Extracting Information from Biological Networks

by

Leonid Alexandrovich Chindelevitch

B.S., McGill University (2006)

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author .....
.....
Department of Mathematics
May 3, 2010

Certified by...
.....
Bonnie Berger
Professor of Applied Mathematics
Thesis Supervisor

Accepted by
.....
Michel X. Goemans
Chairman, Applied Mathematics Committee

Accepted by .
.....
Bjorn Poonen
Chairman, Department Committee on Graduate Students

Extracting Information from Biological Networks

by

Leonid Alexandrovich Chindelevitch

Submitted to the Department of Mathematics
on May 3, 2010, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Systems biology, the study of biological systems in a holistic manner, has been catalyzed by a dramatic improvement in experimental techniques, coupled with a constantly increasing availability of biological data. The representation and analysis of this heterogeneous data is facilitated by the powerful abstraction of biological networks. This thesis examines several types of these networks and looks in detail at the kind of information their analysis can yield.

The first part discusses protein interaction networks. We introduce a new algorithm for the pairwise alignment of these networks. We show that these alignments can provide important clues to the function of proteins as well as insights into the evolutionary history of the species under examination.

The second part discusses regulatory networks. We present an approach for validating putative drug targets based on the information contained in these networks. We show how this approach can also be used to discover drug targets.

The third part discusses metabolic networks. We provide new insights into the structure of constraint-based models of cell metabolism and describe a methodology for performing a complete analysis of a metabolic network. We also present an implementation of this methodology and discuss its application to a variety of problems related to the metabolism of bacteria.

The final part describes an application of our methodology to Mycobacterium tuberculosis, the pathogen responsible for almost 2 million deaths around the world every year. We introduce a method for reconciling metabolic network reconstructions and apply it to merge the two published networks for tuberculosis. We analyze the merged network and show how it can be refined based on available experimental data to improve its predictive power. We conclude with a list of potential drug targets.

Thesis Supervisor: Bonnie Berger
Title: Professor of Applied Mathematics

Acknowledgements

First and foremost, I would like to thank my advisor, Bonnie Berger, for her guidance and mentorship throughout this long journey, for her faith in my ability to succeed, and for her unwavering support. I would also like to thank Aviv Regev for the exceptional opportunity to collaborate with the Broad Institute on infectious disease research, for her patience, and for the many important skills I have learned from her.

My heartfelt thanks also go to:

My mom and dad, for always being there for me and pushing me to do my best.

Michel Goemans, for sharing his vast knowledge of algorithms and complexity.

Mathieu Blanchette, for introducing me to the field of computational biology.

Deborah Hung, for introducing me to the research and clinical aspects of TB.

Marie Turner, for giving me the opportunity to be part of her tuberculosis clinic.

Michael Scnhnall-Levin and Chung-Shou Liao, for being amazing collaborators.

Patrice Macaluso, for being our research group's guardian angel, and the other members of the group, in particular Jérôme Waldispühl, Patrick Schmid, Rohit Singh, Raghu Hosur, Luke Hutchison, Michael Baym, Irene Kaplow, Mike Yu.

The members of Deborah Hung's lab, especially Sarah Stanley, for everything.

Joseph Lehár and Daniel Ziemek, for being my supervisors on the dark side.

My funders, the Natural Sciences and Engineering Research Council of Canada.

My friends, especially Alexey Spiridonov, Maksim Maydanskiy, Srikant Sarangi, Ararat Harutyunyan, Amy Rossman, Karola Mészáros, Aquene Freechild, Premnandhini Satgunam, and my partners in a variety of crimes committed at MIT.

Previous Publications of this Work

Portions of Part I have appeared in the proceedings of the 2010 Pacific Symposium on Biocomputing [1], co-authored with Chung-Shou Liao and Bonnie Berger. They are reprinted with the kind permission of the editors.

Part II has appeared as an unpublished technical report submitted to Pfizer, Inc. in the summer of 2009. It is included with the kind permission of Enoch Huang and Daniel Ziemek.

Part III is based on a paper currently in preparation, co-authored with Aviv Regev and Bonnie Berger.

Part IV is based on a paper currently in preparation, co-authored with Sarah Stanley, Deborah Hung, Aviv Regev and Bonnie Berger.

Contents

Chapter 1

Introduction

In this thesis, we investigate the general problem of extracting information from biological networks. This problem requires a combination of mathematical analysis, computational methods and biological interpretation. In addition, because biological networks are never complete or error-free, it is important to address the question of how their shortcomings can be taken into account in the analysis. Finally, it is important to consider the interaction between the results of an analysis and the results of biological experiments.

