal sets of metabolites [173].

Finally, additional inconsistencies with the solution space shape might occur, caused by stoichiometry, flux bounds, and additional linear constraints. These can be simply tested for by trial and error and noticing infeasibilities of the network (see LP infeasibilities, Figure 2.5). We advise for the use of *cplex* and its conflict refiner to detect and resolve LP infeasibilities present in the model. This was an useful tool when working with C2M2NF(S), and for detecting appropriate linear constraints as will be described in the next section. In fact, in *clingo[LP]*, the conflict refiner (see section 5.5) uses this knowledge to detect exactly the minimal constraints in conflict.

The model is very well curated, in fact it was curated by Jean-Pierre Mazat by hand multiple times so that no inconsistencies appear. However, there are still inconsistencies that only appear when summing ≈ 60 reactions together (Listing A.12), which is not something that can be seen by eye. In conclusion, we did not manage to solve all of the inconsistencies.

It goes without saying that if even small hand-curated models can contain inconsistencies of this scale, automatically generated genome-scale models should not be used without caution. The MEMOTE [169] tool for assessing quality of genome-scale models contains stoichiometry consistency checking, using the method by Gevorgyan. However, in practice, MEMOTE automatically skips it for the reasons mentioned above: the computation of inconsistencies is too expensive on large models. Thus, we advocate for the use of GAMES, which runs much faster (Figure 3.7).

Another network curation and compression process was devised for the bacterial models of *Staphylococcus aureus* and *Pseudomonas aeruginosa*, which are actual genome-scale metabolic models, where the effects of growth medium restriction, and network compression, are more apparent (chapter 4).

3.8.3 Devising an EFMs analysis from exometabolomics data

For the following section, we are going to base ourselves on the hypothesis presented in Hypothesis 3.8.3. We retrieved exometabolomics data provided by Jain and coauthors [32]. This dataset is called NCI-60 CORE data

for COnsumption (uptake flux) or RElease (production flux) data, and it represents 60 tumoural cell lines, from the following categories of tumoral tissues: Colon, Leukemia, Lung, Prostate, Ovarian, Breast, Melanoma, Central Nervous System, and Renal. Similarly to an analysis by Mazat [238], we computed the mean fluxes data across all 60 cell lines. The mean and standard deviation of the data is represented in Table 3.3.

Hypothesis 3.8.3 – Experimentally observed cancer-cells flux distributions fit linearly to EFMs
Given exometabolomics flux data D in fmol/cell/h, we suppose that if our model is well-constructed, there must be a flux balance distribution F in arbitrary units for which exchange fluxes fit linearly to the flux data from D. Considering E a set of EFMs in arbitrary units, F can be expressed as a linear combination of EFMs in E. Thus an EFM $e \in E$ is a linear fit to exometabolomics data D if and only if for every exchange reaction that is active in e, exchange fluxes fits linearly with the data D. We seek to find the EFM e with optimal fit to D.

The experimental data from Jain and coauthors includes an estimation of exchange fluxes for a total of 60 cancer cellular lines in fmol / cell / h. Considering the standard deviation and mean exchange fluxes of the cellular lines, we separated the global experimental observations: uptake, secretion, or either, into two categories: *hard constraints*, what we force as an input constraint for our computation, and *expected observations*, inputs we expect to observe in the minimal pathways, but do not force. The resulting constraint data is reported in Table 3.3.

aspefm was used to compute EFMs in the study. Warburg effect (glucose uptake and lactate production) and glutamine uptake were included as hard constraints, while the rest of NCI-60 observations were included as expected observations. Hard constraints and expected observations were modelled as logical constraints. Additional linear constraints were added. This allows for enumerating a smaller subset of minimal pathways observed with EFMs. A size constraint was added as well. The corresponding constraints are detailed in the next section.

Now let us summarize the methods for our study. We developed the C2M2NFS model as an extension of the C2M2NF model. The model comprises 150 reactions. In accordance with constraints describing Warburg effect, collagen production and inflammatory markers and growth factor VEGF-A synthesis, a subset of 747 EFMs was computed with *aspefm*. Then, the best EFM was selected according to classic linear regressions of the solutions to the mean exometabolomics data from Jain and colleagues using Python package Scikit-Learn [244]. A detailed look at our analysis workflow is presented in Figure 3.8.

3.8.4 Biological constraints for our analysis

Logical constraints given in input to *aspefm* are detailed in Table 3.3. The constraints are separated into two types: *hard constraints*, which are forced inputs for the computation, and *expected observations*, not forcing any input but forbidding the opposite observation. For example, the three hard constraints are: glucose must be consumed, lactate must be produced, glutamine must be consumed. And an example of expected observation would be aspartate should be consumed, *ie.* we forbid aspartate production.

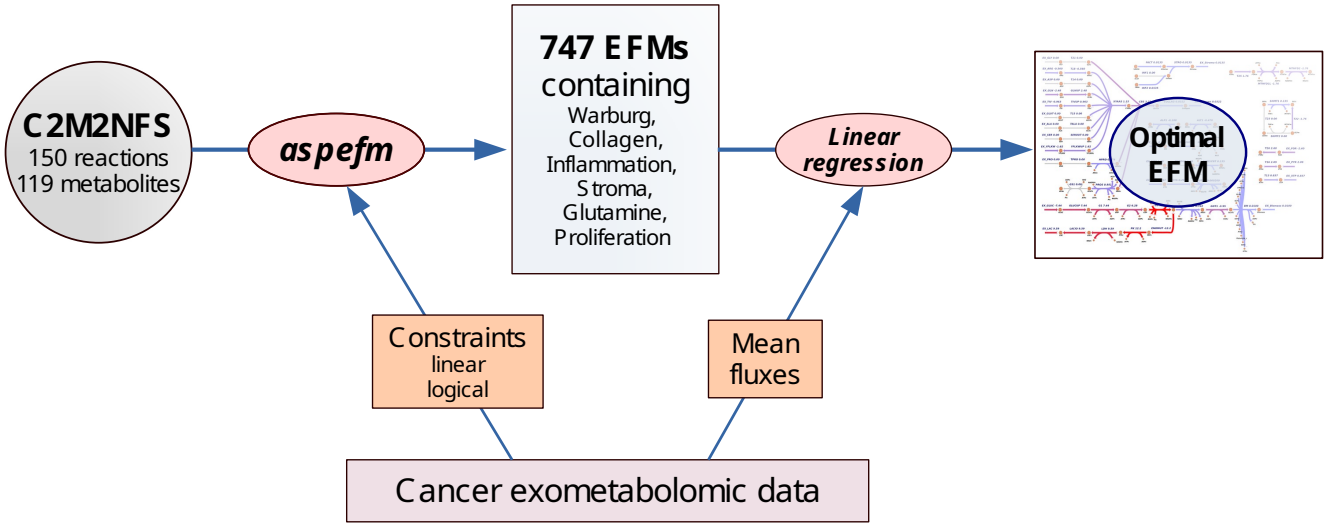


Figure 3.8: Methods for analysis of C2M2NFS. Cancer exometabolomic data from the NCI-60 cell lines - from Jain and coauthors - is used to produce constraints for aspefm computation and for selecting the optimal EFM once the program instances were stopped.

Metabolite	Glucose	Lactate	Glutamine	Glutamate	Serine	Glycine	Alanine	Proline	Asp Asn	Arginine
Mean +/- SD cancer cell exchange flux interval	-326.87 +/- 196.12	442.20 +/- 289.40	-82.48 +/- 56.20	13.54 +/- 16.99	-11.57 +/- 7.05	0.96 +/- 2.97	15.89 +/- 13.34	1.21 +/- 1.49	-3.33 +/- 3.39	-4.90 +/- 4.44
Corresponding constraint	**−**	**+**	**−**	+	−	+/-	+	+/-	−	−

Metabolite	TIV (Thr, Ile, Val)	YFLKW (Tyr, Phe, Leu, Lys, Trp)	XTP (Nucleotides)	Pyruvate	Formate	Histidine	Cysteine	Methionine
Mean +/- SD cancer cell exchange flux interval	-14.90 +/- 7.99	-19.80 +/- 10.57	0.10 +/- 0.22	Uncalibrated data	Data missing	Data missing	0.05 +/- 0.08	-2.11 +/- 1.23
Corresponding constraint	−	−	+/-	+	+/-	−	+/-	+/-

Table 3.3: Mean +/- SD exchange fluxes intervals from NCI-60 exometabolomics data, and resulting hard constraints and expected observations from the model, for glucose, lactate, XTP, pyruvate, formate, and amino acids. Minus (-) symbolizes uptake while plus (+) symbolizes secretion. Experimental observations are separated into two kinds: hard constraints (**bold** font) and expected observations (normal font), indicating different types of logical constraints.

Hard constraints are encoded as direct logical constraints: “reaction must be active”, meaning “reaction flux must be non null” - equation *(1)* - meanwhile expected observations (+) and (–) are encoded by forbidding reactions going in the opposite direction - equation *(2)* (see equation 3.39).

$$\begin{aligned} &(1)\ \forall R \in HardConstraints, && \nu_{R_{fwd}} > 0 \leftrightarrow z_{R_{fwd}} && (R_{fwd}, R_{bwd}) \in Reversibles \\ &(2)\ \forall R \in ExpectedObservations, && \nu_{R_{bwd}} = 0 \leftrightarrow \neg z_{R_{bwd}} && (R_{fwd}, R_{bwd}) \in Reversibles \end{aligned} \tag{3.39}$$

Given that $\nu_R\ \forall r \in R$ represents the flux of each reaction R in the set of all reactions R, and that $z_R\ \forall r \in R$ are *aspefm* indicator Boolean variables of when a reaction is active, *ie.* when its flux is non null. Logical constraints are enforced by setting *aspefm* Boolean indicators to be $\{True\}$ or $\{False\}$, taking advantage of the hybrid nature of the solver.

For example, the hard constraint “lactate must be secreted” will be encoded by the constraint “flux going into forwards reaction of lactate production must be strictly positive”, while the expected observation “alanine must be secreted” will have the constraint “flux going into backwards reaction of alanine production must be null”.

Although the metabolic model is calibrated so that metabolite flux could be correlated to units of metabolite flux in fmol/cell/h, it is best not to assume any units for the flux going into pathways, as EFMs are minimal vectorial solutions, the extreme rays of the model’s flux solution space. As a consequence, we decided linear constraints should be arbitrary as well.

Namely, the constraints we added are described by the following equations: *(3)* bounds each flux by the arbitrary value of 15 *(4)* forbids total flux to surpass the arbitrary value of 500, *(5)* constrains the PMFm metabolite flux to under 1, thus keeping mitochondrial proton-motive force among realistic values (0 to 1 V), *(6)* constrains biomass production flux to over 0.01, as to keep a minimal amount of flux going into tumoral growth, *(7)* constrains collagen production flux to be over biomass production flux, *(8)* constrains inflammatory marker response to be over biomass production flux, finally *(9)* and constrains glucose uptake flux to be over 0.1 (low glucose uptake conditions), or over 1 (high glucose uptake conditions) depending on the aspefm program instance (see equation equation 3.40).

$$\begin{aligned} &(3)\ \forall r \in R, \nu_r < 15 \\ &(4)\ \sum_{r \in R} \nu_r < 500 \\ &(5)\ \nu_{PMF} \leq 1 \\ &(6)\ \nu_{BIOMASS} \geq 0.01 \\ &(7)\ \nu_{COLLAGEN} > \nu_{BIOMASS} \\ &(8)\ \nu_{STROMA} > \nu_{BIOMASS} \\ &(9)\ \nu_{GLUCUP} \geq 0.01 \end{aligned} \tag{3.40}$$

Given that $\nu_r\ \forall r \in R$ represents the flux of each reaction r in the set of all reactions R.

In addition, a size constraint for number of reactions of EFMs was added :

$$(10)\ card\{r \in R \mid v_r \neq 0\} < 60 \tag{3.41}$$

The size constraint of equation *(10)* forces EFMs size to be below 60 *active* reactions - reactions with non-null fluxes (see equation 3.41). This bound was chosen to correspond to about 10 reactions more than the smallest retrieved EFM solution. Such a size constraint takes full use of the *aspefm* solver's constraint programming origins.

Elementary Flux Modes are computed with *aspefm*, a logic programming tool based on *clingo*'s SAT solver technology [211, 210] and leading linear programming tool *cplex* [W8]. For each solution found by aspefm, the rank test is performed [125]. If the rank of the submatrix indexed by the current solution mode is equal to the number of active reactions minus one, then that mode is elementary. If not, then the solution should be excluded and another one should be searched.

This method is relatively computationally fast with fewer constraints, but struggles on largely constrained models as this one, as the majority of solutions found will be combinations of EFMs such that constraints are respected, or Minimal Constrained Flux Modes (MCFMs) [157].

In regards to this issue, the computation was set to run for longer, in our case 3.5 days, before being stopped. Decomposition methods were envisaged to retrieve EFMs from MCFMs, but the idea was abandoned as it was deemed just as computationally expensive, especially considering said resulting EFMs would very likely not respect our imposed constraints.

Multiple instances of aspefm were launched at the same time to compensate for its lack of parallelization. All executions are non-deterministic due to the SAT-solver's random decisions, meaning overlap of solutions between instances is not guaranteed. Once all executions were stopped, all results were gathered and 560 duplicates were removed.

3.8.5 Finding the optimal EFMs with linear regression

After running enumeration of EFMs with aspefm, 747 unique minimal pathways were obtained. These metabolic pathways span a large diversity of possible inputs and outputs, including the varying flux values of internal metabolic reactions. The statistics of fluxes of the principal reactions of interest are plotted in Figure 3.10. A complete view of the statistics of exchange fluxes in our EFMs can be found in Table A.4.

All obtained EFMs show Warburg effect, tumoral growth, production of collagen and either inflammation markers IL1β or TNFα, representing cell inflammation, and growth factor VEGF-A, representing neoangiogenesis. And in particular, as asked by our constraints, more flux is going into collagen synthesis and factors recruitment than biomass. As well, all EFMs either respect observations from Table 3.3, or do not input any flux into the exchange reactions.

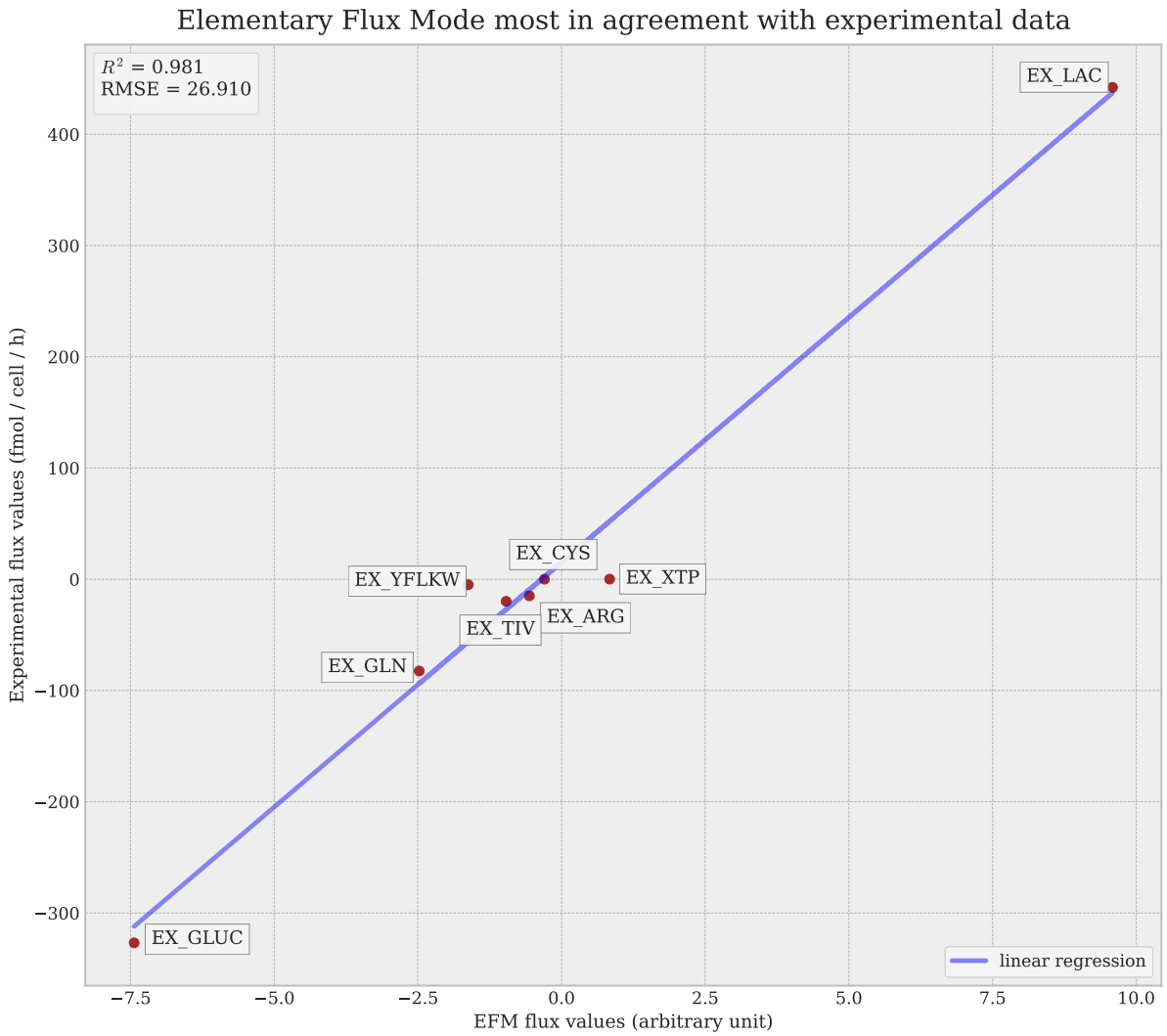


Figure 3.9: Linear regression of the EFM with the best fit to mean flux data from all NCI-60 cancer lines

Aside from glutamine which is a forced input of our model, the only amino acids clusters observed to be consumed in all minimal pathways for tumoral growth, production of collagen and either inflammation markers were cysteine, histidine, TIV and YFLKW. Other amino acids, such as arginine, serine, aspartate and asparagine, might or might not appear to be consumed in EFMs. Similarly, secreted-only amino acids such as glutamate, alanine often did not appear to be produced. An interesting point of note is the high variability of exchange fluxes EX_FOR (SD: 2.40) and EX_GLY (SD: 3.36). Along with methionine (SD: 0.60), and serine (SD: 3.30), which can only be consumed, these four fluctuating exchange reactions appear related to usage of the tetrahydrofolate (THF) cycle.

In order to select a single specific pathway of interest for our analysis, we took all elementary modes from our analysis and correlated their non-null uptake fluxes to the mean uptake flux among all 60 NCI-60 cancer cell lines, as described in Table 3.3. The pathway with the best fit to the experimental data (R^2 of 0.98, RMSE of 26.9) was represented in Figure 3.11. The corresponding linear regression analysis was represented in Figure 3.9. Visualization of the reactions was done through the EscherPy Python package [191].

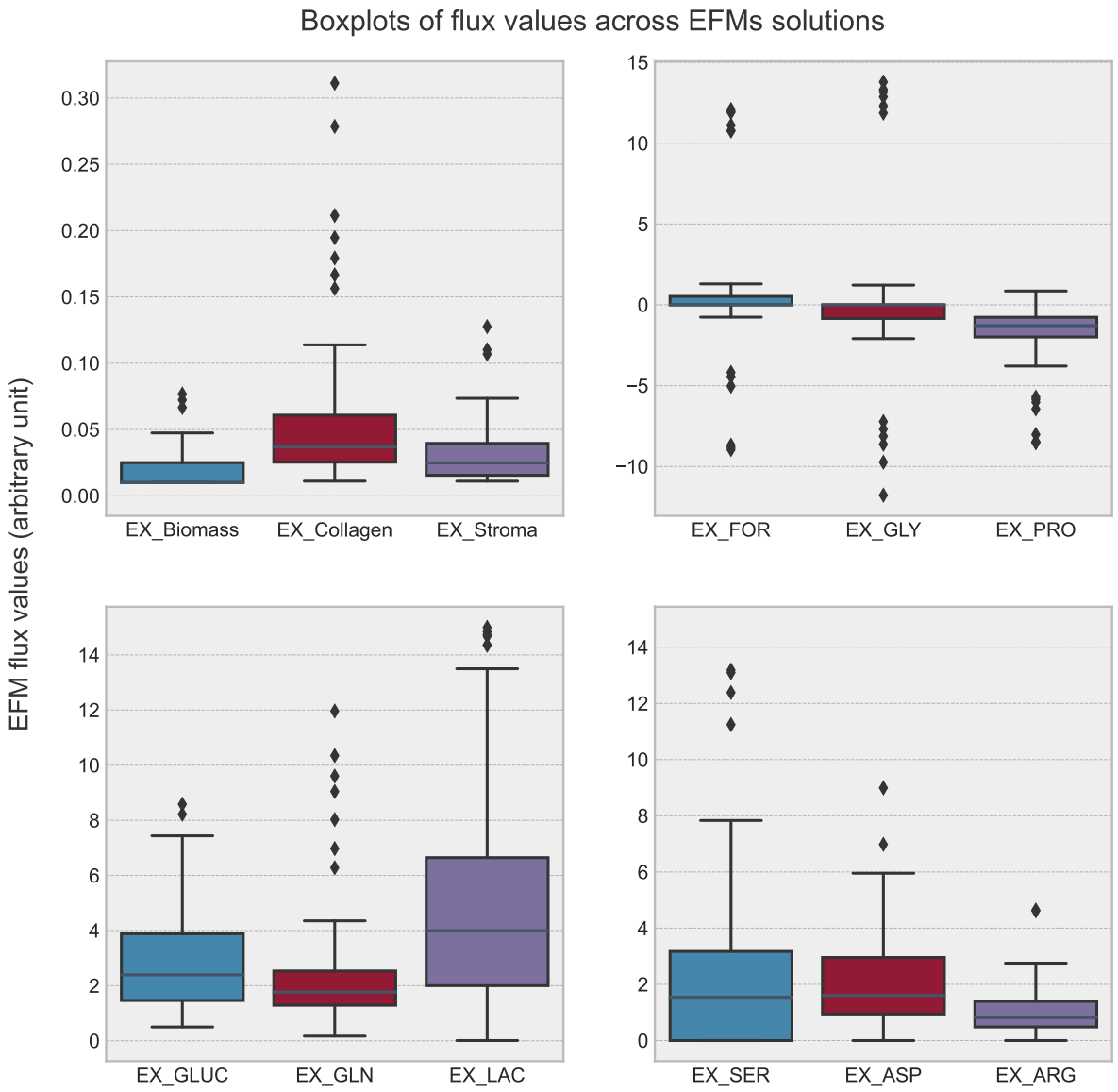


Figure 3.10: Statistics of our principal reactions of interest among all 747 EFMs. EX_Biomass: tumoral growth, EX_Collagen: collagen production, EX_Stroma: recruitment of inflammation markers IL1β or TNFα, and growth factor VEGF-A, EX_GLUC: glucose consumption, EX_GLN: glutamine consumption, EX_LAC: lactate production, EX_FOR: formate consumption or production, EX_GLY: glycine consumption or production, EX_PRO: glycine consumption or production, EX_SER: serine consumption, EX_ASP: aspartate and asparagine consumption, EX_ARG: arginine consumption. In the case of EX_FOR, EX_GLY, EX_PRO, negative values represent consumption, and positive values represent production.

This pathway is characterized by the secretion of TNFα as the inflammatory response marker, VEGF-A as a representant of neoangiogenesis, consumption of the following amino acids: glutamine, cysteine, histidine, arginine, TIV, and YFLKW; collagen production and Warburg effect. Nucleotide synthesis is performed above biomass requirements which results in nucleotide secretion. The tetrahydrofolate cycle is used through cytosolic reaction SHMT1, and mitochondrial reactions MTHFD1L, MTHFD2.

No external consumption of glycine or proline is observed, indicating that in the case of this elementary pathway, amino acids glycine and proline, going into collagen synthesis and composing about 50% of the collagen content, are synthesized de novo, purely through other metabolic reactions and catabolism of other amino acids.

In particular, fluxes of notice include high glutamine consumption (-2.48), which is converted into 2.01 units of glutamate through nucleotide synthesis by the NUC reaction (0.87). Glutamate is also obtained from α-ketoglutarate (αKG) using the GOT1 reaction (-3.95). From these 5.96 units of glutamate, 3.25 units go into mitochondria and get converted into αKG, from which 0.24 units are transformed to citrate through reverse tricarboxylic acid cycle usage, then to oxaloacetate and acetyl-CoA, contributing to biomass lipids production. Among the remaining cytosolic glutamate units, 0.72 units are converted into serine by SERSYNT, 0.95 units are used for proline production, 0.48 units are used for alanine synthesis, 0.43 units are used for VEGF-A and TNFβ production, 0.11 for collagen formation. And from the produced serine units, 0.13 units are converted into glycine through the use of SMHT1.

Thus, glutamate can be converted into proline by reaction PROS, hydroxyproline through reaction HPRO, and glycine through reaction SERSYNT and then use of the THF cycle with reaction SHMT1 to convert serine into glycine, making up for the three principal collagen consituants.

We believe that the best EFM fit not importing either of those collagen consituants and making use of glutamate and SERSYNT to produce proline, serine and glycine de novo is a result of major importance shown by our model and our methodology. As R^2 decreases, the use of SERSYNT may be replaced by the uptake of glycine and serine. Equivalently, proline and other amino acids might be imported from the extracellular medium.

3.8.6 Proposing a schematic model of the tumoural stroma

By taking the EFM most in accordance with physiological data, which displays rates of collagen production and inflammatory markers synthesis constrained to be above biomass production, we achieve a new methodology in constrained-based modelling, vastly different from the usual hypothesis that reaction fluxes in the cell only contribute to optimizing its growth. Our methodology is able to highlight, in accordance with the experimental data, which amino acids are the most pivotal for the synthesis of our four proteins of interest – collagen, VEGF-A, IL1β, and TNFα.

Jain and coauthors found that glycine and THF cycle held a pivotal role in cancer cell metabolism [32]. Since we select the best EFM according to their exometabolomics data, our model's solutions also corroborate that hypothesis. Additionally, we found that glycine, hydroxyproline and proline, the three major components of collagen, could be synthesized endogeneously solely from glutamine, which is converted to glutamate in our model by using the nucleotide synthesis reaction. Glutamate is easily converted into proline through 1-Pyrroline-5-carboxylic acid, and

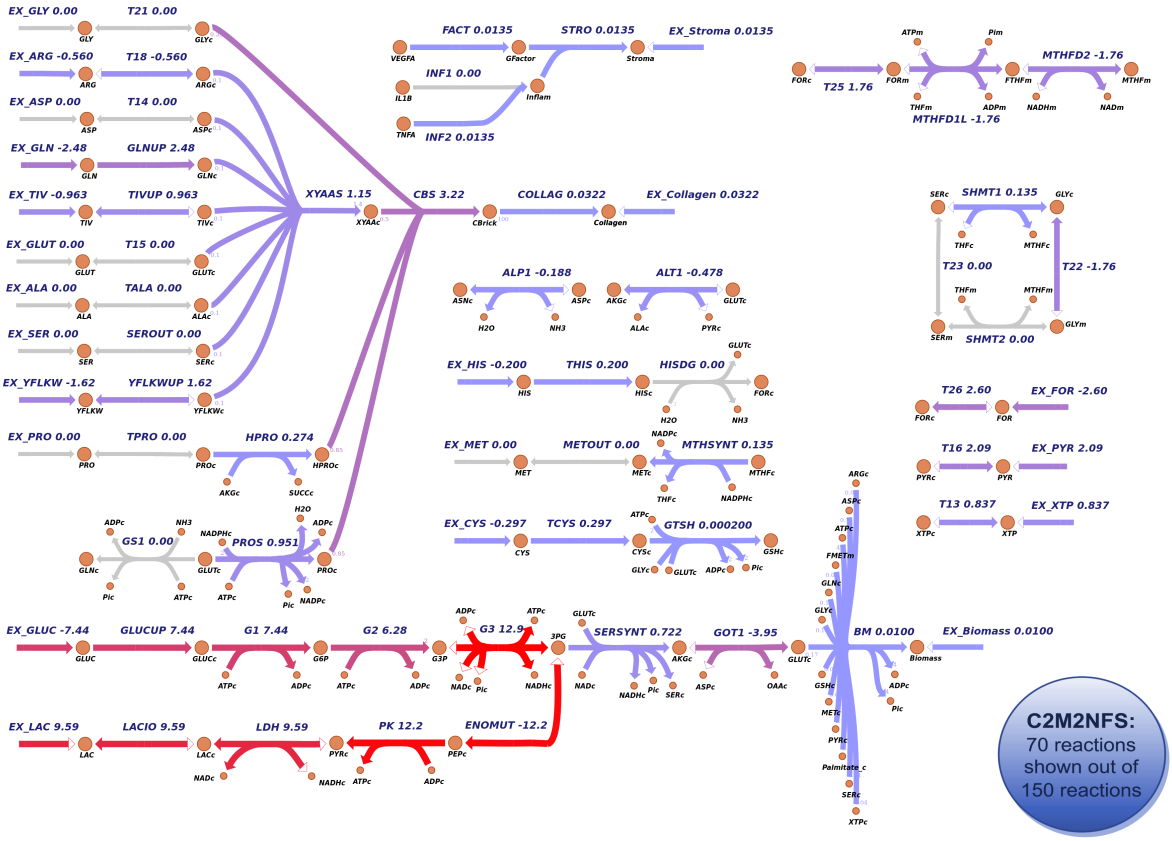


Figure 3.11: EFM with the best fit to mean exchange fluxes from NCI-60 exometabolomics data. 70 reactions of most interest of C2M2NFS are shown, including most cytosolic transporters and some mitochondrial transporters but not mitochondrial TCA Cycle.

then to hydroxyproline [245]. Then, the use of serine synthesis from glutamate and THF cycle allows the cell to obtain its glycine requirement to synthesize collagen, presenting a phenotype similar to the one observed in the stroma around cancer cells. No extracellular proline, glycine or serine import could in fact be needed.

In light of these findings, we devised a graphical model of how the tumoral stroma might be conceived, presented in Figure 3.12. The graphical model includes the main findings observed in our optimal pathway, namely: glucose is fermentated into lactate through glycolysis, and glutamine is converted into glutamate, which is transformed into the main amino acids for collagen production, and into acetyl-CoA lipid bricks helping tumoral growth. Finally, the amino acid pool formed through amino acid biosynthesis and amino acid uptake is used to synthesize inflammation and neoangiogenesis markers, helping to recruit the cells composing the tumoral stroma.

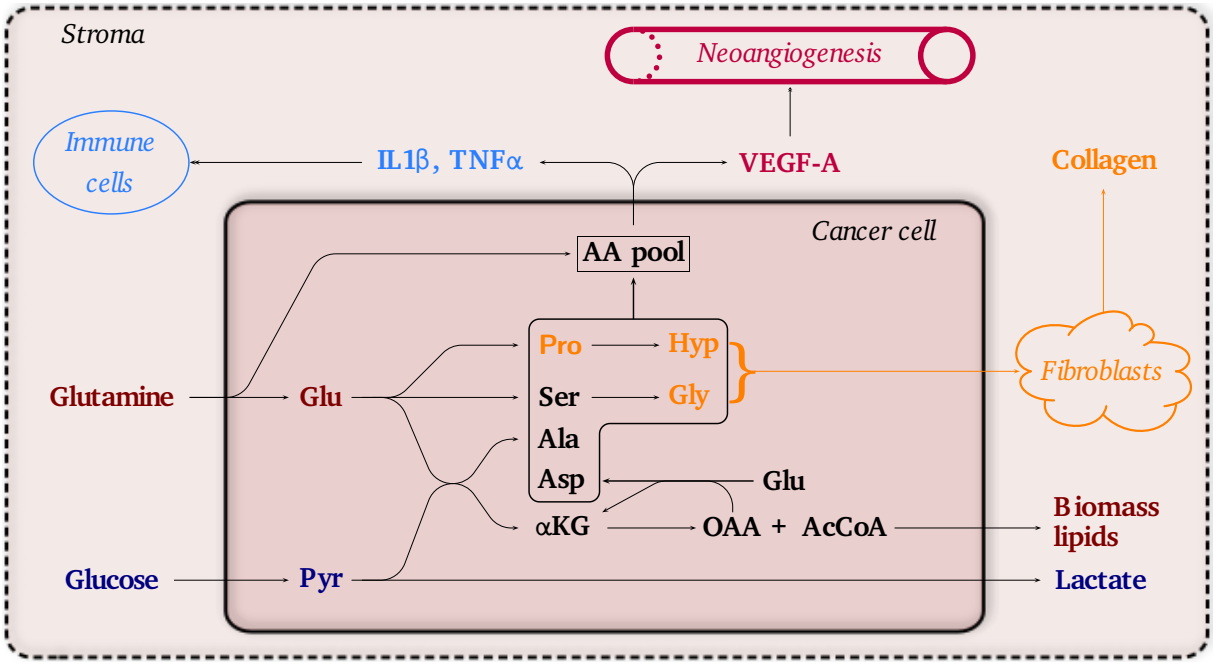


Figure 3.12: Explicative model of tumoral stroma production in light of amino acid metabolism and the Warburg effect. Two parallel pathways are observed, glycolysis and glutaminolysis. Abbreviations are amino acids three letter codes, Hyp: hydroxyproline, AA: amino acids, αKG: α-ketoglutarate, Pyr: pyruvate, OAA: oxaloacetate, AcCoa: acetyl-CoA.

It is interesting to note that the two renowned cancer hallmarks, glycolysis and glutaminolysis, undergo parallel pathways, glycolysis leading to lactate acidifying the tumoral stroma medium and glutaminolysis leading to increased production of biomass lipids and collagen. Whether or not collagen is produced by the main tumoral cell or by cancer-associated fibroblasts [246], and whether or not fibroblasts themselves undergo Warburg effect as well, as has been hypothesized [247, 248], is not of relevance to our unicellular metabolic model. Indeed, if collagen may not be produced by the tissue cell, it follows that the glycine, proline, and hydroxyproline overproduction by our tissue cell will lead to extracellular export of those amino acids into the medium, which will then be picked by fibroblasts to increase their production of collagen. Alternately, our cell model could be representing a cancer-associated fibroblast instead.

3.8.7 Comparing our method to parsimonious Flux Balance Analysis and flux sampling

A parsimonious FBA (pFBA) solution was used for comparison with the linear regressions of EFMs, computed with COBRAPy [72]. As explained previously, FBA is a method which requires optimization of an objective function. Meanwhile, EFMs conduct an unbiased decomposition of a metabolic model into its minimal functional units. The results of the comparison analysis to pFBA is presented in Figure 3.13, Figure 3.14 and Figure 3.15.

For the solution obtained with Parsimonious Flux Balance Analysis, the R^2 fit to experimental data (Figure 3.14) scores as low as the 10% worst EFM solutions (Figure 3.13). The solution is visualized in Figure 3.15. A clear bias is seen towards the objective function, which was set to sum of production fluxes BM, COLLAG and STRO. As an example, the flux of reaction CBS is saturated to 15.0 which is the upper bound set to every flux. No Warburg effect is shown, despite being suggested by the constraints given.

Both inflammation markers IL1β and TNFα are produced, while only one of those must have been necessary at a given time. These characteristics of the COBRApy pFBA solution illustrate the drawbacks of relying on an objective function. Meanwhile, EFMs analysis performs no maximization and returns a diverse panel of minimal pathways that the cell can alternate between at each given time.

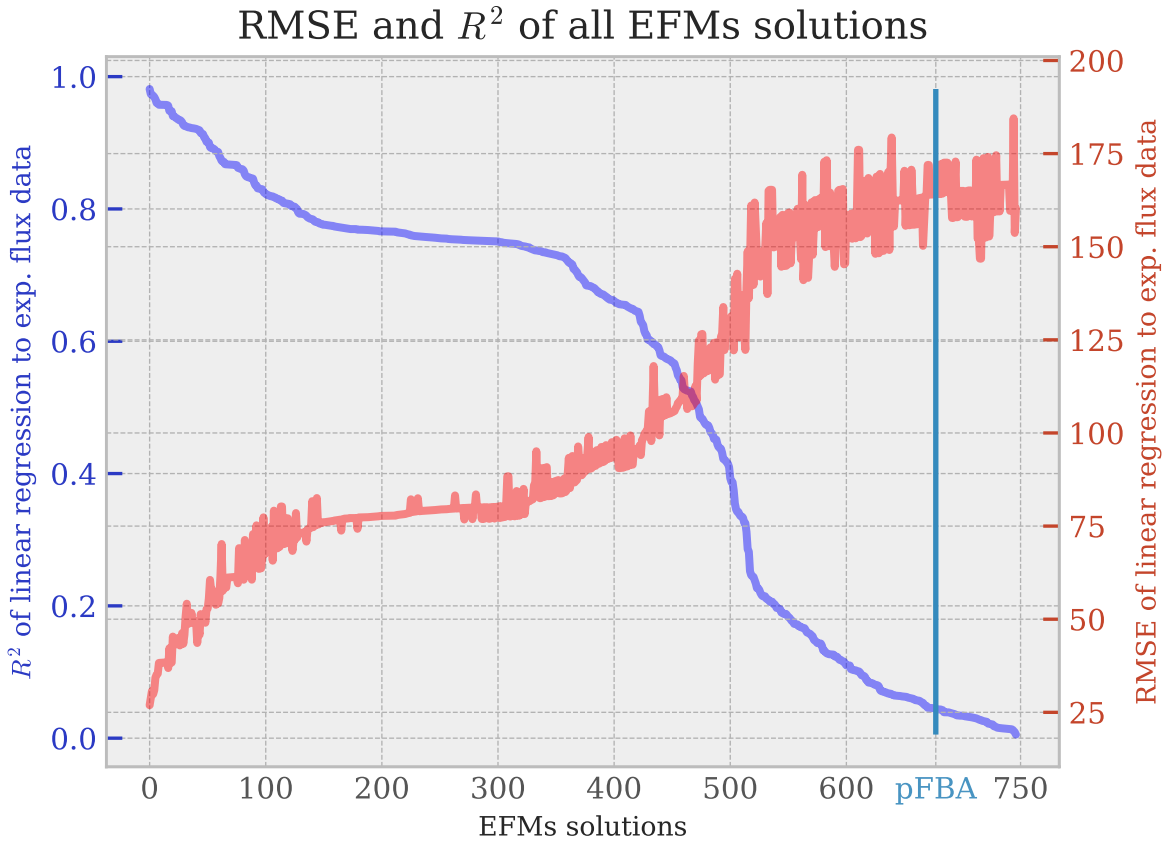


Figure 3.13: RMSE and R^2 scores of linear regressions to exometabolomics data applied to all 747 EFMs solutions. Indicated by a vertical blue line, the R^2 score of the pFBA solution was reported on this graph for comparison with the R^2 of the EFMs solutions

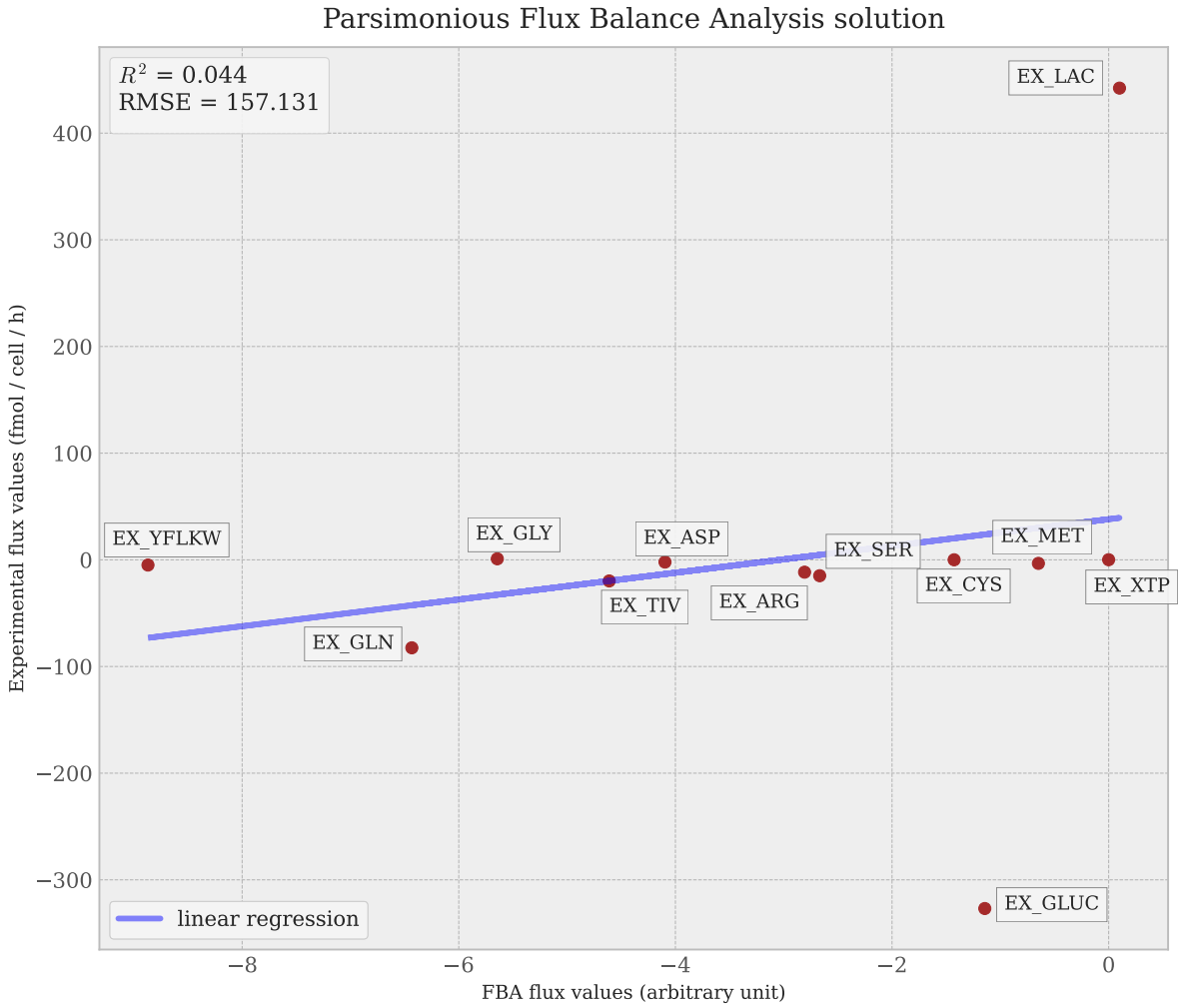


Figure 3.14: Linear regression of experimental flux values to the parsimonious FBA solution

Over the years, interest in the constraint-based modelling community has mostly been oriented more towards Flux Balance Analysis-related methods than Elementary Flux Modes, as the latter are effectively more time-consuming and expensive to compute. However, Flux Balance Analysis has the drawback of partaking in maximization of an objective function, which is an assumption that cannot necessarily correlate with biological observations. The solution obtained from Parsimonious Flux Balance Analysis (pFBA), a bilevel optimization problem with minimization of the sum of reactions on top of maximization of the objective function, was unable to fit to the experimental data (R^2 = 0.044), scoring as low as the ten percent worst Elementary Flux Modes (Figure 3.13, Figure 3.14). As well, a clear bias was seen towards the objective function in the fluxes shown in Figure 3.15, with fluxes being saturated, both inflammation markers being synthesized while only one might have been enough, and three more amino acids were found imported than for the optimal EFM solution. Thus, in the case of this application to cancer, we argue that the maximization approach, optimizing the sum of flux going into tumoral growth, collagen formation and inflammation markers production, is not an adequate answer to our problem.

One might also suggest the use of flux sampling rather than EFMs, with methods such as OptGpSampler [122].

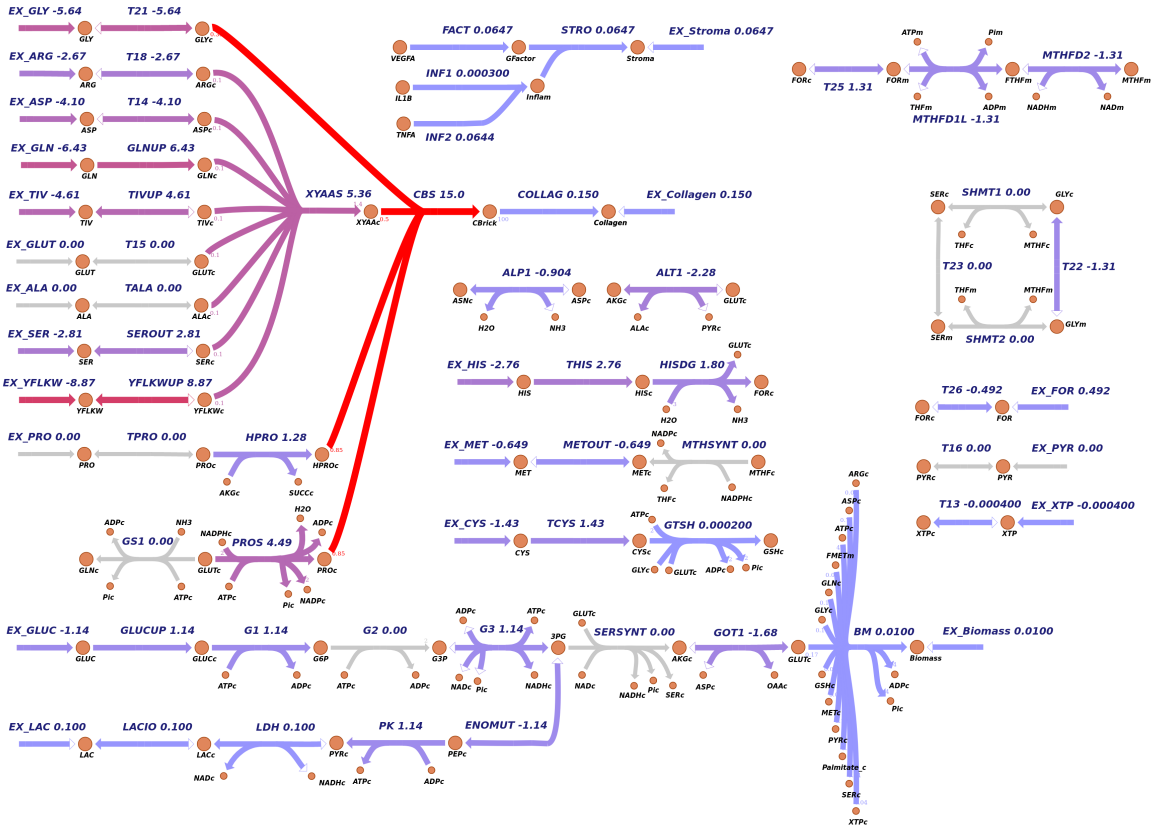


Figure 3.15: Escher visualization of the Parsimonious Flux Balance Analysis optimal solution obtained with the same as constraints as for Elementary Flux Modes Analysis

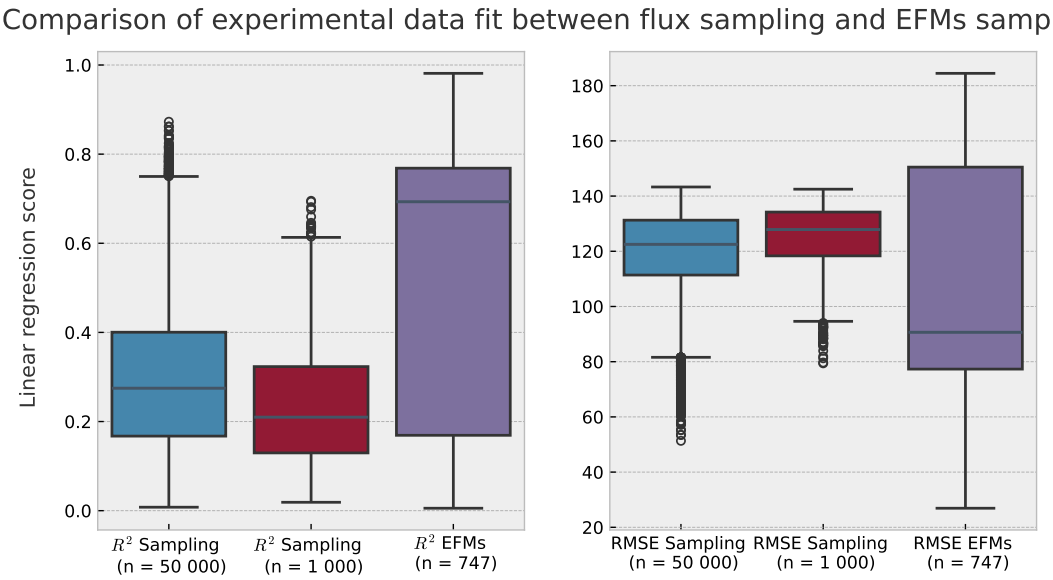


Figure 3.16: Boxplots of regression score values - RMSE and R² for the flux sampling of 1 000 or 50 000 solutions obtained with OptGPSampler compared to EFMs of this study

However, perhaps due to the different nature of solutions sampled, we found that flux sampling did not answer to our problem quite as well as EFMs, achieving a maximum R^2 to experimental data of 0.70 with a sample of 1 000 solutions, and a maximum R^2 of 0.87 with a sample of 50 000 solutions (Figure 3.16). While, with a sample of 747 EFMs, we were effortlessly able to reach a maximum R^2 of 0.98. Therefore, we believe there remains a clear need for Elementary Flux Modes analysis, may it be exhaustive or constrained. Through our findings, by focusing on enumerating subsets of EFMs, and taking the constraint-based approach to its extreme, by adding as many constraints as possible, solutions obtained are also closer to biological reality.

Although our C2M2NFS metabolic model of 150 reactions would be considered of a relatively small scale by today's standards, enumeration of EFMs using EFMTool could not finish or yield any results at all [139]. This due to the way its implemented algorithm, Double Description, works [133]. Previous attempts on networks of around this size had to split the model in multiple smaller networks to complete enumeration [144], or resorted to MILP, which may be restricted by minimizing solution size [140]. Another kind of method emerged for networks over this size using Lexicographic Reverse Search [146]. On the other hand, *aspefm* can yield results on metabolic networks of this size and over, up to thousands, very easily, and it is able to handle biological constraints, which is ultimately the goal in EFM analysis. The tool can yield any new EFM in reasonable time – however, downsides appear from choosing constraints that are too difficult to filter out, which unfortunately includes most linear ones [150, 157]. When that is the case, *aspefm* spends most of its time filtering out solutions that are not EFMs instead of finding EFMs, and its enumeration must thus be eventually stopped with a time limit – in our case 3.5 days. These are points of major improvement for our tool in the future.

3.8.8 Discussing the scope of our analysis

The debate on whether cancer is a genetic disease or merely that cancer cells is unable to oxidize glucose properly and rely on fermentation is still not settled. Warburg postulated that cancer relies heavily on glycolysis and that pyruvate is preferentially converted to lactate instead of continuing to Krebs cycle to be converted to ATP by the oxidation phosphorylation where oxygen is needed [249]. This phenomena - "aerobic glycolysis" - is now recognized as a hallmark of cancer [240, 250]. On the other hand, it has been shown that the stroma plays a key role in tumor development and progression. The stroma is marked by appearance of cytokines such as IL1β and tumor necrose factor TNFα, and growth factor VEGF, causing recruitment of new immune cells and blood vessels [240]. As well, glutamine metabolism is accelerated in cancer cells as opposed to healthy, non-proliferating cells, consequently the demand is larger to meet the energy and biomass production needs [251, 252]. Combining calorie-restricted ketogenic diet with glutamine targeting in late-stage experimental glioblastoma has shown clear therapeutic benefits [253]. The triple helix shape of collagen is made up of three polypeptide chains that are coiled around one another. Glycine, proline, and hydroxyproline residues are prevalent in these chains [241]. The production of hydroxyproline, which is necessary for the stability of the triple helix collagen structure, is facilitated by glutamine [254, 255, 256]. Interestingly, our model finds that endogeous collagen production is possible from glutamine only, with no glycine or proline uptake requirement. This result strongly suggests a possible key role of glutamine in the formation of collagen in cancer.

Mazat and Ransac's model, C2M2NF, which we extended for this study, proves to be a great tool for exploring glutamine metabolism, independently of glucose [237]. Similarly to a study by Mazat [238], we retrieved the exometabolomics cancer cells dataset by Jain and collaborators. In their dataset, Jain and collaborators categorize uptake and secretion fluxes of sixty cell lines by their origin tissue [32]. To determine the EFM best in agreement with experimental data, we took the EFM with best linear regression fit to mean flux values of all cell lines regardless of their origin tissue. In order to further the analysis, we took our optimal EFM, and attempted linear regression against only specific types of cell lines. We found that our EFM showed no specificity to any of the tumor cell lines, achieving highly similar scores in all cases (Table A.5). Alternatively, the issue of tissue specificity, which is of great interest for our study, could be achieved by using larger-scale organ-specific metabolic models [171].

While this EFMs analysis is modelling at a medium to large-scale level, it should be kept in mind that the many selected constraints apply. Selected biological constraints are of major importance for us to keep the number of solutions to manually analyze low. However, the analysis being very constrained means that smaller and larger elementary metabolic pathways also descriptive of biological processes of interest might have been filtered out by our methodology. It should also be noted that the steady-state assumption for intracellular metabolites is a strong hypothesis, ignoring all internal thermodynamic and time-dependant processes at play. Finally, other modellers might consider a smaller-scale level analysis such as the one presented in Braakman and Smith [257] appropriate. Or, alternatively, a larger-scale, extracellular view of the mechanisms in play in collagen formation and recruitment of the multifunctional stroma might be of interest. In particular, metabolic modelling has recently seen a number of advances: the construction of a whole human body metabolic model [171], and particular emphasis on metabolic interactions between cells at the multicellular level [193].

In conclusion, we suggest that the Warburg effect is correlated to the formation of the stroma and particularly to the synthesis of collagen, which plays a key role in cancer progression and metastasis. Metabolic pathways analysis suggests that the collagen production phenotype displayed by fibroblasts and Warburg effect might occur at the same time, and without extracellular import of the macromolecule's main components, glycine and proline. As well, in the process of their synthesis for tumoral growth, amino acids might be recycled into cytokines to recruit immune cells and new blood vessels forming the stroma. Cancer cells act like primary producers of the tumoral ecosystem: their wastes exert an ecological pressure on their environment and modify the surrounding landscape. The rerouting of resources and wastes by the newly formed stroma and vascular network has an impact on the larger scale of the organ and the full organism. By proposing a tumor metabolic model at the unicellular level encompassing the properties of the tumor stroma we open the road for further analyses of the impact of Warburg effect at the tissue level.

Chapter 4

Minimal Cut Sets with *aspefm* reveal new bacterial interactions

Minimal Cut Sets are minimal cuts in metabolic networks disabling certain target functions. They can be used in metabolic engineering contexts as well as simply in the research of reactions essential to a growth medium, of therapeutic relevance. The standard analysis for Minimal Cut Sets is leading to the search of synthetic lethals (SLs) – set of two or more reactions for which removal leads to death of the cell. This is usually limited to sets of four or less reactions. Indeed, it is often hard for a synthetic biologist to deactivate more than three targets at once.

However, *aspefm* excels in finding reaction sets of four or more reactions. In this chapter, we extend the computation of *aspefm* to MCSs, and we broaden the scope of MCSs analysis to include reaction sets of four or more reactions. We demonstrate that these MCSs are of major interest and apply the method to a consortium of two pathogenic bacteria, *Pseudomonas aeruginosa* and *Staphylococcus aureus*. As well, we explore the significance of metabolite exchanges in the context of finding therapeutic solutions against the bacteria.

4.1 Implementation of Minimal Cut Sets in *aspefm*

We can extend our workflow to compute Minimal Cut Sets with *aspefm*. Minimal Cut Sets with the biomass synthesis reactions as the target reaction identify which reaction cuts lead to lethal phenotypes. *aspefm* compared to the other principal computation methods of MCSs presented in Figure 4.1.

aspefm uses the fact that MCSs can be computed as the EFMs of a dual metabolic network, after a conversion process [186] (see subsection 2.12.1). As we can see in Figure 4.2, like with EFMs, we can add any constraints we want, namely specify a size limit for MCSs, specify reactions that are wanted or unwanted – see subsection 3.5.1 for the correponding formulation.

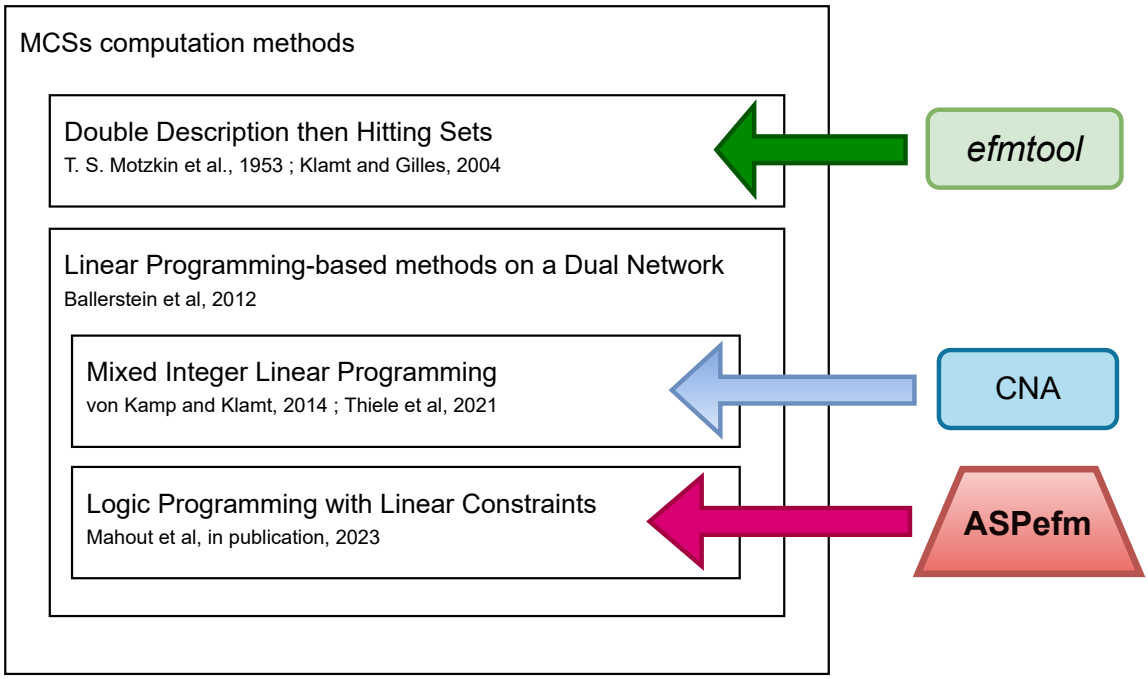


Figure 4.1: *aspefm* compared to the other principal computation methods of MCSs

In *aspefm*, we formulated the problem by making use of the MILP version proposed by von Kamp and Klamt in 2014 [153], which excludes some of the linear variables and constraints introduced by Ballerstein. A notable feature of the method by von Kamp and Klamt was defining an inequality constraint instead of an equality constraint for metabolites of the dual network that were originally irreversible reactions.

All reversible dual reactions are split into two irreversible dual reactions. As in the formalisms defined by Ballerstein and von Kamp, the reactions corresponding to reversibility constraints are the only ones to which subset-minimality applies, meaning the other linear variables are free to be either strictly positive or equal to zero following whether it suits the linear program. We give the LoPLC program constraints in subsection 4.1.2 and the ASP code in Listing A.13.

We first tested the computation of Minimal Cut Sets on *E. coli* core with the biomass as target reaction. We were able to retrieve all 352 MCSs of size three or less on that model in about 21 min. Generally, this is quite a bit slower than computing EFMs on the same network, but that's to be expected as the dual network contains more reactions than the original, meaning there are more logical and linear variables to be taken into consideration for our solvers.

4.1.1 Formalizing the dual stoichiometric matrix

Let us define S the stoichiometry matrix of size $m \times r$, m being the number of metabolites in the metabolites set $\mathcal{M}$ and r being the number of reactions in the reactions set $\mathcal{R}eac$. Let us define the set of reversible reactions $\mathcal{R}ev \subset \mathcal{R}eac$ and the set of target reactions $t \subset \mathcal{R}eac$ to be disabled for MCSs computation.

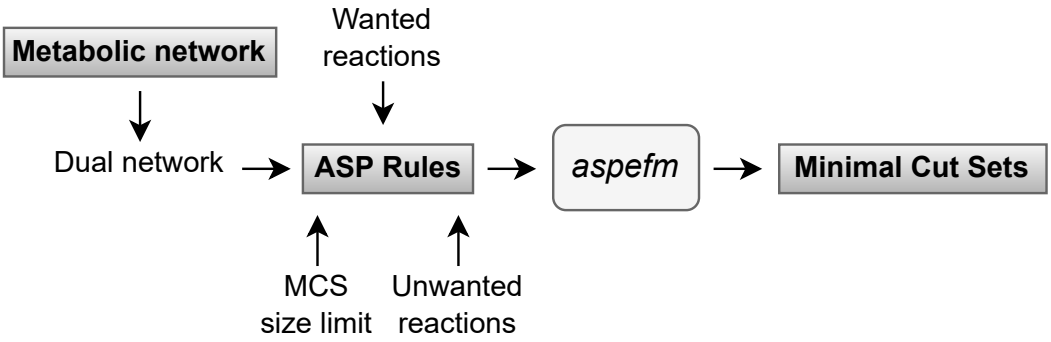


Figure 4.2: Current framework for the computation of MCSs with *aspefm*. Minimal Cut Sets can then be converted into gene knock-outs

In the dual metabolic network, original reactions become metabolites, and original constraints become reactions. If S the primal stoichiometry matrix is of size $m \times r$, with **I** the identity matrix of size $r \times r$ and $-\mathbf{T}$ vector of size $r \times 1$ with values 0 for $j \notin t$ and -1 for $j \in t$, then $\mathcal{D}$ is a dual matrix of size $r \times d$, with $d = m + r + 1$, and defined as $\mathcal{D} = (S^T \ \ \mathbf{I} \ \ -\mathbf{T})$.

The computation of the dual network $\mathcal{D}$ generates a metabolites set $\mathcal{M}_{dual}$ and a reactions set $\mathcal{R}eac_{dual}$. The set $\mathcal{M}_{dual}$ is simply the reactions set $\mathcal{R}eac$ and is thus of size r, while the reactions set $\mathcal{R}eac_{dual}$ is of size d and composed of three different types of reactions: $\mathcal{R}eac_{dual} = \mathcal{S} \cup \mathcal{RC} \cup \mathcal{T}$.

More precisely, all m stoichiometry constraints $\mathcal{S}$ become reversible reactions, r reversibility constraints $\mathcal{RC}$ become reversible if and only if the original reaction is reversible too, while the reactions t to be disabled become one irreversible target reaction $\mathcal{T}$.

After splitting the d dual network reactions into k irreversible reactions, the set $\mathcal{R}_{dual}$ of irreversible reactions is obtained. For clarity, $k = 2m + r + |Rev| + 1$. For MCSs computation, we are concerned with its subset of interest $\mathcal{C}ut \subset \mathcal{R}_{dual}$, corresponding to the split of reactions $\mathcal{RC}$. The set $Rev_{dual} : \mathcal{R}_{dual} \times \mathcal{R}_{dual} \rightarrow \mathcal{R}eac_{dual}$ keeps track of which reactions of $\mathcal{R}_{dual}$ were originally reversible in the reaction set $\mathcal{R}eac_{dual}$. This is the set of forwards backwards reaction pairs. Further ahead, we denote by D the dual matrix of size $k \times r$ after splitting reversible reactions.

4.1.2 Minimal Cut Sets formalization

Given $\mathcal{R}ev \subset \mathcal{R}eac$ the set of reversible reactions of the primal network, D the dual matrix of the stoichiometric matrix S, $\mathcal{M}_{dual}$ the set of dual metabolites, originally reactions of the primal network, $\mathcal{R}_{dual}$ the set of k irreversible dual reactions, $Rev_{dual} \subset \mathcal{R}_{dual} \times \mathcal{R}_{dual}$ indicating which pairs of reactions result from a split, $\mathcal{C}ut \subset \mathcal{R}_{dual}$ the subset of reactions corresponding to directionality constraints of the primal network, and $\mathcal{T} \in \mathcal{R}_{dual}$ a target reaction associated to one or several primal network reactions t that should be disabled, the Minimal Cut Sets problem can be defined as the following:

Problem: Find non trivial subset-minimal affectations to $\{True\}$ of $c_{\overline{r}} \in \mathbb{B}$, $\forall \overline{r} \in \mathcal{C}ut$ such that :

$$\sum_{\overline{r} \in \mathcal{R}_{dual}} D_{\overline{mr}} \times v_{\overline{r}} = 0 \quad \forall \overline{m} \in \mathcal{M}_{dual} \cap \mathcal{R}ev \tag{4.1}$$

$$\sum_{\overline{r} \in \mathcal{R}_{dual}} D_{\overline{mr}} \times v_{\overline{r}} \geq 0 \quad \forall \overline{m} \in \mathcal{M}_{dual} \setminus \mathcal{R}ev \tag{4.2}$$

$$v_{\overline{r}} \geq 0 \quad \forall \overline{r} \in \mathcal{R}_{dual} \tag{4.3}$$

$$z_{\overline{r}} \Leftrightarrow v_{\overline{r}} > 0 \quad \forall \overline{r} \in \mathcal{R}_{dual} \tag{4.4}$$

$$c_{\overline{r}} \Leftrightarrow z_{\overline{r}} \quad \forall \overline{r} \in \mathcal{C}ut \tag{4.5}$$

$$\neg z_{\overline{r}} \vee \neg z_{\overline{r}_{rev}} \ \forall (\overline{r}, \overline{r}_{rev}) \in Rev_{dual} \tag{4.6}$$

$$v \in \mathbb{R}^k,\ z \in \mathbb{B}^k,\ v_{\mathcal{T}} > 0 \tag{4.7}$$

$D_{\overline{mr}}$ denotes the dual matrix D stoichiometry coefficient associated to dual metabolite $\overline{m} \in \mathcal{M}_{dual}$ and dual reaction $\overline{r} \in \mathcal{R}_{dual}$. Equation (1) and (2) represent the steady-state constraint, and is an equality or an inequality whether the metabolite the constraint applies on was originally a reversible or an irreversible reaction. Equation (3) is defining that all reaction fluxes $v_{\overline{r}}$ should be positive or null. Equation (4) associates boolean indicator variables $z_{\overline{r}}$ to active reaction fluxes, meaning reaction with non-null fluxes. Equation (5) defines specific boolean indicator variables $c_{\overline{r}}$ for reaction fluxes corresponding to reactions in $\mathcal{C}ut$. These are the boolean variables that are considered for subset-minimal solutions. The $\mathcal{C}ut$ reactions are the only reactions which flux is of interest: representing the actual reactions in MCSs. Equation (6) forbids flux of two irreversible reactions issued from the split of a reversible one to be non-null. Equation (7) defines the domain of reaction fluxes $v_{\overline{r}}$ $\forall \overline{r} \in \mathcal{R}_{dual}$ as real linear values, the domain of indicator variables $z_{\overline{r}}$ $\forall \overline{r} \in \mathcal{R}_{dual}$ as boolean logic values, and forces the target reaction flux to be non-null.

Taking all of these constraints and searching for subset-minimal affectations of $c_{\overline{r}}$ to $\{True\}$, we obtain the MCSs disabling reaction targets $\mathcal{T}$. Information for dual metabolic network construction and Minimal Cut Sets problem is summarized visually with a complete formalization for genome-scale metabolic models in Figure 4.4. A simpler general look at the framework will also be presented in Figure 4.11.

4.2 Getting back genes from reactions

Minimal Cut Sets might be used to get essential genes, and intervention strategies for metabolic engineering described by gene-knockouts. To retrieve back the minimal cutting sets of genes from the MCSs of reactions, we defined a separate logic problem. It makes use of the GPRs, which are commonly defined in genome-scale models, according to the specification of SBML Level 3 with FBC Level 2 plugin.

4.2.1 Defining a formalism for Minimal Sets of Genes

Let us characterize the set of all genes $\mathcal{G}$, and the set of all Gene-Protein-Reaction association rules $GPRA : \mathcal{R} \mapsto f(\mathcal{P}(\mathcal{G}))$, defining for each reaction $r \in \mathcal{R}$ a Boolean formula $f : \mathbb{B}^{|G|} \mapsto \mathbb{B}$ from a given genes subset $G \subset \mathcal{G}$.

G is a minimal set of genes (MSG) for a reaction subset R if G is subset-minimal and an affectation of all the genes in G to *true* is enough to activate all the reactions in R, according to the GPR rules defined in $GPRA$.

In an EFMs context:

$$\text{MSG}_R = \{G \subset \mathcal{G} \mid \forall r \in R,\ G \text{ satisfies } GPRA(r) \text{ and } \nexists G' \subset G \text{ such that } \forall r \in R,\ G' \text{ satisfies } GPRA(r)\} \tag{4.8}$$

For example, we say that G is a minimal set of genes for an EFM e if an affectation of at least all genes in G to $\{True\}$ is required to activate the EFM e.

The Boolean formulas described by GPRs are monotone, i.e. they do not include the *not* operator. These can be changed to dual form. Considering variables $G = \{g_1, \ldots, g_n\}$, the dual of a formula f written $f(G) = \bigwedge \bigvee g_i$ is a formula f_d such that $f_d(G) = \bigvee \bigwedge\ g_i$, with *and* and *or* operators switched.

The duality property between monotone Boolean functions is such that we can derive the minimal set of genes that this time *deactivates* a set of reactions by simply computing the dual formulae of all GPR relations and obtaining the dual set of GPR rules $GPRA_d(r)\ \forall r$. Therefore, in a Minimal Cut Sets context, G is a minimal set of genes (MSG) for a reaction subset R if G is subset-minimal and an affectation of all the genes in G to *true* is enough to *deactivate* all the reactions in R, according to the GPR rules defined in $GPRA_d$.

In an MCSs context:

$$\text{MSG}_R = \{G \subset \mathcal{G} \mid \forall r \in R,\ G \text{ satisfies } GPRA_d(r) \text{ and } \nexists G' \subset G \text{ such that } \forall r \in R,\ G' \text{ satisfies } GPRA_d(r)\} \tag{4.9}$$

In both cases, this is the same enumeration problem. The problem of enumerating all minimal sets of genes is the problem of enumerating all subset-minimal satisfying assignments of gene Boolean literals to $\{True\}$, which is in at least NP-hard since determining a single satisfying assignment is NP-complete.

The dualization of monotone Boolean functions is a well-studied topic [258]. In many cases, one is interested in necessary components for activation and deactivation of Boolean functions. The same duality property is used for the link between EFMs and MCSs [259, 186, 187]. Thanks to the existing duality properties between Boolean formulae, algorithms for joint generation of EFMs and MCSs exist. Related studies include [259, 155] and [S127].

Minimal sets of genes or MSGs for the obtained reactions MCSs are potential gene knockouts. In order for our subset-minimal computation of genes from reactions to be valid, it is required that each reaction is associated to at least one gene. So for transporter reactions that are not associated to genes, we should add a dummy association, which in fact corresponds to the either backwards or forwards direction of that reaction, since reactions are split for MCSs computation.

In practice, forwards transporter reactions should also be assumed to be dependent on the presence of external metabolites in the medium. Some transporter reactions are also known to be spontaneous and annotated as such in their GPR association rules.

The full problem of getting back potential gene knockouts can be combined with network decompression, which follows the same logic ideas, *i.e.* activating (*resp.* cutting) a reaction subset means activating (cutting) all (one) of its reactions. It can be encoded as a logic program with subset-minimization heuristics in Answer Set Programming.

4.2.2 Illustrating Minimal Sets of Genes with examples

Let us define reactions $\{R1, R2, R3\}$ and the following GPR rules:

$$R1 \rightarrow g1 \wedge g2$$

$$R2 \rightarrow g2 \vee g4$$

$$R3 \rightarrow g3$$

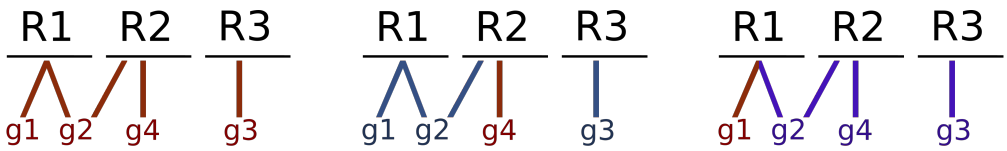


Figure 4.3: Minimal Sets of Genes. (A) An example of reactions and GPRs;
(B) Minimal Set of Genes g = $\{g1, g2, g3\}$ for EFM e = $\{R1, R2, R3\}$;
(C) Minimal Set of Genes g = $\{g2, g4, g3\}$ for MCS c = $\{R1, R2, R3\}$

Intuitively, the minimal set of genes allowing the network to function is $\{g1, g2, g3\}$.

But to be sure, we should compute all minimal sets of genes for subsets of reactions. Searching for minimal affectations to $\{True\}$, we obtain:

MSG$_{\{R1\}} = \{\{g1, g2\}\}$, MSG$_{\{R2\}} = \{\{g2\}, \{g4\}\}$, MSG$_{\{R3\}} = \{\{g3\}\}$

Now let us take EFM e = $\{R1, R2, R3\}$. Minimal affectations to $\{True\}$ gives: MSG$_e = \{\{g1, g2, g3\}\}$

However, if e were to be an MCS c instead, one would be interested into minimal sets of genes that cut network function, instead of minimal functioning units.

To do so, we take the dual functions of all GPR rules before computing MSGs.

$$R1 \rightarrow g1 \vee g2$$
$$R2 \rightarrow g2 \wedge g4$$
$$R3 \rightarrow g3$$

This time, we obtain: $\text{MSG}_{\{R1\}} = \{\{g1\},\{g2\}\}$, $\text{MSG}_{\{R2\}} = \{\{g2,g4\}\}$, $\text{MSG}_{\{R3\}} = \{\{g3\}\}$.

And therefore, $\text{MSG}_c = \{\{g2,g3,g4\}\}$.

Let us further illustrate the use of MSGs for EFMs and MCSs using GPRs from *E. coli* core. This example displays that fumarase (FUM), an enzyme with three isozymes catalyzing that function, is a harder enzyme to deactivate than fumarate reducatase (FRD), an enzyme with four subunits. Indeed, for FRD, deactivating any of the subunits is enough to cut its function, while for FUM, deactivating all three isozymes is required.

$$FRD7 \implies (frdA \wedge frdB \wedge frdC \wedge frdD)$$
$$FUM \implies (fumA \vee fumB \vee fumC)$$

```
reaction("FRD7"). reaction("FUM"). {gene("frdA")}. {gene("frdB")}. {gene("frdC")}.
{gene("frdD")}. {gene("fumA")}. {gene("fumB")}. {gene("fumC")}.

:- reaction("FRD7"); not gene("frdA"; "frdB"; "frdC"; "frdD").
:- reaction("FUM"); not gene("fumA"); not gene("fumB"); not gene("fumC").
#heuristic gene(G). [1, false]
```

The above ASP program yields 7 solutions, including 3 subset-minimal, corresponding to each FUM isozyme.

$$FRD7 \implies (frdA \vee frdB \vee frdC \vee frdD)$$
$$FUM \implies (fumA \wedge fumB \wedge fumC)$$

```
reaction("FRD7"). reaction("FUM"). {gene("frdA")}. {gene("frdB")}. {gene("frdC")}.
{gene("frdD")}. {gene("fumA")}. {gene("fumB")}. {gene("fumC")}.

:- reaction("FRD7"); not gene("frdA"); not gene("frdB"); not gene("frdC"); not gene("frdD").
:- reaction("FUM"); not gene("fumA"; "fumB"; "fumC").

#heuristic gene(G). [1, false]
```

The above ASP program yields 15 solutions, including 4 subset-minimal, corresponding to each FRD subunit.

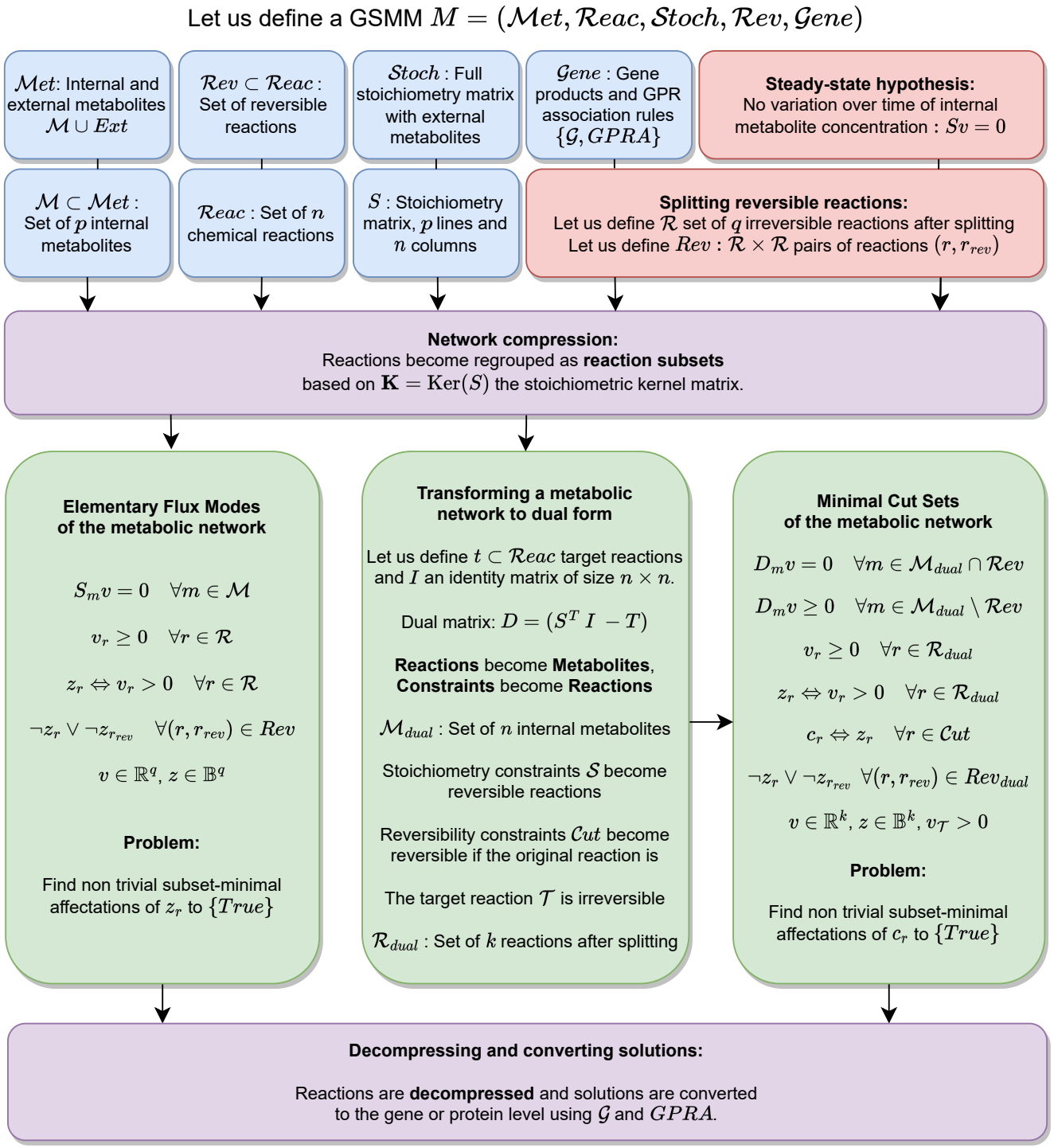


Figure 4.4: Diagram of the problem of computing MCSs and its formulation in a comprehensive view. Minimal Cut Sets are expressed as a dual problem to Elementary Flux Modes. To make use of genome-scale metabolic models (GSMMs), network compression is required, and solutions are converted back to genes using GPRs.

4.3 Application to *S. aureus* and *P. aeruginosa*

Our final *aspefm* application of choice will be to a consortium of highly pathogenic bacteria. *Staphylococcus aureus* and *Pseudomonas aeruginosa* are opportunistic pathogens commonly associated with the skin microbiome and water sources, respectively. The two problematic bacteria are responsible for an estimated 1+ million deaths yearly due in part to widespread antimicrobial resistance [10]. *S. aureus* and *P. aeruginosa* are frequently co-isolated from chronic wounds and cystic fibrosis lungs [260, 11]. Their interactions such as metabolite crossfeeding [261, 262] have been associated with higer resiliancy to antibiotics and worse patient outcomes [263, 264].

The complex nature of their interactions with other bacteria and the environment has motivated a growing number of studies involving consortia of these pathogenic bacteria, whether it is through *in vivo* and *in vitro* models [265, 266], or *in silico* models [267, 268]. Better informed and therefore more effective intervention strategies for treating *S. aureus* and *P. aeruginosa* infections could save millions of lives and billions of dollars in healthcare expenses.

Staphylococcus aureus is a Gram-positive bacterium, member of the *Bacillota* phylum and of the *Staphylococcus* genus. It is known for its cocci form, in circular shape. It is often positive for catalase and it is a *facultative* anaerobe: it can grow without the need for oxygen. *Pseudomonas aeruginosa* is a Gram-negative bacterium, member of the *Pseudomonadota* phylum and of the *Pseudomonas* genus, a catalase positive organism and a *de facto* aerobe, it can grow in anaerobic conditions but it is rather rarely observed to do so. It is known for its rod-shaped form. Both bacteria perform anaerobic respiration using nitrate as an electron acceptor in place of oxygen.

Gram-positive and Gram-negative are indicative of the type of peptidoglycan-made cell wall, surrounding the bacterial membrane. Catalase positive means the bacterium possesses the catalase enzyme, in part used to derive hydrogen peroxide from amino acids, which can then be converted to oxygen. Thus, one can see the two bacteria present characteristics that might be complementary, for instance: *Pseudomonas aeruginosa* might be performing oxidative respiration metabolism while *Staphylococcus aureus* would function with fast anaerobic glycolysis, and so on. Also, the different types of peptidoglycan cell walls make it for more difficult conjoint treatments.

The mucus formed by *Pseudomonas aeruginosa* is often considered a major accelerator of the formation of biofilms and a therapeutic target [S128]. *P. aeruginosa* is considered a multidrug resistant pathogen, and acts more with the role of a catalyst in infections, with lessened virulence, rather than a fast-growing deadly strain like *S. aureus* [267]. In comparison, *S. aureus* as a bacteria presented phenotypes of antibiotic tolerance until recently, where such strains stopped evolving due to selection pressure caused by overusage of antimicrobials [10].

The topic of antimicrobial resistance is very much related to our subject of studying metabolic interactions between the two bacteria. I created an animatic slideshow story illustrating the need for metabolic modellers for better understanding the bacteria and coming up with strong therapeutic solutions in this era of antibacterial overusage. The so-called 'digital story' can be found here: `https://eugloh-network.pageflow.io/maxime-mahout`.

Concerning the bacteria's metabolism, it is now known that *Pseudomonas aeruginosa* presents a reverse diauxie phenotype, or reverse carbon catabolite repression (rCCR), which is the inverse of the 'glucose-first' phenotype from *E. coli* and *S. aureus* [269, 270, 271], termed classic carbon catabolite repression (cCCR) [271]. For example,

in McGill et al [271], the researchers present a metabolic model analysis that fits well to their experimental observations, that is, that the order of growth medium substrate preference for *P. aeruginosa* is the following: (1) amino acids, (2) citrate, (3) succinate, (4) lactate, (5) acetate and in last, (6) glucose.

S. aureus is characterized by its complex regulation, surprisingly involving a non-coding RNA as one of the central parts, RNAIII [S129]. The two central protein regulators of *S. aureus* are CcpA, reacting to glucose and other classical carbon sources (the regulator is involved in cCCR) [S130, S129, S131], and CodY, reacting to branched-chain amino acids [S132, S133] and GTP [272]. Transcriptional regulation for *P. aeruginosa* and *S. aureus* has not yet been completely elucidated and is well less studied than the reference bacterial organism *E. coli*, meaning it can hardly be incorporated in systems biology modelling methods as of now. There have been attempts to elucidate the transcriptional regulation of *S. aureus* using an approach based on analysis of transcriptomic data and FBA-related methods for validation [S134, S135].

Various co-living mechanisms appear in biofilms between the two bacteria. One of them is the release of pyocyanin by *P. aeruginosa*, in order for *P. aeruginosa* to limit *S. aureus* growth [267]. In a chronic wound, other mechanisms include spatial adaptation towards oxygen presence – for *P. aeruginosa* in particular, while *S. aureus* can colonize anoxic regions of the wound. For instance, a nonexhaustive summary of the chronic wound model and its corresponding discoveries presented by Phalak et al in 2016 [267] is presented in Figure 4.5.

Finally, the bacteria are involved in various metabolite exchanges, including syntrophy or metabolite cross-feeding, a term which is used when bacteria are thought to be cooperating with each other in a symbiosis way. This is the core of our study in this chapter of the thesis. We aim to present a new application of MCSs, able to reveal novel metabolite exchanges. To do so, we devise a MCSs analysis on a model of a consortium of these two bacteria: *S. aureus* and *P. aeruginosa*. Finding therapeutic targets to the whole consortium model would result in therapeutic targets that prevent growth and metabolite exchanges of both bacteria.

4.3.1 Genome-scale metabolic model selection

Several models were available for *Staphylococcus aureus* and *Pseudomonas aeruginosa*. For example, the N315 model of *Staphylococcus aureus* from 2005 [164] and the iMO1086 *Pseudomonas aeruginosa* PAO1 model from 2008 [273] are both very well-curated central metabolism models and would have made for very good choices. Instead though, we opted to choose more recent models, and models that contained a large number of reactions, in order to illustrate *aspefm*'s capacity to handle large-size models.

The choice for the *Staphylococcus aureus* model was made according to a study by Renz and Dräger [170]. Lots of pre-processing and curating work was done to make models available for use, notably in the case of the *P. aeruginosa* one, which had unreadable ModelSEED IDs, which are very hard to work with [36]. We also worked on having all the GPRs in the SBML models be associated to UniProt entries, which we will use later in the analysis.

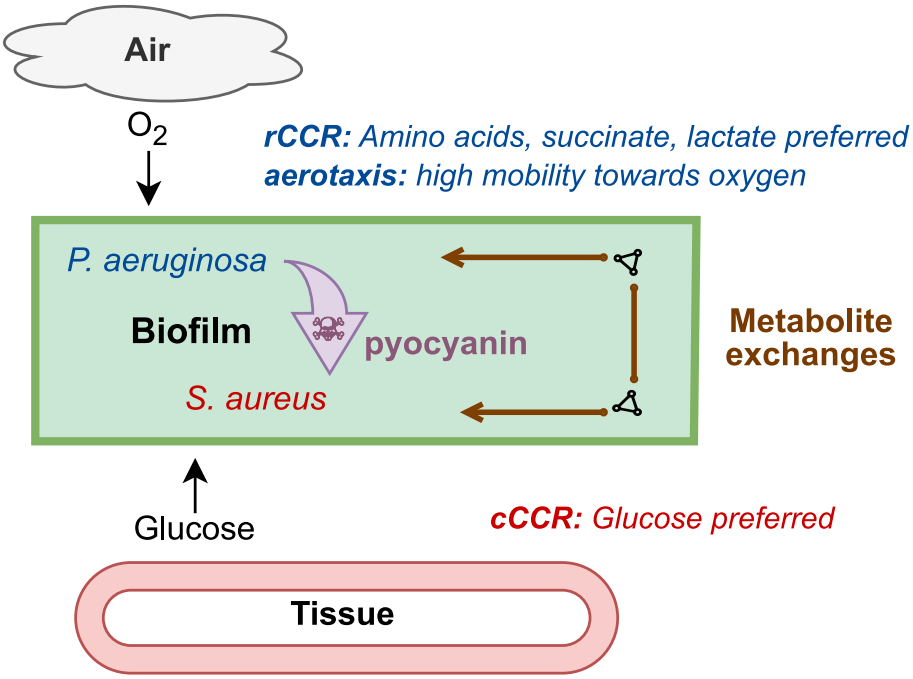


Figure 4.5: *P. aeruginosa* and *S. aureus* biofilm model figure redrawn from Phalak et al, 2016 [267] and expanded upon. The *P. aeruginosa* and *S. aureus* catabolism phenotypes: rCCR (reverse carbon catabolite repression) and cCCR (classic carbon catabolite repression) are presented, as explained in McGill et al, 2021 [271].*P. aeruginosa* mostly colonizes regions towards the oxygen using aerotaxis, and manages its survival by secreting pyocyanin to reduce *S. aureus* strain growth.

We used metabolomics studies to refine the *Staphylococcus aureus* models exchanges lower and upper bounds, including the ones in [S136, S137, S138]. This revealed blocked exchange bounds for *S. aureus* metabolism byproducts in the original *iYS854* such as for example Formate and Butanediol. For *Pseudomonas aeruginosa*, we performed less curation, and trusted the modellers' exchange bounds.

As well, we corrected the *iYS854* model, by adding back transporters for the main purines: adenine and guanine. The existence of such a transporter, while it has seemingly not been conclusively proved, according to our last literature review, it should be supported by evidence of response from the bacterium to guanine depletion [272].

4.3.2 Devising a MCSs analysis on *S. aureus* and *P. aeruginosa*

In order to help understand the multilayered MCSs analysis we perform on *S. aureus* and *P. aeruginosa*, we will detail our two main hypotheses: Hypothesis 4.3.2 and Hypothesis 4.3.1. We also provide definitions related to the analysis in Definition 4.3.1 and a related illustration in Figure 4.12.

To clarify, we will be comparing our tool *aspefm* with two other methods: *cnapy* and *cobamp*. These tools are MILP-based, and they perform well on the so-called MCSs of small size. On the other hand, we believe that, while our tool performs worse on small-size MCSs, it would perform better on the large-size ones.

And in fact, we believe that the so-called MCSs of large size are the ones that justify the use of such an expansive computation tool such as a MCSs enumerator. To illustrate that second belief, we show an application of our constraint-based methodology to interspecies metabolite exchanges that might circumvent lethal phenotypes conferred by knock-out interventions, using the large-size solutions.

Definition 4.3.1 – Minimal Cut Sets-related definitions
On genome-scale models, we define as MCSs of small size the biomass-targeting MCSs of size 3 or less. This is due to the fact these MCSs could be computed with iterative methods such as synthetic lethals (SLs) computation instead. Contrarily, we define as MCSs of large size the biomass-targeting MCSs of size 4 or more. These are the MCSs we are interested in, as their number is exponentially higher as the size requested augments. Our study aims to constrain the solution space of these MCSs to obtain elements giving us insight for therapeutic applications. In particular, we decide to construct a consortium model of our two bacteria of interest, and look at metabolite exchanges allowing recovery of growth, which are highlighted whenever bacterial models exchange reactions are present within MCSs.

If the general idea of the analysis is not clear from these definitions and hypotheses, we suggest getting an early look at the following figures: Figure 4.10, Figure 4.9, Table 4.1, Figure 4.11, Figure 4.12 before following the article's natural reading order. This final analysis done with *aspefm* highlighted many biological results of significant importance, including highlighting that some actively researched antimicrobial targets could in fact be nullified by metabolite exchanges, and recuperating targets that were most likely to target both bacteria at the same time using artificial-intelligence powered protein structure predictions. It is a very complete work, essential to show the metabolic modelling community that the constraint-based approach not only makes EFMs and MCSs calculations possible thanks to powerful computation methods such as *aspefm*, but also required in the light of driving biological discovery and developing new therapeutic treatments.

Hypothesis 4.3.1 – Antibacterial treatments might be nullified by interspecies metabolite exchanges
Oftentimes, pharamaceutical applications to biofilm treatments in chronic wound infections are studied on a pathogen by pathogen level, forgetting the ecological implications of pathogens forming biofilms together to solidify their antimicrobial resistance capacities. For the purpose of this systems biology analysis, we propose to find antibiotic treatment that would target both bacteria at the same time, rather than only a single bacterium. To do so, we reveal that there are in fact metabolite exchanges that might nullify lethal phenotypes conferred by intervention strategies studied on a single-species level. This is done by studying the small-size MCSs of single species, giving gene knockout interventions, on a larger-size scale using large-size MCSs of the consortium-level.

Hypothesis 4.3.2 – *aspefm* specializes in enumerating MCSs of large size

Based on several observations, both on small-size, toy models, and on large-scale, genome-scale models, we believe that our tool, while struggling with smaller-size MCSs, performs better than its concurrence, MILP-based tools, on large-size MCSs. To illustrate that belief, we build an astonishingly large-sized model for EFMs/MCSs computation standards: a consortium model of around three thousand reactions, result of the combination of two models of about one and a half thousand reactions. Surprisingly, we will show that *aspefm* handled the consortium model perfectly fine, and was able to enumerate solutions regardless of their size. This very promising result will provide the foundation for future applications of *aspefm*, whether for biologically relevant MCSs, or for biologically relevant EFMs, the latter which are usually composed of many reactions.

4.3.3 Introducing our MCSs analysis

From there on, I am presenting our article titled "Logic programming-based Minimal Cut Sets reveal consortium-level therapeutic targets for chronic wound infections", which is in publication process [274]. I present the article's introduction as is, as I belive it presents a great summary of the methods detailed in this thesis. Unlike for the two other articles, I kept the biology field way of putting 'Results' before 'Methods', as in the case of this article the main biological results should be at the front, and I believe the methods alone do not make sense without their application to the results.

Abstract: Minimal Cut Sets (MCSs) identify sets of reactions which, when removed from a metabolic network, disable certain cellular functions. The method has been applied to identify essential genes, and to guide the engineering of organisms for desired phenotypes. The traditional search for MCSs within genome-scale metabolic models (GSMMs) targets cellular growth, identifies reaction sets of varying sizes which result in a lethal phenotype if disrupted, and retrieves a list of corresponding gene, mRNA, or enzyme targets using Gene-Protein-Reaction (GPR) association rules. Using the dual link between MCSs and Elementary Flux Modes (EFMs), our logic-programming based tool *aspefm* was able to compute MCSs of any size from GSMMs in acceptable run times. The tool demonstrated better performance than the mixed-integer linear programming method used by *cnapy*, identifying more than twice as many solutions in the same 1.5 day run time.

We applied the new MCSs methodology to a medically-relevant consortium model of two crossfeeding bacteria, *Staphylococcus aureus* and *Pseudomonas aeruginosa*, which were represented by well-curated GSMMs. *aspefm* constraints were used to bias the computation of MCSs toward exchanged metabolites that could complement lethal phenotypes in an individual species. We found that crossfeeding inosine could complement lethal reaction knock-outs in the purine synthesis, glycolysis, and pentose phosphate pathways of both bacteria. Finally, the results were used to derive a list of promising enzyme targets for consortium-level therapeutic applications that cannot be circumvented via interspecies metabolite exchange.

Introduction: Constraint-based metabolic modelling (CBM) is an emerging systems biology field involving the computational reconstruction and analysis of biological mechanisms at multiple levels [91]. At its core are metabolic networks, hypergraphs described by a set of metabolites and reactions linked to each other by stoichiometric coefficients stored in a stoichiometric matrix. The constraint-based modelling approach calculates metabolite fluxes based on the assumption that the system is at steady-state; therefore, intracellular metabolite production and consumption are balanced over time relevant time intervals.

Flux balance analysis (FBA) is one type of CBM that uses linear optimization to identify solutions to metabolic models, based on an objective function which often involves maximizing the flux through a biomass synthesis reaction [71, 119]. The FBA solution is a flux distribution that predicts cellular phenotype including which enzymes are active and what the magnitude of the flux is through each enzyme. The biomass synthesis reaction accounts for cell growth as observed experimentally [116]. FBA and derived methods are used to make *in silico* phenotype predictions based on changes in the growth medium or based on altering of enzyme activity through gene knockouts or recombinant interventions [161].

Elementary Flux Mode (EFM) analysis is another CBM method that performs an exhaustive enumeration of the edges of the metabolic solution space defined by the stoichiometric matrix; FBA solutions are nonnegative linear combinations of EFMs [236]. The number of EFMs grows exponentially in relation to the number of reactions; counting all EFMs has been proven to be #P-hard [134, 135]. Consequently, enumeration of EFMs from large metabolic models with over 100 reactions is challenging, requiring special computational methods [144], biological constraints such as transcriptional regulation [143] and thermodynamic data [275], and careful model network compression [109, 126, 174].

Building from the set of metabolic reactions encoded in the genome, and progressing to the intricate mechanisms at the protein and enzyme level, CBM contributes to the description of a wide variety of cellular processes. Genome-scale metabolic models (GSMMs), large-scale constraint-based metabolic models computationally generated from genomes of interest are now the norm [92, 276], thanks to increased availability of data and computational power. GSMMs are well suited for identifying putative drug targets through predicting gene and metabolite essentiality [277, 278].

GSMMs have been applied to analyze drug targets in cancerous cells [279, 280] and to treat *Pseudomonas aeruginosa* infections [281]. The methods identified essential reactions and synthetic lethals (SLs) [175]. Synthetic lethals refers to combinations of gene-deletions or enzyme interference targets which prevent growth. While the term initially refered to pairs of genes, it is now used to describe n-tuples of reaction targets. The synthetic lethals may explictly consider both the metabolic potential of the organism and the role of the nutritional environment provided by the extracellular medium [176].

Improved algorithms for computing synthetic lethal strategies have been proposed to speed up the calculation process, such as Fast-SL [177] and Rapid-SL [178]. The computation of synthetic lethals essentially deals with a combinatorial exploration of every possible n-tuple of reactions. Thus on large networks of over a thousand reactions, computation runs slower as the size of n-tuples increases, and becomes impracticable if n-tuples of size

over 4 are of interest [177].

Another method proposed for identifying synthetic lethals, whether n-tuple size is under or over 4, is the computation of Minimal Cut Sets (MCSs), with the biomass synthesis reaction as a target reaction. MCSs are traditionally defined as the 'Hitting Sets' of Elementary Flux Modes (EFMs) [180, 181], and are an exhaustive way of exploring robustness of a network. Setting a certain reaction as target for inactivation, MCSs define all sets of reactions capable of preventing flux through the target reaction [183]. In particular, Minimal Cut Sets have been formalized for metabolic engineering and recombinant strain optimization [182]. Alternately, MCSs have demonstrated remarkable performance identifying synthetic lethals in cancer cells [184].

MCSs suffer the same computational time hindrances as EFMs. The number of possible MCSs grows exponential with the number of reactions [185]. Interestingly, it has been proven that MCSs can be enumerated as the EFMs of a so-called dual metabolic network [186, 187]. As a result, similarly to how Mixed-Integer Linear Programming (MILP) methods were developed for computing the shortest EFMs of a metabolic network, [140, 150], MILP methods for computing the shortest MCSs have been developed [153, 154].

Furthermore, it might be necessary to convert the obtained MCSs into sets of target genes or proteins for biological interpretation. Methods have been developed to incorporate multilevel data, namely the Gene-Protein-Reaction association rules (GPRs) from GSMMs, into the stoichiometric matrix [162]. These solutions have been repurposed for the MCSs computation [184, 188, 189].

Here, we develop a new method for calculating and analyzing MCSs using our *aspefm* tool [89]. The *aspefm* program is a SAT-based method designed to compute subsets of EFMs while respecting user-defined constraints. It differs from the Double Description method, implemented in *EFMTool* [139, 127], which needs to enumerate all solutions before generating results, and from MILP-based methods, implemented in *CNA* [149], *cnapy* [138] and *cobamp* [151], which perform minimization in the size of the reaction set. We have extended the functionality of *aspefm* to the computation of MCSs.

Throughout this work, we distinguish MCSs of small size (reaction n-tuples of size 3 or less including essential reactions, synthetic lethal pairs and synthetic lethal triplets) from MCSs of large size (defined here as reaction n-tuples of size 4 or more). The MCSs of small size are usually the desired reaction sets, they are readily calculated by MILP methods and SLs computation algorithms, and effortlessly converted to gene and enzyme targets using GPRs. On the other hand, the MCSs of large size are less well-studied since they would not necessarily correspond to intervention targets. However, in this study we argue MCSs of large size are critical to study network robustness and interactions. We intend to bridge the gap between these two types of MCSs with our analysis.

aspefm successfully processed an aggregate model of two GSMMs totalling over three thousand reactions: a consortium model of bacteria *S. aureus* and *P. aeruginosa*. The tool efficiently identified solutions of interest, using a wide variety of constraints, in acceptable computation times despite the size of the consortium metabolic network. The *aspefm* application identifies potential, nonobvious, interspecies metabolite exchanges essential for consortium growth and thus identifies promising therapeutic targets for controlling the problematic pathogens.

4.4 Results of our MCSs analysis

4.4.1 Overivew of genome-scale metabolic models for analysis of single species and consortium

Manually curated genome-scale metabolic models (GSMM) of *Staphylococcus aureus* and *Pseudomonas aeruginosa* were selected for our analysis. The *S. aureus* GSMM, *iYS854*, was developed based on *S. aureus* str. JE2 [163]. The GSMM has been used for assessing the validity of experimentally determined transcriptional regulation modulons of *S. aureus* [S134, S135]. The model has been graded as the most accurate *S. aureus* GSMM currently available, according to a study by Renz and Dräger [170].

The *P. aeruginosa* GSMM *iPae1146* is based on *P. aeruginosa* strain PAO1 [282]. The GSMM was used for an high-throughput essentiality analysis [281]. In that study, the model was predicted to have around 97 % accuracy for predicting gene essentiality during growth on Lysogeny Broth (LB) medium.

Both metabolic models were pre-processed and curated for our analysis, as detailed in the Methods. The resulting *iYS854* model includes 1454 reactions, 1338 metabolites, and 866 genes, while the resulting *iPae1146* model includes 1495 reactions, 1283 metabolites, and 1148 genes.

The models were analyzed in an *in silico* extracellular environment defined by CSP chemically-defined medium [269], on which *S. aureus* and *P. aeruginosa* can grow as biofilms *in vitro* [271]. The CSP medium was designed to serve as a simplified analog of chronic wound exudate. The medium was chosen as the base for all predictions of growth and consortial crossfeeding in our study.

A consortium model consisting of *P. aeruginosa* and *S. aureus* was built from *iPae1146* and *iYS854* by adding metabolite exchange reactions with a shared control volume containing the growth medium, as detailed in Figure 4.10. The newly created metabolic network contains 3241 reactions and 2752 metabolites. When the consortium model was constrained by an extracellular environment defined by CSP medium, a total of 57 metabolites were classified as 'external' substrates. 46 external substrates were available to both *S. aureus* and *P. aeruginosa*, one metabolite was exclusively available to *P. aeruginosa*: citrate, and ten metabolites were exclusively available to *S. aureus* including some vitamins and purines.

The network compression process, which is required for MCSs computation, excluded 1296 blocked reactions from the CSP-constrained consortium model and returned a compressed consortium network of 1062 reactions and 600 metabolites. Statistics and results from the construction and compression of the consortium model, and of the individual species models *iYS854* and *iPae1146*, are reported in Table 4.1.

As well, the numbers of MCSs of small size for the consortium and individual species models are reported in Table 4.1. 583 MCSs of size three or less were found for the single species *iPae1146* model; 938 MCSs for the single species *iYS854* model. Two single reactions are essential to the consortium-level model, the uptake of ferrous ions and the secretion of glycolate. These two conditions are necessary for the biomass synthesis reactions of both models.

4.4.2 *aspefm* calculates consortium-level models MCSs regardless of the size of the reaction set

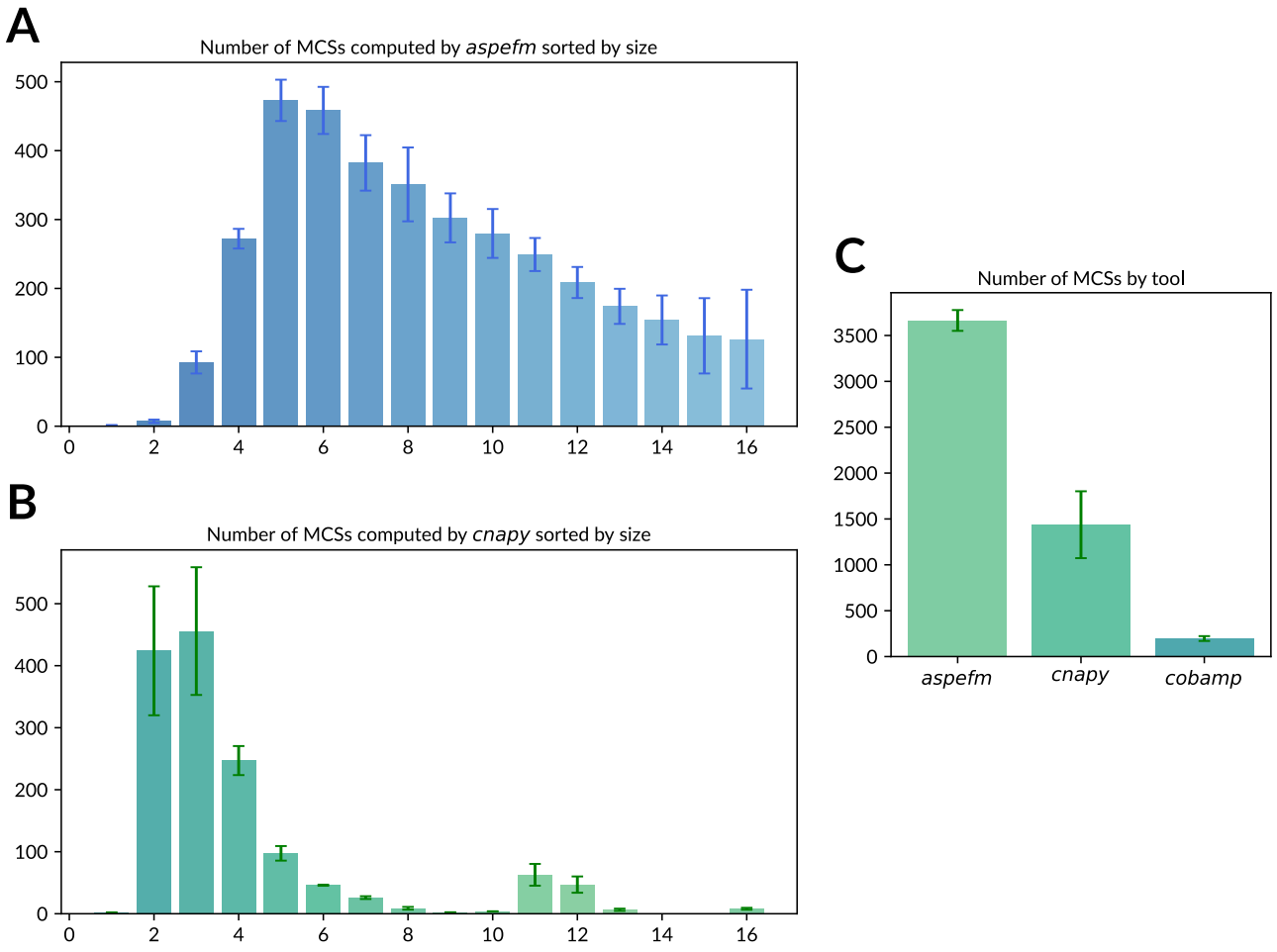


Figure 4.6: Number of MCSs computed by each tool in a simulation on the consortium model set with a time limit of 1.5 days, and limited to MCSs of below 16 reactions. **A**: Size of MCSs computed by *aspefm*, **B**: Size of MCSs computed by *cnapy*, **C**: Number of MCSs computed by each tool. Heights correspond to average numbers for five program executions, error bars represent standard deviation between executions for figure **C**, and half of standard deviation for **A** and **B**.

The performance of our *aspefm* tool was evaluated by computing MCSs from the compressed consortium model comprised of 1062 reactions. The target reactions for the simulation were the biomass synthesis reactions for both single species. The only constraints for the simulation were to identify MCSs of reaction size 16 or less and a maximum run time of 1.5 days.

The performance of *aspefm* was compared with *cnapy* [243] and *cobamp* [151], both are MILP-based MCSs enumeration methods. For each tool, five executions were launched and averaged in Figure 4.6. On average, *aspefm* identified more than twice as many MCSs as *cnapy* in the 1.5 day run time. *aspefm* averaged 3663.8 MCSs while *cnapy* averaged 1437.2 MCSs. Both *aspefm* and *cnapy* were able to enumerate MCSs regardless of the reaction

set size, while *cobamp* was heavily hindered by its forced iterative enumeration approach which started with the smallest cut sets; the method only identified around 196 MCSs on average. Upon decompression, the MCSs found by *aspefm* and *cnapy* reached the order of 10^5 MCSs, illustrating the necessity of network compression.