In this chapter, we introduce and summarize the main contributions of this thesis. The remainder of this document will be divided into four main parts. The first, concerning the alignment of protein interaction networks, is based on a paper presented at the Pacific Symposium for Biocomputing in 2010 [1]. The second, concerning target validation and prediction from regulatory networks, is based on work done at Pfizer in the summer of 2009 and has been accepted as a poster at the 2010 ISMB [2]. The third, concerning the analysis of metabolic networks, is in preparation for journal publication [3]. The fourth, an application of the methods developed in the first three to Mycobacterium tuberculosis, is the result of a collaboration with Aviv

Regev and Deborah Hung's labs at the Broad Institute.

1.1 Protein Interaction Network Alignment

Protein interaction networks encode the way proteins interact with each other, and thus contain a wealth of information about the function of different proteins in various organisms. These networks, however, exhibit variable degrees of annotation. We propose a novel algorithm for aligning a pair of protein interaction networks. This algorithm not only results in a highly accurate prediction of functional orthologs between different species but also provides insights into their evolutionary history. We discuss the application of this algorithm to the protein interaction networks of three species - yeast, fly, and worm - and compare its performance to that of other existing methods.

1.2 Regulatory Networks

Regulatory networks encode information about the mutual influence of different genes and their products - messenger RNA molecules and proteins - inside a cell. These networks can be inferred experimentally or derived from published literature. Causal Reasoning is a framework for analyzing the cause-to-effect relationships these networks contain. Within this framework, we propose an algorithm for inferring the most likely causes of an experimentally observed change in gene expression levels. These causes are postulated to be the targets of the intervention that created the changes in expression. By applying this algorithm to expression data derived from tissues treated with drugs, we provide a method for validating known drug targets and discovering unknown ones.

1.3 Metabolic Networks

Metabolic networks are a representation of all the biochemical reactions that happen inside a cell. The large scale of these networks makes an analysis based purely on kinetics impractical and requires an approach with as few parameters as possible. Constraint-based models of metabolism are one such approach. We discuss various types of analysis that can be performed in a constraint-based model, questioning some of the assumptions made in previous work and developing new insights into the model's structure. Finally, we propose a complete pipeline for the systematic investigation of metabolic networks as well as their refinement based on various experimental data.

1.4 Applications to M. tuberculosis

Mycobacterium tuberculosis, the pathogen responsible for one of the most common infectious diseases around the world, has proven to be an elusive target for therapeutic interventions due to its successful adaptation to the human host. There is a dire need for new approaches to addressing this important public health issue. Using a comprehensive analysis of the metabolic network of this pathogen, we propose a short list of potential drug targets, which will hopefully be the first step towards developing and implementing new antitubercular therapies.

Part I

Protein Interaction Networks

"A person who never made a mistake never tried anything new."
Albert Einstein, theoretical physicist, philosopher and author.

In this part of the thesis, we study the problem of aligning two protein interaction networks. In the first chapter, we motivate the problem of aligning a pair of protein interaction networks and summarize previous work on this problem. In the second chapter, we present PISwap, a novel algorithm for this problem, and describe its implementation. In the third and final chapter, we present experimental results from applying the PISwap algorithm to the protein interaction networks of three different species - yeast, fly, and worm - and discuss its performance as well as some interesting implications.

Chapter 2

Motivation and Previous Work

This chapter presents some background on the alignment of protein-protein interaction (PPI) networks. We begin by motivating the problem of aligning protein interaction networks, and then provide a general survey of existing methods for this problem. We go on to discuss some issues related to IsoRank, one of the most popular protein interaction network alignment algorithms. We conclude by summarizing the key challenges in the field of protein interaction network alignment.

2.1 Problem Motivation

Ever since high-throughput experimental screening techniques such as yeast two-hybrid analysis [5], mass spectrometry [6], and TAP [7] made protein interaction networks available for several species, efforts have been made in the bioinformatics community to extract useful biological information from these networks. One important goal has been to produce accurate alignments of two or more of these networks. The expectation is that this will help in both establishing the biological function of unknown proteins by exhibiting their correspondence with the proteins of another

species with known biological function, as well as providing insight into evolutionary dynamics.

Algorithms for network alignment can be broadly separated into two distinct categories: *local* and *global* algorithms. The distinction is similar to the one made for sequence alignment algorithms. More specifically, local network alignment [8] is concerned with identifying a subnetwork of one species closely matching a subnetwork of another species (or having a given topology). Typically, multiple closely matching subnetworks are identified by such algorithms, which may be mutually inconsistent [9]. On the other hand, global network alignment algorithms attempt to match two or more networks as a whole, and their output is a single mapping between the nodes of the networks [9]. In this part of the thesis, which deals with the global alignment problem, we view this mapping as a bipartite matching, where the nodes on one side of the matching are the proteins from one network, and the nodes on the other side, the ones from the other network.