The MCSs identified using *aspefm* sampled solutions ranging from 1 to 16 reactions with the highest frequency at a reaction size of 5 (Figure 4.6). Meanwhile, the MCSs identified using *cnapy* were biased toward smaller reaction numbers and mainly enumerated solutions with 2-7 reactions with the highest frequency occuring at 3 reactions (Figure 4.6). We believe these differences to be explained by the SAT-based nature of our *aspefm* logic programming tool. *aspefm* enumerates solutions regardless of their size, while the MILP approaches used by *cnapy* and *cobamp* find themselves mainly limited to smaller sets of reactions, of size 3 or less. *aspefm* is thus the tool of choice for MCSs of larger size.

4.4.3 MCSs reveal robustness of consortial metabolite exchanges

Medical infections comprised of both *S. aureus* and *P. aeruginosa* can result in worse patient outcomes and can be more difficult to treat than monocultures. *aspefm* can identify metabolite exchanges between species that would bypass therapeutic strategies targeting only a single species. MCSs of small size from the single species models were tested for lethality at the consortium level to determine if directed crossfeeding interactions or passive metabolite exchange through metabolite leaking could circumvent single species lethalities. Of the 583 MCSs of size three or less for *P. aeruginosa*, 68 were no longer cut sets at the consortium level based on metabolites secreted by *S. aureus*. Meanwhile, of the 938 MCSs of small size for *S. aureus*, 199 cut sets were no longer effective due to metabolite exchanges from *P. aeruginosa*.

The lethal MCSs that were nullified due to metabolite exchanges were verified through additional analyses of the consortium-level model. To accomplish this, MCSs were calculated using *aspefm* by setting the reactions from the original MCSs as ‘wanted reaction‘ constraints while all reactions unnecessary for metabolite exchange as ‘unwanted reaction‘ constraints (see Methods). The verifying MCSs were limited to eight or fewer reactions and a time limit was set to 1.5 days for each computation. In total, 531 compressed consortium model MCSs were computed, ranging in size from 2 to 8 reactions, with the mean and median being 6 reactions and the highest frequency being 7 reactions.

For instance, a MCS of size three that exists for a single species model might be nullified by five different metabolite exchange reactions, resulting in a consortium model MCS of size eight. Theoretically, five interventions on exchanges with the other bacterium would be required in order for the original MCS to regain lethality. Alternatively, if only one metabolite exchange reaction were required to nullify a cut set, then only a single theoretical intervention would be necessary to maintain the lethality of the original MCS.

For each bacterium, the identity of the exchanged metabolites in the consortium model and the number of single-species MCSs they suppress are reported in Figure 4.7. The majority of cut sets nullified due to metabolite exchanges involved purine metabolism, pentose phosphate pathway, and glycolysis. Inosine was a pivotal metabolite in many of those functions, it was able to complement almost half of the identified cut sets for each bacterium.

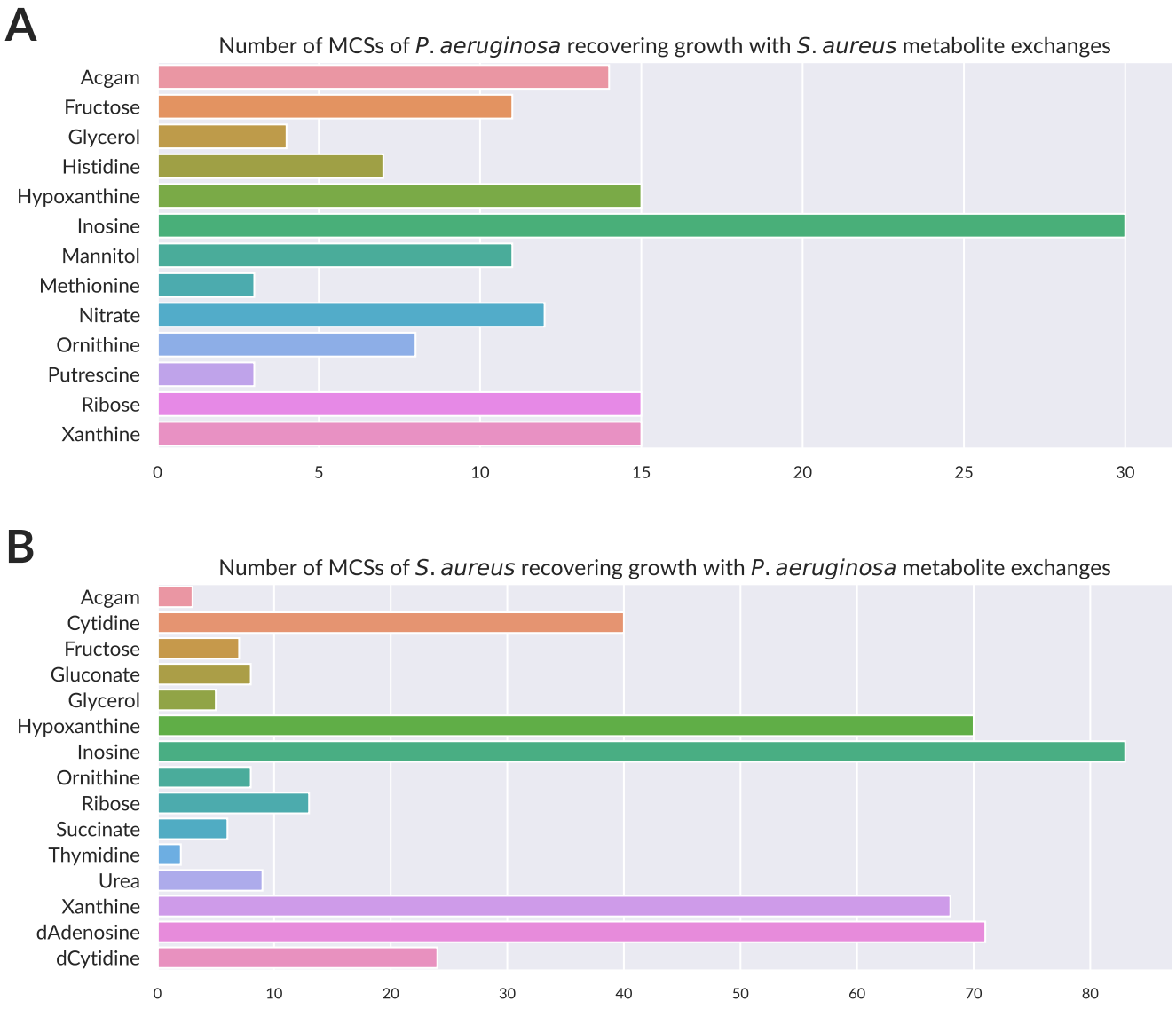


Figure 4.7: Number of single species MCSs recovering growth based on metabolite exchanges from consortium partners. Analysis limited to single-species MCS of reaction size 3 or smaller. **A**: MCSs of *P. aeruginosa* nullified by extracellular metabolite import from *S. aureus*, **B**: MCSs of *S. aureus* nullified by extracellular metabolite import from *P. aeruginosa*. *Abbreviations:* **Acgam**: acetylglucosamine, **d**: desoxy.

Inosine nucleosidase can transform the metabolite into hypoxanthine – a purine – and ribose, which can support many central metabolism transformations.

Overall, purines play a central role in these metabolite exchanges allowing the recovery of growth. It is of note that *S. aureus* contains more purine-related MCSs due to the putative presence of purine transporters, as opposed to *P. aeruginosa* which does not possess genes for such enzymes [S139]. Other notable metabolite exchanges shared by the two bacteria include acetylglucosamine, ribose, fructose and glycerol, complementing glycolysis functions, and urea-related metabolites, complementing urea cycle metabolism functions.

The enzyme N5,N10-methylenetetrahydrofolate dehydrogenase-cyclohydrolase, catalyzing methylenetetrahydrofolate dehydrogenase and methenyltetrahydrofolate cyclohydrolase, has been studied as potential drug target for *P. aeruginosa* [283]. However, these reactions, identified as essential on the single species model, are complemented by purines and histidine exchanged by *S. aureus* on the consortium-level model.

Although the majority of the single-species-level MCSs corresponded to consortium-level MCSs of large size: most recovering growth by at least two distinct possible metabolite exchanges, we also found a few small sized MCSs outliers, including glucosamine-6-phosphate synthase.

Interestingly, while glucosamine-6-phosphate synthase was identified as an essential reaction in both single species models, the lethal phenotype conferred by deleting this reaction would be recovered by a simple exchange of acetylglucosamine between the bacteria. Therefore, efforts using therapeutic agents which inhibit this enzyme [S140, S141] would also need to consider the potential role of acetylglucosamine found in the environment or leaked by bacteria in the consortia which are not inhibited by the therapeutic agent.

4.4.4 MCSs identify multi-species, consortium-level intervention targets

The effects of therapeutic agents that target a single species can, in some cases, be bypassed through metabolite exchange from other consortia species. From single-species MCSs, our analysis illustrated how specific consortium-level metabolite leakage or crossfeeding events could enable growth recovery. This identified MCSs that should be excluded from further analysis, while the remaining MCSs could be more promising targets for therapeutic intervention.

In order to retrieve consortium-level intervention targets, we selected MCSs that were mutually shared by the two bacteria. As well, the corresponding enzyme targets were retrieved from MCSs for protein structure studies. It is hypothesized that proteins with similar structure could be targeted simultaneously by the same therapeutic agent and thus target the consortium by conjointly blocking both *S. aureus* and *P. aeruginosa*. Out of 515 *iPae1146* MCSs and 739 *iYS854* MCSs of small size that were not nullified by metabolite exchanges, 65 MCSs were shared by both bacteria (Table 4.1).

The structures of the proteins associated with the shared 65 MCSs were analyzed for similarity. The protein structure analyses ruled out several MCSs since the target enzymes for *S. aureus* and *P. aeruginosa* were not deemed similar enough for concurrent therapeutic intervention. Enzymes were represented by their GSMMs gene assign-

ments, ie. GPRs (Gene-Protein-Reaction gene products), and were retrieved from MCSs with a logic programming extension of our *aspefm* procedure. Interspecies relative enzyme structure similarity was measured by Root Mean Square Deviation (RMSD) of atom positions in Ångström, protein structure alignments were computed on AlphaFold structure predictions [43]. The remaining shared MCSs were filtered using an additional criterion: the *S. aureus* and *P. aeruginosa* protein structures were compared to any potential human homologues. MCSs with enzymes that were deemed too similar to human homologues were removed. As a result, a ranking of the most promising MCSs for antibiotics discovery in human is presented in Figure 4.8.

From the 65 MCSs common to both bacteria and their corresponding enzymes, 23 of them were considered good potential drug targets for elimination of the bacterial consortium in human (Fig. 3B). The most promising enzymes to simultaneously target both *S. aureus* and *P. aeruginosa* included thirteen enzymes from nucleotide metabolism, lipid synthesis, aromatic amino acid biosynthesis, bacterial cell wall construction, amino sugar metabolism and folate biosynthesis (Figure 3A and 3C).

At the top of the ranking, beta-ketoacyl-ACP synthase III had high protein structure similarity between the two bacteria (interspecies RMSD: 1.21 Å) and fortunously, there did not seem to be human homologues. However, there exist functionally related isozymes beta-ketoacyl-ACP synthase I and II [S142] with high similarity with human homologues, which were thus excluded by our procedure.

Some example cell wall synthesis enzymes include undecaprenyl-disphosphatase (interspecies RMSD: 1.37) and UDP-N-acetylmuramoyl-L-alanine synthetase (RMSD: 1.81). Both enzymes are found only in the bacteria and not in human. Amino sugar metabolism included two enzymes for which there are human homologues: GlmU [S143] and phosphoglucomutase [S144], however the protein structures were considered dissimilar enough by our filtering procedure for these enzymes to be considered targetable.

Other enzymes with high therapeutic potential to treat *S. aureus* and *P. aeruginosa* consortia are four enzymes from the aromatic amino acid biosynthesis pathway: chorismate synthase (RMSD: 1.22), 3-phosphoshikimate 1-carboxyvinyltransferase (RMSD: 1.35), shikimate kinase (RMSD: 1.91) and 3-dehydroquinate synthase (RMSD: 2.03). This biosynthesis pathway is not present in mammals [S145]. In fact, an inhibitor of 3-phosphoshikimate 1-carboxyvinyltransferase, glyphosate, is commonly used as an herbicide [S146].

Additionally, out of the 23, 11 environment-dependent, bacterial-only enzyme targets were identified (Fig. 3B). These enzymes are derived from 2-3 reaction MCSs containing a transporter for an amino acid found in the growth medium and an amino acid biosynthesis reaction, which becomes essential in absence of that amino acid. Were the bacteria to be grown in an environment lacking the amino acids associated with these MCSs, an inhibitor of the enzyme targets would be effective.

These MCSs provide detailed, systemic insight into which amino acid biosynthesis reactions are the most important for *S. aureus* and *P. aeruginosa* growth, and therefore which amino acids biosynthesis pathways are the most promising targets. Our eleven MCSs correspond to six important amino acids, among two classes, aromatic amino acids (tryptophan, histidine, phenylalanine) and branched-chain amino acids (isoleucine, leucine, valine). These

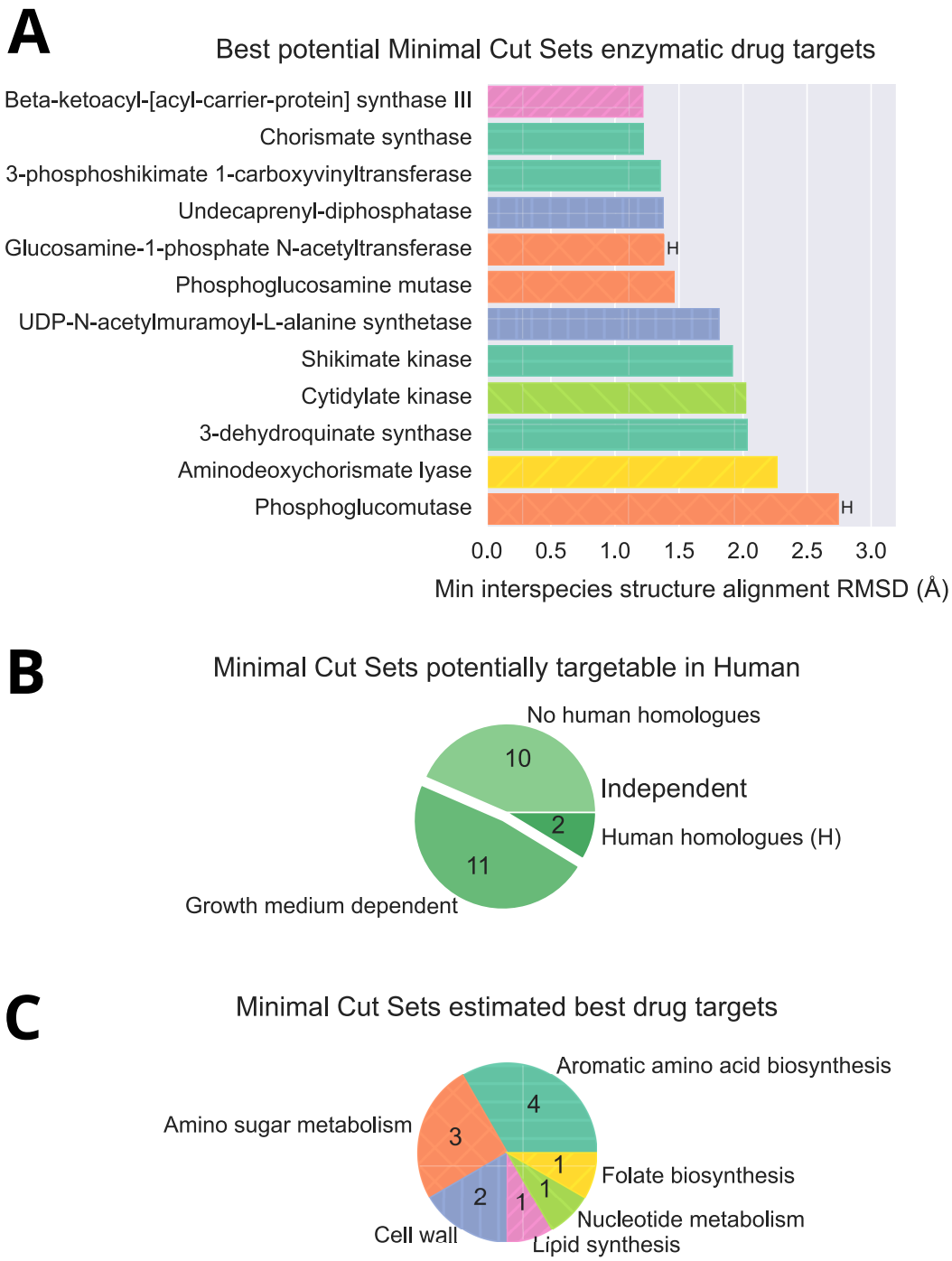


Figure 4.8: Consortium MCSs estimated to be targetable with a single ligand after: elimination of cut sets recovering viability through metabolite exchanges, selection of MCSs with metabolic functions in common to both bacteria, elimination of *P. aeruginosa* and *S. aureus* targets whose protein structures do not match. From the 23 MCSs targetable in human (in **B**), only 12 are independent of the growth medium (shown in **A** and **C**). Enzymes indicated with '**H**' were found to have human homologues. **A**: Ordinates labels show targetable enzymes. Bars heights are Root Mean Square Deviation (RMSD) of atom positions, measured in Ångström, resulting from protein structural alignment of GPR pairs. **B**: Categories and pie distribution of MCSs potentially targetable in human. **C**: Associated metabolism groups and pie distribution for each targetable enzyme in **A**.

essential pathways are in accordance with previous studies and might be druggable [S147]. More insight into the growth medium dependent MCSs, as well as structure alignments and human homologues, is given in Figure 4.13 and Figure 4.14.

4.5 Methods of the MCSs analysis

Minimal Cut Sets were computed on *Staphylococcus aureus* and *Pseudomonas aeruginosa* GSMMs *iYS854* and *iPae1146* , as well as on a consortium model containing the two models and reactions to model crossfeeding. The tools used are either *aspefm*, our tool, for which we detail the methods further, or *cnapy* and *cobamp*. The Python module representing *cnapy* is *StrainDesign* [243]. We provide code for the analysis at `https://github.com/maxm4/paSAmcs/`.

An overview of the complete analyzes performed in Table 4.1, Figure 4.7 and Figure 4.8 is found in Figure 4.9. MCSs of small size of *P. aeruginosa* and *S. aureus* were tested for growth recovery on the consortium model. For MCSs which lost their lethal phenotype on the consortium, MCSs of large size were retrieved explaining which metabolite exchanges allow recovery of growth. For the remaining MCSs of small size, MCSs in common between the bacteria were retrieved and analyzed for the search of possible new antibacterial agents. This analysis is further detailed in Figure 4.11, Figure A.3, Figure A.4.

4.5.1 Metabolic models pre-processing and curating

Both models went into a first phase of pre-processing. *iPae1146* did not adhere to the new reaction and metabolite identifiers standards defined by the BiGG Models Database from UC San Diego [37], and instead used SEED compounds and SEED reactions names [36], which would have made working with the model overly impractical, so the identifiers were replaced. The Equilibrator Python API helped with making assocations between SEED identifiers and BiGG identifiers [19].

A pyocyanin transporter was added to *iPae1146*, as *P. aeruginosa* is known to secrete pyocyanin in presence of *S. aureus* [284, 267]. As well, new transporter reactions were added to *iYS854* to match the Chemically Defined Medium used in Halsey et al. [285], and such that the well-studied WTA-null non-lethal *Staphylococcus aureus* mutant $\Delta tarO$ could be accounted for [286], in order to resolve the ambiguities raised by Seif et al [163].

Accuracy of the models was estimated using the MEMOTE community tool for assessing GSMM quality [169]. *iPae1146* scored low at 23 %, mainly due to its lack of annotations, while *iYS854* scored at 75%. In addition, network topology issues were reported in Table 4.1. The *iPae1146* model contains a smaller number of exchange reactions, about twice as many blocked reactions, and a significantly higher number of reactions implicated in stoichiometrically balanced cycles.

Metabolic models	**P.A.**	**S.A.**	**Consortium**
MODEL STATISTICS			
Number of reactions	1495	1454	3241
Number of metabolites	1283	1338	2752
Number of genes	1148	866	2014
Number of external metabolites and exchange reactions	171	239	293
Number of reactions usable for crossfeeding	/	/	410 (239 + 171)
MEMOTE AND CONSISTENCY ANALYSIS RESULTS			
Universally blocked reactions	637	293	1031
Orphan metabolites (consumed but not produced)	54	39	93
Dead-end metabolites (produced but not consumed)	84	61	145
Reactions in stoichiometrically balanced cycles	253	46	697
Extracellular metabolites without exchange reactions	29	38	/
COMPRESSION RESULTS ON CSP MEDIUM			
CSP medium metabolites	47	56	57
Blocked reactions on CSP medium	673	485	1296
Number of compressed reactions	470	505	1062
Number of metabolites of the compressed network	252	440	600
MINIMAL CUT SETS OF SIZE THREE OR LESS			
Essential reactions on CSP medium	183	309	2
Synthetic lethal pairs of reactions	184	282	over $50 \cdot 10^3$
Synthetic lethal triplets of reactions	216	347	over $50 \cdot 10^3$
Total number of MCSs of size three or less	583	938	unknown
AFTER TESTS FOR CONSORTIUM GROWTH RECOVERY			
Single-species MCSs nullified by consortium metabolite exchanges	68	199	/
Single-species MCSs remaining lethal on the consortium	515	739	/
Consortium-level MCSs (potential enzyme targets)	65		/

Table 4.1: *Pseudomonas aeruginosa* (P.A.), *Staphylococcus aureus* (S.A.) and consortium model statistics. Models were individually curated and compressed, then essential reactions, synthetic lethal pairs, and synthetic lethals triplets were computed on the single species models and on the dual species consortium model. MCSs computations of synthetic lethal pairs and triplets on the consortium model were too expansive and thus stopped after one week.

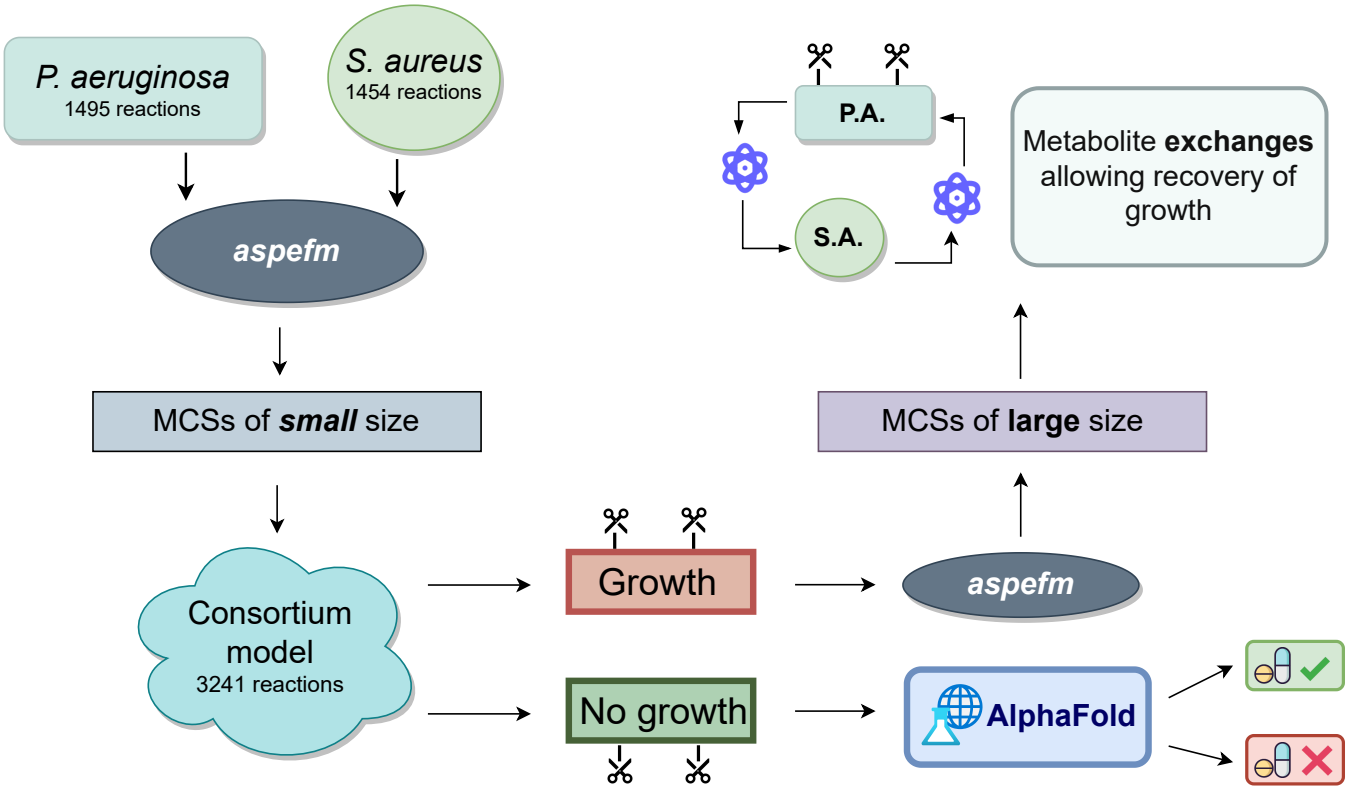


Figure 4.9: Overview of the main study. MCSs are separated into two kinds: small size and large size. The former are computed on single-species models, then tested for growth recovery on the consortium-level model. For MCSs which are no longer lethal on the consortium model, larger size MCSs are computed, revealing metabolite exchanges. For MCSs that are still lethal on the consortium, interspecies protein structure alignments of AlphaFold structures are performed to estimate their druggability.

As well, 29 extracellular metabolites from *iPae1146* and 38 from *iYS854* were found to lack exchange reactions. Notable metabolites from these lists, which are thus excluded from the possible metabolic interactions between bacteria, include formate for *iPae1146* and citrate for *iYS854*.

The models were constrained to CSP Chemically Defined Medium. For all 47 CSP medium metabolites metabolized by *P. aeruginosa* and all 56 CSP medium metabolites metabolized by *S. aureus*, exchanges lower flux bounds were set to arbitrary flux values in accordance to their relative quantity in the medium.

4.5.2 Consortium model construction and analysis

A consortium model of *P. aeruginosa* and *S. aureus* models was constructed using exchange reactions of both models as means for crossfeeding. All reactions of the models were subtitled with '**PA**' or '**SA**' depending on where they came from. The original extracellular compartment of the models were remade into the extracellular compartment of *P. aeruginosa* and the extracellular compartment of *S. aureus*, and an additional extracellular compartment was added, with all previously exchange reactions now yielding a metabolite into that compartment.

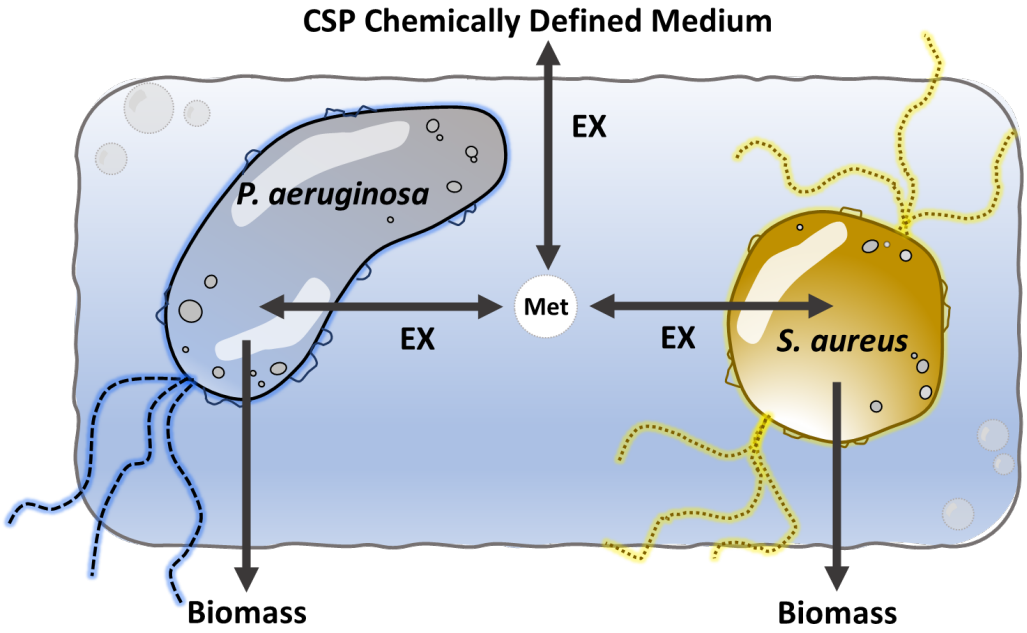


Figure 4.10: Diagram of the *P. aeruginosa* and *S. aureus* consortium model in CSP Chemically Defined Medium, including a view of metabolite exchange mechanisms. Extracellular metabolites are symbolized by "**Met**". Exchange reactions are symbolized by "**EX**".

Additional boundary exchange reactions were added to all newly created extracellular metabolites using COBRAPy [72], and the new exchange reactions fluxes were constrained to correspond to the CSP Chemically Defined Medium [271]. Finally, all reactions that were previously exchange reactions of the *iPae1146* and *iYS854* GSMMs became reactions that could be used for crossfeeding.

We set arbitrary flux bounds of [-20, 20] for all crossfeeding reactions, in accordance with the minimum possible uptake flux in CSP Medium, which is set to -20 mmol/gDW/hr for O_2, as is standard for that metabolite in COBRA models. A view of the consortium model, including crossfeeding in CSP Chemically Defined Medium, is presented in Figure 4.10.

Note that for computation of MCSs – and thus essential reactions, synthetic lethal pairs and triplets – on a model alone, its biomass is taken as the target reaction. As such, when looking for which crossfeeding reactions complement a cut set of *P. aeruginosa* or *S. aureus* alone, only one of the two consortium biomass reactions should be taken, the targeted biomass reaction in question. The full procedure for computing MCSs revealing crossfeeding interactions with *aspefm*, from the sets of lethal MCSs with growth recovery of either bacterium, is presented in Figure A.4.

Separately, for modelling growth on the consortium model, the FBA objective reaction is defined as the sum of both biomass reactions. Thus, for computation of MCSs which are lethal to the whole consortium model (Figure 4.6), both biomasses were taken as the target reactions.

4.5.3 Network compression

To help with computation efficiency, models are compressed using the network compression procedure developed by the von Kamp team, as part of the *pip* package `efmtool_link` from the Klamt lab [W16]. The tool relies on excluding blocked reactions and correcting reversibilities through Flux Variability Analysis [71], then applying a nullspace-based compression method from *EFMTool* [139].

The general principle behind nullspace-based compression was introduced in *METATOOL* [109], and later re-explored in [126] and [174]. Linearly dependent lines from the stoichiometric kernel are regrouped into a same reaction. This allows reactions that always operate together, *i.e.* their fluxes are linearly dependent to each other, to be regrouped into the same reaction subsets. For MCSs analysis, the linear coefficient factor between reactions in the same subset is of no importance, thus MCSs decompression is trivial.

4.5.4 Network dualization

As described in Ballerstein et al [187], it is possible to describe the problem of computing MCSs as the problem of computing particular EFMs on a dual metabolic network, meaning that the original network has to undergo a dualization conversion procedure.

We formulated the problem by making use of the MILP version proposed by von Kamp and Klamt in 2014 [153], which excludes some of the linear variables and constraints introduced by Ballerstein. A notable feature of this method was defining an inequality constraint instead of an equality constraint for metabolites of the dual network that were originally irreversible reactions.

All reversible dual reactions are split into two irreversible dual reactions. As in the formalisms defined by Ballerstein and von Kamp, the reactions corresponding to reversibility constraints are the only ones to which subset-minimality applies, meaning the other linear variables are free to be either strictly positive or equal to zero following whether it suits the linear program.

4.5.5 Using *aspefm* to compute MCSs

The procedure is applied using *aspefm*, our Answer Set Programming (ASP) constraint propagation method. The *aspefm* tool is distributed at `https://github.com/maxm4/aspefm`.

Given the compressed metabolic network in dual form, *aspefm* defines a logic program in Answer Set Programming able to enumerate all or only a subset of Minimal Cut Sets. ASP is a logic programming language, meaning it requires a declarative logic program to be defined as input. It is adapted for combinatorial problems thanks to its SAT-based solving and automatically searches for solutions, the so-called answer sets. SAT refers to the well-studied Boolean satisfiability problem. The solver used by *aspefm* is *clingo* extended to linear constraints through the *clingo*[LP] [210] interface and a *cplex* backend.

Subset-minimal solutions are obtained through the set minimization heuristics of *clingo*, *aspefm*'s solver. Unlike in a Mixed Integer Linear Programming (MILP) formalization, no minimization on the size of the solutions searched is performed. Thus, solutions of large size may be returned by the solver in reasonable time. Once a solution is found, it is added as a negative constraint for the next computation. New solutions are thus found depending on the order of search and enumeration.

An algorithmic amelioration was made to the *clingo*[LP] code. The algorithm for finding core conflicts from conflicting linear constraints was formerly implemented in recursive pure Python. We replaced it by *cplex*'s internal conflict refiner function [W8]. The code runs about 10-times faster on small and on larger models.

Another amelioration was made for direct enumeration of solutions with linear constraints, which are known to modify the solution space and thus its minimal solutions [150, 157]. A solution checker was implemented and it is called to verify the minimality and validity of the solution, in the case of MCSs with biomass as a target, it is a simple FBA call. This linear programming call should be very fast in computation time compared to the overall cost of the combinatorial exploration.

As with EFMs computation, additional constraints can be added to *aspefm*, only yielding a subset of all possible solutions. These constraints need to be expressed on the reactions from the compressed dual network, or undergo a conversion process if these relate to the original network.

4.5.6 Adding constraints to *aspefm*

Using *aspefm*'s input format, it is possible to add constraints to the computation of MCSs. Let us consider $\mathcal{C}ut$ the set of all reactions and c_r Boolean variables representing if a reaction is cut or not. For example a size constraint, meaning that cut sets above a certain size P will not be computed :

$$\text{Card}\{c_r \mid c_r = 1,\ r \in \mathcal{C}ut\} < P \tag{4.10}$$

Supposing we have a non-empty list of "unwanted reactions" of interest $U \subset \mathcal{C}ut$. The following negative Boolean constraint can be added:

$$\bigwedge \neg\, c_r \quad \forall r \in U \tag{4.11}$$

This will specify to the solver to only compute MCSs containing none of those reactions.

Supposing we have a non-empty list of "wanted reactions" of interest $W \subset \mathcal{C}ut$. The following positive Boolean constraint can be added:

$$\bigwedge c_r \quad \forall r \in W \tag{4.12}$$

This will specify to the solver to only compute MCSs containing all of those reactions. Note that since these postive Boolean inputs translate into adding linear constraints bounding the flux cone, our solution checker implemented in *aspefm* must be called in that case.

4.5.7 Retrieving information at the gene and protein level

Gene and protein level information was reviewed as part of the model curation process. Gene products defined in the SBML models were modified to all match UniProt or TrEMBL entries [34], in particular for *iYS854*. *iYS854* genes used new locus tags that yielded no UniProt query results, so genes were renamed to their old locus tags. UniProt entries were seeked for using UniProt BLAST. For the gene products with no match in the str. JE2 strain, an homologue from a close *S. aureus* strain was used. In the process, new RDF annotations were added for future modellers.

For both models, overall Boolean Gene-Protein-Reaction rules (GPRs) were simplified into canonical Disjunctive Normal Form (DNF), which helps with exhaustive enumeration of gene knock-outs from sets of reactions. Minimal sets of genes corresponding to the MCSs were computed using ASP logic programming with subset-minimization heuristics, as the problem can once again be expressed as minimal Boolean affectations to {*True*}.

Minimal sets of genes are non-trivial subset-minimal affectations to {*True*} respecting the GPRs Boolean formulae. However, as underlined in Machado et al [162], some gene products are ubiquitous, appearing in multiple reactions, meaning that the minimal sets of genes obtained from lethal MCSs of reactions might not necessarily be minimal in the number of genes to be knockout.

In order to better describe information at the gene or protein level, it is required that each reaction is associated to at least one gene. So for transporter reactions that are not associated to genes, we added a dummy association, which in fact corresponds to the either backwards or forwards direction of that reaction, since reactions are split for MCSs computation. In practice, almost all reactions lacking genes are transporters, and forwards transporter reactions should also be assumed to be dependent on the presence of external metabolites in the medium. Some transporter reactions are also known to be spontaneous and annotated as such in their GPR association rules.

4.5.8 Retrieving enzyme cuts targeting both bacteria for therapeutic action in human

To derive the sets of MCSs in common to both bacteria, reactions in MCSs were summed together, and MCSs from both models which had the same total mass balance equation were kept. The sets of MCSs in common were then converted into the corresponding minimal sets of genes with ASP logic programming, using GPRs derived from each bacterial model.

Protein structure models were retrieved using the AlphaFold entries [43] associated to UniProt genes, and structure alignment was performed using algorithm FATCAT 2.0 [287]. All retrieved genes using the model's GPRs could be associated to UniProt entries, but very few of these entries had known crystallized structures in PDB. As such, only AlphaFold entries were compared together.

Enzyme targets with good interspecies protein structure alignments were tested for existence of human homologues in the UniProt database using their E.C. Number [15]. For enzymes with no human homologues, enzymes were kept as possible targets. For enzymes which had human homologues, the corresponding AlphaFold structure predictions were retrieved, and further protein structure alignments were performed to check druggability.

4.5.9 Interspecies protein structure alignment of enzyme targets

Protein structures with a FATCAT alignment RMSD above 3 Ångström were considered dissimilar between species. Excluded enzymes comprise Glutamyl-tRNA synthetase and reductase, and in particular, 3-dehydroquinate dehydratase, for which we confirmed through UniProt and InterPro that between *P. aeruginosa* and *S. aureus*, the enzymes have very different protein domains. Overall, we found that this threshold for scores of alignment between AlphaFold structure predictions was a useful indicator of whether or not the proteins were similar between species.

Many MCSs would equivalently include exchange reactions or transporter reactions. Thus, we called these MCSs 'growth medium dependent'. These are possible drug targets, but only in auxotrophic conditions, when amino acids are depleted from the medium. Although these cut sets might be difficult to use as therapeutic targets, we found these cut sets to be at least informative in the sense that if only the medium is depleted then the enzymes become essential and a drug targeting them would have effect. These are also indicator of the main enzymes relating to a particular amino acid metabolism.

4.5.10 Estimating quality of enzyme targets for therapeutic applications to humans

Enzyme targets with good interspecies protein structure alignments were tested for existence of human homologues in the UniProt database using their E.C. Number. For enzymes with no human homologues, enzymes were kept as possible targets. For enzymes which had human homologues, the corresponding AlphaFold structure predictions were retrieved.

FATCAT alignments between their *S. aureus* and *P. aeruginosa* structures and all human homologues were performed. Then, for each alignment, three categories were considered, based on FATCAT scores: 'Structurally equivalent': $(RMSD < 3)$ and $(score < 10^{-6})$, 'Structurally similar': $(3 \leq RMSD < 5)$ and $(10^{-6} \leq score < 10^{-3})$, 'Structurally dissimilar': $(RMSD \geq 5)$ and $(score \geq 10^{-3})$. Then, only enzymes with strictly less than 50% human homologues which had 'Structurally equivalent' alignments with both bacteria were kept as possible targets.

Very few alignments ended up in the ‘Structurally similar‘ category, but the inclusion of the FATCAT score this time in addition to RMSD allowed to rule out alignments with low RMSD but high score, and inversely. An example of enzyme with only one human homologue and such an alignment falling in ‘Structurally similar‘ but not ‘Structurally equivalent‘ is glucosamine-1-phosphate N-acetyltransferase. The bacterial enzymes have different protein domains than its human counterparts.

Most of the enzymes with human homologues were eliminated through this procedure, the analysis excluded thirteen potentially not targetable enzymes out of sixteen. Overall, most enzymes which alignments were classified as ‘Structurally equivalent‘ had identical protein domains between human and bacteria, despite the very large phylogenetic distance, thus making for dangerous therapeutic targets.

Finally, we decided to exclude the target nucleoside diphosphate kinase (ATP:UDP) (36.73% ’Structurally equivalent’, 18 similar out of 49 homologues), even though it was scored the highest in terms of interspecies protein structure alignment, as we believe targeting this enzyme would not be viable in human cells. As a result, only two enzymes with human homologues were kept at the end of the analysis. We represent the results of the analysis in Figure 4.8, Figure 4.14, and Figure 4.13.

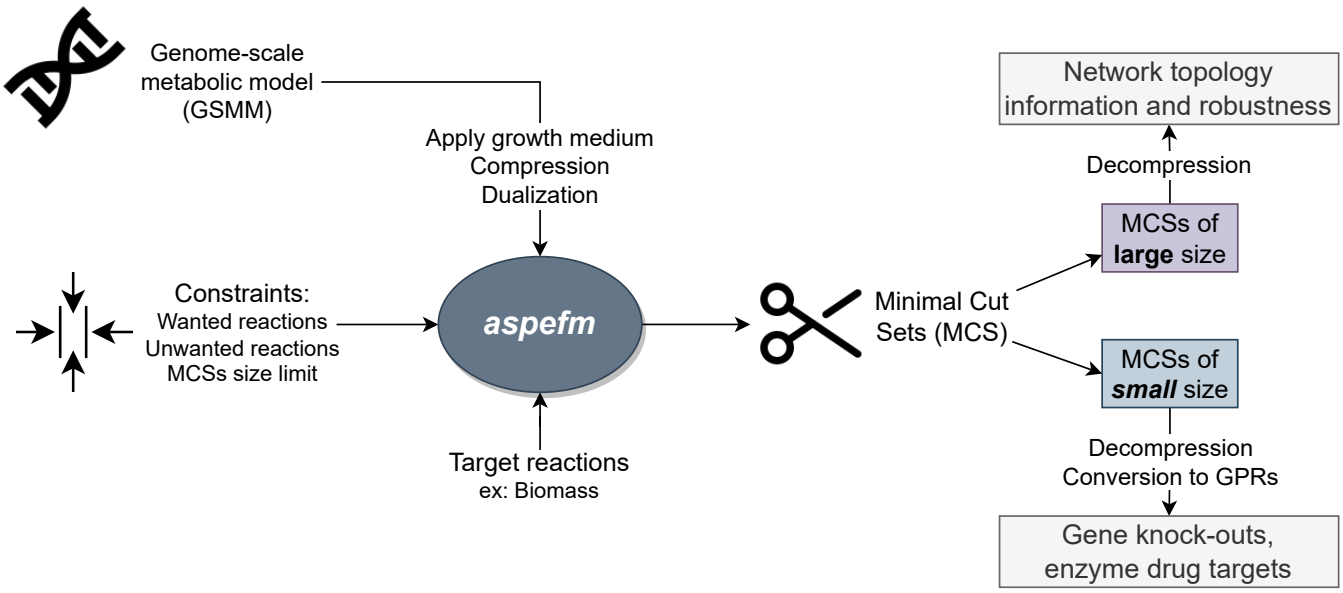


Figure 4.11: *aspefm* framework applied on the GSMMs of *Staphylococcus aureus*, *Pseudomonas aeruginosa*, and on the consortium model.

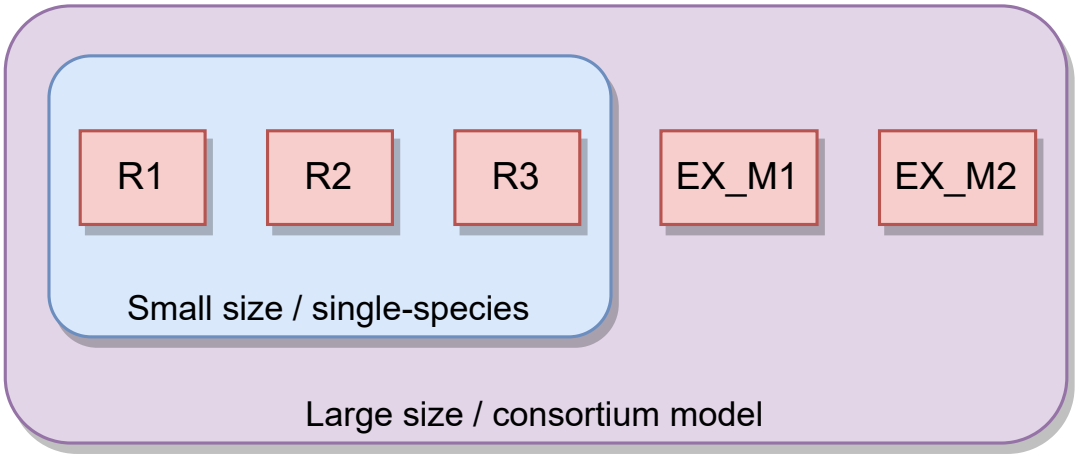


Figure 4.12: Minimal Cut Sets of small size and of large size on single-species models and a consortium model

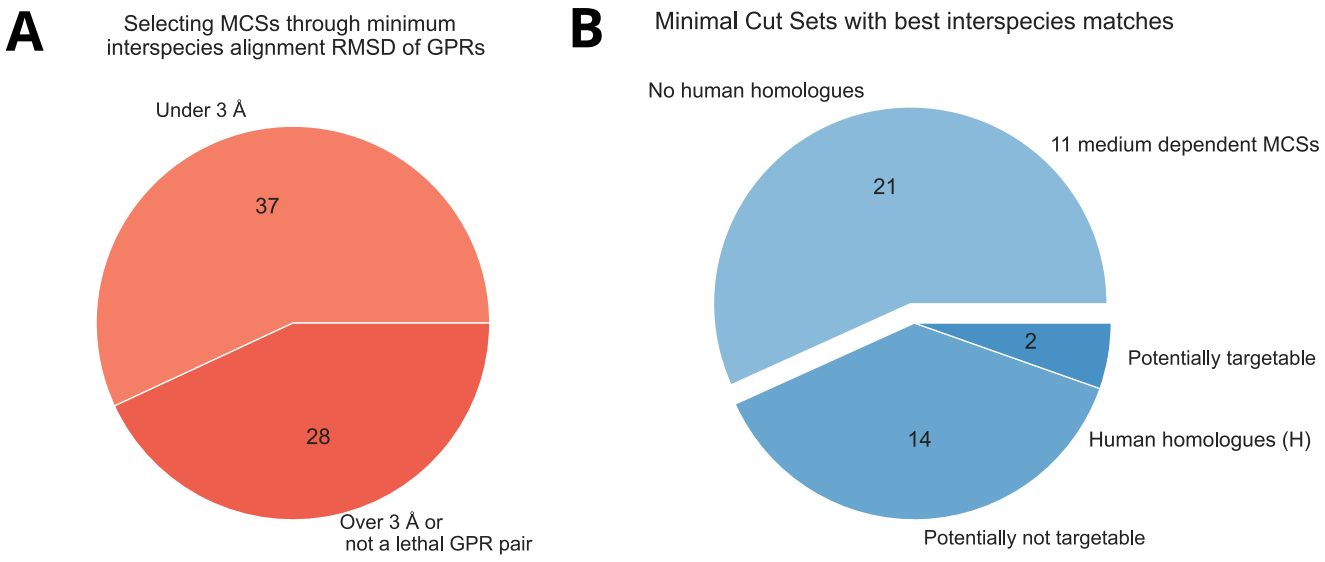


Figure 4.13: **A**: Analysis of GPRs and RMSD scores of interspecies protein structure alignments for the enzyme targets from GPRs, corresponding to the 65 MCSs. **B**: Summary of the search of human homologues, leaving 23 'good' potential targets, including 11 medium dependent ones, out of the 37 considered targets.

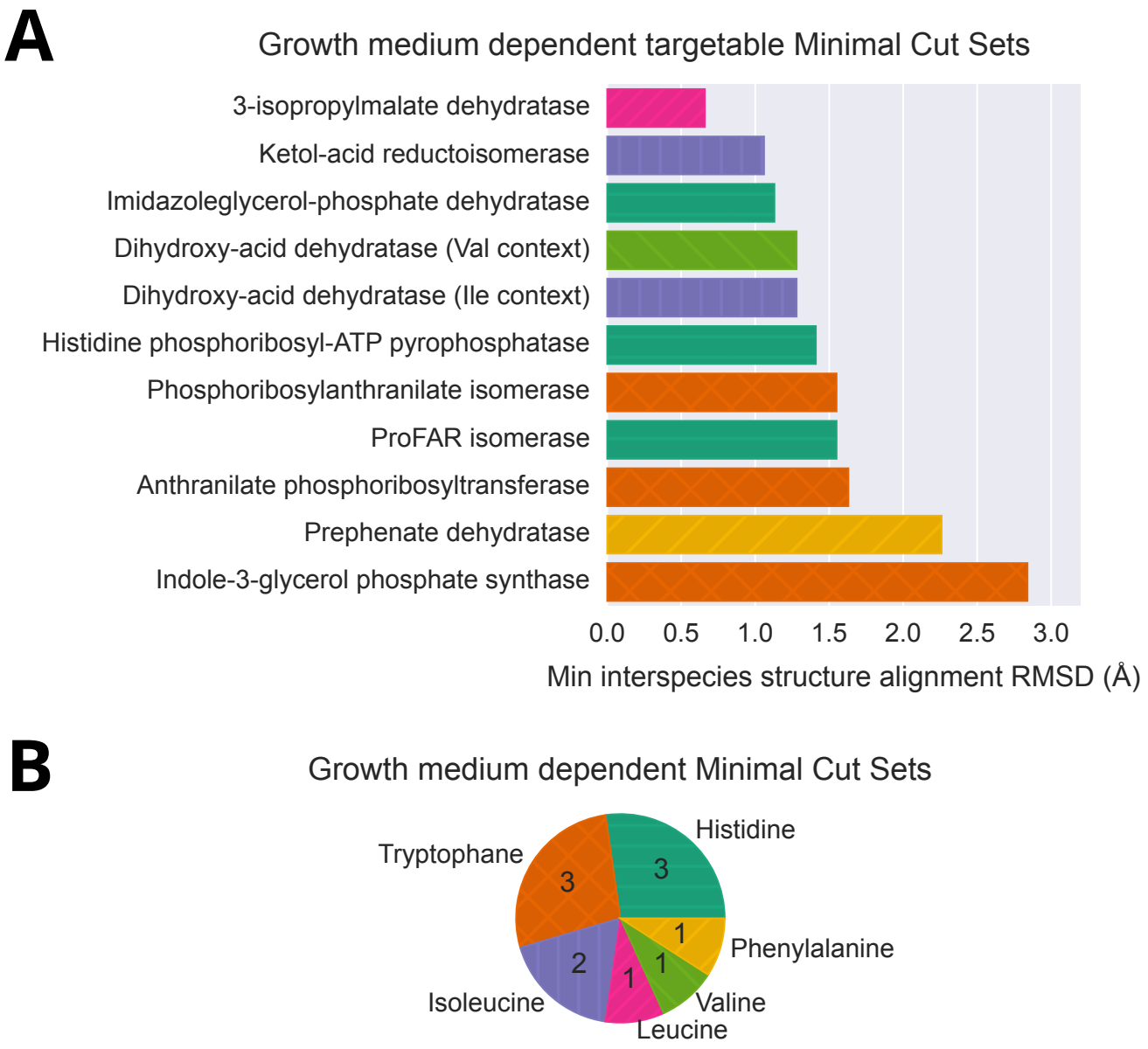


Figure 4.14: **A**: Enzyme targets dependant on the presence of amino acids in the growth medium. Bars heights are minimum Root Mean Square Deviation (RMSD) of atom positions, measured in Ångström, resulting from protein structural alignment of GPR pairs. **B**: Associated amino acids and pie distribution for each MCSs in **A**.

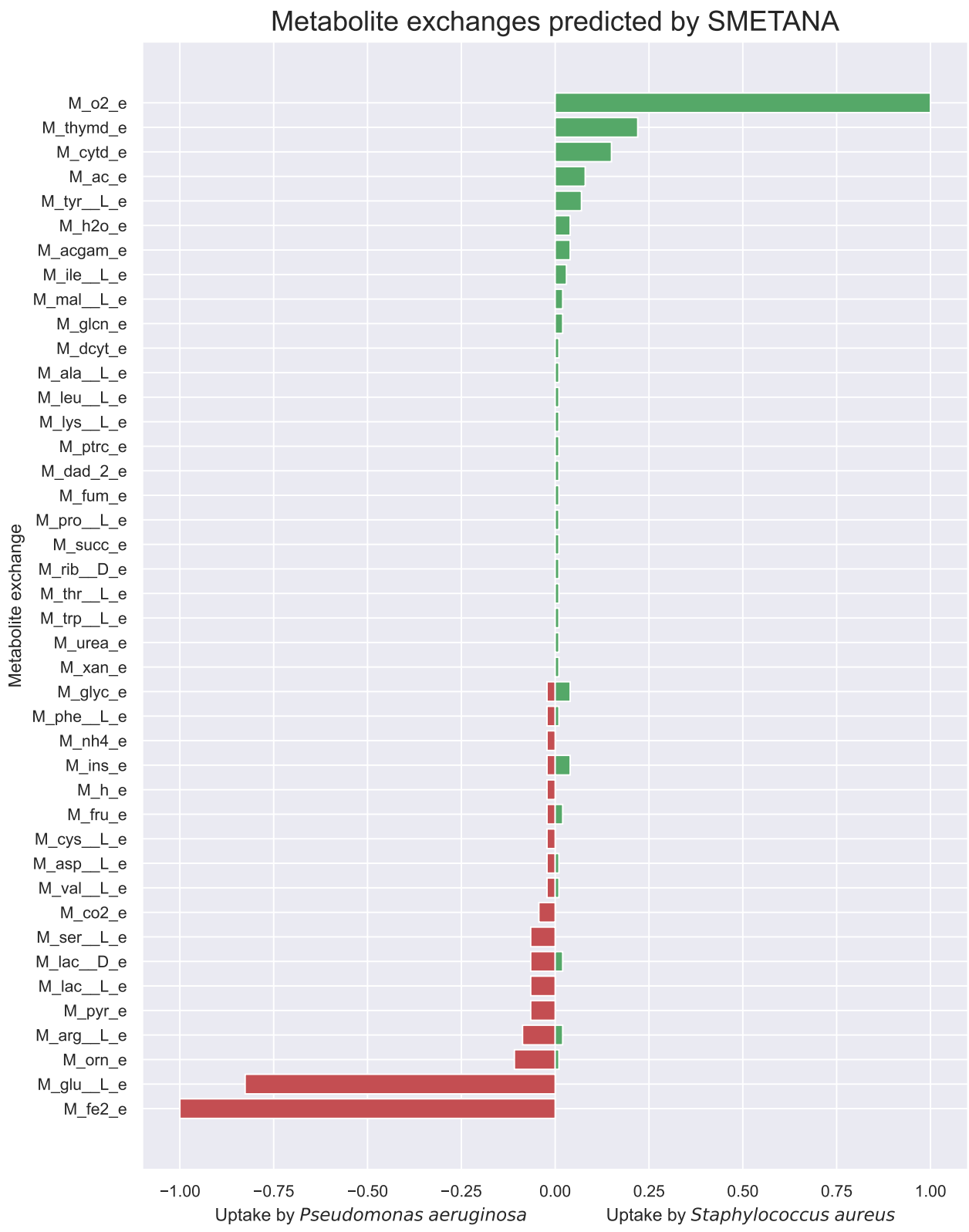


Figure 4.15: Estimated potentials for metabolite exchanges between the two bacteria according to SMETANA. Abscissae are SMETANA scores, multiplied by -1 when S.A. is the donor instead of P.A.

4.6 Discussing the scope of our analysis

aspefm is a powerful and promising tool for metabolic systems analysis, able to compute EFMs while respecting any applied constraints, whether they are logical or linear, and it can be extended at will by the *clingo* Python interface [201, 213]. Here, we extend the tool to the computation of MCSs from metabolic models of single and multispecies systems. MCSs are mathematical objects with great engineering, ecological, and therapeutic potential as they identify combinations of reactions within large, highly connected networks that have outsized abilities to influence phenotype. As illustrated in Figure 4.6, automatic exploration of MCS solutions is possible. Additionally, *aspefm* has the advantage of identifing more than twice as many MCSs as the concurrent MCSs tools within a reasonable runtime and it is not restricted by the size of MCSs in number of reactions. Note that the enumeration of these subsets of solutions with *aspefm* as well as concurrent method *cnapy* is non-deterministic, as demonstrated by the variability in MCSs sizes.

In order to make full use of the enumeration capacities of our *aspefm* tool, we devised a MCSs analysis to study potential crossfeeding interactions within the consortium. The tool was able to enumerate solutions on the consortium-level with ease, despite the network comprising about three thousand reactions. Usually, MCSs of size three or less are often considered the most biological relevant. This logic follows the argument that disrupting more than three different genes, or interfering with more than three mRNA targets or the drugging of more than three enzymes at the same time is challenging. MCSs of small size are readily enumerated from single-species metabolic models, and can be performed with existing SLs enumeration tools or with MCS enumeration tools. However, we argue that MCSs of large size can provide valuable information on network robustness and fragility as well as potential metabolite exchanges, whether for single species or consortia analyses. Compared to other methods, *aspefm* showed better performance for enumeration of MCSs of large size on the consortium model.

Our analysis proposed the usage of MCSs of small size on single-species models as constraints for the computation of MCSs of large size on a consortium-level (Figure 4.9). Single-species MCSs were tested for the recovery of growth on the consortium model, and if metabolite exchanges permitted growth recovery, then MCSs of large size were computed, giving light into the metabolite exchanges in question. To do so, all reactions that were not metabolite exchange reactions were set as negative Boolean inputs, and MCSs of small size were used as positive Boolean inputs, a type of input constraint which is not to our knowledge possible to integrate with other MCSs tools. We implemented an additional FBA check for every MCSs yielded by the solver to verify the minimality of the solutions.

Therefore we present the utility of an exhaustive yet constrained metabolic pathways analysis, through application of biological relevant constraints, as we have presented previously for EFM analysis [89]. Rather than a complete enumeration of all cut sets, which like EFM enumeration is only achievable on modestly sized reaction networks, we enumerate a subset of MCSs while answering a specific biological question. Though, the procedure is unfortunately limited by enumeration capabilities of our tool, which wasn't capable to conclude on whether all solutions were enumerated, and needed to be restricted by a time limit of 1.5 days.

This new application of MCSs highlights the potential role of metabolite exchanges and metabolite crossfeeding on consortium functioning and resiliance to theapeutic efforts (Figure 4.7). We strengthened the study by combining the analysis with drug target predictions for the single-species MCSs that did not regain growth by consortial metabolite exchanges (Figure 4.8). Meylan and coauthors have showed strong evidence that antibiotic tolerance might be affected by the impact of metabolite exchanges, in particular they showed that uptake of fumarate/glyoxylate by *P. aeruginosa* respectively increases/decreases its tolerance to aminoglycosides [288, 289].

Metabolite exchanges can also be retrieved on consortium models through using FBA models and EFMA [290, 291], and we believe those exchanges to be representative of at least three possible mechanisms: cross-feeding (bacterial co-operation), metabolite leakage (overflow of metabolites necessary for biomass production), and simply recuperation from the bacterial medium (a biofilm medium is composed not only of living bacteria but also of necromass) [292]. Interestingly, we compared our tool to SMETANA, a tool for estimating growth-dependent species exchanges in bacterial consortia [293]. We could retrieve that 23% of our highlighted metabolite exchanges were not predicted by SMETANA, as expected since their study of metabolite exchanges is not performed after cutting several reactions. The analysis performed by SMETANA is given in Figure 4.15.

Our study positions itself in recent efforts from the metabolic modelling community in bringing metabolic models to the cellular consortium level [193]. There is a growing interest in multi-species models such as AGORA for modelling of gut microbiota [294], and multicellular models such as whole-body human models [171]. However, we found that even the well-curated models presented in our study were not exempt from curation and modelling errors. For example, *iPae1146* lacked reactions for aminoacyltransferase reactions, which are notably different for *Pseudomonas aeruginosa*, although including these reactions is now standard [37]. Interestingly, the aminoacyltransferases reactions were present on the smaller *P. aeruginosa* PAO1 model *iMO1056* from 2008 [273], but imported from *E. coli* pathways, rather than *P. aeruginosa*. Another instance included the lack of a functioning formate transporter for *iPae1146* and of a citrate transporter for *iYS854*, although these metabolites might have significant interspecies exchange potential.

Data from UniProt [34] and AlphaFold [43] was used to predict therapeutic targets based on protein structure similarities. Protein structure data was not available for many of the enzymes of interest which limited the mapping of the specific protein sequence to a model structure. We therefore used the application to quantify similarities of the predicted structure of two protein sequences rather than the overall accuracy of the 3D structure. If the predicted protein structures were deemed similar enough, it is hypothesized that an inhibitor would be more likely to be a ligand to both. AlphaFold derived preemptive ranking for further analysis and provided a uniform method of applying the computational workflow. Enhanced predictions will be possible with better crystallisation of *S. aureus* and *P. aeruginosa* protein structures, and through further analysis of the enzyme targets with DrugBank [44] for existence of medically approved inhibitor ligands.

Finally, a relevant point of discussion is the conversion of enzymes to gene and reaction data, ie. GPRs. GPRs are AND and OR Boolean rules, respectively symbolizing complexes and isozymes are transformed into reactions of their own. Subunits numbers are unfortunately not incorporated in the standard specification of metabolic models

[69], though this might eventually change thanks to the development of models incorporating resource allocation data [166]. Previous studies have incorporated GPRs into the stoichiometric matrix [162, 189]. This has the downside of making flux go through factice gene reactions. In contrast, our approach for converting reactions to proteins using GPRs made use of logic programming, to keep illustrating its application – as GPRs are Boolean logic rules – and to keep expanding the *aspefm* framework. GPRs are of major importance when analyzing the enzymes gene or protein data, and were useful for us when retrieving AlphaFold entries [43]. They were however of lesser importance for MCSs of large size.

To summarize this study, we argue that both categories of MCSs: small size and large size, equally served their purposes, when used with the biomass synthesis reaction as target. A subset of all large sized MCSs was able to reveal consortium-level metabolite exchanges that could only be observed after deletion of one, two, or three pathways; and, just as importantly, the essential reactions and synthetic lethals for which there was still no consortium growth could be analyzed for their ability to be drug targets. We thus propose that there is a strong need for MCSs enumeration tools such as *aspefm*, and for metabolic modelling methods as a whole in the context of microbial ecology, medical intervention and drug discovery.

In conclusion, the medical treatment of *Staphylococcus aureus* and *Pseudomonas aeruginosa* is a major challenge negatively influencing the lives of millions of patients every year. In this study, we develop, analyze, and demonstrate useful applications of a novel metabolic modelling strategy based on logic programming and the exhaustive enumeration of biochemical pathways in metabolic networks. The algorithms were able to predict nonintuitive metabolic exchanges between multiple bacteria in a consortium-level model illustrating applications to the field of microbial ecology. Additionaly, the systems biology algorithms were able to identify rational sets of therapeutic targets that can inhibit the growth of single species and also inhibit the growth of crossfeeding consortia while avoiding host enzymes with similar protein structures. We identify a list of promising enzyme targets, which share protein structure similarities, and can potentially be targetted by the same therapeutic agent simultaneously and thus shut down activity of the entire consortia.

Chapter 5

Perspectives and conclusions

In addition to logical and linear constraints, we attempt to propose with *aspefm* a complete formalism capable of integrating any possible constraint. To make this possible, we have to extend our *aspefm* tool with other features, which we refer to here as *asepfm* extensions.

For instance, a nonexhaustive list of the kinds of constraints that we aim to integrate with *aspefm* is presented in Figure 5.1. Some of the constraints ideas presented here and not explained earlier will be developed in this chapter.

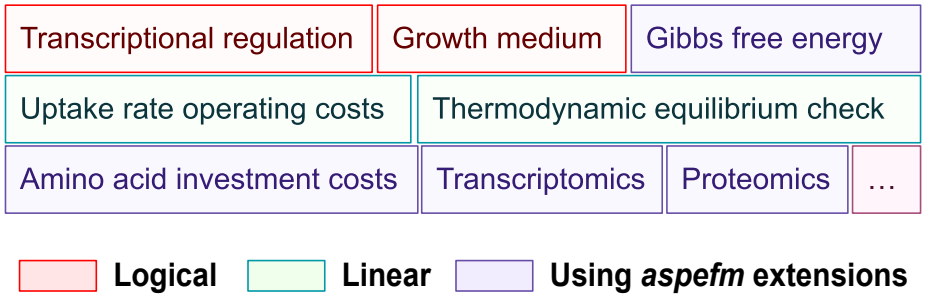


Figure 5.1: Types of constraints: logical, linear, and using *aspefm* extensions

In this chapter, we will present the core interface for incorporating new features to *aspefm*, the theory propagator. As well, we will present ameliorations to *aspefm*, and new kinds of constraints and features that could be integrated to the tool and applied to new biological problems in the future.

Of note, the decomposition of FBA solutions into EFMs is an extension with very high significance. As well, the EFMChecker and MCSChecker extensions were used for our publications, respectively used in the applications of section 3.8 and section 4.3.

5.1 Encoding *aspefm* extensions

As mentioned before, *clingo*[LP] uses an interface called theory propagator to encode the linear constraints and communicate with the LP solver *cplex*. This extension to *clingo* is in fact defined in a Python file called `clingoLP.py`, implementing the "Theory propagator" Python interface.

The theory propagator can act at four levels: *init*, *propagate*, *undo*, and *decide*, related to the propagation of Boolean literals within the SAT-based Conflict Driven Constraint Learning (CDCL) procedure [201]. This is an interface, that each new propagator should implement. The functions are the following: *init* allows one to define ASP atoms to be tracked during literal propagation, and to make initialization procedures dependent on command line arguments and on the ASP atoms in the system after grounding, *decide* allows one to check and possibly change the next propagated literal, *i.e.* heuristic behaviour; *propagate* is called each time a literal gets propagated, it deals with either incomplete solutions called partial assignments or complete assignments, and allows one to add clauses during propagations; *undo* is called when there is a backtrack, and a literal is no longer being propagated.

To encode an extension to *clingo[LP]*, several approaches could be taken. One might implement an additional theory propagator, and have *clingo* run with two propagators, the LP propagator, and the extension propagator. Or, alternately, one might directly integrate the extension into the LP propagator. After testing a bit with the first approach, we concluded that taking the second approach lead us to faster computation times and, in fact, better modularity. In Figure 5.2, we illustrate how the linear solving theory is integrated to *clingo*, making the tool *clingo[LP]*.

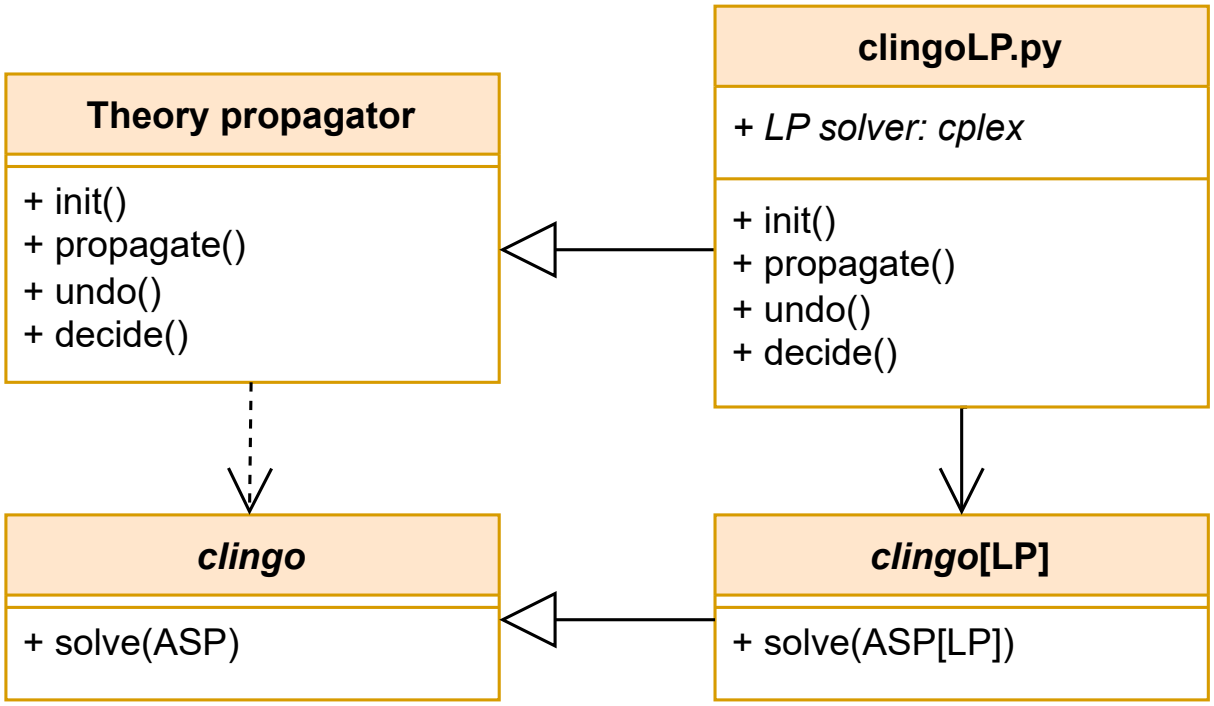


Figure 5.2: *clingo[LP]* as a theory propagator for clingo. Uses UML representation of inheritance. Dotted arrow indicates optional usage. Solid arrow indicates required usage.

The modulable extension system we envisioned was first illustrated with Binary Decision Diagrams (BDD), a well-known Boolean logic formalism and knowledge compilation technique [207, 295]. We called our extension BDD-Extension or BDDChecker, and it was used to encode Transcription Regulation Networks (see section 3.6), to test validity of solutions regarding transcriptional regulation. This was incorporated into the propagation of literals, each time a literal was propagated, we could call the BDD representation of the Boolean formula to check whether regulation is respected or not. More information is provided in French in section B.1.

An illustration is given in Figure B.3. Like *clingo[LP]*, which 'theory' relies on calling the LP solver *cplex* as an oracle to determine whether Boolean literal solutions are valid or invalid, we had the `BDDExtension` make calls to the BDD Python module *dd*. When solutions were found not valid with the BDD, *nogoods*, clauses of negative literals, were added to the computation, so that these partial assignments might never appear again. For EFM computation, and with negative Boolean constraints such as transcriptional regulation, doing so poses no problem, as if a subset of the support of an EFM does not respect such a desired property, then all EFMs containing that subset will still not respect that property [127, 143]. This is not true of the opposite, *i.e.* positive constraints and positive literals.

Since we decided to only keep a single propagator, and have modulable extensions to *clingo[LP]* instead, in our extension formalism, we define the same functions, providing a similar interface, except it is called during the propagation of *clingo[LP]* instead. While the `BDDExtension` was first encoded as a propagator, it could be converted into an extension without changing anything to the mechanisms and the results, and it even provided better performance. Content from those results, we implemented a flexible list of extensions system, illustrated in Figure 5.3.

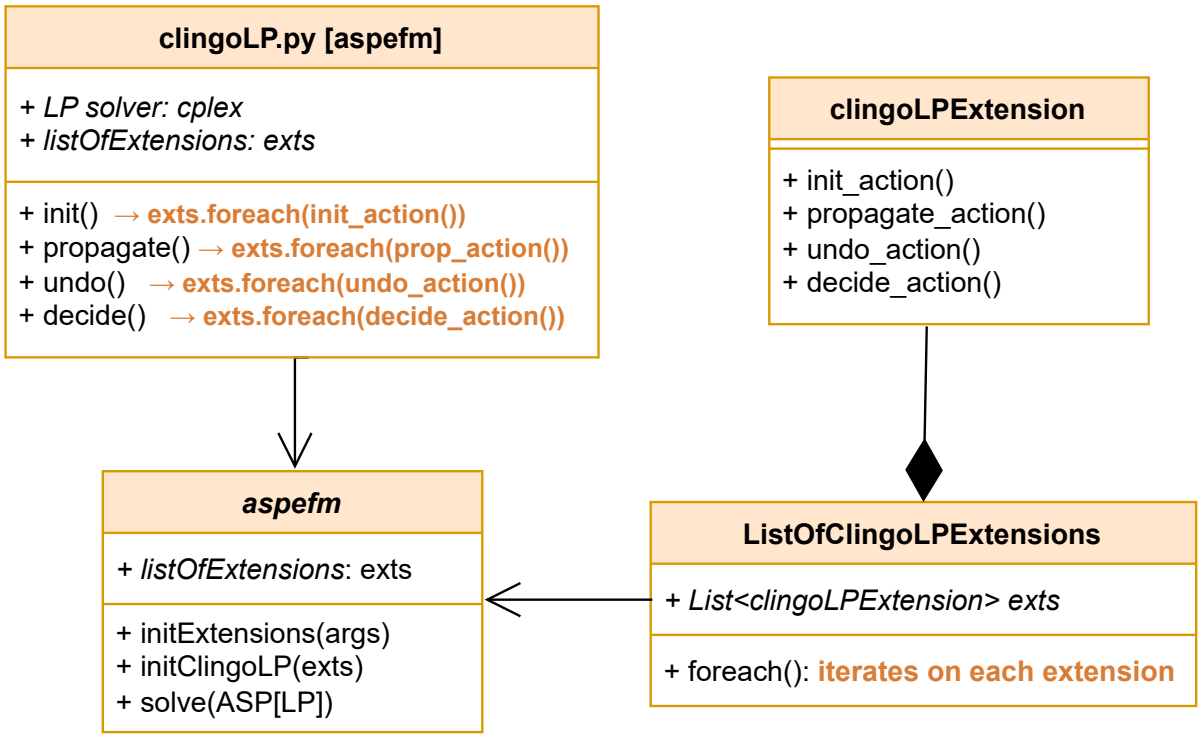


Figure 5.3: Flexible list of extensions system for *aspefm*.
Uses UML representation of aggregation. Solid arrows indicates usage.

5.2 EFMChecker and MCSChecker

In order to filter out MCFMs which appear when adding positive constraints, we devised an EFMChecker system and a MCSChecker as extensions which can be added to the ListOfExtensions (see Figure 5.3). Oftentimes one isn't simply interested in adding a single constraint: one wants to add several linear constraints which are positive, although not too many or else computation would be impossible, then many negative Boolean constraints; and finally even other kind of constraints that could help filtering out solutions, integrated into *aspefm* extensions, by adding nogoods during the propagation of *clingo[LP]* (see Figure 5.1).

For this reason, the ListOfExtensions system is of great utility, as one can add an EFMChecker to remove false solutions, while at the same time having another extension reduce the exploration space of EFMs thanks to the addition of nogoods. Note that in *aspefm*'s enhanced version of *clingo[LP]*, each *init*, *propagate*, *undo*, *decide* iteratively calls the corresponding actions defined in each extension of the list (see Figure 5.3). The order of which the extensions are called and initialized in ListOfExtensions therefore matters.

As their names suggest, EFMChecker is an extension of *clingoLP* that can filter out solutions that are not EFMs, and MCSChecker is an extensions that can filter out solutions that are not MCSs. EFMChecker performs the rank check or kernel test, as defined in Theorem 2.7.2, while MCSChecker instead performs a simpler call to FBA with the target reaction, since we opted for von Kamp's formulation rather than Ballerstein's (subsection 2.12.2). For this reason, these two extensions require data to be provided in command line parameters: for EFMChecker the S-matrix is required for the rank check, while for MCSChecker the SBML file is demanded, for quick use with COBRAPy.

A very notable feature of EFMChecker and MCSChecker extensions is that unlike the BDDChecker, they do not check the state of partial assignments. Instead, they only check the state of complete assignments, that is, assignments that are such that 100% of the literals have been propagated. Therefore, no computation time is wasted on partial assignments, as the test for the percentage of propagated literals simply always returns $\{False\}$ in these cases. Then, nogoods are only added when complete EFM or MCS solutions are retrieved, unfortunately filtering them out after they have already been calculated. However, the benefit of having *aspefm* only return correct EFMs or MCSs, with no post-processing filtering step as was observed in subsection 3.7.3, overthrows that inconvenience.

Extensions are encoded as Python files (Figure 5.5). If we specify no extension in argument to the command line, the ListOfExtensions is initialized empty and *clingo[LP]* runs as normal. We compared the execution times of standard *aspefm* and *aspefm* with the presence of checker extensions for the same problem instances and the extensions did surprisingly not drastically decrease the computation times, suggesting that the complexity of the check calls are much lower than the complexity of finding new solutions.

5.3 Decomposition of FBA solutions in EFMs

The decomposition of FBA solutions into EFMs needs a further restructuration of *clingoLP*, we named the new tool *clingoDCMP*. We applied the decomposition of FBA solutions to the C2M2NF network by Jean Pierre Mazat [238].

So far the results are promising and we believe the development of such a method would greatly help the constraint-based modelling community. The FBA flux vector data chosen for the EFM decomposition on C2M2NF originated from the exometabolomics data from Jain et al [32, 238]. We illustrated our program in Figure 5.4.

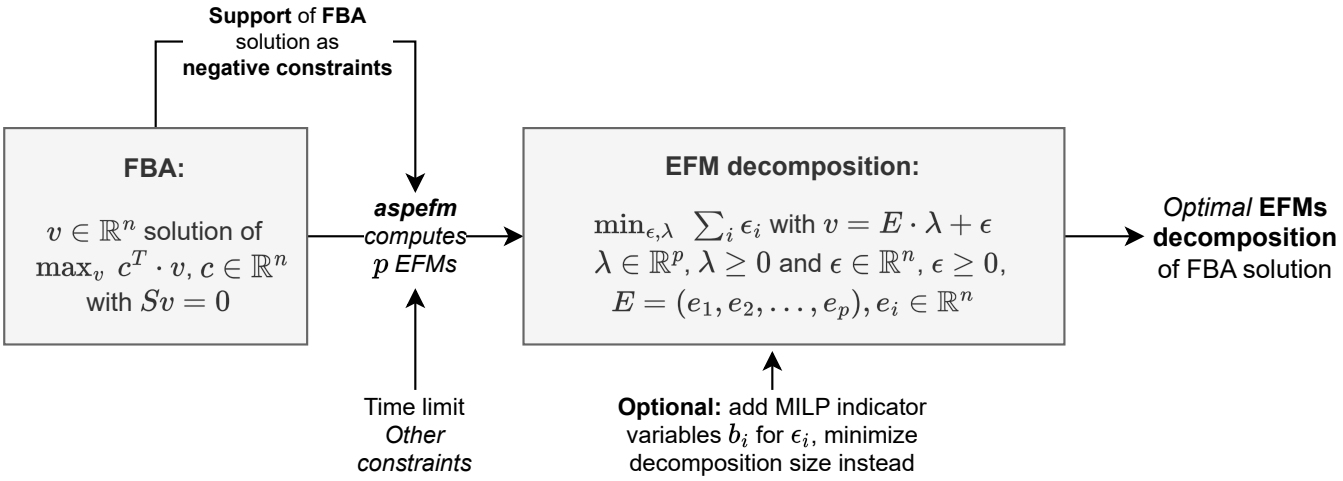


Figure 5.4: Method for decomposition of FBA solutions into EFMs with *aspefm*

In fact, for a single FBA solution, there might be an infinite number of EFMs decompositions that matches with the flux distribution. Or alternately, there might be no perfectly matching decomposition, considering these are real-valued solutions prone to numerical errors, and the numerical algorithms used to compute EFMs and FBA solutions might not be coordinated. The advantage of using *aspefm* on genome-scale models is that, since there are many possible EFMs decompositions, we could simply perform a constrained enumeration of EFMs until one decomposition is found, and this event becomes the stopping condition of our enumeration. This decomposition must therefore be chosen very close to the FBA solution. Indeed, unless we ensure complete enumeration, we could not truly know how close to the *global* optimal our *local* optimal solution is at the time when we stopped EFMs enumeration. Nevertheless, we believe that this application would be of great utility.

We were inspired by the ideas from Jean-Marc Schwartz [131] to decompose a flux distribution into EFMs, based on quadratic programming, but instead, we transformed the problem into a standard linear regression, except with only positive coefficients. The problem then becomes minimizing the sum of ϵ residual values, instead of the squared sum of coefficients (Figure 5.4). We implemented the linear regression test in *gurobi* [W9], and while it worked fine, we struggled into transforming the problem into a MILP that minimizes decomposition size instead of minimizing residuals, due to *gurobi*'s MILP performance being lacking. Minimizing decomposition size as our local optima, although great for analysis, is therefore a harder criterion to envision on genome-scale models. Note that when testing the MILP, we impose a fixed very small bound for the sum ϵ to stay close to the optimal sum of residuals. As well, we independently tried finding EFMs decompositions from the C2M2NF EFMs respecting our constraints using appropriate linear regression techniques for sparse matrix and positive coefficients such as lasso regression using Scikit-learn [296, 244].

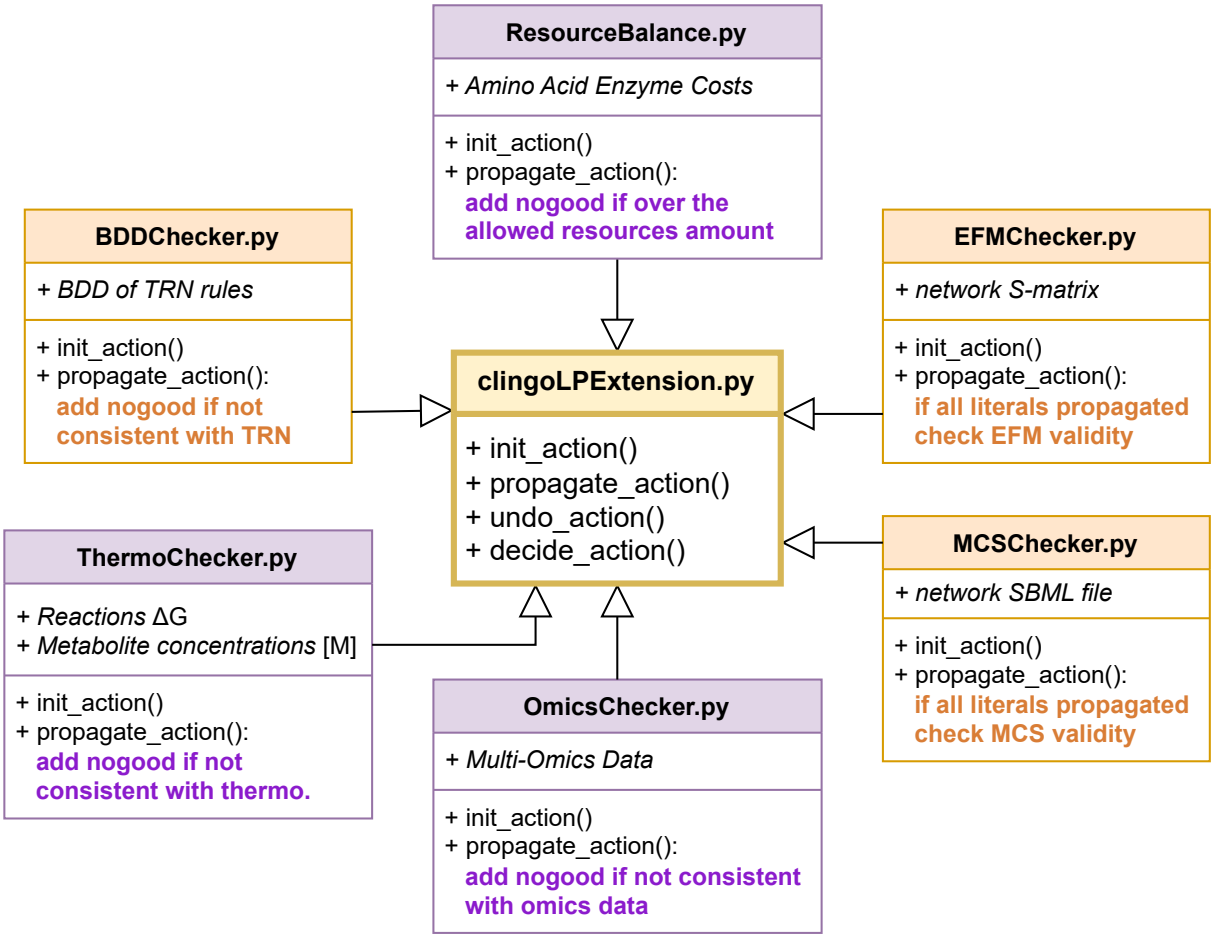


Figure 5.5: Examples of *aspefm* extensions. Uses UML representation of inheritance. Class attributes of extensions are also data required as command line parameters of *aspefm*.

5.4 Further extensions to *aspefm*

To extend our tool further, we envision the integration of thermodynamics data, omics-data, and resource allocation data. To begin, resource allocation data would help us make a three-dimensional Pareto space as described in [225], and therefore, using data from *E. coli* core, we could complete the analysis. Resource allocation data, obtainable from RBAPy [165], and usually expressed in count of nitrogen atoms or of amino acids for each enzyme, can be used to further constrain our model both before and during the computation. For example, implementing our ideas on a specific *aspefm* extension, we could perform a check on the support of partial solutions each time a literal is propagated. If from the data associated to the reactions – deduced from GPRs and RBAPy generated annotations – we can infer that producing the enzymes for all of these reactions is more costly that the total amount of resources that we decided our cell disposes with, then we should add a *nogood* on that solution, because all EFMs containing that partial solution would also be too expansive.

Alternately, the same could be done for omics-data, for example, transcriptomics data. If we have two or more enzymes, and according to transcriptomics the probability of these enzymes to be expressed at the same time is very low, then a *nogood* can be added on the support of the partial solution, because we believe that an EFM containing these two or more enzymes at the same time should not be experimentally observed. This can easily be applied to proteomics as well, since GPRs encompasses the gene and protein level both. Like with the BDD usage, these constraints can be applied dynamically: there is no need to compute beforehand every possible data check, since those can be done during propagation in function of the currently propagated literals.

Finally, following the ideas from tEFMA (thermodynamic EFM analysis) [275, 297], we would like to implement checks of thermodynamics, including computing free energies $\Delta\ G$ from metabolite concentrations $[M]$ (such as those obtained in metabolomics or fluxomics data). This idea was started to be implemented, however it was less clear if two reactions were thermodynamically inconsistent meant that any EFMs containing those two reactions was also inconsistent, so this was a hurdle. This should however be looked further.

While the need to solve a linear program at every propagated literal sounds very time-consuming, we think these new extensions ideas can give insights for further developing the enumeration of EFMs at the genome-scale, especially since they allow the integration of even more constraints. Since these ideas correspond to new kinds of constraints, and are still at the experimental stage, we represented them differently on Figure 5.5, in *purple*. In contrast, extensions that were finished, in *orange*, were distributed on GitHub: https://github.com/maxm4/aspefm.

5.5 Performances of *clingo[LP]*

We made several ameliorations to the code of *clingo[LP]*. The code of *clingo[LP]* is written in pure Python, and we found it to be impossible to parallellize on multiple threads. In particular, one function caused a hugely significant loss of time, due to being encoded as a recursive Python function. This appeared to us when we were forced to update the maximum recursion limit when dealing with genome-scale models. The concerned function was the subset-minimalization process for finding 'core conflicts', or Irreducibly Inconsistent Sets (IISs) [S148] in standard

Linear Programming [W20]. Instead, we replaced the 'core_conflict' function by the `'cplex.get_confl()'` function from the CPLEX API, invoking a special tool from IBM CPLEX © called the 'conflict refiner' [W8]. This is substantially faster and allowed us to improve the computation time greatly.

Previously, we mentioned that extensions are called at each literal propagation. In fact, this is not exactly true. Since we are working with a problem of subset-minimality, propagation of negative literals is supposed the default, while propagation of positive literals are the major event. In the previous *clingo[LP]* code, one would run linear programs even when only negative literals were propagated. By adding a simple check of having at least one positive propagated literal in *propagate* before running *solve*, we gained another significant time factor for the computation of EFMs. As well, we could call *propagate_action* of our extensions when positive literals are propagated only. Since the literals are watched, everything should automatically update when *undo* events occur. Note however, that this modification should only be valid because in *aspefm* we are computing subset-minimal solutions.

However, we are neither *clingo[LP]* nor SAT heuristics experts. We believe a more thourough recoding of *clingo[LP]* might be required in order to revisit whether all parts of the code are pertinent, and for us to fully tailor the code to *aspefm*. To do so, we think contacting the ASP experts from Potsdam University and collaborating with them will be the next step towards a truly clean code, validated by specialists. As well, we believe a full rewrite of the code in Cython as well as a better handling of the program in regards to the Python Global Interpreter Lock (GIL) will help one to parallelize the code. Indeed, *clingo* natively supports threads, but when using multiple threads *clingo[LP]* cannot run properly. This should also be explored with people with knowledge in parallel computing, and in linear programming. We refer to subsection A.6.7 for comments about how performance improved since our first article.

5.6 Conclusions

In summary, the problem of Elementary Flux Modes computation on genome-scale models is particularly difficult, necessitating the use of constrained enumeration methods, in order to integrate multi-omics data, and yield biologically relevant solutions upon request. EFMs have been known to correspond to optimal phenotypes observed in a cell, and the need for methods enumerating specific subset-minimal solutions will keep growing in the future.

We were able to express the computation of EFMs as a logic program with linear constraints, using Answer Set Programming, and Linear Programming solvers. It relies on enumerating minimal Boolean affectations to true, through having predefined heuristics biasing the literal propagation. We named our program *aspefm*, and we illustrated its capacity to integrate multivariate constraints on a well-known central metabolic network of *Escherichia coli*. We were able to reduce the number of EFMs of that network from 226 *million* to less than 1500, and further reduced the optimal phenotypic states to 5 EFMs, showing that making sense of these EFMs is possible.

aspefm is able to integrate any additional constraint, linear, or logical, to the computation of EFMs, and even other types of data can be integrated if proper extensions are coded through the high modularity of the ASP state-of-the art solver *clingo*. However, certain constraints modify the solution space and do not guarantee minimality of solutions anymore. These non-minimal solutions are directly filtered out during the computation thanks to particular extensions of our tool.

aspefm was extended to the computation of Minimal Cut Sets, minimal cuts of reaction flux in a metabolic network. Using genome-scale networks, MCSs are usually associated to essential reactions, and essential genes using Boolean rules of association between genes and enzymes. However, we argue that MCSs methods should rather be focused on the computation of larger-size MCSs, as we found those MCSs were able to reveal cross-feeding interactions between two pathogenic bacteria, *Staphylococcus aureus* and *Pseudomonas aeruginosa*. In particular, our method handles large-size solutions particularly well.

Through a sufficiently constrained enumeration, one can correlate EFMs solutions to flux distributions, using methods such as FBA flux decomposition or simple linear regression analysis to exometabolomics data. We illustrated such a constrained enumeration on a central metabolic network of the human cell, extended for incorporation of tumoral stroma, and we attempted to hypothesize therapeutic applications to cancer.

In conclusion, we believe *aspefm* constitutes a promising leading Logic Programming with Linear Constraints tool for enumeration of subsets of EFMs and MCSs, especially as it belongs among the first to achieve constrained enumeration of EFMs and MCSs on genome-scale models. Through further amelioration of the tool's performance, and developement of an user-friendly interface, we believe our tool might become a standard in the computation of minimal fluxes and cuts for the analysis of metabolic pathways in systems biology.

Appendix A

Addititional methods and results

A.1 Example of metabolic network

```
-METEXT
S biomass fuel1 fuel2

-CAT
R01 : S + ATP => M1 + ADP
R02 : M1 = M2
R03 : M2 + ADP => M3 + ATP
R04 : M3 = M4
R05 : M4 = M5
R06 : M3 => M6
R07 : M6 => M7
R08 : M7 => 0.5 M5 + 0.5 M8
R09 : M8 => fuel1
R10 : M1 = M9
R11 : M9 => M2
R12 : M9 => M10
R13 : M9 + ADP => M11 + ATP
R14 : M11 => 0.7 M12 + 0.3 M13
R15 : M13 => fuel2
R16 : M12 => M5
R17 : M5 => biomass
R18 : M3 => 2.0 M8
```

Listing A.1: Example of toy model written in METATOOL format

A.2 State-of-the art algorithms

Function: FBA($m, obj\ [default : biomass]$)

Data: Metabolic model m, Objective obj

Result: Objective value sol

Solution reaction fluxes v

Type: Linear Programming in iterative code

$S \leftarrow m.Stoch$; `// get S-matrix`

$(lb, ub) \leftarrow m.Bounds$; `// bounds of` $m.\mathcal{R}eac$

find $sol =$ ***max*** $obj \times v$ **such that**

- $Sv = 0$;
- $lb_j \leq v_j \leq ub_j$;
- $v \in \mathbb{R}^n$;

return sol, v;

Algorithm A.1: Flux Balance Analysis

Data: Metabolic model m, Objective obj

Result: Objective values $o1$, $o2$, Solution reaction fluxes v

Get k, S, lb, ub from $Irrev(m)$, *i.e.* m with reversibles split into two irrev. reactions;

find $o1 =$ ***min*** $\sum_{j=1}^{k} v_j$ **such that**

- **find** $o2 =$ ***max*** $obj \times v$ **such that**
  - $Sv = 0$;
  - $0 \leq v_j \leq ub_j$;
  - $v \in \mathbb{R}^k$;
- $v_{obj} = o2$;

return $o1, o2, v$;

Algorithm A.2: Parsimonious FBA

Function: FVA($m, obj\ [biomass], \mu\ [90\%]$)

Data: Metabolic model m, Objective obj

Result: Minimal and maximal fluxes v_{min}, v_{max}

Get n, S, lb, ub from m;

$v_{min} \leftarrow \mathbb{0}^n$; $v_{max} \leftarrow \mathbb{0}^n$;

$(sol, v) \leftarrow$ FBA(m, obj) ;

for $r \in m.Reac$ **do**

- `// for each FBA reaction flux`
- **find** $v_{max}(r) =$ ***max*** v_r ***and*** $v_{min}(r) =$ ***min*** v_r
  - $Sv = 0$;
  - $obj \times v \geq \mu \times sol$;
  - $lb_j \leq v_j \leq ub_j$;
  - $v \in \mathbb{R}^n$;

return v_{min}, v_{max};

Algorithm A.3: Flux Variability Analysis

Data: Metabolic model m, Parameters p

Result: Time simulation t of external metabolite concentrations $[X]$, Vectors sol, v

for t *increasing* ***from*** 0 ***to*** $p.MaxTime$ **do**

- $[X]_t$ = ODE($[Xt]_{t-1}$); `// or` $p.[X]_0$ `if` t_0
- updateExc($m.Exchanges.Bounds$, $[X]_t$) ;
- **if** *regulated* **then**
  - R_t = TRN(R_{t-1}); `// or` $p.R_0$ `if` t_0
  - updateRgx($m.RegulatedRx.Bounds$, R_t);
- $(sol_t, v_t) \leftarrow$ FBA(m) ;

Algorithm A.4: (Regulated) dynamic FBA

Data: Matrix A, defining a polyhedral cone $C = \{x \in \mathbb{R}^d \mid Ax = 0, x \geq 0\}$

Result: Matrix R, extreme rays of $P = \{x \in \mathbb{R}^d \mid x = Rc, c \geq 0\}$

$R \leftarrow K$; `// row-echelon kernel matrix,` $N(A) = \{x \mid Ax = 0\}$

$\rho \leftarrow \{\}$; `// indices of already considered non-negativity constraints`

while $\rho \neq \{1, 2, \cdots, d\}$ **do**

- $j \leftarrow$ choose from $\{1, 2, \cdots, d\}$ if not in ρ;
- $\tau_> \leftarrow \{i \mid R_{j,i} > 0\}$;
- $\tau_0 \leftarrow \{h \mid R_{j,h} = 0\}$;
- $\tau_< \leftarrow \{k \mid R_{j,k} < 0\}$;
- $\tau_{adj} \leftarrow \{(i,k) \mid (i,k) \in (\tau_> \times \tau_<) : R_{*i}$ is adjacent to $R_{*k}\}$;
- $R_{new} \leftarrow []$;
- **foreach** $(i,k) \in \tau_{adj}$ **do**
  - $p \leftarrow R_{*i}$;
  - $q \leftarrow R_{*k}$;
  - $r_{(ik)} \leftarrow p_j\ q - q_j\ p$;
  - append column $r_{(ik)}$ to R_{new};
- $R \leftarrow [[R_{*i}], [R_{*h}], R_{new}]$ with $i \in \tau_>$ and $h \in \tau_0$;
- $\rho \leftarrow \rho \cup \{j\}$;

Algorithm A.5: Double Description

Function: knock_out(m, r)

Data: Metabolic model m, reaction r

Result: Metabolic model m

$rxn \leftarrow$ get reaction r in m ;

if rxn *is reversible* **then**

    // unset lower and upper bound

    $rxn.lb \leftarrow 0$;

    $rxn.ub \leftarrow 0$;

else

    $rxn.ub \leftarrow 0$;

update reaction rxn in m ;

return m;

Algorithm A.6: Reaction knock-out

Function: essential_reactions(m)

Data: Metabolic model m

Result: List of essential reactions sl

$sl \leftarrow \{\}$;

$eps \leftarrow 10^{-5}$;

for $r \in m.\mathcal{R}eac$ **do**

    $mtmp \leftarrow m$;

    $mtmp \leftarrow$ knock_out$(mtmp, r)$;

    $sol \leftarrow$ FBA$(mtmp)$;

    if $sol < eps$ **then**

        $sl \leftarrow sl + \{r\}$;

return sl;

Algorithm A.7: Essential reactions

Data: Metabolic model m

Result: List of synthetic lethal pairs sl

$sl \leftarrow \{\}$;

$eps \leftarrow 10^{-5}$;

for $r1 \in m.\mathcal{R}eac$ **do**

    for $r2 \in m.\mathcal{R}eac - r1$ **do**

        $mtmp \leftarrow m$;

        $mtmp \leftarrow$ knock_out$(mtmp, r1)$;

        $mtmp \leftarrow$ knock_out$(mtmp, r2)$;

        $sol \leftarrow$ FBA$(mtmp)$;

        if $sol < eps$ **then**

            $sl \leftarrow sl + \{r1, r2\}$;

    return sl;

Algorithm A.8: Synthetic lethal pairs

Data: Metabolic model m

Result: List of synthetic lethal triplets sl

$sl \leftarrow \{\}$; $eps \leftarrow 10^{-5}$;

for $r1 \in m.\mathcal{R}eac$ **do**

    for $r2 \in m.\mathcal{R}eac - r1$ **do**

        for $r3 \in m.\mathcal{R}eac - \{r1, r2\}$ **do**

            $mtmp \leftarrow m$;

            $mtmp \leftarrow$ knock_out$(mtmp, r1)$;

            $mtmp \leftarrow$ knock_out$(mtmp, r2)$;

            $mtmp \leftarrow$ knock_out$(mtmp, r3)$;

            $sol \leftarrow$ FBA$(mtmp)$;

            if $sol < eps$ **then**

                $sl \leftarrow sl + \{r1, r2, r3\}$;

return sl;

Algorithm A.9: Synthetic lethal triplets

A.3 *aspefm* and application to toy models

```
% Borne supérieure pour les valeurs de flux
#const nb=5000.

% Toutes les réactions sont irréversibles: elles ont un flux positif ou nul
&dom{0..nb} = flux(R) :- reaction(R).

% Deux réactions irréversibles issues de la séparation d'une réaction
% réversible sont mutuellement exclusives
:- support(R1); support(R2); 1 {reversible(R1, R2); reversible(R2, R1)} 1;
   reaction(R1); reaction(R2).

% Au moins une réaction doit être utilisée
:- not support(R) : reaction(R).

% Pour chaque métabolite, la somme pondérée par la stoechiométrie
% des flux de chaque réaction le consommant ou le produisant est nulle
&sum{C*flux(R) : stoichiometry(M, R, C), reaction(R)} = 0 :- metabolite(M).

% Définition du support, atomes dont la présence indiquent un flux non nul
support(R) :- &sum{flux(R)} > 0; reaction(R).

% Heuristique sur les atomes support pour la minimisation
#heuristic support(R). [1, false]
```

Listing A.2: *aspefm*.lp4 main ASP program code for computation of EFMs

```
reaction(r1; r1_rev; t1; t2; t3; t4).
reversible(r1, r1_rev).
metabolite(a;b).

% A -> B
stoichiometry(a,r1,-1).
stoichiometry(b,r1,1).

% B -> A
stoichiometry(a,r1_rev,1).
stoichiometry(b,r1_rev,-1).

% -> A
stoichiometry(a,t1,1).

% -> B
stoichiometry(b,t2,1).

% B ->
stoichiometry(b,t4,-1).

% A ->
stoichiometry(a,t3,-1).
```

Listing A.3: ASP program code for encoding the toy metabolic model

```
clingo version 5.6.2
Reading from clingoLP.py ...
Solving...

LP Solver output
(0.0, {'flux(t2)': 0.0, 'flux(t4)': 0.0, 'flux(r1)': 0.0, 'flux(t3)': 0.01, 'flux(t1)': 0.01, 'flux(r1_rev)': 0.0})
Answer: 1
support(t1) support(t3)

LP solver calls: 6   Time cplex :  0.0036232471466064453

LP Solver output
(0.0, {'flux(t2)': 0.01, 'flux(t4)': 0.01, 'flux(r1)': 0.0, 'flux(t3)': 0.0, 'flux(t1)': 0.0, 'flux(r1_rev)': 0.0})
Answer: 2
support(t2) support(t4)

LP solver calls: 8   Time cplex :  0.004602670669555664


LP Solver output
(0.0, {'flux(t2)': 0.01, 'flux(t4)': 0.0, 'flux(r1)': 0.0, 'flux(t3)': 0.01, 'flux(t1)': 0.0, 'flux(r1_rev)': 0.01})
Answer: 3
support(r1_rev) support(t2) support(t3)

LP solver calls: 11   Time cplex :  0.006068229675292969


LP Solver output
(0.0, {'flux(t2)': 0.0, 'flux(t4)': 0.01, 'flux(r1)': 0.01, 'flux(t3)': 0.0, 'flux(t1)': 0.01, 'flux(r1_rev)': 0.0})

Answer: 4
support(r1) support(t1) support(t4)
SATISFIABLE

LP solver calls: 15   Time cplex :  0.007972478866577148

Models       : 4
Calls        : 1
Time         : 0.468s (Solving: 0.13s 1st Model: 0.05s Unsat: 0.00s)
CPU Time     : 1.299s
```

Listing A.4: ASP *clingo*[LP] output for the toy metabolic model

```
% Valeurs des constantes d'équilibre apparentes
keq(t2, 1). keq(t3, -1).
keq(r1, 1). keq(r1_rev, -1).
keq(t1, 1). keq(t4, 1).
% Règle thermodynamique
&sum{K*flux(R) : reaction(R), keq(R, K)} > 0.
```

Listing A.5: Illustration of thermodynamics constraints encoding in ASP

A.4 *aspefm* and application on CSP2001

```
%%% Note: Ce fichier doit être lancé avec covert_palsson_constr.lp4
%%% Commenter et décommenter les lignes de ce fichier avec "%"
%% Final additional constraints, altering EFM enumeration
%% Exclude inconsistent elementary modes
:- inconsistent.
%% Environment specification
env("Carbon1").
env("Carbon2").
env("Oxygen").
env("H").
env("F").
```

Listing A.6: Environment settings for growth medium of Covert and Palsson, 2003

```
%Computation of constraints without altering the EFM enumeration
%Constraints from Covert and Palsson, 2003
%Regulation proteins
regprotein("RPo2") :- not env("Oxygen"). %RPo2 IF NOT Oxygen
regprotein("RPc1") :- env("Carbon1"). %RPc1 IF Carbon1
regprotein("RPh") :- support("Th"). %RPh IF Th
regprotein("RPb") :- support("R2_rev"). %RPb IF R2_Rev
%Reaction regulation rules
repressed("R2") :- regprotein("RPb"). %R2 IF NOT RPb
repressed("R7") :- regprotein("RPb"). %R7 IF NOT RPb
repressed("R8") :- regprotein("RPh"). %R8 IF NOT RPh
repressed("R5b") :- not regprotein("RPo2"). %R5b IF RPO2
repressed("R5a") :- regprotein("RPo2"). %R5a IF NOT RPO2
repressed("Rres") :- regprotein("RPo2"). %Rres IF NOT RPO2
repressed("Tc2") :- regprotein("RPc1"). %Tc2 IF NOT RPc1
%envs specification
:- support("Tc1"); not env("Carbon1").  % SUP IF ENV
:- support("Tc2"); not env("Carbon2").
:- support("Th"); not env("H").
:- support("Tf"); not env("F").
:- support("To2"); not env("Oxygen").
%Remove CWA for envs
environment("Carbon1"; "Carbon2"; "H"; "F"; "Oxygen").
env(E) :- env(E); environment(E).
not env(E) :- not env(E); environment(E).
%Consistency
enzyme("R5a"; "R5b"; "R7"; "R8"; "Rres"; "Tc2").
inconsistent :- repressed(E); support(E); enzyme(E).
%Clingo show instructions
#show env/1.
#show regprotein/1.
#show repressed/1.
#show inconsistent/0.
```

Listing A.7: Transcription Regulation Network from Covert and Palsson, 2003

```
% External metabolites
external("Eext";"Dext";"Carbon2";"Hext";"Carbon1";"Biomass";"Fext";"Oxygen").
% Internal metabolites
metabolite("A";"ATP";"B";"NADH";"C";"F";"G";"D";"E";"H";"O2").
% Reversible reactions
reversible("R8","R8_rev";"R2","R2_rev").
% All reactions
reaction("R1";"R2";"R2_rev";"R3";"R4";"R5a";"R5b";"R6";
"R7";"R8";"R8_rev";"Rres";"Tc1";"Tc2";"Tf";"Td";"Te";"Th";"To2";"Growth").
% Stoichiometry
stoichiometry("A","R1",-1). stoichiometry("ATP","R1",-1).
stoichiometry("B","R1",1). stoichiometry("B","R2",-1).
stoichiometry("ATP","R2",2). stoichiometry("NADH","R2",2).
stoichiometry("C","R2",1). stoichiometry("B","R3",-1).
stoichiometry("F","R3",1). stoichiometry("C","R4",-1).
stoichiometry("G","R4",1). stoichiometry("G","R5a",-1).
stoichiometry("C","R5a","0.8"). stoichiometry("NADH","R5a",2).
stoichiometry("G","R5b",-1). stoichiometry("C","R5b","0.8").
stoichiometry("NADH","R5b",2). stoichiometry("C","R6",-1).
stoichiometry("ATP","R6",2). stoichiometry("D","R6",3).
stoichiometry("C","R7",-1). stoichiometry("NADH","R7",-4).
stoichiometry("E","R7",3). stoichiometry("G","R8",-1).
stoichiometry("ATP","R8",-1). stoichiometry("NADH","R8",-2).
stoichiometry("H","R8",1). stoichiometry("NADH","Rres",-1).
stoichiometry("O2","Rres",-1). stoichiometry("ATP","Rres",1).
stoichiometry("Carbon1","Tc1",-1). stoichiometry("A","Tc1",1).
stoichiometry("Carbon2","Tc2",-1). stoichiometry("A","Tc2",1).
stoichiometry("Fext","Tf",-1). stoichiometry("F","Tf",1).
stoichiometry("D","Td",-1). stoichiometry("Dext","Td",1).
stoichiometry("E","Te",-1). stoichiometry("Eext","Te",1).
stoichiometry("Hext","Th",-1). stoichiometry("H","Th",1).
stoichiometry("Oxygen","To2",-1). stoichiometry("O2","To2",1).
stoichiometry("C","Growth",-1). stoichiometry("F","Growth",-1).
stoichiometry("H","Growth",-1). stoichiometry("ATP","Growth",-10).
stoichiometry("Biomass","Growth",1).

% Reversible reactions stoichiometry
stoichiometry("B","R2_rev",1).
stoichiometry("ATP","R2_rev",-2).
stoichiometry("NADH","R2_rev",-2).
stoichiometry("C","R2_rev",-1).
stoichiometry("G","R8_rev",1).
stoichiometry("ATP","R8_rev",1).
stoichiometry("NADH","R8_rev",2).
stoichiometry("H","R8_rev",-1).
```