2.2 Survey of Previous Work

A number of techniques for the problem of PPI network alignment exist. These include NetworkBLAST-M [10], Græmlin 2.0 [11], IsoRank [9], IsoRankN [12], PATH [13], and others [8, 14, 15, 16, 17, 18, 19]. Some of these methods can handle more than two networks. NetworkBLAST-M computes a local alignment by greedily finding regions of high local conservation based on inferred phylogeny. Græmlin 2.0, in contrast, computes a global alignment by training how to infer networks from phylogenetic relationships on a known set of alignments and then optimizing the learned objective function on the set of all networks. IsoRank uses spectral graph theory to first find pairwise alignment scores across all pairs of networks, which is obtained by a spectral method on a product graph; IsoRank then uses these scores in a greedy

algorithm to produce the final alignment. The more recent IsoRankN uses a different method of spectral clustering on the induced graph of pairwise alignment scores.

2.3 Issues Specific to IsoRank

IsoRank [9] is an algorithm similar in nature to the Google PageRank algorithm [20]. However, the rankings it computes apply to pairs of nodes (one from each network being aligned) and are not an end in itself, but rather a measure of the affinity of the nodes that is then used to extract a one-to-one alignment with high overall affinity. More precisely, the rankings are the solution of the eigenvalue problem

$$R = (\alpha)AR + (1 - \alpha)E, \tag{2.1}$$

where E is a normalized vector of pairwise BLAST scores [21] for each pair of nodes and α is a tradeoff parameter between sequence and topology information (also used in the PISwap algorithm presented in the following chapter). The matrix A is the Markov matrix for a random walk on the graph of pairs of nodes. It has a non-zero entry in row $[i, j]$ and column $[u, v]$ if and only if $(i, u) \in E_1, (v, j) \in E_2$, given by

$$A[i, j][u, v] := \frac{w(i, u)w(j, v)}{\left(\sum_{r \in N(u)} w(r, u)\right)\left(\sum_{q \in N(v)} w(q, v)\right)} \tag{2.2}$$

Here, $w(x, y)$ denotes the weight of the edge (x, y) (usually taken to be 1 unless more precise information is available), $N(x)$ is the *neighborhood* of x (the set of vertices *adjacent* to it) and E_1, E_2 are the edge sets of the two networks being aligned.

There are several technical issues specific to IsoRank that we briefly discuss below. They are described in detail in an unpublished manuscript [22].

The first issue is that the eigenvalue problem uses the power method, rather

than the much more efficient Lanczos iteration [23] (Google's PageRank algorithm apparently has the same problem). One reason to choose the power method over the Lanczos iteration could be the need for parallel processing, but that does not seem to be necessary for IsoRank.

The second, much more serious issue is that, unlike the Google PageRank algorithm, IsoRank operates on an undirected graph. However, the solution of the basic eigenvalue problem (equation (2.1) with $\alpha = 1$) is simply a vector with entries proportional to the total weight of each node,

$$R_v := \sum_{u \in N(v)} w(u, v). \tag{2.3}$$

The graph on which IsoRank operates is the Kronecker (tensor) product of the two original graphs [24], so the solution of equation (2.1) with $\alpha = 1$ is the entry-wise product of the total weights of the nodes,

$$R = R_1 \otimes R_2. \tag{2.4}$$

It follows that the greedy algorithm for computing the maximum affinity matching (as well as the maximum bipartite matching algorithm by the *rearrangement inequality* [25]) will simply match the nodes of the two networks by their degree (or total weight). This effectively makes the algorithm local, rather than global, as the only connectivity information used is the degree of each node.

The final issue for IsoRank is the method by which the parameter α is chosen. Because the sequence and the topology information are fully coupled, α cannot be chosen in an a priori way and needs to be adjusted experimentally. On the other hand, in the PISwap algorithm, we are able to provide a principled choice of α that can be directly incorporated into the algorithm.

2.4 Challenges

An important limitation of all protein interaction network alignment methods is the incomplete coverage of the networks as well as the large fraction of false positive interactions. These problems can be addressed in a variety of ways, but none so far has been thought to successfully overcome those limitations. Nevertheless, the question of how much information can be extracted from the alignment of far-from-perfect networks is still both theoretically valid and practically significant.

As pointed out in a recent paper [13], one of the main difficulties faced by network alignment algorithms is the lack of an accurate gold standard for evaluation purposes. This makes comparing two algorithms objectively extremely difficult. There are, of course, a number of proteins in different species that are known to be functional orthologs. However, experience shows that every alignment algorithm will get some of them right and some of them wrong. The practice of including statements of the form "on this particular family of proteins, our algorithm performs extremely well" is both counterproductive (not every alignment algorithm gets evaluated on the same set of known orthologs) as well as misleading (there is no guarantee that the proteins in a closely related family will be better aligned by the algorithm in question than by any of the other ones).

A partial solution was proposed by the authors of the IsoRank algorithm [9]: a concept called *functional coherence* that is discussed in the chapter on Results. Unfortunately, this measure, a score from 0 to 1 which can be computed automatically using the Gene Ontology [26] terms for each protein, also suffers from a number of drawbacks. It is a global score for an alignment, unsuitable for investigating the influence of individually aligned pairs due to its non-additive nature, and seems to depend on the number of annotations for each protein. Finally, it is not necessarily equal to 1 for even a perfect alignment (Alex Levin, personal communication).