Listing A.8: Metabolic network of Covert and Palsson, 2003

A.5 Stoichiometric consistency

```
% Adaptation and translation in ASP of the MILP from Gevorgyan, 2008 at section 3.2
% Computing a subset minimal answer set corresponding to all unconserved metabolites

% Domain upper bound for mass values
#const nb=5000.

% Tells clingo that it is okay if the input file does not contain these predicates
reversible(do_not_use, do_not_use).

% External metabolites are metabolites too in this formalism
metabolite(M) :- external(M).

% Declare metabolite masses m_i
&dom{0..nb} = mass(M) :- metabolite(M).

% Stoichiometry matrix transposed multiplied by metabolite masses must equal 0
&sum{C*mass(M) : stoichiometry(M, R, C), metabolite(M)} = 0 :- reaction(R); not reversible(_, R).

% Compute conserved metabolites
conserved(M) :- &sum{mass(M)} > 0; metabolite(M).

% Compute unconserved metabolites
unconserved(M) :- not conserved(M); metabolite(M).

% Metabolites are assumed conserved
#heuristic unconserved(M). [1, false]

#show unconserved/1.
```

Listing A.9: Compute minimal unconserved metabolites

```
Model analyzed...
At least one error found.

We detected a mass imbalance
:  -> B

from the following reaction isolation set.

1. R2: C -> A + 2.00 B
2. R1: A + B -> C

------------------------------------------------------------------------
------------------------------------------------------------------------

These uni-uni reactions created mass-equivalence.
(The chemical species within a curly bracket have the same atomic mass.)

{C=D} is inferred by:
3. R3: C -> D

------------------------------------------------------------------------

Based on the uni-uni reactions above, we create mass-equivalent pseudo reactions.

(pseudo 1.) R2: {C=D} -> 2.00 {B} + {A}
(pseudo 2.) R1: {B} + {A} -> {C=D}

------------------------------------------------------------------------

An operation between the pseudo reactions:
1.00 * R2 + 1.00 * R1

will result in empty reactant with zero mass:

:  -> {B}
```

Listing A.10: Stoichiometric inconsistency of the example in equation 2.21

```
%Adaptation and translation in ASP of the MILP from Gevorgyan, 2008 at section 3.3
%Takes a choice of unconserved metabolites {unconserved(M) : unconsmet(M)} in additional input
%Takes a stoichiometric kernel of non zero coefficients leftkernel(M, R, C) and columns column(R)

% Domain upper bound for mass values
#const nb=5000.

% External metabolites are metabolites too in this formalism
metabolite(M) :- external(M).

% Declare metabolite masses m_i
&dom{0..nb} = mass(M) :- metabolite(M).

% Stoichiometry matrix transposed multiplied by metabolite masses must equal 0
&sum{C*mass(M) : leftkernel(M, R, C), metabolite(M)} = 0 :- column(R).

% Compute faulty stoichiometry for unconserved metabolites
min_stoch(M) :- &sum{mass(M)} > 0; metabolite(M).

% Unconserved metabolites should be in min_stoichiometries
:- not min_stoch(M), unconserved(M).

% Metabolites are assumed conserved
#heuristic min_stoch(M). [1, false]

#show min_stoch/1.
```

Listing A.11: Compute minimal net stoichiometries

```
We detected a mass imbalance from the following reaction isolation set.

1. R_TIVDG: M_TIVc + 0.70 M_AKGc + 1.33 M_NADc + 0.70 M_Q -> 0.70 M_GLUTc + 1.33 M_NADHc + 0.70 M_QH2
2. R_PL1: 8.00 M_ACoAc + 7.00 M_ATPc + 14.00 M_NADPHc + 7.00 M_HCO3 -> M_Palmitate_c
        + 7.00 M_ADPc + 7.00 M_Pic + 14.00 M_NADPc + 8.00 M_CoAc + 7.00 M_CO2
3. R_NAGS_ACY: 2.00 M_GLUTm + M_ACoAm + M_NADHm + 3.00 M_ATPm -> M_ORNm + M_CoAm + M_NADm + 3.00 M_ADPm + 3.00 M_Pim + M_AKGm + M_ACETm
4. R_ASYNT: 3.00 M_ADPm + 3.00 M_Pim + 8.00 M_DPH + 8.00 M_DPSI -> 3.00 M_ATPm
5. R_CPS1_OTC: M_NH3 + M_HCO3 + M_ORNm + 2.00 M_ATPm -> M_CITRm + 2.00 M_ADPm + 2.00 M_Pim
6. R_ORNT2: M_ORNc -> M_ORNm + M_DPH
7. R_RC2: M_SUCCm + M_Q -> M_MALm + M_QH2
8. R_PP1: M_G6P + 2.00 M_NADPc -> M_R5P + 2.00 M_NADPHc + M_CO2
9. R_RC1: M_NADHm + M_Q -> M_NADm + M_QH2 + 4.00 M_DPH + 4.00 M_DPSI
10. R_ACS: M_ACETm + 2.00 M_ATPm + M_CoAm -> M_ACoAm + 2.00 M_ADPm + 2.00 M_Pim
11. R_NUC: M_R5P + 2.40 M_GLNc + M_ASPc + 0.50 M_GLYc + M_FTHFc + 8.70 M_ATPc + 0.20 M_NADc + 0.75 M_NADPHc + 0.50 M_Q
        -> M_XTPc + 2.40 M_GLUTc + M_MALc + M_THFc + 8.70 M_ADPc + 8.70 M_Pic + 0.20 M_NADHc + 0.75 M_NADPc + 0.50 M_QH2
12. R_ANT: M_ATPm + M_ADPc + M_DPSI -> M_ATPc + M_ADPm
13. R_GOT2: M_GLUTm + M_OAAm -> M_ASPm + M_AKGm
14. R_MDH1: M_MALc + M_NADc -> M_OAAc + M_NADHc
15. R_PDH: M_PYRm + M_NADm -> M_ACoAm + M_NADHm + M_CO2
16. R_ASS1_ASL_FH: M_CITRc + M_ASPc + 2.00 M_ATPc -> M_ARGc + M_MALc + 2.00 M_ADPc + 2.00 M_Pic
17. R_CL: M_CITc + M_ATPc + M_CoAc -> M_ACoAc + M_OAAc + M_ADPc + M_Pic
18. R_IDH2: M_CITm + M_NADPm -> M_AKGm + M_NADPHm + M_CO2
19. R_PEPCK2: M_OAAm + M_ATPm -> M_PEPm + M_ADPm + M_CO2
20. R_IDH3: M_CITm + M_NADm -> M_AKGm + M_NADHm + M_CO2
21. R_ME2: M_MALm + M_NADm -> M_PYRm + M_NADHm + M_CO2
22. R_CS: M_ACoAm + M_OAAm -> M_CITm
23. R_MDH2: M_MALm + M_NADm -> M_OAAm + M_NADHm
24. R_PYC: M_PYRm + M_HCO3 + M_ATPm -> M_OAAm + M_Pim + M_ADPm
25. R_MTHFD2: M_MTHFm + M_NADm -> M_FTHFm + M_NADHm
26. R_PP2: 3.00 M_R5P -> 2.00 M_G6P + M_G3P
27. R_SERSYNT: M_3PG + M_GLUTc + M_NADc -> M_SERc + M_AKGc + M_NADHc + M_Pic
28. R_T2: M_AKGc + M_MALm -> M_AKGm + M_MALc
29. R_NIG: M_DPSI -> 4.00 M_DPH
30. R_G1: M_GLUCc + M_ATPc -> M_G6P + M_ADPc
31. R_GLUD1: M_GLUTm + M_NADm -> M_AKGm + M_NADHm + M_NH3
32. R_GG3: 2.00 M_G3P -> M_G6P + M_Pic
33. R_PEPCK1: M_OAAc + M_ATPc -> M_PEPc + M_ADPc + M_CO2
34. R_T4: M_GLUTc + M_ASPm + M_DPH + M_DPSI -> M_GLUTm + M_ASPc
35. R_T3: M_MALm + M_Pic -> M_MALc + M_Pim
36. R_SLP: M_AKGm + M_NADm + M_Pim + M_ADPm -> M_SUCCm + M_NADHm + M_CO2 + M_ATPm
37. R_ALDH1L2: M_FTHFm + M_NADPm -> M_THFm + M_NADPHm + M_CO2
38. R_SHMT1: M_SERc + M_THFc -> M_GLYc + M_MTHFc
39. R_ALDH1L1: M_FTHFc + M_NADPc -> M_THFc + M_NADPHc + M_CO2
40. R_GLS1: M_GLNm -> M_GLUTm + M_NH3
41. R_ORNT1: M_ORNc + M_CITRm -> M_ORNm + M_CITRc + M_DPH
42. R_T5: M_Pic + M_DPH -> M_Pim
43. R_T7: M_PEPm + M_CITc + M_DPH + M_DPSI -> M_PEPc + M_CITm
44. R_ME1: M_MALc + M_NADPc -> M_PYRc + M_NADPHc + M_CO2
45. R_ARGASE: M_ARGc -> M_UREAc + M_ORNc
46. R_MTHFD1_1: M_MTHFc + M_NADPc -> M_FTHFc + M_NADPHc
47. R_SHMT2: M_SERm + M_THFm -> M_GLYm + M_MTHFm
48. R_G2: M_G6P + M_ATPc -> 2.00 M_G3P + M_ADPc
49. R_GS1: M_GLUTc + M_NH3 + M_ATPc -> M_GLNc + M_ADPc + M_Pic
50. R_T9: M_GLUTc + M_DPH -> M_GLUTm
51. R_T6: M_PYRc + M_DPH -> M_PYRm
52. R_MTHFD1_2: M_FTHFc + M_ADPc + M_Pic -> M_THFc + M_ATPc + M_FORc
53. R_PK: M_PEPc + M_ADPc -> M_PYRc + M_ATPc
54. R_T1: M_CITm + M_MALc -> M_CITc + M_MALm + M_DPH
55. R_NNT: M_NADHm + M_NADPm + M_DPH + M_DPSI -> M_NADm + M_NADPHm
56. R_GG4: M_G6P -> M_GLUCc + M_Pic
57. R_GOT1: M_GLUTc + M_OAAc -> M_ASPc + M_AKGc
58. R_MTHFD2L: M_MTHFm + M_NADPm -> M_FTHFm + M_NADPHm
59. R_ATPASE: M_ATPc -> M_ADPc + M_Pic
60. R_T18: M_ARGc -> M_ARG
61. R_T21: M_GLYc -> M_GLY
62. R_T22: M_GLYc -> M_GLYm
63. R_GLUCUP: M_GLUC -> M_GLUCc
```

Listing A.12: Stoichiometric inconsistency of the C2M2NF model by Jean-Pierre Mazat

A.6 *E. coli* core analysis

A.6.1 ASP Programs

Description of every ASP file provided in Supplementary File S4 of [89]:

- `solve[LP].lp4` *i.e.* `aspefm.lp4` : Program implementing the computation of EFMs under constraints. Works with any network and constraints encoded in ASP as presented in Section 3.4.1.
- `orth_ecoli_core.lp4` : ASP translation of the network, using the encoding established above.
- `orth_ecoli_core_atp.lp4` : ASP translation of the network with modified biomass.
- `ecoli_core_regul.lp4` : Full translation of the *E. coli* core transcriptional regulation network.
- `ecoli_core_additional_constraints.lp4` : Additional constraints for the *E. coli* core network, including environments, thermodynamic constraints and operating costs constraints.

In addition, we used the former standalone implementation of clingo[LP] as a Python script.

Here are the options we used to launch our tool:

```
clingo [LP] [Network] [Constraints] aspefm.lp4 -c nstrict=0
                                    --heuristic Domain --enum-mode domRec
                                    -c accuracy=10 -c epsilon="(1,1)"
```

A.6.2 Additional Python Code

In Supplementary File S5 we provide *Jupyter Notebooks* [S123] computing the Pareto optimal pathways with Escher and the plots presenting the EFMs sorted by biomass uptake rate as in figures/ 3.5 and 3.6, figures/ A.1 and A.2.

We also include Python *pickle* data structures containing the EFMs and MCFMs presented in Tables 3.2 and A.1 as *pandas* data frames. The notebook requires the use of Python modules *pandas*, *pickle*, *matplotlib*, *scipy* and *escher*.

A.6.3 Pareto Optimal Pathways of *E. coli*

The Pareto optimal pathways for the *E. coli* core model were visualized in HTML format with the tool Escher [191]. We represent a total of five EFMs for the network without formate regulation. The pathways for both regulation conditions are presented in a ZIP file attached in Supplementary Files.

A.6.4 *E. coli* Biomass Modifications

The *E. coli* core biomass coefficients were modified so that they included ATP maintenance requirements. The biomass now required 139 moles of ATP, for an amount of 42.55 C moles of biomass and an *E. coli* doubling time of 40 minutes. In particular, the experimentally-known elemental composition of *E. coli* was used (see 1.4.4).

Mass balance was performed using the master equations presented in subsection 1.4.4. We also modified accordingly the H^+ and H_2O coefficients in order for the number of electrons and hydrogen and oxygen moles in the biomass to be closer to typical *E. coli* values. The calculations are detailed in the Supplementary Excel file.

A.6.5 Pareto Optimal Pathways of *E. coli* with the Adjusted Biomass

Integrating the maintenance energy into the biomass reaction resulted in higher resource operating costs, as expected. To ensure relevant EFMs were identified, the operating cost bounds were increased to an O_2 operating cost less than 1.4 O_2 moles per biomass C mole and a glucose operating cost less than 14 C moles per biomass C mole. In addition, maintenance reaction *ATPM* from the model was disabled. The results are presented in Table A.1.

The modified biomass returned a different number of EFMs, resulting in a Pareto front of 5 EFMs for the network with standard regulation (Figure A.1), and 9 EFMs for the network without formate metabolism regulation (Figure A.2).

In addition, disabling the formate regulation resulted this time in only an ~ 4 fold increase in the number of EFMs. The chosen bounds and modifications to the model biomass have a strong impact on the bidimensional substrate operating cost space geometry.

Table A.1: Number of EFMs retrieved on the modified *E. coli* core network depending on culturing conditions for the adjusted biomass. Computation time given within brackets. Disabling the formate regulation returned EFMs for both aerobic and anaerobic conditions in a single execution.

		Standard Regulation	**No Formate Regulation**
Processing	Aerobic conditions	4273 EFMs [2362s]	16411 EFMs [8005s]
	Anaerobic conditions	930 EFMs [469s]	
Post-processing	Filtered out MCFMs	36 MCFMs	137 MCFMs
	Pareto optimal in biomass yield	5 EFMs	9 EFMs

The Pareto optimal pathways for the modified model were visualized in HTML format with the tool Escher. We represent a total of nine EFMs for the network with modified biomass and formate regulation. The pathways are presented in a ZIP file attached in Supplementary Files.

The nature of the Pareto optimal pathways are the same as for the original biomass reaction: no byproducts for the top left EFM, then as O_2 availability decreases, EFMs start producing acetate, acetate, and formate and, finally, acetate, formate, and ethanol under anaerobic conditions.

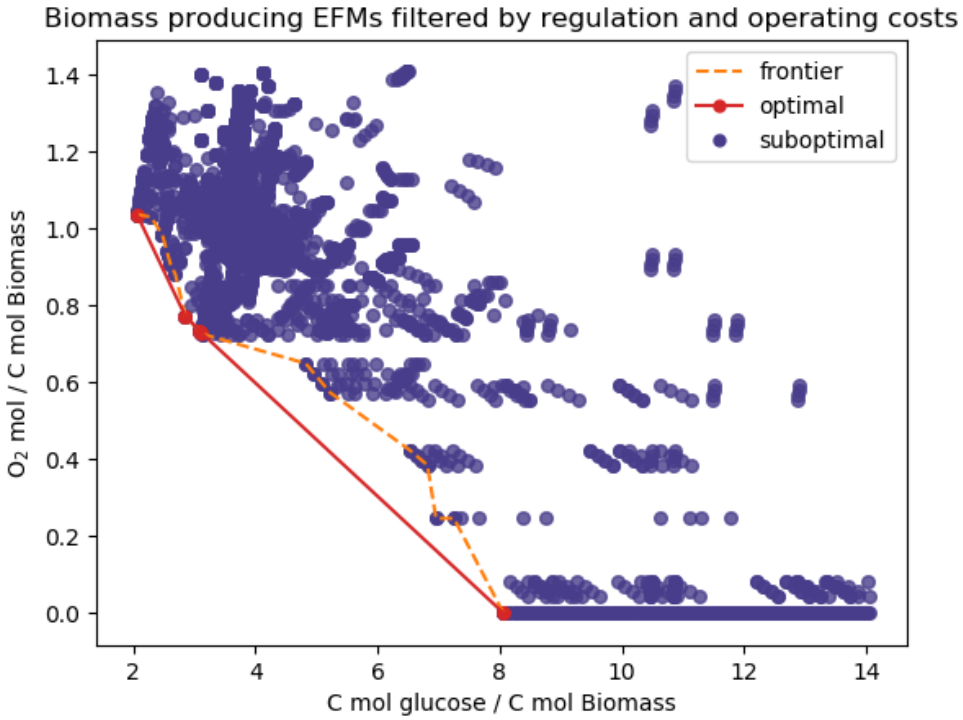


Figure A.1: *E. coli* core EFMs sorted by carbon/biomass and oxygen/biomass operating costs. Biomass was modified to include ATP maintenance. Regulation constraints are as described in Orth et al. 2010.

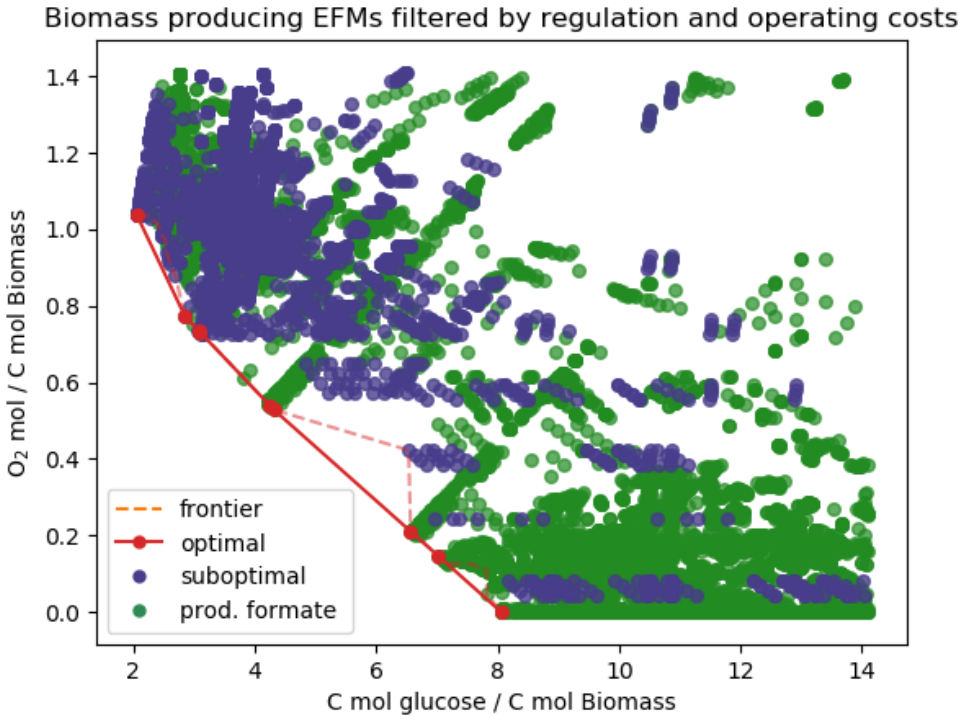


Figure A.2: *E. coli* core EFMs sorted by carbon/biomass and oxygen/biomass operating costs. Biomass was modified to include ATP maintenance. Regulation constraints allow production of formate in aerobic conditions.

A.6.6 Additional Results

Table A.2: Additional results observed for the original biomass. Computation time given within brackets.

Constraints	**Filtered out MCFMs**			**EFMs and MCFMs**		
With regulation and environment	O_2	No O_2	Formate	O_2	No O_2	Formate
No additional constraints	0	0	0	4027 [1314 s]	1459 [602 s]	28256 [5572 s]
Biomass-producing	0	0	0	2746 [833 s]	1355 [436 s]	24324 [6281 s]
Biomass-producing Thermodynamic data	0	0	0	2746 [901 s]	1355 [471 s]	24324 [6843 s]
Biomass-producing Yields ($O_2 < 0.7$) ($C < 7$)	39	0	119	1157 [560 s]	363 [220 s]	11136 [4884 s]
Biomass-producing Thermo and Yields	39	0	119	1157 [542 s]	363 [232 s]	11136 [5318 s]

Table A.3: Additional results observed for the revised biomass; BP: Biomass-Producing. Computation time given within brackets.

Constraints		**Filtered out MCFMs**			**EFMs and MCFMs**		
With regulation and environment		O_2	No O_2	Formate	O_2	No O_2	Formate
ATPM	No additional constraints	0	0	0	8354 [2518 s]	1260 [473 s]	33499 [6676 s]
	Biomass-producing	3	0	3	7076 [2939 s]	1156 [428 s]	29570 [8697 s]
No ATPM	No additional constraints	0	0	0	7735 [2337s]	1228 [428s]	32098 [6474s]
	Biomass-producing	3	0	3	6656 [2948 s]	1140 [441 s]	28795 [8664 s]
	BP Thermodynamic data	3	0	3	6656 [3027 s]	1140 [458 s]	28795 [8744 s]
	BP Yields ($O_2 < 1.4$) ($C < 14$)	36	0	137	4309 [2369 s]	930 [473 s]	16548 [7904 s]
	BP Thermo and yields	36	0	137	4309 [2362 s]	930 [469 s]	16548 [8005 s]

A.6.7 Comments on *E. coli* core Additional Results

The supplementary tables Table A.2 and Table A.3 are provided as they were written and published in the now 3-year old article [89]. Since then, major ameliorations have been implemented into *aspefm*, including the EFMChecker (section 5.2). The great news is with the latest version, computation times are now reduced by at least 5-fold.

Here are some examples of how computation times changed: the 4027 aerobic EFMs are now found in 178s; when adding biomass production we get to 2746 EFMs in 240s, and with yields we get 1118 EFMs in 98s. Keep in mind these results are *with* the EFMChecker extension. For anaerobic conditions, all 1459 EFMs are retrieved in 65s, all 1355 EFMs are retrieved in 124s, and all 363 EFMs are retrieved in 44s. The pattern we can see here is adding the positive constraint alone does *not* actually actively decrease the computation time. This is interesting to keep in mind for future works. Further, we have the 11017 formate-producing EFMs we plotted retrieved in 965s. On to the other table, we have 7735 aerobic EFMs in condition 'No ATPM' computed in 160s, with in addition the biomass-producing condition we have 6656 aerobic EFMs in 486s, and for the longest tested computation time we have 28795 formate and biomass-producing EFMs retrieved in 2623s.

Like the Supplementary Tables, I only give these new results for informative purposes. They do not constitute a legitimate attempt of mine at benchmarking, as it does only represent *single* executions, not even an average, and *clingo*'s execution patterns are *highly* non-deterministic. For example, thermodynamic equilibrium constraints almost always seem to increase computation times no matter how many runs are launched, which is why I thought they might be relevant in the Appendix tables, even if not filtering out any EFMs.

These two tables have always presented quite some oddities. Formate-producing EFMs not requiring to combine aerobic and anaerobic conditions is a result of a clunky encoding of some transcriptional and environmental regulation rules in the transcriptional regulation network, and if I were to redo it today I would try to avoid this as it brings confusion, even though it was convenient not having to combine different executions and conditions.

However a more striking oddity would be the presence of MCFMs that are not EFMs when only adding a single positive constraint: asking for biomass production, in Table A.3, conditions 'Regulation and environment' and 'No ATP'. This always confused me as MCFMs that are not EFMs should appear after forcing two positive constraints, not just one. And I thouroughly checked and curated the TRN so that it would not have any positive constraint. Having retested it with the latest version, I can confirm that these MCFMs are not obtained anymore, and this was most likely a problem with the maximum flux bounds of every reaction, which impose positive constraints. Indeed, I am able to reproduce the issue when setting bounds to $[0, 5000]$ with '-c nb = 5000', as was used back then.

In any case, all data used for the analysis are provided in the article's supplementary material so one should be able to reproduce and recalculate the number of EFMs in the now smaller computation times with the newest version of *aspefm* if interested. For further validation of this analysis, one should gather the exhaustive set of 226.3×10^6 EFMs and check how many EFMs of each condition there really are. This would be a supplementary step to confirm how accurate of a method *aspefm* is at finding the subsets of EFMs. We did not attempt this procedure because of the high memory requirements.

A.7 Tumoural stroma analysis

	minimum	maximum	median	mean	standard deviation
EX_GLY	-11.7760	13.7858	0.0000	0.0361	3.3644
EX_ARG	-5.0745	0.0000	-0.8145	-1.0733	0.9929
EX_ASP	-12.8867	0.0000	-1.6067	-2.2061	1.9422
EX_GLN	-12.0169	-0.1697	-1.7679	-2.4997	2.2157
EX_TIV	-8.7957	-0.7183	-1.6904	-2.2185	1.6127
EX_GLUT	0.0000	6.2446	0.0000	0.1707	0.6644
EX_ALA	0.0000	4.9655	0.0000	0.0640	0.4226
EX_SER	-14.7054	0.0000	-1.5427	-2.5081	3.2967
EX_YFLKW	-14.8778	-1.3146	-3.0614	-4.0092	2.7661
EX_PRO	-8.5352	0.8575	-1.2870	-1.6464	1.5906
EX_HIS	-7.7354	-0.1635	-0.5035	-1.5421	1.8491
EX_MET	-1.2754	4.8228	-0.2290	-0.2083	0.6016
EX_CYS	-2.8155	-0.2429	-0.5519	-0.7241	0.5206
EX_FOR	-8.9515	12.0801	0.0000	0.4891	2.4040
EX_PYR	0.0000	4.8270	0.0000	0.1721	0.6349
EX_XTP	-0.0035	1.4832	0.3022	0.2963	0.2810
EX_GLUC	-8.5815	-0.5000	-2.3908	-2.8593	1.8851
EX_LAC	0.0091	14.9990	3.9917	4.7118	3.4683
EX_Biomass	0.0100	0.0879	0.0100	0.0185	0.0144
EX_Stroma	0.0110	0.1276	0.0248	0.0323	0.0236
EX_Collagen	0.0110	0.3125	0.0367	0.0583	0.0575

Table A.4: Statistics (minimum, maximum, median, standard deviation) of the main exchange fluxes for all 747 Warburg effect EFMs, negative values indicates consumption, positive values indicates production

	Number of cell lines	Linear regression RMSE	Linear regression R^2
All cell lines	60	26.9102	0.9813
Colon	7	28.7879	0.9812
Leukemia	6	19.3316	0.9745
Lung	9	40.9490	0.9797
Prostate	2	33.1043	0.9789
Ovarian	7	25.9862	0.9818
Breast	6	31.3463	0.9803
Melanoma	9	23.2846	0.9818
Central Nervous System	6	19.9142	0.9789
Renal	8	28.8023	0.9770

Table A.5: Scores of linear regression fit to the mean flux values of different types of cancer cell lines in the NCI-60 cancer cell lines data, applied to the EFM with best regression fit to the global mean flux values of all 60 cell lines

A.8 *aspefm* for the analysis of MCSs of *P. aeruginosa* and *S. aureus*

```
% mcs[LP].lp4 - Von Kamp formulation
% Logic Program in Clingo[LP] - Clingcon ASP format
% Finds fluxes that belongs to the stoichiometric matrix kernel
% Defines an heuristic for them to be enumerated subset minimal
% Used to compute Minimal Cut Sets in metabolic networks

% Domain upper bound for flux values
#const nb=20000.

% Tells clingo that it is okay if the input file does not contain these predicates
reversible(do_not_use, do_not_use).

% Rule A: Since all reactions are irreversible, they must have a nonnegative flux
&dom{0..nb} = flux(R) :- reaction(R).

% Rule B: Two irreversible reactions issued from the splitting of
% one reversible reaction are mutually exclusive
:- support(R1); support(R2); 1 {reversible(R1, R2); reversible(R2, R1)} 1;
   reaction(R1); reaction(R2).

% Rule C : At least one reaction that is not the target reaction must be used
:- not support(R) : reaction(R).
:- not cutset(R) : reaction(R).

% Rule D : For each metabolite, the sum, weighted by the stoichiometry,
% of fluxes of all active reactions is null
&sum{C*flux(R) : stoichiometry(M, R, C), reaction(R)} >= 0 :- metabolite(M); mirrev(M).
&sum{C*flux(R) : stoichiometry(M, R, C), reaction(R)} = 0 :- metabolite(M); not mirrev(M).

% Compute support
support(R) :- &sum{flux(R)} > 0; reaction(R).

% Atoms in which we are interested in
cutset(R) :- support(R); interest(R).

% Support minimization heuristic
#heuristic cutset(R). [1, false]

% Show support atoms
#show cutset/1.
```

Listing A.13: *aspmcs*.lp4 main ASP program code for computation of MCSs

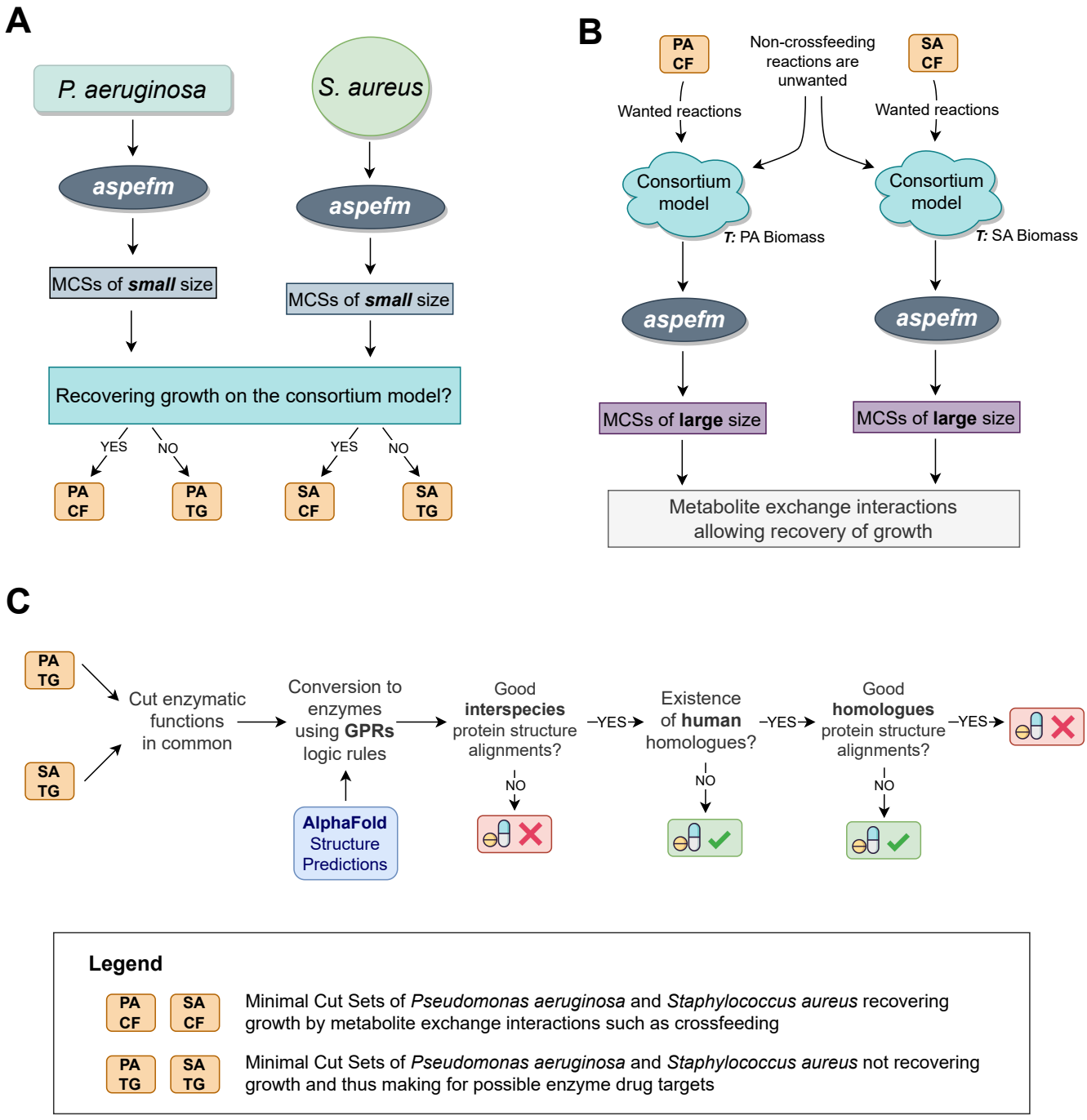


Figure A.3: Summary of the methods performed in the MCSs study using *aspefm*.
A: Computation of all MCSs of small size on the single-species models; **B**: Computation of subsets of MCSs of large size on the consortium model, explaining the metabolite exchanges allowing recovery of growth; **C**: Search for new therapeutic targets using protein structure alignments of AlphaFold predictions. Abbreviations: ***T***: target reaction, **PA**: *Pseudomonas aeruginosa*, **SA**: *Staphylococcus aureus*.

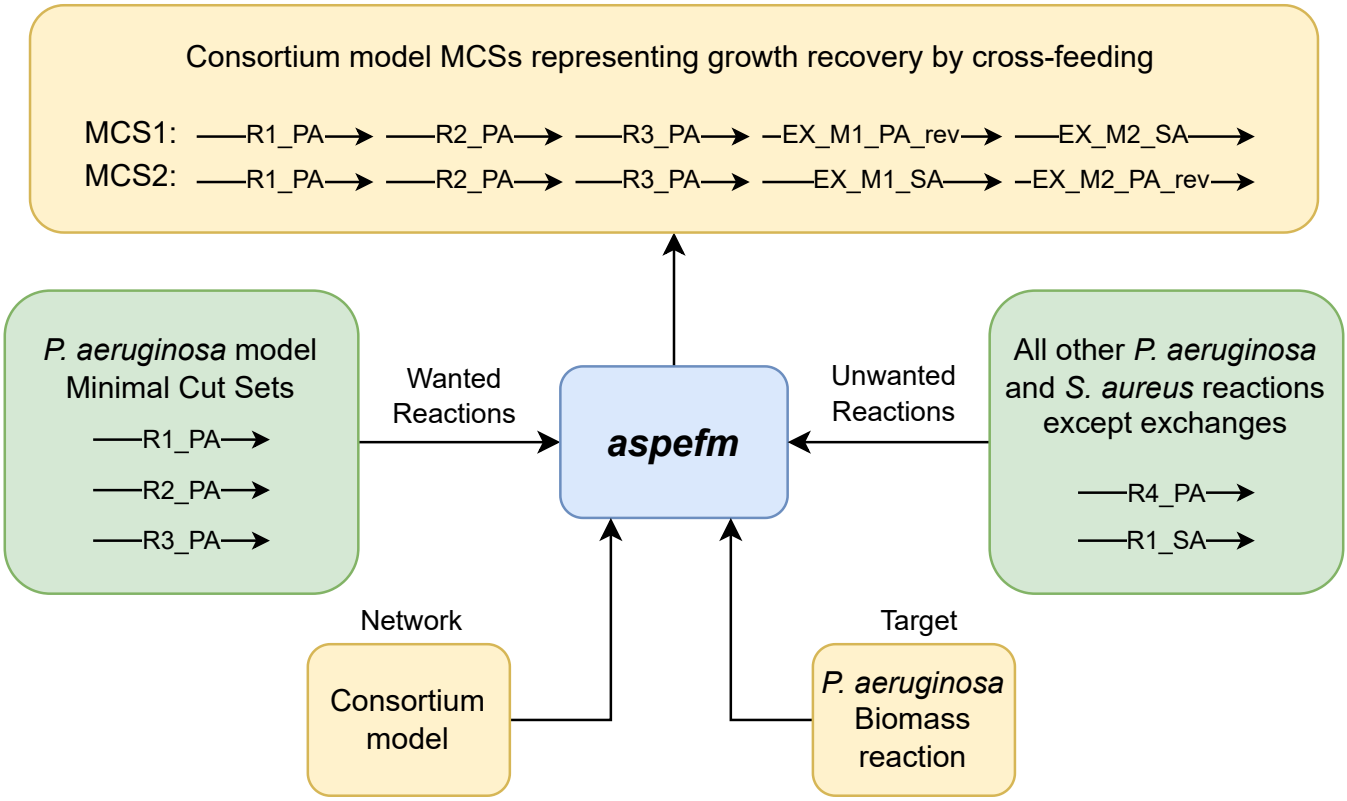


Figure A.4: Procedure for computing MCSs revealing metabolite exchanges with *aspefm*, from the sets of lethal MCSs of either bacteria with growth recovery on the consortium model (illustrated for *P. aeruginosa*).
Exchange reactions are denoted by "**EX**". Backwards direction of a reaction is denoted by "rev". MCSs with growth recovery on the consortium model are selected as wanted reactions constraints, while all other metabolic reactions except exchanges usable for crossfeeding are set as unwanted reactions. After running *aspefm*, output consortium model MCSs represent growth recovery by crossfeeding of external metabolites.
Displayed on the figure, crossfeeding of external metabolites M1 and M2 complement lethality of cutset {R1, R2, R3} of *P. aeruginosa*, thus the new cut sets obtained on the consortium model with *P. aeruginosa*'s biomass reaction as target have: {R1, R2, R3} as base, then either consumption of M1 by PA or production of M1 by SA needs to be cut, and then either consumption of M2 by PA or production of M2 by SA needs to be cut (two examples of such cut sets shown out of four).

Appendix B

French: Compilation de connaissances avec les Binary Decision Diagrams

B.1 Binary Decision Diagrams

Pour illustrer la compilation de connaissances, nous allons détailler une méthode pionnière, les Diagrammes de Décision Binaire ou BDD. Le principe de compilation de connaissances repose sur l'existence de langages de compilation cibles qui ont des propriétés particulières telles que la possibilité de compter et d'énumérer des solutions en temps polynômial [207].

Par exemple, un BDD est une représentation compacte des solutions d'une formule booléenne. Toute formule booléenne peut être représentée par un BDD. Il est alors possible de vérifier en temps linéaire sur le BDD si une affectation de valeurs aux variables booléennes est une solution de la formule booléenne. La complexité du problème est transférée à l'étape de compilation de la formule booléenne en BDD, qui peut prendre beaucoup de temps.

Soit $X = \{x_1, x_2, \ldots, x_n\}$ un ensemble de variables booléennes. Une interprétation $I \in \mathbb{B}^n$ de X est une affectation de chaque variable booléenne de X soit à la valeur Vrai ($\top$), soit à la valeur Faux ($\bot$).

Soit une formule booléenne $\phi : \mathbb{B}^n \mapsto \mathbb{B}$. ϕ associe une interprétation de X soit à Vrai, soit à Faux. On appelle solutions S de ϕ les interprétations de X qui sont vraies par la formule ϕ.

Un Diagramme de Décision Binaire pour une formule booléenne ϕ est un graphe orienté acyclique (DAG) tel que chaque interprétation possible I de X correspond à un chemin sur le graphe qui mène soit à la valeur Vrai, soit à la valeur Faux.

Il y a donc deux types de nœuds: les nœuds terminaux qui sont les valeurs Vrai et Faux et les nœuds non-terminaux qui correspondent aux variables booléennes.

Ainsi, chaque nœud possède un label : soit Vrai ou Faux pour les deux nœuds terminaux, soit une variable booléenne (on peut avoir plusieurs nœuds pour la même variable booléenne).

Un nœud non terminal possède toujours deux arcs sortants: l'arc `low` qui correspond au cas où la variable prend la valeur Faux, et l'arc `high` pour le cas où la variable prend la valeur Vrai.

On peut de plus définir un ordre sur les variables de X, par exemple généralement les variables sont numérotées de $1, 2, \ldots,$ à n.

Un Diagramme de Décision Binaire est dit Ordonné (OBDD) si à tout moment du graphe les variables respectent un ordre donné $\{x_1 < x_2 < \cdots < x_n\}$, c'est-à-dire que si $i < j$, alors un nœud étiqueté x_i ne peut être un descendant d'un nœud étiqueté x_j.

On décrit maintenant chaque nœud par un triplet $(var, low, high)$: var est le label du nœud, low est le label du nœud fils par l'arc sortant `low` et $high$ est le label du nœud fils par l'arc sortant `high`.

Un BDD est dit Réduit (ROBDD) [295] si:

- chaque nœud est dit unique : il n'existe pas deux nœuds distincts avec le même triplet $(var, low, high)$
- il n'existe pas de nœud $(var, low, high)$ tel que $low = high$ (on arrive à la même variable que var soit Vrai ou Faux, il s'agit d'un test redondant)

Exemple: soit ϕ_1 une formule booléenne sur $\{t1, t2, t3, t4\}$ telle que

$$\phi_1 = (t1 \Rightarrow \neg t2) \wedge (t3 \vee t4)$$

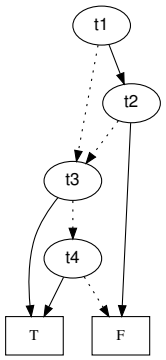


Figure B.1: ROBDD pour la formule $\phi_1 = (t1 \Rightarrow \neg t2) \wedge (t3 \vee t4)$

Un exemple de ROBDD pour la formule ϕ_1 est représenté en Figure B.1. Les arcs `low` sont représentés en pointillés tandis que les arcs `high` sont représentés en trait plein. Par exemple, pour l'interprétation $\{\bot, \bot, \top, \bot\}$, $t1$ est affecté à Faux donc on suit l'arc en pointillés, puis on affecte $t3$ à Vrai ce qui nous amène en suivant le trait plein au nœud Vrai. Il s'agit d'une solution. Notons que ici la valeur de $t2$ et $t4$ n'ont pas influé sur le résultat.

Pour l'interprétation $\{\bot, \top, \bot, \bot\}$, $t1$ est affecté à Faux, $t3$ est affecté à Faux, $t4$ est affecté à Faux donc on arrive au nœud Faux. Il ne s'agit pas d'une solution. Effectivement, on peut voir que cette solution ne respecte pas l'expression booléenne $(t3 \vee t4)$ dans la formule ϕ_1.

Communément, ce que l'on appelera BDD par la suite désignera en fait des diagrammes qui sont réduits et ordonnés (ROBDDs).

En suivant ce principe, nous allons donc chercher à compiler nos formules booléennes correspondant aux réseaux de régulation en BDDs.

Le calcul des modes de flux n'est pas pour l'instant considéré car les contraintes de stœchiométrie nécessaires sont numériques, or les BDDs ne nous permettent que de regarder les contraintes booléennes.

On cherche donc pour l'instant à tester si un ensemble de réactions actives, issu du calcul des modes de flux, respecte ou non les contraintes de nos réseaux de régulation.

En d'autres termes, nous allons compiler un BDD du réseau de régulation de telle sorte que si notre mode de flux est conforme à la régulation, son support correspondra à un chemin du BDD qui amène au nœud Vrai. Inversement, s'il n'est pas conforme, le chemin amènera au nœud Faux. La formule booléenne considérée sera la conjonction de toutes les règles du réseau.

B.1.1 Application

Il existe un certain nombre d'opérations possibles sur un BDD. Nous allons détailler celles qui vont nous intéresser: `COUNT` et `RESTRICT` [295].

- Supposons qu'on possède un BDD u pour une expression ϕ. L'opération `COUNT`(u) nous renvoie le nombre de solutions S de ϕ. Cette opération est réalisée en temps linéaire en la taille de u, autrement dit $O(|u|)$.
- Supposons qu'on possède un BDD u pour une expression ϕ, une variable $x \in X$ et une valeur de vérité $b \in \mathbb{B}$. L'opération `RESTRICT`(u, x, b) nous renvoie alors un BDD $u[x = b]$ pour la sous-expression $\phi[x = b]$ tel que la variable x a été affectée à la valeur b. Cette opération est réalisée en temps linéaire en la taille de u, autrement dit $O(|u|)$.

Appelons $\texttt{LET}(u, V, B)$ l'opération qui consiste à obtenir un BDD $u[V = B]$ tel que un ensemble de k variables $V \subset X$ ont été affectées à k valeurs de vérités $B \in \mathbb{B}^k$. Cette opération consiste à effectuer k fois l'opération `RESTRICT`. Elle est donc toujours en temps polynômial $O(|u| \cdot k)$.

Créons maintenant un BDD u_r corrrespondant au réseau de régulation d'un réseau métabolique. Les réactions sont des variables booléennes de notre réseau de régulation et donc de notre BDD.

Pour un ensemble K de k réactions données, il est alors possible de vérifier en temps polynômial s'il existe une solution dans laquelle toutes les réactions sont affectées à Vrai ($\top$). Il suffit de vérifier que l'opération $\texttt{COUNT}(\texttt{LET}(u_r, K, \top^k))$ renvoie un résultat strictement positif.

Pour chaque mode élémentaire de flux (EFM) trouvé, il est donc possible de vérifier sur le BDD en temps polynômial s'il respecte la régulation.

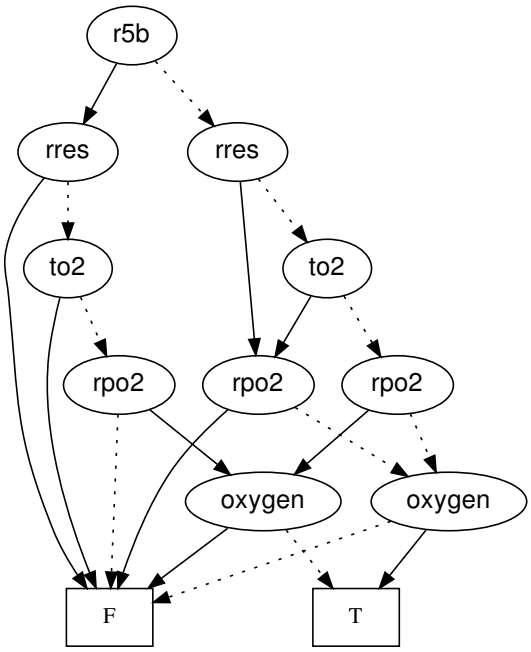


Figure B.2: ROBDD compilant des contraintes de régulation de CSP2001

On peut observer sur la Figure B.2 un BDD compilant les contraintes issues du CSP2001 présentées précédemment avec ASP (voir section 3.6 and section A.4). Appelons ce BDD u_1.

Il est possible d'observer directement que les réactions $r5b$ et $rres$ ne peuvent pas apparaître ensemble dans un même EFM. En effet, en suivant les traits pleins depuis les nœuds $r5b$ et $rres$, on arrive au nœud Faux.

En revanche, si l'on se tient à ce BDD qui ne représente qu'une petite partie du réseau de régulation complet, tout EFM ne contenant aucune des 3 réactions $\{r5b, rres, to2\}$ est une solution, que ce soit en aérobie ou en anaérobie. Il existe alors 2 solutions possibles pour les variables restantes, 2 chemins alternatifs : $\{rpo2 : \top,\ oxygen : \bot\}$ et $\{rpo2 : \bot,\ oxygen : \top\}$.

En d'autres termes, le nombre de solutions renvoyé par $\texttt{COUNT}(\texttt{LET}(u_1, \{r5b, rres\}, \{\top, \top\}))$ est 0. Le nombre de solutions renvoyé par $\texttt{COUNT}(\texttt{LET}(u_1, \{r5b, rres, to2\}, \{\bot, \bot, \bot\}))$ est 2.

B.1.2 Implémentation

Pour tester les BDDs, nous utilisons la librairie *dd* en Python. Cette librairie interface la librairie standard *cudd* pour les BDD réalisée en langage C, en utilisant Cython. Le code Python exécuté appelle donc du code C, ce qui permet des optimisations majeures en temps de calcul.

```
from dd import BDD
bdd = BDD() # Nouveau Manager de BDD
bdd.declare(variables) # Déclaration des variables
function = bdd.add_expr(expression) # Déclaration de la formule booléenne
restricted_function = bdd.let({'R1': True}, function) # LET
nb_sols = restricted_function.count() # COUNT
```

Nous allons désormais détailler l'intégration du BDD dans le solveur ASP. L'idée consistera à faire un appel aux fonctions `LET` et `COUNT` sur notre BDD lorsque ASP décide quelles réactions vont être actives pour éliminer prématurément les ensembles de réactions incompatibles.

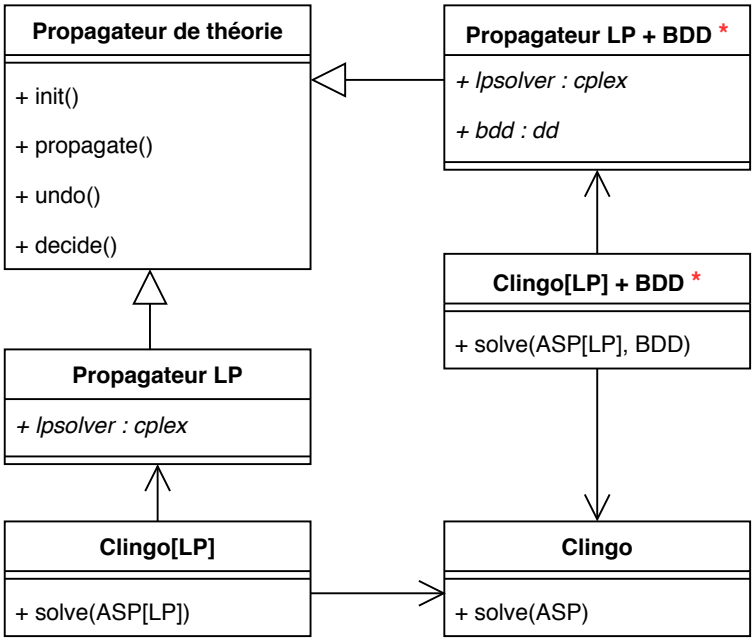


Figure B.3: Schéma UML simplifié des outils utilisés. En rouge: Nouveau code apporté au programme.

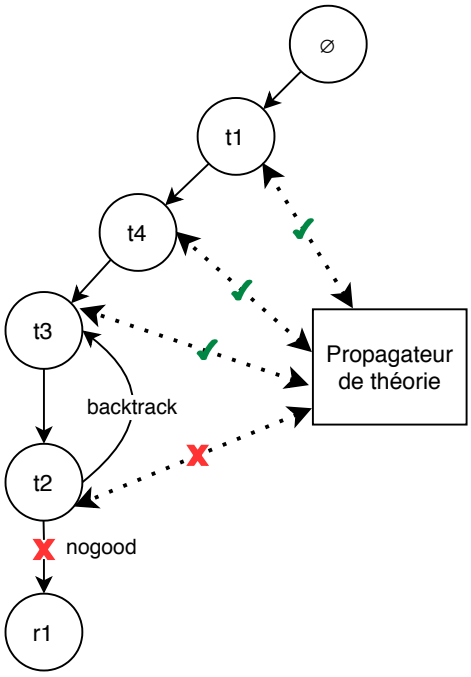


Figure B.4: Illustration générale de l'intégration d'un BDD avec le propagateur de théorie, prenant la formule Booléenne $\phi_1 = (t1 \Rightarrow \neg t2) \wedge (t3 \vee t4)$. A chaque propagation de littéral, un appel est réalisé au BDD, utilisant les méthodes `LET` et `COUNT` afin de vérifier en temps polynômial si l'affectation respecte ou non la formule.

B.1.3 Intégration dans *clingo*

En Figure B.4, nous proposons une illustration générale de l'intégration dans clingo des BDD avec le propagateur de théorie, prenant pour example la formule Booléenne de la Figure B.1. Dans le contexte ci-dessous, chaque flèche standard est une affectation à vrai ou "propagate", et "backtrack" fait référence à un évènement "undo". Enfin, "nogood" fait référence à un ajout d'une clause négative sur les littéraux restants $t2$ et $r1$ ensemble, ajout qui, comme l'appel au BDD, est effectué à l'intérieur de l'implémentation de la fonction "propagate".

Afin de tester les avantages de la compilation de connaissances, nous avons décidé d'intégrer les BDDs dans notre implémentation en clingo[LP]. Pour cela, nous pouvons utiliser l'interface des *propagateurs de théorie* de clingo. Un propagateur de théorie est une structure externe à clingo capable d'inspecter chaque affectation et ajouter des nogoods pendant la résolution ASP.

L'interface de propagateur de théorie est disponible depuis la version 5 du solveur clingo [201], et peut être implémentée en Python. Il s'agit de la même interface qui est utilisée pour intégrer la programmation linéaire dans clingo[LP].

Nous avons d'abord tenté d'utiliser deux propagateurs, un pour les programmes LP cplex et un pour les BDD, ce qui est possible avec clingo 5. En revanche, nous n'étions pas satisfaits du résultat, car le propagateur BDD semblait être appelé après le propagateur LP.

Nous avons donc finalement modifié le code de clingo[LP] afin d'y intégrer le propagateur pour le BDD directement avant tout appel au solveur linéaire, et ainsi couper les solutions incompatibles avec la régulation le plus tôt possible.

L'interface de propagateur définit des méthodes suivantes :

- **"init"** : l'initialisation du propagateur, on doit y définir les atomes à observer, on obtient alors les littéraux correspondants.
- **"propagate"** : cette méthode est appelée chaque fois qu'un littéral observé est affecté à Vrai, on a de plus contrôle sur la résolution et on peut ajouter des clauses ou des nogoods (clauses de littéraux négatifs) selon la valeur des littéraux observés.
- **"undo"** : cette méthode est appelée chaque fois que l'on "annule" un précédent "propagate", c'est à dire qu'on assigne à Faux un littéral qu'on avait précédemment affecté à Vrai. Cette méthode ne donne pas de contrôle de la résolution.
- **"decide"** : cette méthode est appelée lorsque le solveur doit décider quels littéraux assigner à Vrai, elle peut être utilisée pour implémenter des heuristiques.

Pour clarifier, si le littéral correspondant à l'atome observé est vrai, alors l'atome observé est présent dans la solution, et inversement.

Dans notre cas, les atomes à observer sont les atomes `support(R)`. Chaque fois qu'un littéral est affecté à Vrai ("propagate"), on ajoute la réaction `R` correspondante dans l'ensemble, et chaque fois que cette décision est annulée ("undo"), on le retire de l'ensemble.