Chapter 3

The PISwap Algorithm

In this chapter we introduce the PISwap algorithm for pairwise alignment of protein interaction networks. We begin by providing a slightly different formulation of the problem, which will prove to be more convenient. We then describe the algorithm as well as the ideas that inspired its conception. We go on to establish that its running time is pseudo-polynomial in the size of the input and conclude by providing some details of its implementation.

3.1 Problem Formulation

We consider the global alignment of a pair of protein-protein-interaction (PPI) networks. Each network is represented by a graph whose vertices correspond to proteins, and there is an undirected edge between two vertices if the corresponding proteins have been found to interact with each other.

Given a pair of PPI networks and a list of pairwise sequence similarities between proteins in the two networks computed according to some criterion, the specific aim of global alignment is to find a mapping between the proteins of the two networks that

best represents conserved biological function. We formulate this network alignment problem as a graph-theoretical problem.

Let $G_X = (X, E_X)$ and $G_Y = (Y, E_Y)$ be two PPI networks in which $e_x = (x, x') \in E_X$ and $e_y = (y, y') \in E_Y$, with $x, x' \in X$ and $y, y' \in Y$, represent interaction between two proteins in G_X and G_Y respectively.

Suppose $G = (X \cup Y, E)$ is an edge-weighted bipartite graph in which each edge $e = (x, y) \in E$ is associated with a nonnegative edge weight $s(e)$, which represents the sequence similarity between $x \in X$ and $y \in Y$. That is, $s : E \to \mathbb{Z}^+$ denotes a sequence similarity function on the edges of G, where sequence similarity on an edge, a pair of proteins, could be, for instance, the BLAST Bit-value of the sequences as retrieved from Ensembl [30].

A matching $M \subseteq E$ of G is defined to be a subset of edges such that no two edges in M share an endpoint. In addition, given a matching M, we let a function $t : E \to \mathbb{Z}^+$ be a topology similarity function with respect to M if for an edge $e = (x, y)$, $t(e)$ represents the topology similarities between the neighborhoods of $x \in X$ and $y \in Y$. The topology similarity function represents the number of edges in G_X and G_Y conserved by the matching M.

More precisely, for an edge $e = (x, y)$, $t(e)$ is the number of edges between the neighborhoods of x and y, $N(x)$ of G_X and $N(y)$ of G_Y respectively, which are also in the matching M, i.e. $t(e) = |\{(x', y') \in M | x' \in N(x) \text{ and } y' \in N(y)\}|$.

Our objective is to find a matching M that maximizes the following weight function w:

$$w(M) = \sum_{e \in M} \{\alpha t(e) + (1 - \alpha) s(e)\}, \tag{3.1}$$

where $\alpha \in [0, 1]$ is a parameter controlling the importance of the network topology similarities relative to sequence similarities. This is equivalent to the objective function introduced by Singh et al. [9].

The above weight function contains two terms: the topology similarity function t and the sequence similarity function s. Tuning the parameter α allows us to change the relative importance of PPI network data in finding the optimal global alignment. For instance, $\alpha = 0$ implies no network data will be used, while $\alpha = 1$ indicates only network data will be used.

We will use a local search approach to pairwise PPI network alignment in a manner similar to `2-Opt` as follows: when given a maximum weighted bipartite matching M^* in $G = (X \cup Y, E)$, we define a subset $\text{prefer}_Y(x)$ for each $x \in X$, which consists of the c highest-weighted neighbors of x in Y (where the weight of a neighbor of x is given by its sequence similarity to x). Similarly, for every $y \in Y$, a vertex subset $\text{prefer}_X(y) \subseteq X$ is similarly defined to consist of the c highest-weighted neighbors of y in X. Here, c is some relatively small integer chosen ahead of time. It can be shown [31] that $c = 20$ suffices for most practical applications.

Our aim is to repeatedly find a candidate $e' = (u, v)$, $v \in \text{prefer}_Y(x)$, $u \in \text{prefer}_X(y)$ to swap with $e = (x, y)$, where e, $e' \in M^*$, such that the weight of the new matching, $w((M^* \setminus \{e, e'\}) \cup \{e_1, e_2\})$, where $e_1 = (x, v)$ and $e_2 = (u, y)$ are the edges obtained by swapping e and e', is higher than $w(M^*)$.

Just as in IsoRank [9], we seek to maximize an objective function consisting of two terms, one accounting for sequence similarity between the proteins that get matched to each other, and the other one, for the number of interactions preserved by the matching. Our search space is the set of all matchings. We use the parameter $\alpha \in [0, 1]$ to control the relative importance of the topology information with respect to the sequence information. This formulation of the problem is known to be NP-hard, and even APX-hard [27], which means that it does not admit a polynomial-time approximation scheme unless P = NP. Since the size of the search space (i.e. the number of possible matchings) grows exponentially with the number of proteins in each

network, we use a local search technique adapted from other NP-hard optimization problems, which is described in detail in the following section.

3.2 Algorithmic results

The main idea of our algorithm for the global alignment of pairwise PPI networks is a two-phase approach to searching for a matching that maximizes our objective function. The special case of our problem with $\alpha = 0$ (i.e. one where only sequence information is used) is a variant of the *linear assignment problem*, which is that of finding a maximum-weight matching in a bipartite graph.

Hence the first stage of our algorithm finds a maximum-weight matching in the bipartite graph obtained by joining pairs of proteins in the two networks, where the weight of an edge is given by the sequence similarity of the two proteins that it joins. This matching can be obtained by the well-known *Hungarian algorithm* in polynomial time [35] and sped up with extensive use of priority queues and decomposition techniques [34].

In the second stage of our algorithm, we apply the local search method, which is widely used in the combinatorial optimization field, to iteratively improve the initial matching while taking into account both the sequence score and the topology score of the matching.

From a variety of local search methods, we make use of the `2-Opt` algorithm, which was first proposed by Croes [29]. The `2-Opt` algorithm is one of the most famous heuristics for the well-known *Traveling Salesman Problem* [36]. Given a set of cities, the Traveling Salesman Problem is to find an ordering of cities that minimizes the total length of the tour when visiting all the cities in some order and returning to the starting city. The basic concept of the `2-Opt` algorithm is simple. A move deletes two edges of the original tour, thus breaking the tour into two paths, and

then reconnects those paths by swapping these edges.

Local search algorithms do not appear to perform well from a theoretical point of view. Papadimitriou and Steiglitz [38] have shown that no local search algorithm (like `2-Opt`) that takes polynomial time per move can guarantee a constant approximation ratio for TSP unless P = NP. In addition, it has been shown that a sequence of exponential moves might be required by `2-Opt` before halting [37], and an analogous result [28] has been extended to `3-Opt` and k-`Opt`.

Although the worst-case analysis of the `2-Opt` algorithm is pessimistic, the average-case analysis is considerably more optimistic. A significant discovery [28] has shown that the expected approximation ratio to the optimum is bounded by a constant. A similar improvement with respect to running time has been obtained as well. That is, the expected number of moves is polynomially bounded. Furthermore, `2-Opt` outperformed almost all the local search and greedy algorithms in experimental results for TSP [31]. More precisely, `2-Opt` (or k-`Opt`) gave better final tours than other local search algorithms for TSPLIB instances [31] with respect to both approximation ratio and running time.

The technique of iterative edge swaps is, however, not limited to TSP. It is also the basis of an algorithm for graph randomization, which attempts to produce a random graph with a given degree distribution [32]. It can be shown [33] that the Markov chain defined by this method converges to the uniform distribution on the set of all connected simple graphs with the given degree distribution, thus giving an exact algorithm.

Figure 3-1 illustrates a matching on a pair of small networks transformed by an edge swap. It is important to note that an edge swap generally changes the conserved edges, provided these edges are incident to (i.e. share a vertex with) each of the two edges being swapped.

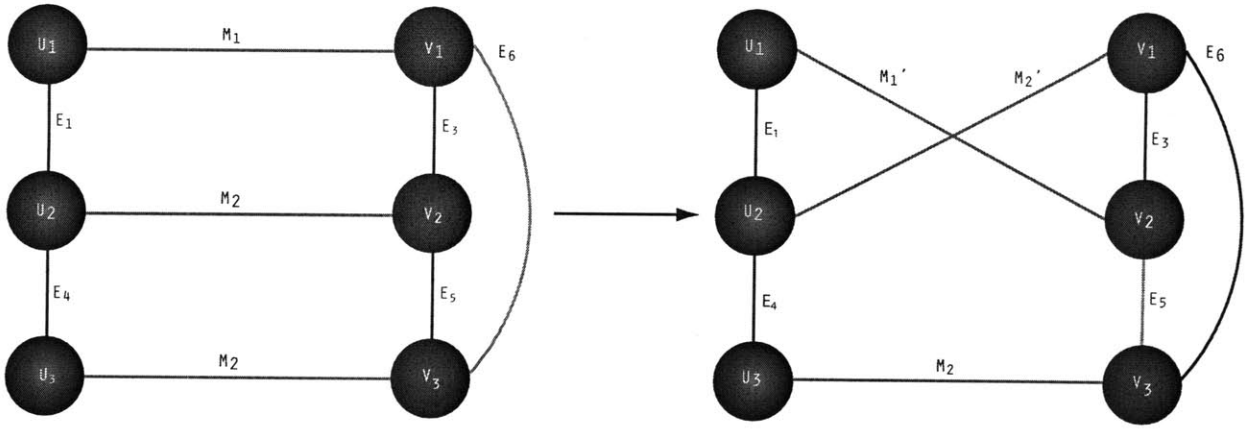

Figure 3-1: The situation before and after a possible edge swap. The edges M_1 and M_2 of the matching are swapped out and replaced by M_1' and M_2'. The matching is illustrated in red, conserved edges are black, and non-conserved edges are light blue.