A chaque "propagate", l'ensemble de réactions stocké correspond donc aux réactions présentes dans la solution actuellement calculée. On peut donc faire des requêtes à notre BDD pour vérifier si ces réactions peuvent apparaître ensemble dans une même solution. Le BDD nous répond en temps polynômial, et s'il n'y a pas de solution possible contenant ces réactions, on ajoute alors un nogood sur les littéraux.

Par exemple, avec notre implémentation de BDD en *dd* Python, nous faisons un simple appel à la méthode *count* afin de compter le nombre de chemins possibles en assignant toutes les réactions à Vrai avec *let*. S'il n'y a aucun chemin possible (*count* = 0), alors l'EFM correspondant n'est pas acceptable (voir subsection B.1.1): on ajoute un nogood.

On souhaite que le BDD nous aide au niveau des clauses booléennes disjonctives ou conjonctives de plus de deux litéraux. Les règles de régulation de nos réseaux de type "P $\implies$ Q" sont plutôt bien gérées par ASP. De ce fait, l'utilisation du BDD qui est une interface extérieure pourrait devenir une perte de temps. Lorsque nous avons suffisament de clauses, nous pensons que le BDD peut être plus rapide que ASP dans la gestion des contraintes.

B.1.4 Résultats

Nous avons compilé en BDDs nos deux réseaux de régulation étudiés, CSP2001 et ECOLICORE, puis nous avons compilé deux BDDs des EFMs solutions sur le réseau ECOLICORE.

Les caractéristiques de chaque BDD ainsi que leur temps de construction sont présentés en Table B.1.

Réseau	CSP2001	ECOLICORE	ECOLICORE	
Type	Régulation		Solutions	
Environnement	Encodés dans le BDD		Minimal	Enrichi
Nombre total d'EFMs possibles	jusqu'à 80	$226.3 \cdot 10^6$	4027	40551
Nombre de nœuds	438	5576	6752	18521
Nombre de variables	29	252	154	154
Nombre de solutions	5250	$1.1 \cdot 10^{37}$	4027	40551
Temps de construction	699 µs	17.3 ms	2.17 s	32.7 s

Table B.1: Caractéristiques de chaque BDD

Comme pour ASP, le réseau de régulation de CSP2001 nous a permis de tester la méthode. Nous retrouvons bien pour chaque environnement différent les mêmes résultats que ceux présentés dans l'article de Covert et Palsson.

Les environnements sont modifiés en faisant un appel à *let* pour affecter en conséquence les valeurs des variables correspondant aux métabolites externes. Les métabolites externes qui sont dans l'environnement sont affectés à Vrai, les autres métabolites externes à Faux.

On a donc un unique BDD compilant tous les environnements. Selon l'environnement on récupère le sous-BDD correspondant.

Pour le réseau ECOLICORE, nous avons testé les deux environnements suivants avec le BDD de régulation : l'environnement minimal {Glucose, Pi, H, H2O, CO2, NH4, O2} et l'environnement enrichi {Glucose, Pi, H, H2O, CO2, NH4, O2, Lactose, Pyruvate}.

En plus des réseaux de régulations, nous avons compilé deux BDDs des solutions du réseau ECOLICORE. En effet, les EFMs, ou du moins leurs supports, peuvent s'exprimer par une clause conjonctive de littéraux positifs.

Le premier BDD de solutions correspond aux EFMs de l'environnement minimal tandis que le second correspond aux EFMs de l'environnement enrichi. Si l'on donne ces solutions comme contraintes à clingo[LP], le solveur pourra alors nous recalculer les mêmes EFMs.

B.1.5 Comparaison

Nous comparons maintenant l'efficacité des contraintes compilées en BDD par rapport aux mêmes contraintes en ASP évaluées directement pendant la résolution. Pour cela, nous prenons les règles de calcul des EFMs en ASP[LP], le réseau ECOLICORE et les contraintes de régulation traduites soit en ASP soit en BDD. Dans le cas des contraintes BDD, nous utilisons notre version modifiée de clingo[LP] (voir subsection B.1.3) qui effectue des appels aux BDD via la librairie *dd*. Des résultats moyennés sur 5 éxécutions pour chaque condition sont présentés en Table B.2.

ECOLICORE		Environnement minimal	Environnement enrichi
Contraintes BDD	Nombre d'EFMs	4027	40651
	Temps (s)	1211.48	8973.29
	LPs résolus	51414	252777.4
	Nogoods ajoutés	22	20
Contraintes ASP	Nombre d'EFMs	4027	40325.6
	Temps (s)	1218.81	8914.80
	LPs résolus	43804	251836.4

Table B.2: Comparaison de la performance des méthodes avec les contraintes de régulation

On n'observe pas de différence significative pour le temps de calcul entre les contraintes ASP ou BDD, quelque soit l'environnement proposé. Le nombre de LPs résolus par clingo[LP] est cependant légèrement plus faible pour les contraintes ASP que pour les contraintes BDD.

Comme expliqué précédemment, cela est dû aux règles de régulation de type "P $\implies$ Q", qui sont très bien gérées à la fois par clingo et par le BDD. Notons que le nombre de nogoods ajoutés est très faible, ainsi avoir compilé le réseau en BDD n'apporte pas ici d'aide précieuse.

En revanche, sur nos BDDs des solutions (voir subsection B.1.4) – avec plus de nœuds et qui ont pris significativement plus de temps à construire – nous allons pouvoir observer des différences majeures. En effet, les solutions sont encodées sous forme de clauses conjonctives de nombreux littéraux positifs.

Nous présentons en Table B.3 les résultats d'une seule exécution pour chaque condition, avec les solutions comme contraintes. On observe un nombre conséquent de nogoods ajoutés par le BDD pour chaque environnement.

De plus, face aux clauses conjonctives, les contraintes ASP font appel au solveur LP près de 10 fois plus que les contraintes BDD. Le temps de calcul est lui aussi significativement plus élevé pour les contraintes ASP.

ECOLICORE		Environnement minimal	Environnement enrichi
Contraintes BDD	Nombre d'EFMs	4027	40088
	Temps (s)	611	4239
	LPs résolus	37416	233179
	Nogoods ajoutés	652	7752
Contraintes ASP	Nombre d'EFMs	4027	40551
	Temps (s)	1746	27599
	LPs résolus	300796	2239529

Table B.3: Comparaison des méthodes avec les solutions en tant que contraintes

Bien que ce test nécessite en premier lieu la connaissance des solutions, il montre les avantages possibles apportés par la compilation des contraintes en BDD par rapport à une résolution entièrement en ASP.

En effet, ces résultats laissent à penser que nous aurions observé un gain de temps par rapport à l'utilisation de ASP si nous avions compilé en BDD un réseau de régulation transcriptionnelle avec plus de règles de types clauses conjonctives.

List of Algorithms

List of Listings

Bibliography

[1] Iris Fry. The origins of research into the origins of life. *Endeavour*, 30(1):24–28, March 2006. ISSN 0160-9327. doi: 10.1016/j.endeavour.2005.12.002. URL https://www.sciencedirect.com/science/article/pii/S016093270600007X.

[2] Eugene V. Koonin and Petro Starokadomskyy. Are viruses alive? The replicator paradigm sheds decisive light on an old but misguided question. *Studies in history and philosophy of biological and biomedical sciences*, 59: 125–134, October 2016. ISSN 1369-8486. doi: 10.1016/j.shpsc.2016.02.016. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5406846/.

[3] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. The RNA World and the Origins of Life. In *Molecular Biology of the Cell. 4th edition*. Garland Science, 2002. URL https://www.ncbi.nlm.nih.gov/books/NBK26876/.

[4] Frank AL Anet. The place of metabolism in the origin of life. *Current Opinion in Chemical Biology*, 8(6):654–659, December 2004. ISSN 1367-5931. doi: 10.1016/j.cbpa.2004.10.005. URL https://www.sciencedirect.com/science/article/pii/S1367593104001371.

[5] George Plopper and Diana Bebek Ivankovic. *Principles of Cell Biology*. Jones & Bartlett Learning, LLC, Burlington, 2020. ISBN 978-1-284-14984-5.

[6] Nick Lane, John F. Allen, and William Martin. How did LUCA make a living? Chemiosmosis in the origin of life. *BioEssays*, 32(4):271–280, 2010. ISSN 1521-1878. doi: 10.1002/bies.200900131. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bies.200900131.

[7] J. G. Salway and D. K. Granner. *Metabolism at a Glance Ed. 3*. John Wiley & Sons, 2013. ISBN 978-1-4051-0716-7. URL https://univ.scholarvox.com/book/88839012.

[8] Gerhard Michal. Roche Biochemical Pathways Wall Charts by Gerhard Michal, January 2018. URL https://zenodo.org/record/4446230.

[9] Lawrence E. Hunter. Cytoplasm and Metabolism. In *The Processes of Life : An Introduction to Molecular Biology*, pages 77–82. MIT Press, 2012. ISBN 978-0-262-01305-5. URL https://unr-ra-scholarvox-com.docelec.univ-lyon1.fr/catalog/book/docid/88841796?searchterm=metabolic%20pathways.

[10] Antimicrobial Resistance Collaborators. Global burden of bacterial antimicrobial resistance in 2019: a systematic analysis. *The Lancet*, 399(10325):629–655, February 2022. ISSN 0140-6736. doi: 10.1016/S0140-6736(21)02724-0. URL https://www.sciencedirect.com/science/article/pii/S0140673621027240.

[11] Heather G. Ahlgren, Andrea Benedetti, Jennifer S. Landry, Joanie Bernier, Elias Matouk, Danuta Radzioch, Larry C. Lands, Simon Rousseau, and Dao Nguyen. Clinical outcomes associated with Staphylococcus aureus and Pseudomonas aeruginosa airway infections in adult cystic fibrosis patients. *BMC Pulmonary Medicine*, 15(1):67, December 2015. ISSN 1471-2466. doi: 10.1186/s12890-015-0062-7. URL https://bmcpulmmed.biomedcentral.com/articles/10.1186/s12890-015-0062-7.

[12] Alejandro Schcolnik-Cabrera, Alma Chávez-Blanco, Guadalupe Domínguez-Gómez, and Alfonso Dueñas-González. Understanding tumor anabolism and patient catabolism in cancer-associated cachexia. *American Journal of Cancer Research*, 7(5):1107–1135, May 2017. ISSN 2156-6976. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5446478/.

[13] Donald Voet and Judith G. Voet. *Biochemistry*. J. Wiley & Sons, New York, 2nd ed edition, 1995. ISBN 978-0-471-58651-7. URL http://catdir.loc.gov/catdir/toc/onix02/94049605.html.

[14] Daniel L. Purich. *Enzyme Kinetics : Catalysis & Control : A Reference of Theory and Best-Practice Methods*. Elsevier Science, 2010. ISBN 978-0-12-380924-7. URL https://univ.scholarvox.com/book/88812171#.

[15] IUBMB International Union of Biochemistry and Molecular Biology. Nomenclature Committee. *Enzyme nomenclature 1992 : recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes*. San Diego : Published for the International Union of Biochemistry and Molecular Biology by Academic Press, 1992. ISBN 978-0-12-227164-9 978-0-12-227165-6. URL http://archive.org/details/enzymenomenclatu0000inte_d6c2.

[16] Pauline M. Doran. *Bioprocess engineering principles, second edition*. EngineeringPro collection. Academic Press, Waltham, Mass., 2nd ed edition, 2013. ISBN 978-0-12-220851-5. URL http://www.books24x7.com/marc.asp?bookid=50921.

[17] Athel Cornish-Bowden. *Fundamentals of Enzyme Kinetics*. Wiley-VCH, Weinheim, 4., auflage edition, 2013. ISBN 978-3-527-66548-8. URL https://nbn-resolving.org/urn:nbn:de:101:1-2014081611156.

[18] Vladimir Leskovac. *Comprehensive Enzyme Kinetics*. Springer Science & Business Media, March 2003. ISBN 978-0-306-46712-7. doi: 10.1007/b100340.

[19] A. Flamholz, E. Noor, A. Bar-Even, and R. Milo. eQuilibrator–the biochemical thermodynamics calculator. *Nucleic Acids Research*, 40:D770–D775; DOI:10.1093/nar/gkr874, 2012. doi: 10.1093/nar/gkr874.

[20] Elad Noor, Hulda S. Haraldsdóttir, Ron Milo, and Ronan M. T. Fleming. Consistent Estimation of Gibbs Energy Using Component Contributions. *PLOS Computational Biology*, 9(7):e1003098, July 2013. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1003098. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003098.

[21] David L. Nelson and Michael Cox. *Lehninger principles of biochemistry (8th Edition)*. Macmillan International, 2021. ISBN 978-1-319-38149-3.

[22] R. Eisenthal and A. Cornish-Bowden. Prospects for antiparasitic drugs. The case of Trypanosoma brucei, the causative agent of African sleeping sickness. *The Journal of Biological Chemistry*, 273(10):5500–5505, March 1998. ISSN 0021-9258. doi: 10.1074/jbc.273.10.5500.

[23] Dorotea Dudaš, Ulrike Wittig, Maja Rey, Andreas Weidemann, and Wolfgang Müller. Improved insights into the SABIO-RK database via visualization. *Database*, 2023:baad011, January 2023. ISSN 1758-0463. doi: 10.1093/database/baad011. URL https://doi.org/10.1093/database/baad011.

[24] Sandra Placzek, Ida Schomburg, Antje Chang, Lisa Jeske, Marcus Ulbrich, Jana Tillack, and Dietmar Schomburg. BRENDA in 2017: new perspectives and new tools in BRENDA. *Nucleic Acids Research*, 45(D1): D380–D388, January 2017. ISSN 0305-1048. doi: 10.1093/nar/gkw952. URL https://doi.org/10.1093/nar/gkw952.

[25] B. Nagel, H. Dellweg, and L. M. Gierasch. Glossary for chemists of terms used in biotechnology (IUPAC Recommendations 1992). *Pure and Applied Chemistry*, 64(1):143–168, January 1992. ISSN 1365-3075. doi: 10.1351/pac199264010143. URL https://www.degruyter.com/document/doi/10.1351/pac199264010143/html.

[26] Peter F. Stanbury, Allan Whitaker, and Stephen J. Hall. Microbial growth kinetics. In *Principles of Fermentation Technology*. Butterworth-Heinemann, August 2016. ISBN 978-0-444-63408-5.

[27] Amit Varma and Bernhard O. Palsson. Metabolic Flux Balancing: Basic Concepts, Scientific and Practical Use. *Bio/Technology*, 12(10):994–998, October 1994. ISSN 1546-1696. doi: 10.1038/nbt1094-994. URL https://www.nature.com/articles/nbt1094-994.

[28] Jeffrey D. Orth, Ronan M. T. Fleming, and Bernhard Ø. Palsson. Reconstruction and use of microbial metabolic networks: the core escherichia coli metabolic model as an educational guide. *EcoSal*, 28:245–248, 2010. doi: 10.1128/ecosal.10.2.1.

[29] Virginie Orgogozo, Baptiste Morizot, and Arnaud Martin. The differential view of genotype–phenotype relationships. *Frontiers in Genetics*, 6, 2015. ISSN 1664-8021. doi: 10.3389/fgene.2015.00179. URL https://www.frontiersin.org/articles/10.3389/fgene.2015.00179.

[30] Yehudit Hasin, Marcus Seldin, and Aldons Lusis. Multi-omics approaches to disease. *Genome Biology*, 18(1):83, May 2017. ISSN 1474-760X. doi: 10.1186/s13059-017-1215-1. URL https://doi.org/10.1186/s13059-017-1215-1.

[31] Andrew R. Joyce and Bernhard Ø Palsson. The model organism as a system: integrating 'omics' data sets. *Nature Reviews Molecular Cell Biology*, 7(3):198–210, March 2006. ISSN 1471-0080. doi: 10.1038/nrm1857. URL https://www.nature.com/articles/nrm1857.

[32] Mohit Jain, Roland Nilsson, Sonia Sharma, Nikhil Madhusudhan, Toshimori Kitami, Amanda L. Souza, Ran

Kafri, Marc W. Kirschner, Clary B. Clish, and Vamsi K. Mootha. Metabolite profiling identifies a key role for glycine in rapid cancer cell proliferation. *Science*, 336(6084):1040–1044, 2012. doi: 10.1126/science.1218595.

[33] Uwe Sauer. High-throughput phenomics: experimental methods for mapping fluxomes. *Current Opinion in Biotechnology*, 15(1):58–63, February 2004. ISSN 0958-1669. doi: 10.1016/j.copbio.2003.11.001. URL https://www.sciencedirect.com/science/article/pii/S0958166903001800.

[34] The UniProt Consortium. UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531, January 2023. ISSN 0305-1048. doi: 10.1093/nar/gkac1052. URL https://doi.org/10.1093/nar/gkac1052.

[35] Minoru Kanehisa, Michihiro Araki, Susumu Goto, Masahiro Hattori, Mika Hirakawa, Masumi Itoh, Toshiaki Katayama, Shuichi Kawashima, Shujiro Okuda, Toshiaki Tokimatsu, and Yoshihiro Yamanishi. KEGG for linking genomes to life and the environment. *Nucleic Acids Research*, 36(suppl_1):D480–D484, January 2008. ISSN 0305-1048. doi: 10.1093/nar/gkm882. URL https://doi.org/10.1093/nar/gkm882.

[36] Samuel M D Seaver, Filipe Liu, Qizhi Zhang, James Jeffryes, José P Faria, Janaka N Edirisinghe, Michael Mundy, Nicholas Chia, Elad Noor, Moritz E Beber, Aaron A Best, Matthew DeJongh, Jeffrey A Kimbrel, Patrik D'haeseleer, Sean R McCorkle, Jay R Bolton, Erik Pearson, Shane Canon, Elisha M Wood-Charlson, Robert W Cottingham, Adam P Arkin, and Christopher S Henry. The ModelSEED Biochemistry Database for the integration of metabolic annotations and the reconstruction, comparison and analysis of metabolic models for plants, fungi and microbes. *Nucleic Acids Research*, 49(D1):D575–D588, January 2021. ISSN 0305-1048. doi: 10.1093/nar/gkaa746. URL https://doi.org/10.1093/nar/gkaa746.

[37] Zachary A. King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A. Lerman, Ali Ebrahim, Bernhard O. Palsson, and Nathan E. Lewis. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic acids research*, 44(D1):D515–D522, January 2016. ISSN 1362-4962. doi: 10.1093/nar/gkv1049. URL https://pubmed.ncbi.nlm.nih.gov/26476456.

[38] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, J. Wang, and the rest of the SBML Forum. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg015. URL https://doi.org/10.1093/bioinformatics/btg015.

[39] Rahuman S. Malik-Sheriff, Mihai Glont, Tung V. N. Nguyen, Krishna Tiwari, Matthew G. Roberts, Ashley Xavier, Manh T. Vu, Jinghao Men, Matthieu Maire, Sarubini Kananathan, Emma L. Fairbanks, Johannes P. Meyer, Chinmay Arankalle, Thawfeek M. Varusai, Vincent Knight-Schrijver, Lu Li, Corina Duenas-Roca, Gaurhari Dass, Sarah M. Keating, Young M. Park, Nicola Buso, Nicolas Rodriguez, Michael Hucka, and Henning Hermjakob.

BioModels—15 years of sharing computational models in life science. *Nucleic Acids Research*, November 2019. doi: 10.1093/nar/gkz1055. URL https://doi.org/10.1093/nar/gkz1055.

[40] Finja Büchel, Nicolas Rodriguez, Neil Swainston, Clemens Wrzodek, Tobias Czauderna, Roland Keller, Florian Mittag, Michael Schubert, Mihai Glont, Martin Golebiewski, Martijn van Iersel, Sarah Keating, Matthias Rall, Michael Wybrow, Henning Hermjakob, Michael Hucka, Douglas B. Kell, Wolfgang Müller, Pedro Mendes, Andreas Zell, Claudine Chaouiya, Julio Saez-Rodriguez, Falk Schreiber, Camille Laibe, Andreas Dräger, and Nicolas Le Novère. Path2Models: large-scale generation of computational models from biochemical pathway maps. *BMC systems biology*, 7:116, November 2013. ISSN 1752-0509. doi: 10.1186/1752-0509-7-116.

[41] Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I. Aladjem, Sarala M. Wimalaratne, Frank T. Bergman, Ralph Gauges, Peter Ghazal, Hideya Kawaji, Lu Li, Yukiko Matsuoka, Alice Villéger, Sarah E. Boyd, Laurence Calzone, Melanie Courtot, Ugur Dogrusoz, Tom C. Freeman, Akira Funahashi, Samik Ghosh, Akiya Jouraku, Sohyoung Kim, Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryanin, Douglas B. Kell, Chris Sander, Herbert Sauro, Jacky L. Snoep, Kurt Kohn, and Hiroaki Kitano. The Systems Biology Graphical Notation. *Nature Biotechnology*, 27(8):735–741, August 2009. ISSN 1546-1696. doi: 10.1038/nbt.1558. URL https://www.nature.com/articles/nbt.1558.

[42] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, January 2000. ISSN 0305-1048. doi: 10.1093/nar/28.1.235. URL https://doi.org/10.1093/nar/28.1.235.

[43] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, Augustin Žídek, Tim Green, Kathryn Tunyasuvunakool, Stig Petersen, John Jumper, Ellen Clancy, Richard Green, Ankur Vora, Mira Lutfi, Michael Figurnov, Andrew Cowie, Nicole Hobbs, Pushmeet Kohli, Gerard Kleywegt, Ewan Birney, Demis Hassabis, and Sameer Velankar. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 50(D1):D439–D444, January 2022. ISSN 0305-1048. doi: 10.1093/nar/gkab1061. URL https://doi.org/10.1093/nar/gkab1061.

[44] David S. Wishart, Yannick D. Feunang, An C. Guo, Elvis J. Lo, Ana Marcu, Jason R. Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082, January 2018. ISSN 1362-4962. doi: 10.1093/nar/gkx1037.

[45] Athel Cornish-Bowden. Making systems biology work in the 21st century. *Genome Biology*, 6(4):317, March 2005. ISSN 1474-760X. doi: 10.1186/gb-2005-6-4-317. URL https://doi.org/10.1186/gb-2005-6-4-317.

[46] Mihajlo D. Mesarović. Systems Theory and Biology—View of a Theoretician. In M. D. Mesarović, editor,

Systems Theory and Biology, pages 59–87, Berlin, Heidelberg, 1968. Springer. ISBN 978-3-642-88343-9. doi: 10.1007/978-3-642-88343-9_3.

[47] Alain Friboulet and Daniel Thomas. Systems Biology—an interdisciplinary approach. *Biosensors and Bioelectronics*, 20(12):2404–2407, June 2005. ISSN 0956-5663. doi: 10.1016/j.bios.2004.11.014. URL https://www.sciencedirect.com/science/article/pii/S0956566304005639.

[48] Marc W. Kirschner. The Meaning of Systems Biology. *Cell*, 121(4):503–504, May 2005. ISSN 0092-8674, 1097-4172. doi: 10.1016/j.cell.2005.05.005. URL https://www.cell.com/cell/abstract/S0092-8674(05)00447-2.

[49] Hiroaki Kitano. Systems Biology: A Brief Overview. *Science*, 295(5560):1662–1664, March 2002. doi: 10.1126/science.1069492. URL https://www.science.org/doi/full/10.1126/science.1069492.

[50] René Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563–585, December 1973. ISSN 0022-5193. doi: 10.1016/0022-5193(73)90247-6. URL https://www.sciencedirect.com/science/article/pii/0022519373902476.

[51] Markus W. Covert. *Fundamentals of Systems Biology: From Synthetic Circuits to Whole-cell Models*. CRC Press, October 2017. ISBN 978-1-4987-2847-8.

[52] Uri Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. CRC Press, July 2019. ISBN 978-1-00-000132-7.

[53] François Jacob and Jacques Monod. Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3(3):318–356, June 1961. ISSN 00222836. doi: 10.1016/S0022-2836(61)80072-7. URL https://linkinghub.elsevier.com/retrieve/pii/S0022283661800727.

[54] Hans V. Westerhoff and Bernhard O. Palsson. The evolution of molecular biology into systems biology. *Nature Biotechnology*, 22(10):1249–1252, October 2004. ISSN 1546-1696. doi: 10.1038/nbt1020. URL https://www.nature.com/articles/nbt1020.

[55] Olaf Wolkenhauer. Systems biology: The reincarnation of systems theory applied in biology? *Briefings in Bioinformatics*, 2(3):258–270, September 2001. ISSN 1467-5463. doi: 10.1093/bib/2.3.258. URL https://doi.org/10.1093/bib/2.3.258.

[56] Douglas B. Kell. Systems biology, metabolic modelling and metabolomics in drug discovery and development. *Drug Discovery Today*, 11(23):1085–1092, December 2006. ISSN 1359-6446. doi: 10.1016/j.drudis.2006.10.004. URL https://www.sciencedirect.com/science/article/pii/S1359644606004181.

[57] Douglas B. Kell and Stephen G. Oliver. Here is the evidence, now what is the hypothesis? The complementary roles of inductive and hypothesis-driven science in the post-genomic era. *BioEssays: News and Reviews in Molecular, Cellular and Developmental Biology*, 26(1):99–105, January 2004. ISSN 0265-9247. doi: 10.1002/bies.10385.

[58] Ferric C. Fang and Arturo Casadevall. Reductionistic and Holistic Science. *Infection and Immunity*, 79(4):

1401–1404, April 2011. ISSN 0019-9567. doi: 10.1128/IAI.01343-10. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3067528/.

[59] Jens Nielsen. Systems Biology of Metabolism. *Annual Review of Biochemistry*, 86:245–275, June 2017. ISSN 1545-4509. doi: 10.1146/annurev-biochem-061516-044757.

[60] Jens Nielsen and Michael C. Jewett. Impact of systems biology on metabolic engineering of Saccharomyces cerevisiae. *FEMS Yeast Research*, 8(1):122–131, February 2008. ISSN 1567-1356. doi: 10.1111/j.1567-1364.2007.00302.x. URL https://doi.org/10.1111/j.1567-1364.2007.00302.x.

[61] Rosemary Yu and Jens Nielsen. Big data in yeast systems biology. *FEMS Yeast Research*, 19(7):foz070, 2019. doi: 10.1093/femsyr/foz070.

[62] D. Ewen Cameron, Caleb J. Bashor, and James J. Collins. A brief history of synthetic biology. *Nature Reviews Microbiology*, 12(5):381–390, May 2014. ISSN 1740-1534. doi: 10.1038/nrmicro3239. URL https://www.nature.com/articles/nrmicro3239.

[63] Hiroaki Kitano. Systems biology: toward system-level understanding of biological systems. *Foundations of systems biology*, pages 1–36, 2001.

[64] David Fell. *Understanding the control of metabolism*. January 1997. ISBN 1 85578 047 X.

[65] Hans V. Westerhoff, Jan-Hendrik S. Hofmeyr, and Boris N. Kholodenko. Getting to the inside of cells using metabolic control analysis. *Biophysical Chemistry*, 50(3):273–283, June 1994. ISSN 0301-4622. doi: 10.1016/0301-4622(93)E0095-M. URL https://www.sciencedirect.com/science/article/pii/0301462293E0095M.

[66] Athel Cornish-Bowden. Putting the Systems Back into Systems Biology. *Perspectives in Biology and Medicine*, 49(4):475–489, 2006. ISSN 1529-8795. doi: 10.1353/pbm.2006.0053. URL https://muse.jhu.edu/pub/1/article/204463.

[67] Athel Cornish-Bowden, María Luz Cárdenas, Juan-Carlos Letelier, Jorge Soto-Andrade, and Flavio Guíñez Abarzúa. Understanding the parts in terms of the whole. *Biology of the Cell*, 96(9):713–717, 2004. ISSN 1768-322X. doi: 10.1016/j.biolcel.2004.06.006. URL https://onlinelibrary.wiley.com/doi/abs/10.1016/j.biolcel.2004.06.006.

[68] Katrin Hübner, Sven Sahle, and Ursula Kummer. Applications and trends in systems biology in biochemistry. *The FEBS Journal*, 278(16):2767–2857, 2011. ISSN 1742-4658. doi: 10.1111/j.1742-4658.2011.08217.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1742-4658.2011.08217.x.

[69] Brett G. Olivier and Frank T. Bergmann. SBML Level 3 Package: Flux Balance Constraints version 2. *Journal of Integrative Bioinformatics*, 15(1):1–39, 2018. ISSN 1613-4516. doi: 10.1515/jib-2017-0082.

[70] J. Kyle Medley, Kiri Choi, Matthias König, Lucian Smith, Stanley Gu, Joseph Hellerstein, Stuart C. Sealfon, and Herbert M. Sauro. Tellurium notebooks—An environment for reproducible dynamical modeling in systems

biology. *PLOS Computational Biology*, 14(6):e1006220, June 2018. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006220. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006220.

[71] Jeffrey D. Orth, Ines Thiele, and Bernhard Ø. Palsson. What is flux balance analysis? *Nature Biotechnology*, 28(3):245–248, 2010. ISSN 1546-1696. doi: 10.1038/nbt.1614. URL https://doi.org/10.1038/nbt.1614.

[72] Ali Ebrahim, Joshua A. Lerman, Bernhard O. Palsson, and Daniel R. Hyduke. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Systems Biology*, 7(1):74, August 2013. ISSN 1752-0509. doi: 10.1186/1752-0509-7-74. URL https://doi.org/10.1186/1752-0509-7-74.

[73] Markus W. Covert, Eric M. Knight, Jennifer L. Reed, Markus J. Herrgard, and Bernhard O. Palsson. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92–96, 2004. doi: 10.1038/nature02456. URL https://doi.org/10.1038/nature02456.

[74] Avlant Nilsson and Jens Nielsen. Genome scale metabolic modeling of cancer. *Metabolic Engineering*, 43(Pt B):103–112, September 2017. ISSN 1096-7184. doi: 10.1016/j.ymben.2016.10.022.

[75] Almut Heinken, Geeta Acharya, Dmitry Ravcheev, Johannes Hertel, Malgorzata Nyga, Onyedika Emmanuel Okpala, Marcus Hogan, Stefanía Magnúsdóttir, Filippo Martinelli, German Preciat, Janaka Edirisinghe, Christopher Henry, Ronan M.T. Fleming, and Ines Thiele. AGORA2: Large scale reconstruction of the microbiome highlights wide-spread drug-metabolising capacitie. Technical report, 2020. URL https://doi.org/10.1101/2020.11.09.375451.

[76] Daniel Machado, Oleksandr M. Maistrenko, Sergej Andrejev, Yongkyu Kim, Peer Bork, Kaustubh R. Patil, and Kiran R. Patil. Polarization of microbial communities between competitive and cooperative metabolism. *Nature Ecology & Evolution*, 5(2):195–203, February 2021. ISSN 2397-334X. doi: 10.1038/s41559-020-01353-4. URL https://www.nature.com/articles/s41559-020-01353-4.

[77] Timothy S. Gardner, Charles R. Cantor, and James J. Collins. Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403(6767):339–342, January 2000. ISSN 1476-4687. doi: 10.1038/35002131. URL https://www.nature.com/articles/35002131.

[78] Matthias Heinemann and Sven Panke. Synthetic biology—putting engineering into biology. *Bioinformatics*, 22 (22):2790–2799, November 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl469. URL https://doi.org/10.1093/bioinformatics/btl469.

[79] Gregory Stephanopoulos. Synthetic Biology and Metabolic Engineering. *ACS Synthetic Biology*, 1(11):514–525, November 2012. doi: 10.1021/sb300094q. URL https://doi.org/10.1021/sb300094q.

[80] Gregory Stephanopoulos and Anthony J. Sinskey. Metabolic engineering — methodologies and future prospects. *Trends in Biotechnology*, 11(9):392–396, September 1993. ISSN 0167-7799. doi: 10.1016/0167-7799(93)90099-U. URL https://www.sciencedirect.com/science/article/pii/016777999390099U.

[81] Gregory Stephanopoulos, Aristos A. Aristidou, and Jens Nielsen. *Metabolic Engineering: Principles and Methodologies*. Elsevier, October 1998. ISBN 978-0-08-053628-6.

[82] Jae Sung Cho, Gi Bae Kim, Hyunmin Eun, Cheon Woo Moon, and Sang Yup Lee. Designing Microbial Cell Factories for the Production of Chemicals. *JACS Au*, 2(8):1781–1799, August 2022. doi: 10.1021/jacsau.2c00344. URL https://doi.org/10.1021/jacsau.2c00344.

[83] Jens Nielsen and Jay D. Keasling. Engineering Cellular Metabolism. *Cell*, 164(6):1185–1197, March 2016. ISSN 00928674. doi: 10.1016/j.cell.2016.02.004. URL https://linkinghub.elsevier.com/retrieve/pii/S0092867416300708.

[84] Sang Yup Lee, Hyun Uk Kim, Tong Un Chae, Jae Sung Cho, Je Woong Kim, Jae Ho Shin, Dong In Kim, Yoo-Sung Ko, Woo Dae Jang, and Yu-Sin Jang. A comprehensive metabolic map for production of bio-based chemicals. *Nature Catalysis*, 2(1):18–33, January 2019. ISSN 2520-1158. doi: 10.1038/s41929-018-0212-4. URL https://www.nature.com/articles/s41929-018-0212-4.

[85] Jay D. Keasling. Synthetic biology and the development of tools for metabolic engineering. *Metabolic Engineering*, 14(3):189–195, May 2012. ISSN 1096-7176. doi: 10.1016/j.ymben.2012.01.004. URL https://www.sciencedirect.com/science/article/pii/S1096717612000055.

[86] Rongming Liu, Marcelo C. Bassalo, Ramsey I. Zeitoun, and Ryan T. Gill. Genome scale engineering techniques for metabolic engineering. *Metabolic Engineering*, 32:143–154, November 2015. ISSN 1096-7176. doi: 10.1016/j.ymben.2015.09.013. URL https://www.sciencedirect.com/science/article/pii/S1096717615001238.

[87] Hidde De Jong. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. Technical report, 2000. URL https://hal.inria.fr/inria-00072606.

[88] Misbah Razzaq, Loïc Paulevé, Anne Siegel, Julio Saez-Rodriguez, Jérémie Bourdon, and Carito Guziolowski. Computational discovery of dynamic cell line specific Boolean networks from multiplex time-course data. *PLOS Computational Biology*, 14(10):1–23, October 2018. doi: 10.1371/journal.pcbi.1006538. URL https://doi.org/10.1371/journal.pcbi.1006538.

[89] Maxime Mahout, Ross P. Carlson, and Sabine Peres. Answer Set Programming for Computing Constraints-Based Elementary Flux Modes: Application to Escherichia coli Core Metabolism. *Processes*, 8(12):1649, December 2020. ISSN 2227-9717. doi: 10.3390/pr8121649. URL https://www.mdpi.com/2227-9717/8/12/1649.

[90] Markus W. Covert, Nan Xiao, Tiffany J. Chen, and Jonathan R. Karr. Integrating metabolic, transcriptional regulatory and signal transduction models in Escherichia coli. *Bioinformatics*, 24(18):2044–2050, September 2008. ISSN 1367-4803. doi: 10.1093/bioinformatics/btn352. URL https://doi.org/10.1093/bioinformatics/btn352.

[91] Bernhard Ø. Palsson. *Systems biology: Constraint-based reconstruction and analysis*. Cambridge University Press, 2015. ISBN 978-1-316-23994-0. URL https://books.google.fr/books?id=QNBpBgAAQBAJ.

[92] Nathan D. Price, Jason A. Papin, Christophe H. Schilling, and Bernhard O. Palsson. Genome-scale microbial in silico models: the constraints-based approach. *Trends in Biotechnology*, 21(4):162–169, April 2003. ISSN 0167-7799. doi: 10.1016/S0167-7799(03)00030-1.

[93] Markus W. Covert and Bernhard O. Palsson. Constraints-based models: Regulation of Gene Expression Reduces the Steady-state Solution Space. *Journal of Theoretical Biology*, 221(3):309–325, 2003. doi: 10.1006/jtbi.2003.3071. URL https://doi.org/10.1006/jtbi.2003.3071.

[94] Markus W. Covert and Bernhard Ø Palsson. Transcriptional Regulation in Constraints-based Metabolic Models ofEscherichia coli. *Journal of Biological Chemistry*, 277(31):28058–28064, 2002. doi: 10.1074/jbc.m201691200. URL https://doi.org/10.1074/jbc.m201691200.

[95] Bernhard Palsson. The challenges of in silico biology. *Nature Biotechnology*, 18(11):1147–1150, November 2000. doi: 10.1038/81125. URL https://doi.org/10.1038/81125.

[96] Michel Minoux. *Programmation mathématique: théorie et algorithmes*. Editions Technique & Documentation : Lavoisier, Paris, 2e éd edition, 2008. ISBN 978-2-7430-1000-3. URL http://catalogue.bnf.fr/ark:/12148/cb412483685.

[97] George B Dantzig. Origins of the simplex method. In *A history of scientific computing*, pages 141–151. 1990.

[98] Irvin J. Lustig and Jean-François Puget. Program Does Not Equal Program: Constraint Programming and Its Relationship to Mathematical Programming. *Interfaces*, 31(6):29–53, December 2001. ISSN 0092-2102. doi: 10.1287/inte.31.6.29.9647. URL https://pubsonline.informs.org/doi/10.1287/inte.31.6.29.9647.

[99] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, Chichester Weinheim, nachdr. edition, 2011. ISBN 978-0-471-98232-6.

[100] Günter M. Ziegler. Polytopes, Polyhedra, and Cones. In Günter M. Ziegler, editor, *Lectures on Polytopes: Updated Seventh Printing of the First Edition*, Graduate Texts in Mathematics, pages 27–50. Springer, New York, NY, 1995. ISBN 978-1-4613-8431-1. doi: 10.1007/978-1-4613-8431-1_1. URL https://doi.org/10.1007/978-1-4613-8431-1_1.

[101] Florian A. Potra and Stephen J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1):281–302, December 2000. ISSN 0377-0427. doi: 10.1016/S0377-0427(00)00433-7. URL https://www.sciencedirect.com/science/article/pii/S0377042700004337.

[102] Richard M. Karp. Reducibility among Combinatorial Problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, The IBM Research Symposia Series, pages 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

[103] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, January 2005. ISSN 0167-6377. doi: 10.1016/j.orl.2004.04.002. URL https://www.sciencedirect.com/science/article/pii/S0167637704000501.

[104] Juan Pablo Vielma. Mixed Integer Linear Programming Formulation Techniques. *SIAM Review*, 57(1):3–57, 2015. ISSN 0036-1445. doi: 10.1137/130915303. URL https://www.jstor.org/stable/24248517.

[105] Ignacio E. Grossmann and Zdravko Kravanja. Mixed-integer nonlinear programming techniques for process systems engineering. *Computers & Chemical Engineering*, 19:189–204, June 1995. ISSN 0098-1354. doi: 10.1016/0098-1354(95)87036-9. URL https://www.sciencedirect.com/science/article/pii/0098135495870369.

[106] Evgeni V. Nikolaev, Anthony P. Burgard, and Costas D. Maranas. Elucidation and Structural Analysis of Conserved Pools for Genome-Scale Metabolic Reconstructions. *Biophysical Journal*, 88(1):37–49, January 2005. ISSN 0006-3495. doi: 10.1529/biophysj.104.043489. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1305013/.

[107] Steffen Klamt, Utz-Uwe Haus, and Fabian Theis. Hypergraphs and Cellular Networks. *PLoS Computational Biology*, 5(5):e1000385, May 2009. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000385. URL https://dx.plos.org/10.1371/journal.pcbi.1000385.

[108] R. Carlson and F. Srienc. Fundamental Escherichia coli biochemical pathways for biomass and energy production: identification of reactions. *Biotechnology and Bioengineering*, 85(1):1–19, January 2004. doi: 10.1002/bit.10812.

[109] T Pfeiffer, I Sanchez-Valdenebro, J C Nuno, F Montero, and S Schuster. METATOOL: for studying metabolic networks. *Bioinformatics*, 15(3):251–257, 1999. ISSN 1367-4803. doi: 10.1093/bioinformatics/15.3.251. URL https://doi.org/10.1093/bioinformatics/15.3.251.

[110] Nathan D. Price, Jennifer L. Reed, and Bernhard Ø Palsson. Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nature Reviews Microbiology*, 2(11):886–897, November 2004. ISSN 1740-1534. doi: 10.1038/nrmicro1023. URL https://www.nature.com/articles/nrmicro1023.

[111] A. Varma and B. O. Palsson. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type Escherichia coli W3110. *Applied and environmental microbiology*, 60(10):3724–3731, October 1994. ISSN 0099-2240. doi: 10.1128/aem.60.10.3724-3731.1994. URL https://pubmed.ncbi.nlm.nih.gov/7986045.

[112] Laurent Heirendt, Sylvain Arreckx, Thomas Pfau, Sebastián N. Mendoza, Anne Richelle, Almut Heinken, Hulda S. Haraldsdóttir, Jacek Wachowiak, Sarah M. Keating, Vanja Vlasov, Stefania Magnusdóttir, Chiam Yu Ng, German Preciat, Alise Žagare, Siu H. J. Chan, Maike K. Aurich, Catherine M. Clancy, Jennifer Modamio, John T. Sauls, Alberto Noronha, Aarash Bordbar, Benjamin Cousins, Diana C. El Assal, Luis V. Valcarcel, Iñigo Apaolaza, Susan Ghaderi, Masoud Ahookhosh, Marouen Ben Guebila, Andrejs Kostromins, Nicolas Sompairac, Hoai M. Le, Ding Ma, Yuekai Sun, Lin Wang, James T. Yurkovich, Miguel A. P. Oliveira, Phan T. Vuong, Lemmer P. El Assal, Inna Kuperstein, Andrei Zinovyev, H. Scott Hinton, William A. Bryant, Francisco J. Aragón Artacho, Francisco J. Planes, Egils Stalidzans, Alejandro Maass, Santosh Vempala, Michael Hucka, Michael A. Saunders, Costas D. Maranas, Nathan E. Lewis, Thomas Sauter, Bernhard Ø Palsson, Ines Thiele, and Ronan M. T.

Fleming. Creation and analysis of biochemical constraint-based models using the COBRA Toolbox v.3.0. *Nature Protocols*, 14(3):639–702, March 2019. ISSN 1750-2799. doi: 10.1038/s41596-018-0098-2. URL https://www.nature.com/articles/s41596-018-0098-2.

[113] Benjamin J. Bornstein, Sarah M. Keating, Akiya Jouraku, and Michael Hucka. LibSBML: An API library for SBML. *Bioinformatics (Oxford, England)*, 24(6):880–881, 2008. ISSN 1367-4803. doi: 10.1093/bioinformatics/btn051. URL http://dx.doi.org/10.1093/bioinformatics/btn051.

[114] Adam M Feist and Bernhard O Palsson. The biomass objective function. *Current Opinion in Microbiology*, 13 (3):344–349, June 2010. ISSN 1369-5274. doi: 10.1016/j.mib.2010.03.003. URL 10.1016/j.mib.2010.03.003.

[115] J. P. Folsom and R. P. Carlson. Physiological, biomass elemental composition and proteomic analyses of Escherichia coli ammonium-limited chemostat growth, and comparison with iron- and glucose-limited chemostat growth. *Microbiology*, 161(8):1659–1670, 2015. doi: 10.1099/mic.0.000118. URL https://mic.microbiologyresearch.org/content/journal/micro/10.1099/mic.0.000118.

[116] Ashley E. Beck, Kristopher A. Hunt, and Ross P. Carlson. Measuring Cellular Biomass Composition for Computational Biology Applications. *Processes*, 6(5):38, May 2018. ISSN 2227-9717. doi: 10.3390/pr6050038. URL https://www.mdpi.com/2227-9717/6/5/38.

[117] Radhakrishnan Mahadevan, Jeremy S. Edwards, and Francis J. Doyle. Dynamic Flux Balance Analysis of Diauxic Growth in Escherichia coli. *Biophysical Journal*, 83(3):1331–1340, September 2002. ISSN 0006-3495. doi: 10.1016/S0006-3495(02)73903-9. URL https://www.sciencedirect.com/science/article/pii/S0006349502739039.

[118] Markus W. Covert, Christophe H. Schilling, and Bernhard O. Palsson. Regulation of Gene Expression in Flux Balance Models of Metabolism. *Journal of Theoretical Biology*, 213(1):73–88, November 2001. doi: 10.1006/jtbi.2001.2405. URL https://doi.org/10.1006/jtbi.2001.2405.

[119] Robert Schuetz, Lars Kuepfer, and Uwe Sauer. Systematic evaluation of objective functions for predicting intracellular fluxes in Escherichia coli. *Molecular Systems Biology*, 3:119, July 2007. ISSN 1744-4292. doi: 10.1038/msb4100162. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1949037/.

[120] Steinn Gudmundsson and Ines Thiele. Computationally efficient flux variability analysis. *BMC Bioinformatics*, 11(1):489, September 2010. ISSN 1471-2105. doi: 10.1186/1471-2105-11-489. URL https://doi.org/10.1186/1471-2105-11-489.

[121] Helena A. Herrmann, Beth C. Dyson, Lucy Vass, Giles N. Johnson, and Jean-Marc Schwartz. Flux sampling is a powerful tool to study metabolism under changing environmental conditions. *npj Systems Biology and Applications*, 5(1):1–8, September 2019. ISSN 2056-7189. doi: 10.1038/s41540-019-0109-0. URL http://www.nature.com/articles/s41540-019-0109-0.

[122] Wout Megchelenbrink, Martijn Huynen, and Elena Marchiori. optGpSampler: An Improved Tool for Uniformly Sampling the Solution-Space of Genome-Scale Metabolic Networks. *PLOS ONE*, 9(2):e86587, February 2014.

ISSN 1932-6203. doi: 10.1371/journal.pone.0086587. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0086587.

[123] Nathan E. Lewis, Kim K. Hixson, Tom M. Conrad, Joshua A. Lerman, Pep Charusanti, Ashoka D. Polpitiya, Joshua N. Adkins, Gunnar Schramm, Samuel O. Purvine, Daniel Lopez-Ferrer, Karl K. Weitz, Roland Eils, Rainer König, Richard D. Smith, and Bernhard Ø Palsson. Omic data from evolved E. coli are consistent with computed optimal growth from genome-scale models. *Molecular Systems Biology*, 6:390, July 2010. ISSN 1744-4292. doi: 10.1038/msb.2010.47.

[124] R. Mahadevan and C. H. Schilling. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metabolic Engineering*, 5(4):264–276, October 2003. ISSN 1096-7176. doi: 10.1016/j.ymben.2003.09.002.

[125] S. Klamt, J. Gagneur, and A. von Kamp. Algorithmic approaches for computing elementary modes in large biochemical reaction networks. *IEE Proceedings - Systems Biology*, 152(4):249–255(6), December 2005. ISSN 1741-2471. doi: 10.1049/ip-syb:20050035.

[126] Julien Gagneur and Steffen Klamt. Computation of elementary modes: A unifying framework and the new binary approach. *BMC bioinformatics*, 5:175, 2004. doi: 10.1186/1471-2105-5-175.

[127] Marco Terzer. *Large scale methods to enumerate extreme rays and elementary modes*. PhD Thesis, ETH Zurich, 2009.

[128] A. Rezola, L. F. de Figueiredo, M. Brock, J. Pey, A. Podhorski, C. Wittmann, S. Schuster, A. Bockmayr, and F. J. Planes. Exploring metabolic pathways in genome-scale networks via generating flux modes. *Bioinformatics*, 27(4):534–540, 2010. ISSN 1367-4803. doi: 10.1093/bioinformatics/btq681. URL https://doi.org/10.1093/bioinformatics/btq681.

[129] Steffen Klamt, Georg Regensburger, Matthias P. Gerstl, Christian Jungreuthmayer, Stefan Schuster, Radhakrishnan Mahadevan, Jürgen Zanghellini, and Stefan Müller. From elementary flux modes to elementary flux vectors: Metabolic pathway analysis with arbitrary linear flux constraints. *PLOS Computational Biology*, 13: e1005409, April 2017. doi: 10.1371/journal.pcbi.1005409.

[130] Tom J. Clement, Erik B. Baalhuis, Bas Teusink, Frank J. Bruggeman, Robert Planqué, and Daan H. de Groot. Unlocking Elementary Conversion Modes: ecmtool Unveils All Capabilities of Metabolic Networks. *Patterns*, 2(1):100177, January 2021. ISSN 2666-3899. doi: 10.1016/j.patter.2020.100177. URL https://www.sciencedirect.com/science/article/pii/S2666389920302415.

[131] J.-M. Schwartz and M. Kanehisa. A quadratic programming approach for decomposing steady-state metabolic flux distributions onto elementary modes. *Bioinformatics*, 21(Suppl 2):ii204–ii205, 2005. doi: 10.1093/bioinformatics/bti1132.

[132] J.-M. Schwartz and M. Kanehisa. Quantitative elementary mode analysis of metabolic pathways: the example of yeast glycolysis. *BMC Bioinformatics*, 7(1):1–20, 2006. doi: 10.1186/1471-2105-7-186.

[133] Komei Fukuda and Alain Prodon. Double Description Method Revisited. In *Combinatorics and Computer Science*, 1995. doi: 10.1007/3-540-61576-8_77. URL https://doi.org/10.1007/3-540-61576-8_77.

[134] Vicente Acuña, Flavio Chierichetti, Vincent Lacroix, Alberto Marchetti-Spaccamela, Marie-France Sagot, and Leen Stougie. Modes and cuts in metabolic networks: Complexity and algorithms. *Biosystems*, 95(1):51 – 60, 2009. ISSN 0303-2647. doi: 10.1016/j.biosystems.2008.06.015. URL http://www.sciencedirect.com/science/article/pii/S0303264708001469.

[135] Vicente Acuña, Alberto Marchetti-Spaccamela, Marie-France Sagot, and Leen Stougie. A note on the complexity of finding and enumerating elementary modes. *Biosystems*, 99(3):210 – 214, 2010. ISSN 0303-2647. doi: 10.1016/j.biosystems.2009.11.004. URL http://www.sciencedirect.com/science/article/pii/S0303264709002019.

[136] Axel von Kamp and Stefan Schuster. Metatool 5.0: fast and flexible elementary modes analysis. *Bioinformatics*, 22(15):1930–1931, August 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl267. URL 10.1093/bioinformatics/btl267.

[137] M. G. Poolman. ScrumPy: metabolic modelling with Python. *Systems Biology*, 153(5):375–378, September 2006. ISSN 1741-2471. doi: 10.1049/ip-syb:20060010. URL 10.1049/ip-syb:20060010.

[138] Sven Thiele, Axel von Kamp, Pavlos Stephanos Bekiaris, Philipp Schneider, and Steffen Klamt. CNApy: a CellNetAnalyzer GUI in Python for Analyzing and Designing Metabolic Networks. *Bioinformatics*, 2021. doi: 10.1093/bioinformatics/btab828. URL 10.1093/bioinformatics/btab828.

[139] Marco Terzer and Jörg Stelling. Large-scale computation of elementary flux modes with bit pattern trees. *Bioinformatics*, 24(19):2229–2235, 2008. ISSN 1367-4803. doi: 10.1093/bioinformatics/btn401. URL https://doi.org/10.1093/bioinformatics/btn401.

[140] Luis F. de Figueiredo, Adam Podhorski, Angel Rubio, Christoph Kaleta, John E. Beasley, Stefan Schuster, and Francisco J. Planes. Computing the shortest elementary flux modes in genome-scale metabolic networks. *Bioinformatics*, 25(23):3158–3165, 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp564. URL https://doi.org/10.1093/bioinformatics/btp564.

[141] T.S. Motzkin, H. Raiffa, G.L. Thompson, and R.M. Thrall. The double description method. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to theory of games, Vol. 2*. Princeton University Press, 1953.

[142] Martin Morterol. *Méthodes avancées de raisonnement en logique propositionnelle : application aux réseaux métaboliques*. PhD thesis, Université Paris-Sud.

[143] Christian Jungreuthmayer, David E. Ruckerbauer, and Jürgen Zanghellini. regEfmtool: Speeding up elementary flux mode calculation using transcriptional regulatory rules in the form of three-state logic. *Biosystems*, 113 (1):37 – 39, 2013. ISSN 0303-2647. doi: 10.1016/j.biosystems.2013.04.002. URL http://www.sciencedirect.com/science/article/pii/S0303264713000890.

[144] K. A. Hunt, J. P. Folsom, R. L. Taffs, and R. P. Carlson. Complete enumeration of elementary flux modes

through scalable demand-based subnetwork definition. *Bioinformatics*, 30(11):1569–1578, 2014. doi: 10.1093/bioinformatics/btu021.

[145] Sabine Peres, Mario Jolicœur, Cécile Moulin, Philippe Dague, and Stefan Schuster. How important is thermodynamics for identifying elementary flux modes? *PLOS ONE*, 12(2):1–20, 2017. doi: 10.1371/journal.pone.0171440. URL https://doi.org/10.1371/journal.pone.0171440.

[146] Bianca A. Buchner and Jürgen Zanghellini. EFMlrs: a Python package for elementary flux mode enumeration via lexicographic reverse search. *BMC Bioinformatics*, 22(1):547, November 2021. ISSN 1471-2105. doi: 10.1186/s12859-021-04417-9. URL https://doi.org/10.1186/s12859-021-04417-9.

[147] David Avis and Komei Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry*, 8:295–313, 1992. doi: 10.1007/BF02293050.

[148] David Avis and Charles Jordan. Comparative computational results for some vertex and facet enumeration codes. *arXiv:1510.02545v3*, 2016.

[149] Steffen Klamt and Axel von Kamp. An application programming interface for CellNetAnalyzer. *Biosystems*, 105(2):162–168, August 2011. ISSN 0303-2647. doi: 10.1016/j.biosystems.2011.02.002. URL https://www.sciencedirect.com/science/article/pii/S0303264711000402.

[150] Jon Pey and Francisco J. Planes. Direct calculation of elementary flux modes satisfying several biological constraints in genome-scale metabolic networks. *Bioinformatics*, 30(15):2197–2203, 2014. ISSN 1367-4803. doi: 10.1093/bioinformatics/btu193. URL https://doi.org/10.1093/bioinformatics/btu193.

[151] Vítor Vieira and Miguel Rocha. CoBAMP: a Python framework for metabolic pathway analysis in constraint-based models. *Bioinformatics*, 35(24):5361–5362, July 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz598. URL https://doi.org/10.1093/bioinformatics/btz598.

[152] A. Rezola, J. Pey, L. F. de Figueiredo, A. Podhorski, S. Schuster, A. Rubio, and F. J. Planes. Selection of human tissue-specific elementary flux modes using gene expression data. *Bioinformatics*, 29(16):2009–2016, 2013. doi: 10.1093/bioinformatics/btt328.