We now present the pseudocode of our algorithm. Its running-time is analyzed in the following section.

Algorithm [PISwap]

Given a weighted bipartite graph $G = (X \cup Y, E)$ with a maximum weighted matching M^* and parameters α and c,

1. Compute topology similarities $t(e)$ and weight $w(e)$ for each edge $e \in M^*$, where the weight $w(e) = \alpha t(e) + (1 - \alpha)s(e)$;

2. Find a candidate set T consisting of every edge $e = (x, y) \in M^*$ which satisfies the following condition:

   There is $e' = (u, v) \in M^*$ with $v \in \mathit{prefer}_Y(x), u \in \mathit{prefer}_X(y)$ such that, for $e_1 = (x, v)$, $e_2 = (u, y)$,

$$\mathit{swap}(e, e') := \{w(e_1)+w(e_2)+\alpha(t(e_1)+t(e_2))\}-\{w(e)+w(e')+\alpha(t(e)+t(e'))\} > 0; \quad (2)$$

3. **while** $S \neq \emptyset$ **do**

   3-1. Select the pair of edges $e = (x, y) \in T$ and $e' = (u, v)$ which achieve the maximum value of $swap(e, e')$;

   3-2. Swap e, e' with $e_1 = (x, v)$, $e_2 = (u, y)$ to obtain a new matching $(M^* \setminus \{e, e'\}) \cup \{e_1, e_2\}$ and $T = T \setminus \{e, e'\}$;

   3-3. Verify if the new inserted edges e_1, e_2 satisfy the condition (2) above and put them into T if necessary;

   3-4. Update the topology similarities $t(e)$ for each edge $e \in M^*$ with one endpoint in $N(x) \cup N(u)$ and the other, in $N(y) \cup N(v)$ and modify $w(e)$ and $swap(e, e')$;

4. **return** M^*.

3.3 Running-time analysis

In what follows, M^* always denotes the mapping at the current iteration.

First, note that only topology similarities $t(e)$ for each edge $e \in M^*$ incident to a node in $N(x)$, $N(y)$, $N(u)$, and $N(v)$ are changed when we swap the edges (x, y) and (u, v) in M^*. In addition, as we consider the weight difference $w(M^* \setminus \{e, e'\} \cup \{e_1, e_2\}) - w(M^*)$, removing the edge $e = (x, y) \in M^*$ causes the weight loss $w(e)$, but it also causes neighbors in $N(x)$ and $N(y)$ to lose $\alpha t(e)$. Removing the edge $e' = (u, v)$ produces an analogous effect. On the other hand, there is a weight gain $w(e_1)$ as well as $\alpha t(e_1)$ from inserting the edge $e_1 = (x, v)$. Inserting the edge $e_2 = (u, y)$ produces an analogous effect.

Therefore, the weight of the matching M^* increases by the quantity $swap(e, e')$, defined in (2). The inequality in (2) thus ensures that the objective function increases after the swap.

Lemma 1. *Given a a weighted bipartite graph $G = (X \cup Y, E)$ with a maximum weighted matching M^* and parameters α and c, the running time of* PISwap *is pseudo-polynomial.*

Proof.

It is readily seen that the cardinality of a maximum-weight matching M^* is $|M^*| \leq \min\{|X|, |Y|\}$. Note that the preprocessing of PISwap to obtain a maximum weighted matching M^* by the *Hungarian algorithm* takes $O(|M^*|^3)$ time.

Let Δ denote the largest degree of a vertex in G_X and G_Y, i.e. the largest number of neighbors a vertex in $X \cup Y$ can have. Let B denote the largest similarity value for two sequences, i.e. $B = \max_{x \in X, y \in Y}\{s(x, y)\}$.

In Step 1 we compute the topology similarities $t(e)$ and the weights $w(e)$ for each edge $e = (x, y) \in M^*$. Since we consider all possible pairwise combinations between neighbors of x and neighbors of y, this requires $O(|M^*| \times \Delta^2)$ time.

In Step 2 we find the candidate set T. We first compute the subsets $\text{prefer}_Y(x)$ and $\text{prefer}_X(y)$ for each vertex in $X \cup Y$. The running time is bounded by $O(|X| \times |Y|)$ since it requires $O(|Y|)$ (respectively $O(|X|)$) time to find the c highest-weighted neighbors in Y (respectively X) for each vertex $x \in X$ (respectively $y \in Y$), for any constant c.