[153] Axel von Kamp and Steffen Klamt. Enumeration of smallest intervention strategies in genome-scale metabolic networks. *PLOS Computational Biology*, 10(1):1–13, January 2014. doi: 10.1371/journal.pcbi.1003378. URL https://doi.org/10.1371/journal.pcbi.1003378.

[154] Luis Tobalina, Jon Pey, and Francisco J. Planes. Direct calculation of minimal cut sets involving a specific reaction knock-out. *Bioinformatics*, 32(13):2001–2007, July 2016. ISSN 1367-4803. doi: 10.1093/bioinformatics/btw072. URL 10.1093/bioinformatics/btw072.

[155] Hyun-Seob Song, Noam Goldberg, Ashutosh Mahajan, and Doraiswami Ramkrishna. Sequential computation of elementary modes and minimal cut sets in genome-scale metabolic networks using alternate integer linear programming. *Bioinformatics*, 33(15):2345–2353, March 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx171. URL https://doi.org/10.1093/bioinformatics/btx171.

[156] S. Peres, M. Morterol, and L. Simon. SAT-Based Metabolics Pathways Analysis without Compilation. In J.O. Dada P. Mendes and K. Smallbone, editors, *Lecture Note in Bioinformatics*, volume 8859, pages 20–31. Springer International Publishing, 2014. doi: 10.1007/978-3-319-12982-2_2. URL https://doi.org/10.1007/978-3-319-12982-2_2.

[157] M. Morterol, P. Dague, S. Peres, and L. Simon. Minimality of metabolic flux modes under boolean regulation constraints. In *Workshop on constraint-based methods for bioinformatics (WCB)*, 2016.

[158] Nuala A. O'Leary, Mathew W. Wright, J. Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, Alexander Astashyn, Azat Badretdin, Yiming Bao, Olga Blinkova, Vyacheslav Brover, Vyacheslav Chetvernin, Jinna Choi, Eric Cox, Olga Ermolaeva, Catherine M. Farrell, Tamara Goldfarb, Tripti Gupta, Daniel Haft, Eneida Hatcher, Wratko Hlavina, Vinita S. Joardar, Vamsi K. Kodali, Wenjun Li, Donna Maglott, Patrick Masterson, Kelly M. McGarvey, Michael R. Murphy, Kathleen O'Neill, Shashikant Pujar, Sanjida H. Rangwala, Daniel Rausch, Lillian D. Riddick, Conrad Schoch, Andrei Shkeda, Susan S. Storz, Hanzhen Sun, Francoise Thibaud-Nissen, Igor Tolstoy, Raymond E. Tully, Anjana R. Vatsan, Craig Wallin, David Webb, Wendy Wu, Melissa J. Landrum, Avi Kimchi, Tatiana Tatusova, Michael DiCuccio, Paul Kitts, Terence D. Murphy, and Kim D. Pruitt. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Research*, 44(D1):D733–745, January 2016. ISSN 1362-4962. doi: 10.1093/nar/gkv1189.

[159] Peter D. Karp, Daniel Weaver, and Mario Latendresse. How accurate is automated gap filling of metabolic models? *BMC Systems Biology*, 12(1):73, June 2018. ISSN 1752-0509. doi: 10.1186/s12918-018-0593-7. URL https://doi.org/10.1186/s12918-018-0593-7.

[160] Clémence Frioux, Torsten Schaub, Sebastian Schellhorn, Anne Siegel, and Philipp Wanko. Hybrid metabolic network completion. In Marcello Balduccini and Tomi Janhunen, editors, *Logic programming and nonmonotonic reasoning*, pages 308–321, Cham, 2017. Springer International Publishing. ISBN 978-3-319-61660-5. doi: 10.1007/978-3-319-61660-5_28.

[161] Nathan E. Lewis, Harish Nagarajan, and Bernhard O. Palsson. Constraining the metabolic genotype–phenotype relationship using a phylogeny of in silico methods. *Nature Reviews Microbiology*, 10(4):291–305, April 2012. ISSN 1740-1534. doi: 10.1038/nrmicro2737. URL https://www.nature.com/articles/nrmicro2737.

[162] Daniel Machado, Markus J. Herrgård, and Isabel Rocha. Stoichiometric Representation of Gene–Protein–Reaction Associations Leverages Constraint-Based Analysis from Reaction to Gene-Level Phenotype Prediction. *PLOS Computational Biology*, 12(10):e1005140, October 2016. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1005140. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005140.

[163] Yara Seif, Jonathan M. Monk, Nathan Mih, Hannah Tsunemoto, Saugat Poudel, Cristal Zuniga, Jared Broddrick, Karsten Zengler, and Bernhard O. Palsson. A computational knowledge-base elucidates the response

of Staphylococcus aureus to different media types. *PLOS Computational Biology*, 15(1):e1006644, January 2019. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006644. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006644.

[164] Scott A. Becker and Bernhard Ø Palsson. Genome-scale reconstruction of the metabolic network in Staphylococcus aureus N315: an initial draft to the two-dimensional annotation. *BMC microbiology*, 5:8, March 2005. ISSN 1471-2180. doi: 10.1186/1471-2180-5-8. URL 10.1186/1471-2180-5-8.

[165] Anne Goelzer and Vincent Fromion. Resource allocation in living organisms. *Biochemical Society Transactions*, 45(4):945–952, July 2017. ISSN 0300-5127. doi: 10.1042/BST20160436. URL 10.1042/BST20160436.

[166] A. Goelzer and V. Fromion. Bacterial growth rate reflects a bottleneck in resource allocation. *Biochimica et Biophysica Acta (BBA) - General Subjects*, 1810(10):978–988, October 2011. ISSN 0304-4165. doi: 10.1016/j.bbagen.2011.05.014. URL https://www.sciencedirect.com/science/article/pii/S0304416511001267.

[167] A. Goelzer, J. Muntel, V. Chubukov, M. Jules, E. Prestel, R. Nolker, M. Mariadassou, S. Aymerich, M. Hecker, P. Noirot, D. Becher, and V. Fromion. Quantitative prediction of genome-wide resource allocation in bacteria. *Metabolic Engineering*, 32:232–243, 2015. doi: 10/f7x9n3.

[168] Ana Bulović, Stephan Fischer, Marc Dinh, Felipe Golib, Wolfram Liebermeister, Christian Poirier, Laurent Tournier, Edda Klipp, Vincent Fromion, and Anne Goelzer. Automated generation of bacterial resource allocation models. *Metabolic Engineering*, 55:12–22, September 2019. ISSN 1096-7176. doi: 10.1016/j.ymben.2019.06.001. URL 10.1016/j.ymben.2019.06.001.

[169] Christian Lieven, Moritz E. Beber, Brett G. Olivier, Frank T. Bergmann, Meric Ataman, Parizad Babaei, Jennifer A. Bartell, Lars M. Blank, Siddharth Chauhan, Kevin Correia, Christian Diener, Andreas Dräger, Birgitta E. Ebert, Janaka N. Edirisinghe, José P. Faria, Adam M. Feist, Georgios Fengos, Ronan M. T. Fleming, Beatriz García-Jiménez, Vassily Hatzimanikatis, Wout van Helvoirt, Christopher S. Henry, Henning Hermjakob, Markus J. Herrgård, Ali Kaafarani, Hyun Uk Kim, Zachary King, Steffen Klamt, Edda Klipp, Jasper J. Koehorst, Matthias König, Meiyappan Lakshmanan, Dong-Yup Lee, Sang Yup Lee, Sunjae Lee, Nathan E. Lewis, Filipe Liu, Hongwu Ma, Daniel Machado, Radhakrishnan Mahadevan, Paulo Maia, Adil Mardinoglu, Gregory L. Medlock, Jonathan M. Monk, Jens Nielsen, Lars Keld Nielsen, Juan Nogales, Intawat Nookaew, Bernhard O. Palsson, Jason A. Papin, Kiran R. Patil, Mark Poolman, Nathan D. Price, Osbaldo Resendis-Antonio, Anne Richelle, Isabel Rocha, Benjamín J. Sánchez, Peter J. Schaap, Rahuman S. Malik Sheriff, Saeed Shoaie, Nikolaus Sonnenschein, Bas Teusink, Paulo Vilaça, Jon Olav Vik, Judith A. H. Wodke, Joana C. Xavier, Qianqian Yuan, Maksim Zakhartsev, and Cheng Zhang. MEMOTE for standardized genome-scale metabolic model testing. *Nature Biotechnology*, 38(3):272–276, March 2020. ISSN 1546-1696. doi: 10.1038/s41587-020-0446-y. URL https://www.nature.com/articles/s41587-020-0446-y.

[170] Alina Renz and Andreas Dräger. Curating and comparing 114 strain-specific genome-scale metabolic models of Staphylococcus aureus. *NPJ systems biology and applications*, 7(1):30, June 2021. ISSN 2056-7189. doi: 10.1038/s41540-021-00188-4. URL 10.1038/s41540-021-00188-4.

[171] Ines Thiele, Swagatika Sahoo, Almut Heinken, Johannes Hertel, Laurent Heirendt, Maike K. Aurich, and Ronan MT Fleming. Personalized whole-body models integrate metabolism, physiology, and the gut microbiome. *Molecular Systems Biology*, 16(5):e8982, May 2020. ISSN 1744-4292. doi: 10.15252/msb.20198982. URL https://www.embopress.org/doi/10.15252/msb.20198982.

[172] Albert Gevorgyan, Mark G. Poolman, and David A. Fell. Detection of stoichiometric inconsistencies in biomolecular models. *Bioinformatics*, 24(19):2245–2251, October 2008. ISSN 1367-4803. doi: 10.1093/bioinformatics/btn425. URL https://doi.org/10.1093/bioinformatics/btn425.

[173] Woosub Shin and Joseph L Hellerstein. Isolating structural errors in reaction networks in systems biology. *Bioinformatics*, 37(3):388–395, April 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btaa720. URL https://doi.org/10.1093/bioinformatics/btaa720.

[174] Robert Urbanczik and Clemens Wagner. Functional stoichiometric analysis of metabolic networks. *Bioinformatics*, 21(22):4176–4180, 2005. doi: 10.1093/bioinformatics/bti674.

[175] Patrick F Suthers, Alireza Zomorrodi, and Costas D Maranas. Genome-scale gene/reaction essentiality and synthetic lethality analysis. *Molecular Systems Biology*, 5(1):301, January 2009. ISSN 1744-4292. doi: 10.1038/msb.2009.56. URL https://www.embopress.org/doi/full/10.1038/msb.2009.56.

[176] A. P. Burgard, S. Vaidyaraman, and C. D. Maranas. Minimal reaction sets for Escherichia coli metabolism under different growth requirements and uptake environments. *Biotechnology Progress*, 17(5):791–797, 2001. ISSN 8756-7938. doi: 10.1021/bp0100880.

[177] Aditya Pratapa, Shankar Balachandran, and Karthik Raman. Fast-SL: an efficient algorithm to identify synthetic lethal sets in metabolic networks. *Bioinformatics*, 31(20):3299–3305, October 2015. ISSN 1367-4803. doi: 10.1093/bioinformatics/btv352. URL 10.1093/bioinformatics/btv352.

[178] Mehdi Dehghan Manshadi, Payam Setoodeh, and Habil Zare. Rapid-SL identifies synthetic lethal sets with an arbitrary cardinality. *Scientific Reports*, 12(1):14022, August 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-18177-w. URL https://www.nature.com/articles/s41598-022-18177-w.

[179] Xiaotie Deng. Complexity Issues in Bilevel Linear Programming. In Athanasios Migdalas, Panos M. Pardalos, and Peter Värbrand, editors, *Multilevel Optimization: Algorithms and Applications*, Nonconvex Optimization and Its Applications, pages 149–164. Springer US, Boston, MA, 1998. ISBN 978-1-4613-0307-7. doi: 10.1007/978-1-4613-0307-7_6. URL https://doi.org/10.1007/978-1-4613-0307-7_6.

[180] S. Klamt and E. D. Gilles. Minimal cut sets in biochemical reaction networks. *Bioinformatics*, 20:226–234, 2004. doi: 10.1093/bioinformatics/btg395.

[181] C. Berge. *Hypergraphs, volume 45: Combinatorics of finite sets*. North Holland, 1 edition, August 1989. ISBN 0-444-87489-5.

[182] O. Hädicke and S. Klamt. Computing complex metabolic intervention strategies using constrained minimal cut sets. *Metabolic Engineering*, 13(2):204–213, 2011. doi: 10.1016/j.ymben.2010.12.004.

[183] Matthias P. Gerstl, Steffen Klamt, Christian Jungreuthmayer, and Jürgen Zanghellini. Exact quantification of cellular robustness in genome-scale metabolic networks. *Bioinformatics*, 32(5):730–737, November 2015. ISSN 1367-4803. doi: 10.1093/bioinformatics/btv649. URL https://doi.org/10.1093/bioinformatics/btv649.

[184] Iñigo Apaolaza, Edurne San José-Eneriz, Luis Tobalina, Estíbaliz Miranda, Leire Garate, Xabier Agirre, Felipe Prósper, and Francisco J. Planes. An in-silico approach to predict and exploit synthetic lethality in cancer metabolism. *Nature Communications*, 8(1):459, September 2017. ISSN 2041-1723. doi: 10.1038/s41467-017-00555-y. URL 10.1038/s41467-017-00555-y.

[185] Steffen Klamt and Jörg Stelling. Combinatorial Complexity of Pathway Analysis in Metabolic Networks. *Molecular Biology Reports*, 29(1):233–236, March 2002. ISSN 1573-4978. doi: 10.1023/A:1020390132244. URL https://doi.org/10.1023/A:1020390132244.

[186] Kathrin Ballerstein. Logical interaction networks in biology. 2012. doi: 10.3929/ETHZ-A-007558749. URL 10.3929/ETHZ-A-007558749.

[187] Kathrin Ballerstein, Axel von Kamp, Steffen Klamt, and Utz-Uwe Haus. Minimal cut sets in a metabolic network are elementary modes in a dual network. *Bioinformatics*, 28(3):381–387, February 2012. ISSN 1460-2059, 1367-4803. doi: 10.1093/bioinformatics/btr674. URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr674.

[188] Iñigo Apaolaza, Luis Vitores Valcarcel, and Francisco J Planes. gMCS: fast computation of genetic minimal cut sets in large networks. *Bioinformatics*, 35(3):535–537, February 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty656. URL 10.1093/bioinformatics/bty656.

[189] Philipp Schneider, Axel von Kamp, and Steffen Klamt. An extended and generalized framework for the calculation of metabolic intervention strategies based on minimal cut sets. *PLOS Computational Biology*, 16(7):e1008110, July 2020. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1008110. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008110.

[190] Anthony P. Burgard, Priti Pharkya, and Costas D. Maranas. Optknock: A bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and Bioengineering*, 84(6):647–657, 2003. ISSN 1097-0290. doi: 10.1002/bit.10803. URL http://onlinelibrary.wiley.com/doi/abs/10.1002/bit.10803.

[191] Zachary A. King, Andreas Dräger, Ali Ebrahim, Nikolaus Sonnenschein, Nathan E. Lewis, and Bernhard O. Palsson. Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways. *PLoS computational biology*, 11(8):e1004321–e1004321, 2015. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004321. URL https://pubmed.ncbi.nlm.nih.gov/26313928.

[192] Elizabeth Brunk, Swagatika Sahoo, Daniel C. Zielinski, Ali Altunkaya, Andreas Dräger, Nathan Mih, Francesco Gatto, Avlant Nilsson, German Andres Preciat Gonzalez, Maike Kathrin Aurich, Andreas Prlić, Anand Sastry, Anna D. Danielsdottir, Almut Heinken, Alberto Noronha, Peter W. Rose, Stephen K. Burley, Ronan M.T. Fleming,

Jens Nielsen, Ines Thiele, and Bernhard O. Palsson. Recon3D: A Resource Enabling A Three-Dimensional View of Gene Variation in Human Metabolism. *Nature biotechnology*, 36(3):272–281, March 2018. ISSN 1087-0156. doi: 10.1038/nbt.4072. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5840010/.

[193] Samir Giri, Leonardo Oña, Silvio Waschina, Shraddha Shitut, Ghada Yousif, Christoph Kaleta, and Christian Kost. Metabolic dissimilarity determines the establishment of cross-feeding interactions in bacteria. *Current Biology*, 31(24):5547–5557.e6, December 2021. ISSN 0960-9822. doi: 10.1016/j.cub.2021.10.019. URL https://www.sciencedirect.com/science/article/pii/S0960982221014081.

[194] Daniel C. Zielinski, Neema Jamshidi, Austin J. Corbett, Aarash Bordbar, Alex Thomas, and Bernhard O. Palsson. Systems biology analysis of drivers underlying hallmarks of cancer cell metabolism. *Scientific Reports*, 7 (1):41241, January 2017. ISSN 2045-2322. doi: 10.1038/srep41241. URL https://www.nature.com/articles/srep41241.

[195] M.T. Wortel, E. Noor, M. Ferris, F.J. Bruggeman, and W. Liebermeister. Metabolic enzyme cost explains variable trade-offs between microbial growth rate and yield. *PLOS Computational Biology*, 14(2):1–21, February 2018. doi: 10.1371/journal.pcbi.1006010. URL https://doi.org/10.1371/journal.pcbi.1006010.

[196] S. Müller, G. Regensburger, and R. Steuer. Enzyme allocation problems in kinetic metabolic networks: Optimal solutions are elementary flux modes. *Journal of theoretical biology*, 347:182–190, 2014. doi: 10.1016/j.jtbi.2013.11.015.

[197] Cecile Moulin, Laurent Tournier, and Sabine Peres. Combining Kinetic and Constraint-Based Modelling to Better Understand Metabolism Dynamics. *Processes*, 9(10):1701, October 2021. ISSN 2227-9717. doi: 10.3390/pr9101701. URL https://www.mdpi.com/2227-9717/9/10/1701.

[198] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Pearson Education, Limited, Harlow, 2016. ISBN 978-1-292-15396-4.

[199] Alain Colmerauer and Philippe Roussel. The birth of Prolog. In *History of programming languages—II*, pages 331–367. Association for Computing Machinery, New York, NY, USA, January 1996. ISBN 978-0-201-89502-5. URL https://doi.org/10.1145/234286.1057820.

[200] Vladimir Lifschitz. What Is Answer Set Programming?. In *AAAI*, volume 8, pages 1594–1597, 2008. doi: 10.5555/1620270.1620340.

[201] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko. Theory Solving Made Easy with Clingo 5. In *Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016)*. Schloss-Dagstuhl - Leibniz Zentrum für Informatik, 2016. doi: 10.4230/OASIcs.ICLP.2016.2. URL https://drops.dagstuhl.de/entities/document/10.4230/OASIcs.ICLP.2016.2.

[202] Frederic Portoraro. Automated reasoning. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford encyclopedia of philosophy*. Metaphysics Research Lab, Stanford University, summer 2023 edition, 2023. URL https://plato.stanford.edu/archives/sum2023/entries/reasoning-automated/.

[203] Robert Kowalski. Predicate logic as programming language. volume 74 of *IFIP Congr.*, pages 569–574, January 1974.

[204] Curtis Franks. Propositional logic. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford encyclopedia of philosophy*. Metaphysics Research Lab, Stanford University, summer 2023 edition, 2023. URL https://plato.stanford.edu/archives/sum2023/entries/logic-propositional/.

[205] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, May 1971. Association for Computing Machinery. ISBN 978-1-4503-7464-4. doi: 10.1145/800157.805047. URL https://dl.acm.org/doi/10.1145/800157.805047.

[206] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, April 1984. ISSN 0166-218X. doi: 10.1016/0166-218X(84)90081-7. URL https://www.sciencedirect.com/science/article/pii/0166218X84900817.

[207] A. Darwiche and P. Marquis. A Knowledge Compilation Map. *J. Artif. Intell. Res.*, 17:229–264, 2002. doi: 10.1613/jair.989. URL https://doi.org/10.1613/jair.989.

[208] Oliver Ray, Takehide Soh, and Katsumi Inoue. Analyzing Pathways Using ASP-Based Approaches. In Katsuhisa Horimoto, Masahiko Nakatsui, and Nikolaj Popov, editors, *Algebraic and Numeric Biology*, volume 6479 of *Lecture Notes in Computer Science*, pages 167–183, Berlin, Heidelberg, 2012. Springer. ISBN 978-3-642-28067-2. doi: 10.1007/978-3-642-28067-2_10.

[209] M. Gebser, T. Schaub, S. Thiele, B. Usadel, and P. Veber. Detecting inconsistencies in large biological networks with answer set programming. *International Conference on Logic Programming*, pages 130–144, 2008. doi: 10.1007/978-3-540-89982-2_19.

[210] Tomi Janhunen, Roland Kaminski, Max Ostrowski, Sebastian Schellhorn, Philipp Wanko, and Torsten Schaub. Clingo goes linear constraints over reals and integers. *Theory and Practice of Logic Programming*, 17(5-6):872–888, September 2017. ISSN 1471-0684, 1475-3081. doi: 10.1017/S1471068417000242. URL https://www.cambridge.org/core/journals/theory-and-practice-of-logic-programming/article/clingo-goes-linear-constraints-over-reals-and-integers/E314128EA348CB01FC7ECA66C0FE64EF.

[211] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187-188:52 – 89, 2012. ISSN 0004-3702. doi: 10.1016/j.artint.2012.04.001. URL http://www.sciencedirect.com/science/article/pii/S0004370212000409.

[212] Roland Kaminski, Torsten Schaub, and Philipp Wanko. A Tutorial on Hybrid Answer Set Solving with clingo. pages 167–203. 2017. ISBN 978-3-319-61032-0. doi: 10.1007/978-3-319-61033-7_6.

[213] Roland Kaminski, Javier Romero, Torsten Schaub, and Philipp Wanko. How to build your own ASP-based system?! *arXiv:2008.06692 [cs]*, August 2020. URL http://arxiv.org/abs/2008.06692.

[214] Yuliya Lierler. What is answer set programming to propositional satisfiability. *Constraints*, 22(3):

307–337, 2017. ISSN 1572-9354. doi: 10.1007/s10601-016-9257-7. URL https://doi.org/10.1007/s10601-016-9257-7.

[215] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Marius Lindauer, Max Ostrowski, Javier Romero, Torsten Schaub, Sven Thiele, and Philipp Wanko. Potassco user guide. URL https://github.com/potassco/guide/releases/tag/v2.2.0.

[216] Abdelhamid Boudane, Said Jabbour, Badran Raddaoui, and Lakhdar Sais. Efficient SAT-Based encodings of conditional cardinality constraints. In *LPAR*, pages 181–195, 2018.

[217] Joao Marques-Silva and Inês Lynce. Towards Robust CNF Encodings of Cardinality Constraints. In Christian Bessière, editor, *Principles and Practice of Constraint Programming – CP 2007*, Lecture Notes in Computer Science, pages 483–497, Berlin, Heidelberg, 2007. Springer. ISBN 978-3-540-74970-7. doi: 10.1007/978-3-540-74970-7_35.

[218] Mutsunori Banbara, Benjamin Kaufmann, Max Ostrowski, and Torsten Schaub. Clingcon: The next generation. *CoRR*, abs/1705.04569, 2017. URL http://arxiv.org/abs/1705.04569.

[219] Markus W. Covert, Iman Famili, and Bernhard O. Palsson. Identifying constraints that govern cell behavior: a key to converting conceptual to computational models in biology? *Biotechnology and Bioengineering*, 84(7): 763–772, 2003. ISSN 1097-0290. doi: 10.1002/bit.10849. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bit.10849.

[220] L. David and A. Bockmayr. Computing elementary flux modes involving a set of target reactions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(6):1099–1107, 2014. doi: 10.1109/TCBB.2014.2343964.

[221] J. Behre, T. Wilhelm, A. von Kamp, E. Ruppin, and S. Schuster. Structural robustness of metabolic networks with respect to multiple knockouts. *Journal of Theoretical Biology*, 252:433–441, 2008. doi: 10.1016/j.jtbi.2007.09.043.

[222] C. Jungreuthmayer, G. Nair, S. Klamt, and J. Zanghellini. Comparison and improvement of algorithms for computing minimal cut sets. *BMC Bioinformatics*, 14:318, 2013. doi: 10.1186/1471-2105-14-318.

[223] Christian Jungreuthmayer, David E. Ruckerbauer, Matthias P. Gerstl, Michael Hanscho, and Jürgen Zanghellini. Avoiding the Enumeration of Infeasible Elementary Flux Modes by Including Transcriptional Regulatory Rules in the Enumeration Process Saves Computational Costs. *PLOS ONE*, 10(6):1–23, 2015. doi: 10.1371/journal.pone.0129840. URL https://doi.org/10.1371/journal.pone.0129840.

[224] C. T. Trinh, P. Unrean, and F. Srienc. Minimal Escherichia coli Cell for the Most Efficient Production of Ethanol from Hexoses and Pentoses. *Applied and Environmental Microbiology*, 74(12):3634 –3643, 2008. doi: 10.1128/AEM.02708-07.

[225] R. P. Carlson. Metabolic systems cost-benefit analysis for interpreting network structure and regulation.

Bioinformatics, 23(10):1258–1264, March 2007. ISSN 1367-4803. doi: 10.1093/bioinformatics/btm082. URL https://doi.org/10.1093/bioinformatics/btm082.

[226] R. P. Carlson. Decomposition of complex microbial behaviors into resource-based stress responses. *Bioinformatics*, 25(1):90–97, November 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btn589. URL https://doi.org/10.1093/bioinformatics/btn589.

[227] R. P. Carlson and R. L. Taffs. Molecular-level tradeoffs and metabolic adaptation to simultaneous stressors. *Current Opinion In Biotechnology*, 21(5):670–676, October 2010. doi: 10.1016/j.copbio.2010.05.011.

[228] A.E. Beck, K.A. Hunt, H.C. Bernstein, and R.P. Carlson. Chapter 15 - interpreting and designing microbial communities for bioprocess applications, from components to interactions to emergent properties. In Carrie A. Eckert and Cong T. Trinh, editors, *Biotechnology for biofuel production and optimization*, pages 407 – 432. Elsevier, Amsterdam, 2016. ISBN 978-0-444-63475-7. doi: https://doi.org/10.1016/B978-0-444-63475-7.00015-7. URL http://www.sciencedirect.com/science/article/pii/B9780444634757000157.

[229] J. P. Folsom, A. E. Parker, and R. P. Carlson. Physiological and proteomic analysis of escherichia coli iron-limited chemostat growth. *Journal of Bacteriology*, 196(15):2748–2761, 2014. ISSN 0021-9193. doi: 10.1128/JB.01606-14. URL https://jb.asm.org/content/196/15/2748.

[230] A. Provost and G. Bastin. Dynamic metabolic modelling under the balanced growth condition. *Journal of Process Control*, 14(7):717 – 728, 2004. ISSN 0959-1524. doi: 10.1016/j.jprocont.2003.12.004. URL http://www.sciencedirect.com/science/article/pii/S0959152403001392.

[231] Jin Il Kim, Jeffery D. Varner, and Doraiswami Ramkrishna. A hybrid model of anaerobic E. coli GJT001: Combination of elementary flux modes and cybernetic variables. *Biotechnology Progress*, 24(5):993–1006, 2008. doi: 10.1002/btpr.73. URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/btpr.73.

[232] M. R. de Graef, S. Alexeeva, J. L. Snoep, and M. J. Teixeira de Mattos. The steady-state internal redox state (NADH/NAD) reflects the external redox state and is correlated with catabolic adaptation in Escherichia coli. *Journal of Bacteriology*, 181(8):2351–2357, April 1999. doi: 10.1128/jb.181.8.2351-2357.1999.

[233] S. Alexeeva, K. J. Hellingwerf, and M. J. Teixeira de Mattos. Requirement of ArcA for redox regulation in Escherichia coli under microaerobic but not anaerobic or aerobic conditions. *Journal of Bacteriology*, 185(1):204–209, January 2003. doi: 10.1128/jb.185.1.204-209.2003.

[234] S. Peres and V. Fromion. Thermodynamic approaches in flux analysis. In Metabolic Flux Analysis, editor, *Methods in molecular biology*. Springer edition, 2019.

[235] Stefan Schuster and Claus Hilgetag. On elementary flux modes in biochemical reaction systems at steady state. *J. Biol. Syst*, 2:165–182, 1994. doi: 10.1142/s0218339094000131.

[236] Stefan Schuster, Thomas Dandekar, and David A. Fell. Detection of elementary modes in biochemical networks : A promising tool for pathway analysis and metabolic engineering. *Trends in Biotechnology*, 17:53–60, 1999. doi: 10.1016/S0167-7799(98)01290-6.

[237] Jean-Pierre Mazat and Stéphane Ransac. The fate of glutamine in human metabolism. The interplay with glucose in proliferating cells. *Metabolites*, 9(5):81, 2019. doi: 10.3390/metabo9050081.

[238] Jean-Pierre Mazat. One-carbon metabolism in cancer cells: A critical review based on a core model of central metabolism. *Biochemical Society Transactions*, 49(1):1–15, 2021. doi: 10.1042/BST20190008.

[239] Maxime Mahout, Laurent Schwartz, Romain Attal, Ashraf Bakkar, and Sabine Peres. Metabolic modelling links Warburg effect to collagen formation, angiogenesis and inflammation in the tumoural stroma. *In submission*, 2023.

[240] Douglas Hanahan and Robert A. Weinberg. Hallmarks of Cancer: The Next Generation. *Cell*, 144(5):646–674, March 2011. ISSN 0092-8674. doi: 10.1016/j.cell.2011.02.013. URL https://www.sciencedirect.com/science/article/pii/S0092867411001279.

[241] Sylvie Ricard-Blum. The Collagen Family. *Cold Spring Harbor Perspectives in Biology*, 3(1):a004978, January 2011. ISSN , 1943-0264. doi: 10.1101/cshperspect.a004978. URL http://cshperspectives.cshlp.org/content/3/1/a004978.

[242] John A. M Ramshaw, Naina K Shah, and Barbara Brodsky. Gly-X-Y Tripeptide Frequencies in Collagen: A Context for Host–Guest Triple-Helical Peptides. *Journal of Structural Biology*, 122(1):86–91, January 1998. ISSN 1047-8477. doi: 10.1006/jsbi.1998.3977. URL https://www.sciencedirect.com/science/article/pii/S1047847798939776.

[243] Philipp Schneider, Pavlos Stephanos Bekiaris, Axel von Kamp, and Steffen Klamt. StrainDesign: a comprehensive Python package for computational design of metabolic networks. *Bioinformatics*, 38(21):4981–4983, November 2022. ISSN 1367-4803. doi: 10.1093/bioinformatics/btac632. URL https://doi.org/10.1093/bioinformatics/btac632.

[244] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. ISSN 1533-7928. URL http://jmlr.org/papers/v12/pedregosa11a.html.

[245] Wei Liu, Anne Le, Chad Hancock, Andrew N. Lane, Chi V. Dang, Teresa W.-M. Fan, and James M. Phang. Reprogramming of proline and glutamine metabolism contributes to the proliferative and metabolic responses regulated by oncogenic transcription factor c-MYC. *Proceedings of the National Academy of Sciences of the United States of America*, 109(23):8983–8988, June 2012. ISSN 1091-6490. doi: 10.1073/pnas.1203244109.

[246] Erik Sahai, Igor Astsaturov, Edna Cukierman, David G. DeNardo, Mikala Egeblad, Ronald M. Evans, Douglas Fearon, Florian R. Greten, Sunil R. Hingorani, Tony Hunter, Richard O. Hynes, Rakesh K. Jain, Tobias Janowitz, Claus Jorgensen, Alec C. Kimmelman, Mikhail G. Kolonin, Robert G. Maki, R. Scott Powers, Ellen Puré, Daniel C. Ramirez, Ruth Scherz-Shouval, Mara H. Sherman, Sheila Stewart, Thea D. Tlsty, David A. Tu-

veson, Fiona M. Watt, Valerie Weaver, Ashani T. Weeraratna, and Zena Werb. A framework for advancing our understanding of cancer-associated fibroblasts. *Nature Reviews Cancer*, 20(3):174–186, March 2020. ISSN 1474-1768. doi: 10.1038/s41568-019-0238-1. URL https://www.nature.com/articles/s41568-019-0238-1.

[247] Stephanos Pavlides, Diana Whitaker-Menezes, Remedios Castello-Cros, Neal Flomenberg, Agnieszka K. Witkiewicz, Philippe G. Frank, Mathew C. Casimiro, Chenguang Wang, Paolo Fortina, Sankar Addya, Richard G. Pestell, Ubaldo E. Martinez-Outschoorn, Federica Sotgia, and Michael P. Lisanti. The reverse Warburg effect: aerobic glycolysis in cancer associated fibroblasts and the tumor stroma. *Cell Cycle (Georgetown, Tex.)*, 8(23): 3984–4001, December 2009. ISSN 1551-4005. doi: 10.4161/cc.8.23.10238.

[248] Christos Sazeides and Anne Le. Metabolic Relationship Between Cancer-Associated Fibroblasts and Cancer Cells. In *The Heterogeneity of Cancer Metabolism [Internet]. 2nd edition*. Springer, May 2021. doi: 10.1007/978-3-030-65768-0_14. URL https://www.ncbi.nlm.nih.gov/books/NBK573675/.

[249] Thomas N. Seyfried, Roberto E. Flores, Angela M. Poff, and Dominic P. D'Agostino. Cancer as a metabolic disease: implications for novel therapeutics. *Carcinogenesis*, 35(3):515–527, March 2014. ISSN 0143-3334. doi: 10.1093/carcin/bgt480. URL https://doi.org/10.1093/carcin/bgt480.

[250] Robert A. Weinberg. *The Biology of Cancer*. Garland Science, May 2013. ISBN 978-1-317-96346-2.

[251] M R Reynolds, A N Lane, B Robertson, S Kemp, Y Liu, B G Hill, D C Dean, and B F Clem. Control of glutamine metabolism by the tumor suppressor Rb. *Oncogene*, 33(5):556–566, January 2014. ISSN 0950-9232, 1476-5594. doi: 10.1038/onc.2012.635. URL https://www.nature.com/articles/onc2012635.

[252] Anna Halama and Karsten Suhre. Advancing Cancer Treatment by Targeting Glutamine Metabolism-A Roadmap. *Cancers*, 14(3):553, January 2022. ISSN 2072-6694. doi: 10.3390/cancers14030553.

[253] Purna Mukherjee, Zachary M. Augur, Mingyi Li, Collin Hill, Bennett Greenwood, Marek A. Domin, Gramoz Kondakci, Niven R. Narain, Michael A. Kiebish, Roderick T. Bronson, Gabriel Arismendi-Morillo, Christos Chinopoulos, and Thomas N. Seyfried. Therapeutic benefit of combining calorie-restricted ketogenic diet and glutamine targeting in late-stage experimental glioblastoma. *Communications Biology*, 2(1):1–14, May 2019. ISSN 2399-3642. doi: 10.1038/s42003-019-0455-x. URL https://www.nature.com/articles/s42003-019-0455-x.

[254] Robert B. Hamanaka, Erin M. O'Leary, Leah J. Witt, Yufeng Tian, Gizem A. Gökalp, Angelo Y. Meliton, Nickolai O. Dulin, and Gökhan M. Mutlu. Glutamine Metabolism Is Required for Collagen Protein Synthesis in Lung Fibroblasts. *American Journal of Respiratory Cell and Molecular Biology*, 61(5):597–606, November 2019. ISSN 1535-4989. doi: 10.1165/rcmb.2019-0008OC.

[255] Jing Ge, Huachun Cui, Na Xie, Sami Banerjee, Sijia Guo, Shubham Dubey, Stephen Barnes, and Gang Liu. Glutaminolysis Promotes Collagen Translation and Stability via alpha-Ketoglutarate-mediated mTOR Activation and Proline Hydroxylation. *American Journal of Respiratory Cell and Molecular Biology*, 58(3):378–390, March 2018. ISSN 1535-4989. doi: 10.1165/rcmb.2017-0238OC.

[256] Steve Stegen, Kjell Laperre, Guy Eelen, Gianmarco Rinaldi, Peter Fraisl, Sophie Torrekens, Riet Van Looveren, Shauni Loopmans, Geert Bultynck, Stefan Vinckier, Filip Meersman, Patrick H. Maxwell, Jyoti Rai, MaryAnn Weis, David R. Eyre, Bart Ghesquière, Sarah-Maria Fendt, Peter Carmeliet, and Geert Carmeliet. HIF-1alpha metabolically controls collagen synthesis and modification in chondrocytes. *Nature*, 565(7740): 511–515, January 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-0874-3. URL https://www.nature.com/articles/s41586-019-0874-3.

[257] Rogier Braakman and Eric Smith. The compositional and evolutionary logic of metabolism. *Physical Biology*, 10(1):011001, December 2012. ISSN 1478-3975. doi: 2017040521593700. URL https://dx.doi.org/10.1088/1478-3975/10/1/011001.

[258] Michael L. Fredman and Leonid Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21:618–628, 1996. doi: 10.1006/jagm.1996.0062.

[259] Utz-Uwe Haus, Steffen Klamt, and Tamon Stephen. Computing Knock-Out Strategies in Metabolic Networks. *Journal of Computational Biology*, 15(3):259–268, March 2008. doi: 10.1089/cmb.2007.0229. URL 10.1089/cmb.2007.0229.

[260] Maret L. Maliniak, Arlene A. Stecenko, and Nael A. McCarty. A longitudinal analysis of chronic MRSA and Pseudomonas aeruginosa co-infection in cystic fibrosis: A single-center study. *Journal of Cystic Fibrosis*, 15(3): 350–356, May 2016. ISSN 1569-1993. doi: 10.1016/j.jcf.2015.10.014. URL https://www.sciencedirect.com/science/article/pii/S1569199315002581.

[261] Stephanie DeLeon, Allie Clinton, Haley Fowler, Jake Everett, Alexander R. Horswill, and Kendra P. Rumbaugh. Synergistic Interactions of Pseudomonas aeruginosa and Staphylococcus aureus in an In Vitro Wound Model. *Infection and Immunity*, 82(11):4718–4728, November 2014. doi: 10.1128/IAI.02198-14. URL https://journals.asm.org/doi/10.1128/IAI.02198-14.

[262] An Hotterbeekx, Samir Kumar-Singh, Herman Goossens, and Surbhi Malhotra-Kumar. In vivo and In vitro Interactions between Pseudomonas aeruginosa and Staphylococcus spp. *Frontiers in Cellular and Infection Microbiology*, 7, 2017. ISSN 2235-2988. doi: 10.3389/fcimb.2017.00106. URL https://www.frontiersin.org/articles/10.3389/fcimb.2017.00106.

[263] Trevor Dalton, Scot E. Dowd, Randall D. Wolcott, Yan Sun, Chase Watters, John A. Griswold, and Kendra P. Rumbaugh. An In Vivo Polymicrobial Biofilm Wound Infection Model to Study Interspecies Interactions. *PLoS ONE*, 6(11):e27317, November 2011. ISSN 1932-6203. doi: 10.1371/journal.pone.0027317. URL https://dx.plos.org/10.1371/journal.pone.0027317.

[264] Christopher D. Sibley, Kangmin Duan, Carrie Fischer, Michael D. Parkins, Douglas G. Storey, Harvey R. Rabin, and Michael G. Surette. Discerning the Complexity of Community Interactions Using a Drosophila Model of Polymicrobial Infections. *PLoS Pathogens*, 4(10):e1000184, October 2008. ISSN 1553-7374. doi: 10.1371/journal.ppat.1000184. URL https://dx.plos.org/10.1371/journal.ppat.1000184.

[265] Paul W. Woods, Zane M. Haynes, Elin G. Mina, and Cláudia N. H. Marques. Maintenance of S. aureus in Co-culture With P. aeruginosa While Growing as Biofilms. *Frontiers in Microbiology*, 9:3291, 2018. ISSN 1664-302X. doi: 10.3389/fmicb.2018.03291.

[266] Patrícia M Alves, Eida Al-Badi, Cathryn Withycombe, Paul M Jones, Kevin J Purdy, and Sarah E Maddocks. Interaction between Staphylococcus aureus and Pseudomonas aeruginosa is beneficial for colonisation and pathogenicity in a mixed biofilm. *Pathogens and Disease*, 76(1):fty003, February 2018. ISSN 2049-632X. doi: 10.1093/femspd/fty003. URL https://doi.org/10.1093/femspd/fty003.

[267] Poonam Phalak, Jin Chen, Ross P. Carlson, and Michael A. Henson. Metabolic modeling of a chronic wound biofilm consortium predicts spatial partitioning of bacterial species. *BMC Systems Biology*, 10(1): 90, September 2016. ISSN 1752-0509. doi: 10.1186/s12918-016-0334-8. URL https://doi.org/10.1186/s12918-016-0334-8.

[268] Poonam Phalak and Michael A. Henson. Metabolic modelling of chronic wound microbiota predicts mutualistic interactions that drive community composition. *Journal of Applied Microbiology*, 127(5):1576–1593, November 2019. ISSN 1364-5072. doi: 10.1111/jam.14421. URL https://ami.journals.onlinelibrary.wiley.com/doi/full/10.1111/jam.14421.

[269] Yeni P. Yung, S. Lee McGill, Hui Chen, Heejoon Park, Ross P. Carlson, and Luke Hanley. Reverse diauxie phenotype in Pseudomonas aeruginosa biofilm revealed by exometabolomics and label-free proteomics. *NPJ Biofilms and Microbiomes*, 5:31, October 2019. ISSN 2055-5008. doi: 10.1038/s41522-019-0104-7. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6814747/.

[270] Heejoon Park, S. Lee McGill, Adrienne D. Arnold, and Ross P. Carlson. Pseudomonad reverse carbon catabolite repression, interspecies metabolite exchange, and consortial division of labor. *Cellular and Molecular Life Sciences*, 77(3):395–413, February 2020. ISSN 1420-9071. doi: 10.1007/s00018-019-03377-x. URL 10.1007/s00018-019-03377-x.

[271] S. Lee McGill, Yeni Yung, Kristopher A. Hunt, Michael A. Henson, Luke Hanley, and Ross P. Carlson. Pseudomonas aeruginosa reverse diauxie is a multidimensional, optimized, resource utilization strategy. *Scientific Reports*, 11(1):1457, January 2021. ISSN 2045-2322. doi: 10.1038/s41598-020-80522-8. URL http://www.nature.com/articles/s41598-020-80522-8.

[272] Alyssa N. King, Samiksha A. Borkar, David J. Samuels, Zachary Batz, Logan L. Bulock, Marat R. Sadykov, Kenneth W. Bayles, and Shaun R. Brinsmade. Guanine Limitation Results in CodY-Dependent and -Independent Alteration of Staphylococcus aureus Physiology and Gene Expression. *Journal of Bacteriology*, 200(14): e00136–18, June 2018. doi: 10.1128/JB.00136-18. URL https://journals.asm.org/doi/full/10.1128/JB.00136-18.

[273] Matthew A. Oberhardt, Jacek Puchałka, Kimberly E. Fryer, Vítor A. P. Martins dos Santos, and Jason A. Papin. Genome-scale metabolic network analysis of the opportunistic pathogen Pseudomonas aeruginosa

PAO1. *Journal of Bacteriology*, 190(8):2790–2803, April 2008. ISSN 1098-5530. doi: 10.1128/JB.01583-07. URL 10.1128/JB.01583-07.

[274] Maxime Mahout, Ross P. Carlson, Laurent Simon, and Sabine Peres. Logic programming-based Minimal Cut Sets reveal consortium-level therapeutic targets for chronic wound infections. 2023.

[275] M. P Gerstl, C. Jungreuthmayer, and J. Zanghellini. tEFMA: Computing thermodynamically feasible elementary flux modes in metabolic networks. *Bioinformatics*, 31(13):2232–2234, 2015. doi: 10.1093/bioinformatics/btv111.

[276] Edward J. O'Brien, Joshua A. Lerman, Roger L. Chang, Daniel R. Hyduke, and Bernhard Ø Palsson. Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. *Molecular Systems Biology*, 9:693, October 2013. ISSN 1744-4292. doi: 10.1038/msb.2013.52. URL 10.1038/msb.2013.52.

[277] Andrew R. Joyce and Bernhard Ø. Palsson. Predicting Gene Essentiality Using Genome-Scale in Silico Models. In Andrei L. Osterman and Svetlana Y. Gerdes, editors, *Microbial Gene Essentiality: Protocols and Bioinformatics*, Methods in Molecular Biology™, pages 433–457. Humana Press, Totowa, NJ, 2008. ISBN 978-1-59745-321-9. doi: 10.1007/978-1-59745-321-9_30. URL https://doi.org/10.1007/978-1-59745-321-9_30.

[278] Joana C. Xavier, Kiran Raosaheb Patil, and Isabel Rocha. Metabolic models and gene essentiality data reveal essential and conserved metabolism in prokaryotes. *PLoS Computational Biology*, 14(11):e1006556, November 2018. ISSN 1553-734X. doi: 10.1371/journal.pcbi.1006556. URL 10.1371/journal.pcbi.1006556.

[279] Ori Folger, Livnat Jerby, Christian Frezza, Eyal Gottlieb, Eytan Ruppin, and Tomer Shlomi. Predicting selective drug targets in cancer through metabolic networks. *Molecular Systems Biology*, 7(1):501, January 2011. ISSN 1744-4292. doi: 10.1038/msb.2011.35. URL https://www.embopress.org/doi/full/10.1038/msb.2011.35.

[280] Rasmus Agren, Adil Mardinoglu, Anna Asplund, Caroline Kampf, Mathias Uhlen, and Jens Nielsen. Identification of anticancer drugs for hepatocellular carcinoma through personalized genome-scale metabolic modeling. *Molecular Systems Biology*, 10(3):721, March 2014. ISSN 1744-4292. doi: 10.1002/msb.145122.

[281] Anna S. Blazier and Jason A. Papin. Reconciling high-throughput gene essentiality data with metabolic network reconstructions. *PLOS Computational Biology*, 15(4):e1006507, April 2019. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006507. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006507.

[282] Jennifer A. Bartell, Anna S. Blazier, Phillip Yen, Juliane C. Thøgersen, Lars Jelsbak, Joanna B. Goldberg, and Jason A. Papin. Reconstruction of the metabolic network of Pseudomonas aeruginosa to interrogate virulence factor synthesis. *Nature Communications*, 8(1):14631, March 2017. ISSN 2041-1723. doi: 10.1038/ncomms14631. URL 10.1038/ncomms14631.

[283] Thomas C. Eadsforth, Mary Gardiner, Fernando V. Maluf, Stuart McElroy, Daniel James, Julie Frearson, David Gray, and William N. Hunter. Assessment of Pseudomonas aeruginosa N5,N10-Methylenetetrahydrofolate

Dehydrogenase - Cyclohydrolase as a Potential Antibacterial Drug Target. *PLOS ONE*, 7(4):e35973, April 2012. ISSN 1932-6203. doi: 10.1371/journal.pone.0035973. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0035973.

[284] Zoe A. Machan, Graham W. Taylor, Tyrone L. Pitt, Peter J. Cole, and Robert Wilson. 2-Heptyl-4-hydroxyquinoline N-oxide, an antistaphylococcal agent produced by Pseudomonas aeruginosa. *Journal of Antimicrobial Chemotherapy*, 30(5):615–623, November 1992. ISSN 0305-7453. doi: 10.1093/jac/30.5.615. URL https://doi.org/10.1093/jac/30.5.615.

[285] Cortney R. Halsey, Shulei Lei, Jacqueline K. Wax, Mckenzie K. Lehman, Austin S. Nuxoll, Laurey Steinke, Marat Sadykov, Robert Powers, and Paul D. Fey. Amino Acid Catabolism in Staphylococcus aureus and the Function of Carbon Catabolite Repression. *mBio*, 8(1):e01434–16, February 2017. ISSN 2150-7511. doi: 10.1128/mBio.01434-16. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5312079/.

[286] Lincoln W Pasquina, John P Santa Maria, and Suzanne Walker. Teichoic acid biosynthesis as an antibiotic target. *Current Opinion in Microbiology*, 16(5):531–537, October 2013. ISSN 1369-5274. doi: 10.1016/j.mib.2013.06.014. URL https://www.sciencedirect.com/science/article/pii/S1369527413000908.

[287] Zhanwen Li, Lukasz Jaroszewski, Mallika Iyer, Mayya Sedova, and Adam Godzik. FATCAT 2.0: towards a better understanding of the structural diversity of proteins. *Nucleic Acids Research*, 48(W1):W60–W64, July 2020. ISSN 0305-1048. doi: 10.1093/nar/gkaa443. URL https://doi.org/10.1093/nar/gkaa443.

[288] Sylvain Meylan, Caroline B. M. Porter, Jason H. Yang, Peter Belenky, Arnaud Gutierrez, Michael A. Lobritz, Jihye Park, Sun H. Kim, Samuel M. Moskowitz, and James J. Collins. Carbon Sources Tune Antibiotic Susceptibility in Pseudomonas aeruginosa via Tricarboxylic Acid Cycle Control. *Cell Chemical Biology*, 24(2):195–206, February 2017. ISSN 2451-9456, 2451-9448. doi: 10.1016/j.chembiol.2016.12.015. URL https://www.cell.com/cell-chemical-biology/abstract/S2451-9456(16)30476-7.

[289] Sylvain Meylan, Ian W. Andrews, and James J. Collins. Targeting Antibiotic Tolerance, Pathogen by Pathogen. *Cell*, 172(6):1228–1238, March 2018. ISSN 0092-8674, 1097-4172. doi: 10.1016/j.cell.2018.01.037. URL https://www.cell.com/cell/abstract/S0092-8674(18)30114-4.

[290] Kristopher A. Hunt, Ryan M. Jennings, William P. Inskeep, and Ross P. Carlson. Multiscale analysis of autotroph-heterotroph interactions in a high-temperature microbial community. *PLOS Computational Biology*, 14(9):e1006431, September 2018. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006431. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006431.

[291] Kristopher A. Hunt, Ryan M. Jennings, William P. Inskeep, and Ross P. Carlson. Stoichiometric modelling of assimilatory and dissimilatory biomass utilisation in a microbial community. *Environmental Microbiology*, 18(12):4946–4960, 2016. ISSN 1462-2920. doi: 10.1111/1462-2920.13444. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1462-2920.13444.

[292] Nicole Paczia, Anke Nilgen, Tobias Lehmann, Jochem Gätgens, Wolfgang Wiechert, and Stephan Noack.

Extensive exometabolome analysis reveals extended overflow metabolism in various microorganisms. *Microbial Cell Factories*, 11(1):122, September 2012. ISSN 1475-2859. doi: 10.1186/1475-2859-11-122. URL https://doi.org/10.1186/1475-2859-11-122.

[293] Aleksej Zelezniak, Sergej Andrejev, Olga Ponomarova, Daniel R. Mende, Peer Bork, and Kiran Raosaheb Patil. Metabolic dependencies drive species co-occurrence in diverse microbial communities. *Proceedings of the National Academy of Sciences*, 112(20):6449–6454, May 2015. doi: 10.1073/pnas.1421834112. URL https://www.pnas.org/doi/abs/10.1073/pnas.1421834112.

[294] Stefanía Magnúsdóttir, Almut Heinken, Laura Kutt, Dmitry A. Ravcheev, Eugen Bauer, Alberto Noronha, Kacy Greenhalgh, Christian Jäger, Joanna Baginska, Paul Wilmes, Ronan M. T. Fleming, and Ines Thiele. Generation of genome-scale metabolic reconstructions for 773 members of the human gut microbiota. *Nature Biotechnology*, 35(1):81–89, January 2017. ISSN 1546-1696. doi: 10.1038/nbt.3703. URL https://www.nature.com/articles/nbt.3703.

[295] Henrik Reif Andersen. An Introduction to Binary Decision Diagrams. *Lecture notes for Efficient Algorithms and Programs*, 1999.

[296] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1996.tb02080.x. URL https://www.jstor.org/stable/2346178.

[297] M. P. Gerstl, D. E. Ruckerbauer, D. Mattanovich, C. Jungreuthmayer, and J. Zanghellini. Metabolomics integrated elementary flux mode analysis in large metabolic networks. *Scientific Reports*, 5:8930, 2015. doi: 10.1038/srep08930.

Secondary Bibliography

[S1] Jacob Beal, Natalie G. Farny, Traci Haddock-Angelli, Vinoo Selvarajah, Geoff S. Baldwin, Russell Buckley-Taylor, Markus Gershater, Daisuke Kiga, John Marken, Vishal Sanchania, Abigail Sison, and Christopher T. Workman. Robust estimation of bacterial cell count from optical density. *Communications Biology*, 3(1): 1–29, September 2020. ISSN 2399-3642. doi: 10.1038/s42003-020-01127-5. URL https://www.nature.com/articles/s42003-020-01127-5.

[S2] Sunil Nath and John Villadsen. Oxidative phosphorylation revisited. *Biotechnology and Bioengineering*, 112 (3):429–437, 2015. ISSN 1097-0290. doi: 10.1002/bit.25492. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bit.25492.

[S3] Athel Cornish-Bowden. Enzyme specificity: Its meaning in the general case. *Journal of Theoretical Biology*, 108(3):451–457, June 1984. ISSN 0022-5193. doi: 10.1016/S0022-5193(84)80045-4. URL https://www.sciencedirect.com/science/article/pii/S0022519384800454.

[S4] Gregor Mendel. Versuche über Plflanzenhybriden. *Verhandlungen des naturforschenden vereines in brünn*, 4:3–47, 1865.

[S5] Ilona Miko. Gregor Mendel and the Principles of Inheritance. *Nature Education*, page 1(1):134, 2008. URL https://www.nature.com/scitable/topicpage/gregor-mendel-and-the-principles-of-inheritance-593/.

[S6] David Botstein and Neil Risch. Discovering genotypes underlying human phenotypes: past successes for mendelian disease, future approaches for complex disease. *Nature Genetics*, 33(3):228–237, March 2003. ISSN 1546-1718. doi: 10.1038/ng1090. URL https://www.nature.com/articles/ng1090z.

[S7] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, February 2001. ISSN 1476-4687. doi: 10.1038/35057062. URL https://www.nature.com/articles/35057062.

[S8] J. Craig Venter et al. The Sequence of the Human Genome. *Science (New York, N.Y.)*, 291(5507):1304–1351, February 2001. doi: 10.1126/science.1058040. URL https://www.science.org/doi/full/10.1126/science.1058040.