We then find all the edges $e' \in M^*$ satisfying the condition (2) for every edge $e \in M^*$. For every edge $e = (x, y) \in M^*$, there are at most c^2 edges in M^* with one endpoint in $\text{prefer}_X(y)$ and the other endpoint in $\text{prefer}_Y(x)$ that e can be swapped with. The weight difference $swap(e, e')$ can be computed in constant time from the topology similarities and sequence similarities for every edge. Hence Step 2 takes

$O(c^2|M^*|)$ time. Without loss of generality, we assume $c \leq \sqrt{|M^*|}$; otherwise we can check all pairs of edges in M^* to see if they are able to be swapped, at a cost of $O(|M^*|^2)$.

Step 3 is iterative, and we first consider the time complexity of one iteration. The maximum value of $swap(e, e')$ can be found in constant time by using a priority queue. The swap operation also takes constant time. For the two new inserted edges of M^*, we verify if they satisfy the property (2) in $O(c^2)$ time as above. The last step to update the values of $t(e)$, $w(e)$, and $swap(e, e')$. This takes $O(\Delta^2)$ time as above, since only the edges $e \in M^*$ with one endpoint in $N(x) \cup N(u)$ and the other, in $N(y) \cup N(v)$, are affected. Therefore, each iteration requires $O(\max\{c^2, \Delta^2\})$ time.

Finally, consider the number of iterations of the while loop. The total sequence score is an integer and varies between 0 and $|M^*| \times B$, and similarly, the total topology score is an integer and varies between 0 and $|M^*|^2$; the consecutive values of $w(M^*)$ form a strictly increasing sequence whose length is bounded by $(|M^*| \times B + 1) \times (|M^*|^2 + 1) \in O(|M^*|^3 \times B)$. Since step 3 is the dominating one, PISwap runs in $O(\max\{c^2, \Delta^2\} \times B \times |M^*|^3)$ time. □

3.4 Implementation details

The algorithm was implemented in Python 2.5.4 [42] using the NetworkX [43] package. We also used Joris van Rantwijk's GPL implementation of the maximum-weight matching algorithm based on the blossom method for finding augmenting paths and the primal-dual method for finding a matching of maximum weight [44]. All experiments were performed on a Lenovo laptop with a 64-bit architecture running Windows Vista with an Intel Core 2 Duo T9600 CPU and 4 GB of RAM.

Chapter 4

Results and Discussion

This chapter presents the results of applying the PISwap algorithm to actual protein interaction networks. We begin by discussing the algorithm's performance on the protein interaction networks for C.elegans, D.melanogaster and S.cerevisiae. We then suggest how the results of the algorithm can provide interesting evolutionary insights. We conclude with a discussion of the significance of our findings.

4.1 Experimental results

In spite of the analysis presented in the previous chapter, the running-time of our algorithm is actually dominated by the preprocessing step, that of finding a maximum-weight bipartite matching. This is because the running time is cubic in the size of the input, whereas the number of iterations in the main loop of the algorithm was typically between 40 and 120. Thus, the upper bound presented in the analysis is overly pessimistic. We used the value $c = 200$ for our experiments, since this was close to the maximum degree Δ of the input networks (which was 190, 184 and 323 for C.elegans, D.melanogaster and S.cerevisiae, respectively).

In terms of actual time, the initial matching took between 5 and 10 minutes to compute on a single-processor laptop, but it was then saved and used with multiple values of α and c. Each of the runs of the second stage of the algorithm took no more than 15 to 20 seconds to produce the final mapping.

The ratio of α to $1-\alpha$ can be thought of as giving the relative weight of sequence information to topology information. We found that choosing α to make this relative weight half of what it is for the initial mapping was a good rule of thumb. In other words, since the original mapping was based purely on sequence information, we chose α to favor the topology twice as much as the original mapping did, in all the reported experiments. We used both raw BLAST scores and normalized BLAST scores, retrieved from the IsoRank website [45]. The raw BLAST scores used were computed as $s(i,j) = B(i,j) + B(j,i)$, where $B(i,j)$ is the value given by BLAST on input i and j (this is because BLAST may occasionally produce "asymmetric" results). The normalized scores were computed as $\frac{s(i,j)}{\sqrt{s(i,i)s(j,j)}}$, and resulted in values between 0 and 1. Although our algorithm is described as dealing with integers, it is actually capable of handling fractional weights (although the running-time bound proved in the previous chapter does not hold in that case). Also, it is important to note that, since the initial matching was a maximum-weight matching, not all the proteins that had a non-zero similarity to some protein in the other species ended up in the matching — the size of the matching was roughly $0.9\ \min(|V_1|, |V_2|)$, or 90% of the largest possible matching. We chose not to force matches between proteins with zero sequence similarity to all proteins in the other species, since topology alone does not suffice to create a reliable matching between them [9].

Table 1 illustrates the results for the unnormalized input data (the results for the normalized data are qualitatively very similar). It clearly shows that PISwap compares favorably to other algorithms. The abbreviations denote the species used:

CE = C.elegans, DM = D.melanogaster, SC = S.cerevisiae. The figures for the initial matching are provided for comparison purposes only, and it is the final alignments that are output by the algorithm.