[S9] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A. Albers, Eric Banks, Mark A. DePristo, Robert E.

Handsaker, Gerton Lunter, Gabor T. Marth, Stephen T. Sherry, Gilean McVean, and Richard Durbin. The variant call format and VCFtools. *Bioinformatics*, 27(15):2156–2158, August 2011. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr330. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3137218/.

[S10] Leslie A. Pray. DNA Replication and Causes of Mutation. *Nature Education*, page 1(1):214, 2008. URL https://www.nature.com/scitable/topicpage/dna-replication-and-causes-of-mutation-409/.

[S11] Arlindo L. Oliveira. Biotechnology, Big Data and Artificial Intelligence. *Biotechnology Journal*, 14(8):1800613, 2019. ISSN 1860-7314. doi: 10.1002/biot.201800613. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/biot.201800613.

[S12] Erik Pettersson, Joakim Lundeberg, and Afshin Ahmadian. Generations of sequencing technologies. *Genomics*, 93(2):105–111, February 2009. ISSN 0888-7543. doi: 10.1016/j.ygeno.2008.10.003. URL https://www.sciencedirect.com/science/article/pii/S0888754308002498.

[S13] Michael L. Metzker. Sequencing technologies — the next generation. *Nature Reviews Genetics*, 11(1): 31–46, January 2010. ISSN 1471-0064. doi: 10.1038/nrg2626. URL https://www.nature.com/articles/nrg2626.

[S14] Anton Nekrutenko and James Taylor. Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. *Nature Reviews Genetics*, 13(9):667–672, September 2012. ISSN 1471-0064. doi: 10.1038/nrg3305. URL https://www.nature.com/articles/nrg3305.

[S15] Suleyman Aydin. A short history, principles, and types of ELISA, and our laboratory experience with peptide/protein analyses using ELISA. *Peptides*, 72:4–15, October 2015. ISSN 0196-9781. doi: 10.1016/j.peptides.2015.04.012. URL https://www.sciencedirect.com/science/article/pii/S019697811500131X.

[S16] Jackson O. Lay, Rohana Liyanage, Sabine Borgmann, and Charles L. Wilkins. Problems with the "omics". *TrAC Trends in Analytical Chemistry*, 25(11):1046–1056, December 2006. ISSN 0165-9936. doi: 10.1016/j.trac.2006.10.007. URL https://www.sciencedirect.com/science/article/pii/S0165993606002329.

[S17] Albert Hsiao and Michael D. Kuo. High-throughput Biology in the Postgenomic Era. *Journal of Vascular and Interventional Radiology*, 20(7, Supplement):S488–S496, July 2009. ISSN 1051-0443. doi: 10.1016/j.jvir.2009.04.040. URL https://www.sciencedirect.com/science/article/pii/S1051044309003534.

[S18] Vivien Marx. The big challenges of big data. *Nature*, 498(7453):255–260, June 2013. ISSN 1476-4687. doi: 10.1038/498255a. URL https://www.nature.com/articles/498255a.

[S19] John P. A. Ioannidis and Muin J. Khoury. Improving Validation Practices in "Omics" Research. *Science*, 334 (6060):1230–1232, December 2011. doi: 10.1126/science.1211811. URL https://www.science.org/doi/full/10.1126/science.1211811.

[S20] Adam McDermaid, Brandon Monier, Jing Zhao, Bingqiang Liu, and Qin Ma. Interpretation of differential gene expression results of RNA-seq data: review and integration. *Briefings in Bioinformatics*, 20(6):2044–2054, November 2019. ISSN 1477-4054. doi: 10.1093/bib/bby067. URL https://doi.org/10.1093/bib/bby067.

[S21] Sandra Maaß and Dörte Becher. Methods and applications of absolute protein quantification in microbial systems. *Journal of Proteomics*, 136:222–233, March 2016. ISSN 1874-3919. doi: 10.1016/j.jprot.2016.01.015. URL https://www.sciencedirect.com/science/article/pii/S1874391916300185.

[S22] Yuehan Feng, Valentina Cappelletti, and Paola Picotti. Quantitative proteomics of model organisms. *Current Opinion in Systems Biology*, 6:58–66, December 2017. ISSN 2452-3100. doi: 10.1016/j.coisb.2017.09.004. URL https://www.sciencedirect.com/science/article/pii/S2452310017300926.

[S23] Francisco Calderón-Celis, Jorge Ruiz Encinar, and Alfredo Sanz-Medel. Standardization approaches in absolute quantitative proteomics with mass spectrometry. *Mass Spectrometry Reviews*, 37(6):715–737, 2018. ISSN 1098-2787. doi: 10.1002/mas.21542. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/mas.21542.

[S24] Katja Dettmer and Bruce D Hammock. Metabolomics–a new exciting field within the "omics" sciences. *Environmental Health Perspectives*, 112(7):A396–A397, May 2004. ISSN 0091-6765. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1241997/.

[S25] Maroun Bou Khalil, Weimin Hou, Hu Zhou, Fred Elisma, Leigh Anne Swayne, Alexandre P. Blanchard, Zemin Yao, Steffany A.L. Bennett, and Daniel Figeys. Lipidomics era: Accomplishments and challenges. *Mass Spectrometry Reviews*, 29(6):877–929, 2010. ISSN 1098-2787. doi: 10.1002/mas.20294. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/mas.20294.

[S26] Séverine Duvaud, Chiara Gabella, Frédérique Lisacek, Heinz Stockinger, Vassilios Ioannidis, and Christine Durinx. Expasy, the Swiss Bioinformatics Resource Portal, as designed by its users. *Nucleic Acids Research*, 49(W1):W216–W227, July 2021. ISSN 0305-1048. doi: 10.1093/nar/gkab225. URL https://doi.org/10.1093/nar/gkab225.

[S27] Charles E Cook, Mary T Bergman, Guy Cochrane, Rolf Apweiler, and Ewan Birney. The European Bioinformatics Institute in 2017: data coordination and integration. *Nucleic Acids Research*, 46(D1):D21–D29, January 2018. ISSN 0305-1048. doi: 10.1093/nar/gkx1154. URL https://doi.org/10.1093/nar/gkx1154.

[S28] Stephan Fuchs, Henry Mehlan, Jörg Bernhardt, André Hennig, Stephan Michalik, Kristin Surmann, Jan Pané-Farré, Anne Giese, Stefan Weiss, Linus Backert, Alexander Herbig, Kay Nieselt, Michael Hecker, Uwe Völker, and Ulrike Mäder. AureoWiki - The repository of the Staphylococcus aureus research and annotation community. *International Journal of Medical Microbiology*, 308(6):558–568, August 2018. ISSN 1438-4221. doi: 10.1016/j.ijmm.2017.11.011. URL https://www.sciencedirect.com/science/article/pii/S1438422117304629.

[S29] Liam D. H. Elbourne, Sasha G. Tetu, Karl A. Hassan, and Ian T. Paulsen. TransportDB 2.0: a database for exploring membrane transporters in sequenced genomes from all domains of life. *Nucleic Acids Research*, 45(D1):D320–D324, January 2017. ISSN 0305-1048. doi: 10.1093/nar/gkw1068. URL https://doi.org/10.1093/nar/gkw1068.

[S30] Stephane De Cesco, John B. Davis, and Paul E. Brennan. TargetDB: A target information aggregation tool and tractability predictor. *PLOS ONE*, 15(9):e0232644, September 2020. ISSN 1932-6203. doi: 10.1371/journal.pone.0232644. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0232644.

[S31] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000. ISSN 1546-1718. doi: 10.1038/75556. URL https://www.nature.com/articles/ng0500_25.

[S32] The Gene Ontology Consortium. The Gene Ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031, May 2023. ISSN 1943-2631. doi: 10.1093/genetics/iyad031. URL https://doi.org/10.1093/genetics/iyad031.

[S33] Jim Thurmond, Joshua L Goodman, Victor B Strelets, Helen Attrill, L Sian Gramates, Steven J Marygold, Beverley B Matthews, Gillian Millburn, Giulia Antonazzo, Vitor Trovisco, Thomas C Kaufman, Brian R Calvi, and the FlyBase Consortium. FlyBase 2.0: the next generation. *Nucleic Acids Research*, 47(D1):D759–D765, January 2019. ISSN 0305-1048. doi: 10.1093/nar/gky1003. URL https://doi.org/10.1093/nar/gky1003.

[S34] Beatrice Amos, Cristina Aurrecoechea, Matthieu Barba, Ana Barreto, Evelina Y Basenko, Wojciech Bażant, Robert Belnap, Ann S Blevins, Ulrike Böhme, John Brestelli, Brian P Brunk, Mark Caddick, Danielle Callan, Lahcen Campbell, Mikkel B Christensen, George K Christophides, Kathryn Crouch, Kristina Davis, Jeremy DeBarry, Ryan Doherty, Yikun Duan, Michael Dunn, Dave Falke, Steve Fisher, Paul Flicek, Brett Fox, Bindu Gajria, Gloria I Giraldo-Calderón, Omar S Harb, Elizabeth Harper, Christiane Hertz-Fowler, Mark J Hickman, Connor Howington, Sufen Hu, Jay Humphrey, John Iodice, Andrew Jones, John Judkins, Sarah A Kelly, Jessica C Kissinger, Dae Kun Kwon, Kristopher Lamoureux, Daniel Lawson, Wei Li, Kallie Lies, Disha Lodha, Jamie Long, Robert M MacCallum, Gareth Maslen, Mary Ann McDowell, Jaroslaw Nabrzyski, David S Roos, Samuel S C Rund, Stephanie Wever Schulman, Achchuthan Shanmugasundram, Vasily Sitnik, Drew Spruill, David Starns, Christian J Stoeckert, Jr, Sheena Shah Tomko, Haiming Wang, Susanne Warrenfeltz, Robert Wieck, Paul A Wilkinson, Lin Xu, and Jie Zheng. VEuPathDB: the eukaryotic pathogen, vector and host bioinformatics resource center. *Nucleic Acids Research*, 50(D1):D898–D911, January 2022. ISSN 0305-1048. doi: 10.1093/nar/gkab929. URL https://doi.org/10.1093/nar/gkab929.

[S35] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. PubChem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1):D1102–D1109, January 2019. ISSN 0305-1048. doi: 10.1093/nar/gky1033. URL https://doi.org/10.1093/nar/gky1033.

[S36] Kirill Degtyarenko, Paula de Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael

Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Research*, 36(suppl_1):D344–D350, January 2008. ISSN 0305-1048. doi: 10.1093/nar/gkm791. URL https://doi.org/10.1093/nar/gkm791.

[S37] Parit Bansal, Anne Morgat, Kristian B Axelsen, Venkatesh Muthukrishnan, Elisabeth Coudert, Lucila Aimo, Nevila Hyka-Nouspikel, Elisabeth Gasteiger, Arnaud Kerhornou, Teresa Batista Neto, Monica Pozzato, Marie-Claude Blatter, Alex Ignatchenko, Nicole Redaschi, and Alan Bridge. Rhea, the reaction knowledgebase in 2022. *Nucleic Acids Research*, 50(D1):D693–D700, January 2022. ISSN 0305-1048. doi: 10.1093/nar/gkab1016. URL https://doi.org/10.1093/nar/gkab1016.

[S38] Sébastien Moretti, Van Du T Tran, Florence Mehl, Mark Ibberson, and Marco Pagni. MetaNetX/MNXref: unified namespace for metabolites and biochemical reactions in the context of metabolic models. *Nucleic Acids Research*, 49(D1):D570–D574, January 2021. ISSN 0305-1048. doi: 10.1093/nar/gkaa992. URL https://doi.org/10.1093/nar/gkaa992.

[S39] Ron Caspi, Tomer Altman, Richard Billington, Kate Dreher, Hartmut Foerster, Carol A. Fulcher, Timothy A. Holland, Ingrid M. Keseler, Anamika Kothari, Aya Kubo, Markus Krummenacker, Mario Latendresse, Lukas A. Mueller, Quang Ong, Suzanne Paley, Pallavi Subhraveti, Daniel S. Weaver, Deepika Weerasinghe, Peifen Zhang, and Peter D. Karp. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Research*, 42(D1):D459–D471, January 2014. ISSN 0305-1048. doi: 10.1093/nar/gkt1103. URL https://doi.org/10.1093/nar/gkt1103.

[S40] Jaina Mistry, Sara Chuguransky, Lowri Williams, Matloob Qureshi, Gustavo A Salazar, Erik L L Sonnhammer, Silvio C E Tosatto, Lisanna Paladin, Shriya Raj, Lorna J Richardson, Robert D Finn, and Alex Bateman. Pfam: The protein families database in 2021. *Nucleic Acids Research*, 49(D1):D412–D419, January 2021. ISSN 0305-1048. doi: 10.1093/nar/gkaa913. URL https://doi.org/10.1093/nar/gkaa913.

[S41] Typhaine Paysan-Lafosse, Matthias Blum, Sara Chuguransky, Tiago Grego, Beatriz Lázaro Pinto, Gustavo A Salazar, Maxwell L Bileschi, Peer Bork, Alan Bridge, Lucy Colwell, Julian Gough, Daniel H Haft, Ivica Letunić, Aron Marchler-Bauer, Huaiyu Mi, Darren A Natale, Christine A Orengo, Arun P Pandurangan, Catherine Rivoire, Christian J A Sigrist, Ian Sillitoe, Narmada Thanki, Paul D Thomas, Silvio C E Tosatto, Cathy H Wu, and Alex Bateman. InterPro in 2022. *Nucleic Acids Research*, 51(D1):D418–D427, January 2023. ISSN 0305-1048. doi: 10.1093/nar/gkac993. URL https://doi.org/10.1093/nar/gkac993.

[S42] Christian von Mering, Martijn Huynen, Daniel Jaeggi, Steffen Schmidt, Peer Bork, and Berend Snel. STRING: a database of predicted functional associations between proteins. *Nucleic Acids Research*, 31 (1):258–261, January 2003. ISSN 0305-1048. doi: 10.1093/nar/gkg034. URL https://doi.org/10.1093/nar/gkg034.

[S43] Dana Silverbush and Roded Sharan. A systematic approach to orient the human protein–protein interaction network. *Nature Communications*, 10(1):3015, July 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-10887-6. URL https://www.nature.com/articles/s41467-019-10887-6.

[S44] Michael J Buck and Jason D Lieb. ChIP-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments. *Genomics*, 83(3):349–360, March 2004. ISSN 0888-7543. doi: 10.1016/j.ygeno.2003.11.004. URL https://www.sciencedirect.com/science/article/pii/S0888754303003628.

[S45] C. Reder. Metabolic control theory: a structural approach. *Journal of Theoretical Biology*, 135(2):175–201, November 1988. ISSN 0022-5193. doi: 10.1016/s0022-5193(88)80073-0.

[S46] Reinhart Heinrich and Tom A. Rapoport. A Linear Steady-State Treatment of Enzymatic Chains. *European Journal of Biochemistry*, 42(1):89–95, 1974. ISSN 1432-1033. doi: 10.1111/j.1432-1033.1974.tb03318.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1432-1033.1974.tb03318.x.

[S47] A K Groen, R J Wanders, H V Westerhoff, R van der Meer, and J M Tager. Quantification of the contribution of various steps to the control of mitochondrial respiration. *Journal of Biological Chemistry*, 257(6):2754–2757, March 1982. ISSN 0021-9258. doi: 10.1016/S0021-9258(19)81026-8. URL https://www.sciencedirect.com/science/article/pii/S0021925819810268.

[S48] Barbara M Bakker, Hans V Westerhoff, Fred R Opperdoes, and Paul A. M Michels. Metabolic control analysis of glycolysis in trypanosomes as an approach to improve selectivity and effectiveness of drugs. *Molecular and Biochemical Parasitology*, 106(1):1–10, February 2000. ISSN 0166-6851. doi: 10.1016/S0166-6851(99)00197-8. URL https://www.sciencedirect.com/science/article/pii/S0166685199001978.

[S49] Barbara M. Bakker, Hans V. Westerhoff, and Paul A. M. Michels. Regulation and control of compartmentalized glycolysis in bloodstream form Trypanosoma brucei. *Journal of Bioenergetics and Biomembranes*, 27 (5):513–525, October 1995. ISSN 1573-6881. doi: 10.1007/BF02110191. URL https://doi.org/10.1007/BF02110191.

[S50] Barbara M. Bakker, Paul A. M. Michels, Fred R. Opperdoes, and Hans V. Westerhoff. What Controls Glycolysis in Bloodstream Form Trypanosoma brucei? *. *Journal of Biological Chemistry*, 274(21): 14551–14559, May 1999. ISSN 0021-9258, 1083-351X. doi: 10.1074/jbc.274.21.14551. URL https://www.jbc.org/article/S0021-9258(19)73126-3/abstract.

[S51] Jorrit J. Hornberg, Frank J. Bruggeman, Hans V. Westerhoff, and Jan Lankelma. Cancer: A Systems Biology disease. *Biosystems*, 83(2):81–90, February 2006. ISSN 0303-2647. doi: 10.1016/j.biosystems.2005.05.014. URL https://www.sciencedirect.com/science/article/pii/S0303264705001176.

[S52] Marta Cascante, Laszlo G. Boros, Begoña Comin-Anduix, Pedro de Atauri, Josep J. Centelles, and Paul W.-N. Lee. Metabolic control analysis in drug discovery and disease. *Nature Biotechnology*, 20(3):243–249, March 2002. ISSN 1087-0156. doi: 10.1038/nbt0302-243.

[S53] Barbara M Bakker, Heike E Assmus, Frank Bruggeman, Jurgen R Haanstra, Edda Klipp, and Hans Westerhoff. Network-based selectivity of antiparasitic inhibitors. *Molecular biology reports*, 29(1-2):1–5, 2002. doi: 10.1023/A:1020397513646.

[S54] Kyeong Rok Choi and Sang Yup Lee. Systems metabolic engineering of microorganisms for food and cosmetics production. *Nature Reviews Bioengineering*, pages 1–26, June 2023. ISSN 2731-6092. doi: 10.1038/s44222-023-00076-y. URL https://www.nature.com/articles/s44222-023-00076-y.

[S55] Kyeong Rok Choi, Woo Dae Jang, Dongsoo Yang, Jae Sung Cho, Dahyeon Park, and Sang Yup Lee. Systems Metabolic Engineering Strategies: Integrating Systems and Synthetic Biology with Metabolic Engineering. *Trends in Biotechnology*, 37(8):817–837, August 2019. ISSN 0167-7799, 1879-3096. doi: 10.1016/j.tibtech.2019.01.003. URL https://www.cell.com/trends/biotechnology/abstract/S0167-7799(19)30003-4.

[S56] Vasiliy A Portnoy, Daniela Bezdan, and Karsten Zengler. Adaptive laboratory evolution—harnessing the power of biology for metabolic engineering. *Current Opinion in Biotechnology*, 22(4):590–594, August 2011. ISSN 0958-1669. doi: 10.1016/j.copbio.2011.03.007. URL https://www.sciencedirect.com/science/article/pii/S0958166911000590.

[S57] Troy E. Sandberg, Michael J. Salazar, Liam L. Weng, Bernhard O. Palsson, and Adam M. Feist. The emergence of adaptive laboratory evolution as an efficient tool for biological discovery and industrial biotechnology. *Metabolic Engineering*, 56:1–16, December 2019. ISSN 1096-7176. doi: 10.1016/j.ymben.2019.08.004. URL https://www.sciencedirect.com/science/article/pii/S1096717619301533.

[S58] Lily A. Chylek, Leonard A. Harris, Chang-Shung Tung, James R. Faeder, Carlos F. Lopez, and William S. Hlavacek. Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems. *WIREs Systems Biology and Medicine*, 6(1):13–36, 2014. ISSN 1939-005X. doi: 10.1002/wsbm.1245. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wsbm.1245.

[S59] Matthieu Bouguéon, Pierre Boutillier, Jérôme Feret, Octave Hazard, and Nathalie Théret. The Rule-Based Model Approach: A Kappa Model for Hepatic Stellate Cells Activation by TGFB1. In *Systems Biology Modelling and Analysis*, pages 69–126. John Wiley & Sons, Ltd, 2022. ISBN 978-1-119-71660-0. doi: 10.1002/9781119716600.ch4. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119716600.ch4.

[S60] Rui-Sheng Wang, Assieh Saadatpour, and Réka Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Physical biology*, 9(5):055001–055001, 2012. ISSN 1478-3975. doi: 10.1088/1478-3975/9/5/055001.

[S61] Stéphanie Chevalier, Vincent Noël, Laurence Calzone, Andrei Zinovyev, and Loïc Paulevé. Synthesis and Simulation of Ensembles of Boolean Networks for Cell Fate Decision. In Alessandro Abate, Tatjana Petrov, and Verena Wolf, editors, *Computational Methods in Systems Biology*, Lecture Notes in Computer Science, pages 193–209, Cham, 2020. Springer International Publishing. ISBN 978-3-030-60327-4. doi: 10.1007/978-3-030-60327-4_11.

[S62] Petronela Buiga and Jean-Marc Schwartz. Opportunities and Challenges Provided by Boolean Modelling of Cancer Signalling Pathways. In Fabricio Alves Barbosa da Silva, Nicolas Carels, Marcelo Trindade dos Santos, and Francisco José Pereira Lopes, editors, *Networks in Systems Biology: Applications for Disease Mod-*

eling, Computational Biology, pages 199–216. Springer International Publishing, Cham, 2020. ISBN 978-3-030-51862-2. doi: 10.1007/978-3-030-51862-2_9. URL https://doi.org/10.1007/978-3-030-51862-2_9.

[S63] Benjamin A. Hall and Anna Niarakis. Data integration in logic-based models of biological mechanisms. *Current Opinion in Systems Biology*, 28:100386, December 2021. ISSN 2452-3100. doi: 10.1016/j.coisb.2021.100386. URL https://www.sciencedirect.com/science/article/pii/S2452310021000809.

[S64] Nicolas Le Novère. Quantitative and logic modelling of molecular and gene networks. *Nature Reviews Genetics*, 16(3):146–158, March 2015. ISSN 1471-0064. doi: 10.1038/nrg3885. URL https://www.nature.com/articles/nrg3885.

[S65] Karen Eilbeck, Suzanna E. Lewis, Christopher J. Mungall, Mark Yandell, Lincoln Stein, Richard Durbin, and Michael Ashburner. The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biology*, 6(5):R44, April 2005. ISSN 1474-760X. doi: 10.1186/gb-2005-6-5-r44. URL https://doi.org/10.1186/gb-2005-6-5-r44.

[S66] Jon Ison, Matúš Kalaš, Inge Jonassen, Dan Bolser, Mahmut Uludag, Hamish McWilliam, James Malone, Rodrigo Lopez, Steve Pettifer, and Peter Rice. EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, 29(10):1325–1332, May 2013. ISSN 1367-4803. doi: 10.1093/bioinformatics/btt113. URL https://doi.org/10.1093/bioinformatics/btt113.

[S67] Michelle L. Wynn, Nikita Consul, Sofia D. Merajver, and Santiago Schnell. Logic-based models in systems biology: a predictive and parameter-free network analysis method. *Integrative Biology: Quantitative Biosciences from Nano to Macro*, 4(11):1323–1337, November 2012. ISSN 1757-9708. doi: 10.1039/c2ib20193c.

[S68] Christopher Rackauckas and Qing Nie. DifferentialEquations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, 5(1), 2017. doi: 10.5334/jors.151.

[S69] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017. doi: 10.1137/141000671. URL https://doi.org/10.1137/141000671.

[S70] SciPy 1.0 Contributors. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2. URL https://doi.org/10.1038/s41592-019-0686-2.

[S71] Daniel T. Gillespie. A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *Journal of Computational Physics*, 22:403–434, December 1976. ISSN 0021-9991. doi: 10.1016/0021-9991(76)90041-3. URL https://ui.adsabs.harvard.edu/abs/1976JCoPh..22..403G.

[S72] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, December 1977. ISSN 0022-3654. doi: 10.1021/j100540a008. URL https://doi.org/10.1021/j100540a008.

[S73] Ron Milo, Paul Jorgensen, Uri Moran, Griffin Weber, and Michael Springer. BioNumbers—the database of key numbers in molecular and cell biology. *Nucleic Acids Research*, 38(Database issue):D750–D753, January 2010. ISSN 0305-1048. doi: 10.1093/nar/gkp889. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2808940/.

[S74] Cecile Moulin, Laurent Tournier, and Sabine Peres. Using a Hybrid Approach to Model Central Carbon Metabolism Across the Cell Cycle. In Milan Češka and Nicola Paoletti, editors, *Hybrid Systems Biology*, Lecture Notes in Computer Science, pages 132–146, Cham, 2019. Springer International Publishing. ISBN 978-3-030-28042-0. doi: 10.1007/978-3-030-28042-0_9.

[S75] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, UK ; New York, 2004. ISBN 978-0-521-83378-3.

[S76] Hoang Tuy. Bilevel Linear Programming, Multiobjective Programming, And Monotonic Reverse Convex Programming. In *Multilevel Optimization: Algorithms and Applications*, volume 20 of *Nonconvex Optimization and Its Applications*. Springer US, Boston, MA, 1998. ISBN 978-1-4613-7989-8 978-1-4613-0307-7. doi: 10.1007/978-1-4613-0307-7. URL http://link.springer.com/10.1007/978-1-4613-0307-7.

[S77] H. W. Kuhn and A. W. Tucker. Nonlinear Programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, volume 2, pages 481–493. University of California Press, January 1951.

[S78] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[S79] Paolo Baldan, Nicoletta Cocco, Andrea Marin, and Marta Simeoni. Petri nets for modelling metabolic pathways: a survey. *Natural Computing*, 9(4):955–989, December 2010. ISSN 1572-9796. doi: 10.1007/s11047-010-9180-6. URL https://doi.org/10.1007/s11047-010-9180-6.

[S80] Georgios A. Pavlopoulos, Maria Secrier, Charalampos N. Moschopoulos, Theodoros G. Soldatos, Sophia Kossida, Jan Aerts, Reinhard Schneider, and Pantelis G. Bagos. Using graph theory to analyze biological networks. *BioData Mining*, 4(1):10, April 2011. ISSN 1756-0381. doi: 10.1186/1756-0381-4-10. URL https://doi.org/10.1186/1756-0381-4-10.

[S81] Ümit V. Çatalyürek and Cevdet Aykanat. Patoh (partitioning tool for hypergraphs). In *Encyclopedia of parallel computing*, pages 1479–1487. Springer, 2011.

[S82] Umit V. Catalyurek and Cevdet Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on parallel and distributed systems*, 10(7):673–693, 1999. doi: 10.1109/71.780863.

[S83] Laurent Heirendt, Ines Thiele, and Ronan M T Fleming. DistributedFBA.jl: high-level, high-performance flux balance analysis in Julia. *Bioinformatics*, 33(9):1421–1423, May 2017. ISSN 1367-4803, 1367-4811. doi:

10.1093/bioinformatics/btw838. URL https://academic.oup.com/bioinformatics/article/33/9/1421/2908434.

[S84] Hao Wang, Simonas Marcišauskas, Benjamín J. Sánchez, Iván Domenzain, Daniel Hermansson, Rasmus Agren, Jens Nielsen, and Eduard J. Kerkhoven. RAVEN 2.0: A versatile toolbox for metabolic network reconstruction and a case study on Streptomyces coelicolor. *PLOS Computational Biology*, 14(10):e1006541, October 2018. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006541. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006541.

[S85] Kristian Jensen, Joao Cardoso, and Nikolaus Sonnenschein. Optlang: An algebraic modeling language for mathematical optimization. *The Journal of Open Source Software*, 2016. ISSN 2475-9066. doi: 10.21105/joss.00139.

[S86] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018. doi: 10.1080/23307706.2017.1397554.

[S87] João Capela, Davide Lagoa, Ruben Rodrigues, Emanuel Cunha, Fernando Cruz, Ana Barbosa, José Bastos, Diogo Lima, Eugénio C Ferreira, Miguel Rocha, and Oscar Dias. merlin, an improved framework for the reconstruction of high-quality genome-scale metabolic models. *Nucleic Acids Research*, page gkac459, June 2022. ISSN 0305-1048. doi: 10.1093/nar/gkac459. URL https://doi.org/10.1093/nar/gkac459.

[S88] Vítor Pereira, Fernando Cruz, and Miguel Rocha. MEWpy: a computational strain optimization workbench in Python. *Bioinformatics*, 37(16):2494–2496, August 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btab013. URL https://doi.org/10.1093/bioinformatics/btab013.

[S89] Brett Olivier, willigott, Douwe Molenaar, and Bas Teusink. CBMPy release 0.8.4, February 2023. URL https://zenodo.org/record/7679232.

[S90] Stuart M. Harwood, Kai Höffner, and Paul I. Barton. Efficient solution of ordinary differential equations with a parametric lexicographic linear program embedded. *Numerische Mathematik*, 133(4):623–653, August 2016. ISSN 0945-3245. doi: 10.1007/s00211-015-0760-3. URL https://doi.org/10.1007/s00211-015-0760-3.

[S91] David S. Tourigny, Jorge Carrasco Muriel, and Moritz E. Beber. dfba: Software for efficient simulation of dynamic flux-balance analysis models in Python. *Journal of Open Source Software*, 5(52):2342, August 2020. ISSN 2475-9066. doi: 10.21105/joss.02342. URL https://joss.theoj.org/papers/10.21105/joss.02342.

[S92] Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. COPASI—a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074, December 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl485. URL https://doi.org/10.1093/bioinformatics/btl485.

[S93] Jennifer L. Reed, Thuy D. Vo, Christophe H. Schilling, and Bernhard O. Palsson. An expanded genome-scale model of Escherichia coli K-12 (iJR904 GSM/GPR). *Genome Biology*, 4(9):R54, August 2003. ISSN 1474-760X. doi: 10.1186/gb-2003-4-9-r54. URL https://doi.org/10.1186/gb-2003-4-9-r54.

[S94] Jochen Förster, Iman Famili, Patrick Fu, Bernhard Ø Palsson, and Jens Nielsen. Genome-Scale Reconstruction of the Saccharomyces cerevisiae Metabolic Network. *Genome Research*, 13(2):244–253, February 2003. ISSN 1088-9051, 1549-5469. doi: 10.1101/gr.234503. URL https://genome.cshlp.org/content/13/2/244.

[S95] Natalie C. Duarte, Markus J. Herrgård, and Bernhard Ø Palsson. Reconstruction and Validation of Saccharomyces cerevisiae iND750, a Fully Compartmentalized Genome-Scale Metabolic Model. *Genome Research*, 14(7):1298–1309, July 2004. ISSN 1088-9051, 1549-5469. doi: 10.1101/gr.2250904. URL https://genome.cshlp.org/content/14/7/1298.

[S96] Encyclopedia of Mathematics. Monotone Boolean function - Encyclopedia of Mathematics, September 2023. URL https://encyclopediaofmath.org/wiki/Monotone_Boolean_function.

[S97] Sally A. Woods, Steven D. Schwartzbach, and John R. Guest. Two biochemically distinct classes of fumarase in Escherichia coli. *Biochimica et Biophysica Acta (BBA) - Protein Structure and Molecular Enzymology*, 954:14–26, January 1988. ISSN 0167-4838. doi: 10.1016/0167-4838(88)90050-7. URL https://www.sciencedirect.com/science/article/pii/0167483888900507.

[S98] Colton J. Lloyd, Ali Ebrahim, Laurence Yang, Zachary A. King, Edward Catoiu, Edward J. O'Brien, Joanne K. Liu, and Bernhard O. Palsson. COBRAme: A computational framework for genome-scale models of metabolism and gene expression. *PLOS Computational Biology*, 14(7):e1006302, July 2018. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006302. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006302.

[S99] Eduard J Kerkhoven. Advances in constraint-based models: methods for improved predictive power based on resource allocation constraints. *Current Opinion in Microbiology*, 68:102168, August 2022. ISSN 1369-5274. doi: 10.1016/j.mib.2022.102168. URL https://www.sciencedirect.com/science/article/pii/S1369527422000522.

[S100] Lin Liu and Alexander Bockmayr. Regulatory dynamic enzyme-cost flux balance analysis: A unifying framework for constraint-based modeling. *Journal of Theoretical Biology*, 501:110317, September 2020. ISSN 0022-5193. doi: 10.1016/j.jtbi.2020.110317. URL https://www.sciencedirect.com/science/article/pii/S0022519320301727.

[S101] Akira Funahashi, Yukiko Matsuoka, Akiya Jouraku, Mineo Morohashi, Norihiro Kikuchi, and Hiroaki Kitano. CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks. *Proceedings of the IEEE*, 96(8):1254–1265, August 2008. ISSN 1558-2256. doi: 10.1109/JPROC.2008.925458.

[S102] Joost Boele, Brett G. Olivier, and Bas Teusink. FAME, the Flux Analysis and Modeling Environment. *BMC*

Systems Biology, 6(1):8, January 2012. ISSN 1752-0509. doi: 10.1186/1752-0509-6-8. URL https://doi.org/10.1186/1752-0509-6-8.

[S103] Tim Daniel Rose and Jean-Pierre Mazat. FluxVisualizer, a Software to Visualize Fluxes through Metabolic Networks. *Processes*, 6(5):39, May 2018. ISSN 2227-9717. doi: 10.3390/pr6050039. URL https://www.mdpi.com/2227-9717/6/5/39.

[S104] Arnaud Belcour, Clémence Frioux, Méziane Aite, Anthony Bretaudeau, Falk Hildebrand, and Anne Siegel. Metage2Metabo, microbiota-scale metabolic complementarity for the identification of key species. *eLife*, 9: e61968, December 2020. ISSN 2050-084X. doi: 10.7554/eLife.61968. URL https://doi.org/10.7554/eLife.61968.

[S105] Sjoerd Opdam, Anne Richelle, Benjamin Kellman, Shanzhong Li, Daniel C Zielinski, and Nathan E Lewis. A systematic evaluation of methods for tailoring genome-scale metabolic models. *Cell systems*, 4(3):318–329, 2017. doi: 10.1016/j.cels.2017.01.010.

[S106] Brian J. Schmidt, Ali Ebrahim, Thomas O. Metz, Joshua N. Adkins, Bernhard Ø. Palsson, and Daniel R. Hyduke. GIM3E: condition-specific models of cellular metabolism developed from metabolomics and expression data. *Bioinformatics*, 29(22):2900–2908, November 2013. ISSN 1367-4803. doi: 10.1093/bioinformatics/btt493. URL https://doi.org/10.1093/bioinformatics/btt493.

[S107] Scott A. Becker and Bernhard O. Palsson. Context-Specific Metabolic Networks Are Consistent with Experiments. *PLOS Computational Biology*, 4(5):e1000082, May 2008. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000082. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000082.

[S108] Adam P. Arkin, Robert W. Cottingham, Christopher S. Henry, Nomi L. Harris, Rick L. Stevens, Sergei Maslov, Paramvir Dehal, Doreen Ware, Fernando Perez, Shane Canon, Michael W. Sneddon, Matthew L. Henderson, William J. Riehl, Dan Murphy-Olson, Stephen Y. Chan, Roy T. Kamimura, Sunita Kumari, Meghan M. Drake, Thomas S. Brettin, Elizabeth M. Glass, Dylan Chivian, Dan Gunter, David J. Weston, Benjamin H. Allen, Jason Baumohl, Aaron A. Best, Ben Bowen, Steven E. Brenner, Christopher C. Bun, John-Marc Chandonia, Jer-Ming Chia, Ric Colasanti, Neal Conrad, James J. Davis, Brian H. Davison, Matthew DeJongh, Scott Devoid, Emily Dietrich, Inna Dubchak, Janaka N. Edirisinghe, Gang Fang, José P. Faria, Paul M. Frybarger, Wolfgang Gerlach, Mark Gerstein, Annette Greiner, James Gurtowski, Holly L. Haun, Fei He, Rashmi Jain, Marcin P. Joachimiak, Kevin P. Keegan, Shinnosuke Kondo, Vivek Kumar, Miriam L. Land, Folker Meyer, Marissa Mills, Pavel S. Novichkov, Taeyun Oh, Gary J. Olsen, Robert Olson, Bruce Parrello, Shiran Pasternak, Erik Pearson, Sarah S. Poon, Gavin A. Price, Srividya Ramakrishnan, Priya Ranjan, Pamela C. Ronald, Michael C. Schatz, Samuel M. D. Seaver, Maulik Shukla, Roman A. Sutormin, Mustafa H. Syed, James Thomason, Nathan L. Tintle, Daifeng Wang, Fangfang Xia, Hyunseung Yoo, Shinjae Yoo, and Dantong Yu. KBase: The United States Department of Energy Systems Biology Knowledge-

base. *Nature Biotechnology*, 36(7):566–569, August 2018. ISSN 1546-1696. doi: 10.1038/nbt.4163. URL https://www.nature.com/articles/nbt.4163.

[S109] Matteo Mori, Terence Hwa, Olivier C. Martin, Andrea De Martino, and Enzo Marinari. Constrained Allocation Flux Balance Analysis. *PLOS Computational Biology*, 12(6):e1004913, June 2016. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004913. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004913.

[S110] Iván Domenzain, Benjamín Sánchez, Mihail Anton, Eduard J. Kerkhoven, Aarón Millán-Oropeza, Céline Henry, Verena Siewers, John P. Morrissey, Nikolaus Sonnenschein, and Jens Nielsen. Reconstruction of a catalogue of genome-scale metabolic models with enzymatic constraints using GECKO 2.0. *Nature Communications*, 13(1):3766, June 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-31421-1. URL https://www.nature.com/articles/s41467-022-31421-1.

[S111] Daan H. de Groot, Josephus Hulshof, Bas Teusink, Frank J. Bruggeman, and Robert Planqué. Elementary Growth Modes provide a molecular description of cellular self-fabrication. *PLOS Computational Biology*, 16 (1):e1007559, January 2020. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1007559. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1007559.

[S112] Stefan Müller. Elementary growth modes/vectors and minimal autocatalytic sets for kinetic/constraint-based models of cellular growth. *bioRxiv*, page 2021.02.24.432769, February 2021. doi: 10.1101/2021.02.24.432769. URL 10.1101/2021.02.24.432769.

[S113] G. Carrault, M. O. Cordier, R. Quiniou, and F. Wang. Temporal abstraction and inductive logic programming for arrhythmia recognition from electrocardiograms. *Artificial Intelligence in Medicine*, 28(3):231–263, July 2003. ISSN 0933-3657. doi: 10.1016/S0933-3657(03)00066-6. URL https://www.sciencedirect.com/science/article/pii/S0933365703000666.

[S114] Thomas Eiter, Wolfgang Faber, Christoph Koch, Nicola Leone, and Gerald Pfeifer. DLV - A System for Declarative Problem Solving. 2000. doi: 10.48550/arxiv.cs/0003036. URL https://arxiv.org/abs/cs/0003036.

[S115] Mark E. Stickel. A Prolog technology theorem prover: a new exposition and implementation in Prolog. *Theoretical Computer Science*, 104(1):109–128, October 1992. ISSN 0304-3975. doi: 10.1016/0304-3975(92)90168-F. URL https://www.sciencedirect.com/science/article/pii/030439759290168F.

[S116] David Poole. Logic programming for robot control. In *IJCAI*, pages 150–157, 1995.

[S117] José Neves, Tiago Guimarães, Sabino Gomes, Henrique Vicente, Mariana Santos, João Neves, José Machado, and Paulo Novais. Logic Programming and Artificial Neural Networks in Breast Cancer Detection. In Ignacio Rojas, Gonzalo Joya, and Andreu Catala, editors, *Advances in Computational Intelligence*, Lecture Notes in Computer Science, pages 211–224, Cham, 2015. Springer International Publishing. ISBN 978-3-319-19222-2. doi: 10.1007/978-3-319-19222-2_18.

[S118] Ali Sinan Köksal, Viktor Kuncak, and Philippe Suter. Scala to the Power of Z3: Integrating SMT and Programming. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction – CADE-23*, Lecture Notes in Computer Science, pages 400–406, Berlin, Heidelberg, 2011. Springer. ISBN 978-3-642-22438-6. doi: 10.1007/978-3-642-22438-6_30.

[S119] Miquel Bofill, Miquel Palah'i, and Mateu Villaret. A System for CSP Solving through Satisfiability Modulo Theories. pages 303–312, January 2009. ISBN 978-84-692-4600-9.

[S120] Ilkka Niemelä, Patrik Simons, and Tommi Syrjänen. Smodels: A System for Answer Set Programming. *CoRR*, cs.AI/0003033, March 2000.

[S121] Douglas Walton. Rules for Reasoning from Knowledge and Lack of Knowledge. *Philosophia (Ramat Gan)*, 34(3):355–376, 2006. ISSN 0048-3893. doi: 10.1007/s11406-006-9028-6.

[S122] Grigoris Antoniou and Frank van Harmelen. Web Ontology Language: OWL. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 91–110. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-540-92673-3. doi: 10.1007/978-3-540-92673-3_4. URL https://doi.org/10.1007/978-3-540-92673-3_4.

[S123] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. Jupyter Notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016. doi: 10.3233/978-1-61499-649-1-87. URL https://ebooks.iospress.nl/doi/10.3233/978-1-61499-649-1-87.

[S124] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

[S125] Albert Gevorgyan. *Theoretical and Computational Methods for Studying Genome-Scale Metabolic Networks Applied to Streptococcus agalactiae*. PhD thesis, Oxford Brookes University, Cell Systems Modelling Group, 2009. URL https://mudshark.brookes.ac.uk/Publications/thesis?action=AttachFile&do=view&target=Gevorgyan.pdf.

[S126] Heike Assmuss. *Modelling Carbohydrate Metabolism in Potato Tuber Cells*. PhD thesis, Oxford Brookes University, Cell Systems Modelling Group, 2005. URL https://mudshark.brookes.ac.uk/Publications/thesis?action=AttachFile&do=view&target=Assmuss.pdf.

[S127] J. C. Bioch and T. Ibaraki. Complexity of Identification and Dualization of Positive Boolean Functions. *In-*

formation and Computation, 123(1):50–63, November 1995. ISSN 0890-5401. doi: 10.1006/inco.1995.1157. URL 10.1006/inco.1995.1157.

[S128] Sabine Peres and Comet Jean-Paul. Contribution of Computational Tree Logic to Biological Regulatory Networks: Example from Pseudomonas Aeruginosa. In Corrado Priami, editor, *Computational Methods in Systems Biology*, Lecture Notes in Computer Science, pages 47–56, Berlin, Heidelberg, 2003. Springer. ISBN 978-3-540-36481-8. doi: 10.1007/3-540-36481-1_5.

[S129] Greg A. Somerville and Richard A. Proctor. At the Crossroads of Bacterial Metabolism and Virulence Factor Synthesis in Staphylococci. *Microbiology and Molecular Biology Reviews*, 73(2):233–248, June 2009. doi: 10.1128/MMBR.00005-09. URL https://journals.asm.org/doi/full/10.1128/MMBR.00005-09.

[S130] Austin S. Nuxoll, Steven M. Halouska, Marat R. Sadykov, Mark L. Hanke, Kenneth W. Bayles, Tammy Kielian, Robert Powers, and Paul D. Fey. CcpA Regulates Arginine Biosynthesis in Staphylococcus aureus through Repression of Proline Catabolism. *PLOS Pathogens*, 8(11):e1003033, November 2012. ISSN 1553-7374. doi: 10.1371/journal.ppat.1003033. URL https://journals.plos.org/plospathogens/article?id=10.1371/journal.ppat.1003033.

[S131] J. Stülke and W. Hillen. Carbon catabolite repression in bacteria. *Current Opinion in Microbiology*, 2(2): 195–201, April 1999. ISSN 1369-5274. doi: 10.1016/S1369-5274(99)80034-4.

[S132] Logan L. Bulock, Jongsam Ahn, Dhananjay Shinde, Sanjit Pandey, Cleofes Sarmiento, Vinai C. Thomas, Chittibabu Guda, Kenneth W. Bayles, and Marat R. Sadykov. Interplay of CodY and CcpA in Regulating Central Metabolism and Biofilm Formation in Staphylococcus aureus. *Journal of Bacteriology*, 204(7): e00617–21, June 2022. doi: 10.1128/jb.00617-21. URL https://journals.asm.org/doi/full/10.1128/jb.00617-21.

[S133] Ludwig Stenz, Patrice Francois, Katrine Whiteson, Christiane Wolz, Patrick Linder, and Jacques Schrenzel. The CodY pleiotropic repressor controls virulence in gram-positive pathogens. *FEMS Immunology & Medical Microbiology*, 62(2):123–139, July 2011. ISSN 0928-8244. doi: 10.1111/j.1574-695X.2011.00812.x. URL https://doi.org/10.1111/j.1574-695X.2011.00812.x.

[S134] Saugat Poudel, Hannah Tsunemoto, Yara Seif, Anand V. Sastry, Richard Szubin, Sibei Xu, Henrique Machado, Connor A. Olson, Amitesh Anand, Joe Pogliano, Victor Nizet, and Bernhard O. Palsson. Revealing 29 sets of independently modulated genes in Staphylococcus aureus, their regulators, and role in key physiological response. *Proceedings of the National Academy of Sciences*, 117(29):17228–17239, July 2020. doi: 10.1073/pnas.2008413117. URL https://www.pnas.org/doi/full/10.1073/pnas.2008413117.

[S135] Saugat Poudel, Ying Hefner, Richard Szubin, Anand Sastry, Ye Gao, Victor Nizet, and Bernhard O. Palsson. Coordination of CcpA and CodY regulators in Staphylococcus aureus USA300 strains, May 2022. URL https://www.biorxiv.org/content/10.1101/2022.05.25.493525v1.

[S136] Maria Teresa Ferreira, Ana S. Manso, Paula Gaspar, Mariana G. Pinho, and Ana Rute Neves. Effect of

Oxygen on Glucose Metabolism: Utilization of Lactate in Staphylococcus Aureus as Revealed by In Vivo NMR Studies. *PLOS ONE*, 8(3):e58277, March 2013. ISSN 1932-6203. doi: 10.1371/journal.pone.0058277. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0058277.

[S137] Manuel Liebeke, Kirsten Dörries, Daniela Zühlke, Jörg Bernhardt, Stephan Fuchs, Jan Pané-Farré, Susanne Engelmann, Uwe Völker, Rüdiger Bode, Thomas Dandekar, Ulrike Lindequist, Michael Hecker, and Michael Lalk. A metabolomics and proteomics study of the adaptation of Staphylococcus aureus to glucose starvation. *Molecular BioSystems*, 7(4):1241–1253, April 2011. ISSN 1742-2051. doi: 10.1039/C0MB00315H. URL http://pubs.rsc.org/en/content/articlelanding/2011/mb/c0mb00315h.

[S138] Anne Troitzsch, Vu Van Loi, Karen Methling, Daniela Zühlke, Michael Lalk, Katharina Riedel, Jörg Bernhardt, Eslam M. Elsayed, Gert Bange, Haike Antelmann, and Jan Pané-Farré. Carbon Source-Dependent Reprogramming of Anaerobic Metabolism in Staphylococcus aureus. *Journal of Bacteriology*, 203(8):e00639–20, March 2021. doi: 10.1128/JB.00639-20. URL https://journals.asm.org/doi/full/10.1128/JB.00639-20.

[S139] F. Bergmann, A. Mahler, H. Burger-Rachamimov, and D. Diller. Mechanism of uptake of purines by Pseudomonas aeruginosa. *Biochemistry*, 8(2):457–465, February 1969. ISSN 0006-2960. doi: 10.1021/bi00830a001.

[S140] Abdelsattar M. Omar, Saleh Ihmaid, EL-Sayed E. Habib, Sultan S. Althagfan, Sahar Ahmed, Hamada S. Abulkhair, and Hany E. A. Ahmed. The rational design, synthesis, and antimicrobial investigation of 2-Amino-4-Methylthiazole analogues inhibitors of GlcN-6-P synthase. *Bioorganic Chemistry*, 99:103781, June 2020. ISSN 0045-2068. doi: 10.1016/j.bioorg.2020.103781. URL https://www.sciencedirect.com/science/article/pii/S0045206820304090.

[S141] Joanna Stefaniak, Michal G. Nowak, Marek Wojciechowski, Slawomir Milewski, and Andrzej S. Skwarecki. Inhibitors of glucosamine-6-phosphate synthase as potential antimicrobials or antidiabetics–synthesis and properties. *Journal of Enzyme Inhibition and Medicinal Chemistry*, 37(1):1928–1956, 2022. doi: 10.1080/14756366.2022.2096018.

[S142] Caspar Elo Christensen, Birthe B. Kragelund, Penny von Wettstein-Knowles, and Anette Henriksen. Structure of the human beta-ketoacyl [ACP] synthase from the mitochondrial type II fatty acid synthase. *Protein Science : A Publication of the Protein Society*, 16(2):261–272, February 2007. ISSN 0961-8368. doi: 200701221720. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2203288/.

[S143] Mark P. Pereira, Jan E. Blanchard, Cecilia Murphy, Steven L. Roderick, and Eric D. Brown. High-Throughput Screening Identifies Novel Inhibitors of the Acetyltransferase Activity of Escherichia coli GlmU. *Antimicrobial Agents and Chemotherapy*, 53(6):2306–2311, June 2009. doi: 10.1128/aac.01572-08. URL https://journals.asm.org/doi/full/10.1128/aac.01572-08.

[S144] Laura E. Naught, Catherine Regni, Lesa J. Beamer, and Peter A. Tipton. Roles of Active Site Residues in

Pseudomonas aeruginosa Phosphomannomutase/Phosphoglucomutase. *Biochemistry*, 42(33):9946–9951, August 2003. ISSN 0006-2960. doi: 10.1021/bi034673g. URL https://doi.org/10.1021/bi034673g.

[S145] J. R. Coggins, C. Abell, L. B. Evans, M. Frederickson, D. A. Robinson, A. W. Roszak, and A. P. Lapthorn. Experiences with the shikimate-pathway enzymes as targets for rational drug design. *Biochemical Society transactions*, 31(Pt 3):548–552, 2003. doi: 10.1042/bst0310548.

[S146] Lene Nørby Nielsen, Henrik M. Roager, Mònica Escolà Casas, Henrik L. Frandsen, Ulrich Gosewinkel, Kai Bester, Tine Rask Licht, Niels Bohse Hendriksen, and Martin Iain Bahl. Glyphosate has limited short-term effects on commensal bacterial community composition in the gut environment due to sufficient aromatic amino acid levels. *Environmental Pollution*, 233:364–376, February 2018. ISSN 0269-7491. doi: 10.1016/j.envpol.2017.10.016. URL https://www.sciencedirect.com/science/article/pii/S0269749117328099.

[S147] Tathyana M. Amorim Franco and John S. Blanchard. Bacterial Branched-Chain Amino Acid Biosynthesis: Structures, Mechanisms, and Drugability. *Biochemistry*, 56(44):5849–5865, November 2017. ISSN 0006-2960. doi: 10.1021/acs.biochem.7b00849. URL https://doi.org/10.1021/acs.biochem.7b00849.

[S148] J. N. M. van Loon. Irreducibly inconsistent systems of linear inequalities. *European Journal of Operational Research*, 8(3):283–288, November 1981. ISSN 0377-2217. doi: 10.1016/0377-2217(81)90177-6. URL https://www.sciencedirect.com/science/article/pii/0377221781901776.

Software and Web Resources

[W1] National Center for Biotechnology Information (NCBI), 1988. URL https://www.ncbi.nlm.nih.gov/.

[W2] MIT Synthetic Biology Working Group. Synthetic Biology: FAQ, December 2002. URL https://web.archive.org/web/20021212065409/http://syntheticbiology.org/faq.html.

[W3] iGEM Competition. An Introduction to Engineering in Synthetic Biology, July 2023. URL https://web.archive.org/web/20230807070602/https://technology.igem.org/engineering/introduction.

[W4] Loïc Paulevé. BooleanNetworks.jl: Boolean networks in Julia. URL https://github.com/bnediction/BooleanNetworks.jl.

[W5] The MathWorks Inc. MATLAB version: 9.13.0 (R2022b), 2022. URL https://www.mathworks.com.

[W6] Wen-Wei Tseng. Gillespie Algorithm (Julia): Gillespie algorithm in Julia. URL http://web.archive.org/web/20231212195438/https://nextjournal.com/bebi5009/gillespie-julia.

[W7] Mathematics Stack Exchange and Mihir Chaturvedi. Are convex hulls polytopes?, September 2016. URL https://web.archive.org/web/20230524162516/https://math.stackexchange.com/questions/1910921/are-all-convex-hulls-polytopes-too.

[W8] IBM. CPLEX, August 2023. URL https://www.ibm.com/products/ilog-cplex-optimization-studio.

[W9] Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2023. URL https://www.gurobi.com.

[W10] Andrew Makhorin. GLPK—GNU linear programming kit., 2008. URL http://www.gnu.org/software/glpk/glpk.html.

[W11] Mathematics Stack Exchange and user284639. Polyhedra vs Polytope, March 2020. URL https://web.archive.org/web/20230524154826/https://math.stackexchange.com/questions/1663362/polyhedra-vs-polytope.

[W12] The OpenCobra project. Open-source, community-developed code base for COnstraint-Based Reconstruction and Analysis, 2021. URL http://opencobra.github.io/.

[W13] OpenCobra. COBRApy - Constraint-Based Reconstruction and Analysis in Python, June 2023. URL https://github.com/opencobra/cobrapy.

[W14] The cobrapy core team. Flux sampling - COBRAPy ReadTheDocs, 2016. URL https://cobrapy.readthedocs.io/en/latest/sampling.html.

[W15] Daniel Machado. Proper encoding of GPR associations · Issue #11 · opencobra/schema, 2017. URL https://github.com/opencobra/schema/issues/11.

[W16] efmtool_link. from CNApy team, November 2021. URL https://github.com/cnapy-org/efmtool_link.

[W17] Indiana University Information Technology Services. What is a computer's clock speed? URL https://web.archive.org/web/20211008085334/https://kb.iu.edu/d/aekt.

[W18] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. *The Satisfiability Modulo Theories Library (SMT-LIB)*. 2016. URL https://smtlib.cs.uiowa.edu/.

[W19] Jeffrey Orth. dynamic rFBA: Cobra Toolbox, 2008. URL https://github.com/opencobra/cobratoolbox/blob/79084d33e317a540eed6c70a0c9a481beda0289d/src/analysis/rFBA/dynamicRFBA.m.

[W20] IBM Documentation, November 2021. URL https://www.ibm.com/docs/en/icos/20.1.0?topic=conflicts-how-conflict-differs-from-iis.