The number of swaps required was 97 (102) for that between D.melanogaster and S.cerevisiae, 38 (54) for that between C.elegans and S.cerevisiae, and 71 (85) for the mapping between C.elegans and D.melanogaster. In each pair, the first number indicates the unnormalized case and the second, the normalized case. Note that these are significantly higher than the diameter (length of the path between the two most distant nodes) of the networks, which are 14, 11 and 9 for C.elegans, D.melanogaster and S.cerevisiae, respectively, meaning that information relevant for the alignment may have had a chance to propagate to all of the nodes.

To evaluate the output, we first used the HomoloGene database [46] which is thought to contain a reliable orthology mapping, though mainly based on sequence similarity between the proteins and the DNA regions that code for them [46]. We also looked at the number of conserved interactions. It is interesting to note that, compared to the initial matching, the final matching contains the same number of functional orthologs based on HomoloGene as the gold standard (except in the case of C.elegans and D.melanogaster, where the pair (CO5C8.6, CG1826) becomes lost in the swapping process), while at the same time having significantly more conserved interactions (on average, 60% more). This shows that our algorithm is indeed performing its goal of achieving a high topology similarity while retaining an excellent sequence similarity. For comparison, the D.melanogaster - S.cerevisiae alignment produced by the five algorithms (MP [39], MRF [40], IsoRank [9], GA [13] and PATH [41]) studied by Zaslavskiy et al. [13] produce between 36 and 41 orthologous pairs from HomoloGene, compared to 51 for our algorithm (though the additional constraints imposed on the problem studied there make it difficult to perform a fair comparison).

Evaluation of PISwap alignments based on unnormalized sequence data

	DM-SC alignment		CE-SC alignment		CE-DM alignment	
	Initial	Final	Initial	Final	Initial	Final
Number of swaps	0	97	0	38	0	71
HomoloGene pairs	51	51	41	41	819	818
Conserved edges	272	398	106	150	80	154
Functional Coherence	N/A	0.510	N/A	0.172	N/A	0.355

Table 4.1: Results for the unnormalized input data. The abbreviations denote the species used: CE = C.elegans, DM = D.melanogaster, SC = S.cerevisiae. The figures for the initial matching are provided for comparison purposes only, and it is the final alignments that are output by the algorithm.

In addition, we computed the Functional Coherence values, following the method outlined by Singh et al. [9]. The method for computing Functional Coherence can be summarized as follows. First, the GO terms corresponding to each protein are collected. Then, each GO term is mapped to a subset of the so-called standardized GO terms, which in this case are its ancestors at a distance 5 from the root of the GO tree. Finally, the similarity between each pair of aligned proteins is computed as the median of the fractional overlaps of their corresponding sets of standardized GO terms. The Functional Coherence of an alignment is defined as the average pairwise Functional Coherence of the protein pairs that it matches. The values for the D.melanogaster-S.cerevisiae alignment produced by our algorithm was 0.510, comparable to the values of 0.519, 0.509 and 0.522 produced on the same input by IsoRank, GA and PATH, respectively [13]. The functional coherence values of the other pairs were not available for the other algorithms.

4.2 Evolutionary model

Although the edge-swapping technique was originally inspired by the field of combinatorial optimization, one can speculate that it can actually give us insights into the way two networks evolved from a common ancestor. If the networks of two closely related species are being analyzed, it is conceivable that at the outset, the proteins of the two networks are essentially identical in sequence, and hence, their correspondence can be determined exclusively on the basis of sequence information. Suppose, however, that as the two species evolve, a pair of proteins in one of the species have traded functions with one another. In that case, reconstructing the initial correspondence would require precisely an edge swap.

Comparing the network alignment problem to the (simpler) sequence alignment problem, one could say that edge swaps at the network level are the analog of compensatory mutations at the sequence level. One could then argue that, just as compensatory mutations can provide important clues for the evolutionary history of the sequences, function exchanges (represented by edge swaps) can provide important indications for the evolutionary history of the protein interaction networks. Unfortunately, function exchanges are much more difficult to detect than compensatory mutations, since network data is noisy, incomplete and unreliable [9]. Nevertheless, an algorithm such as PISwap could be adapted to estimating the number of function exchange events that have taken place during the evolutionary process.

While evolutionary events other than exchanges of function, such as duplications, insertions and deletions of proteins, certainly take place in biological networks [18], this approach can still yield useful biological insights. In addition, the evolutionary distance between two species could in principle be computed from the number of evolutionary events (including function exchanges) that have taken place and could perhaps provide a more accurate estimate than the (appropriately defined and weighted)

edit distance between two orthologous sequences present in those two species, since it would in some sense encompass all the protein sequences at once.

4.3 Discussion

We presented an algorithm for performing a global alignment of two protein interaction networks. In this algorithm, the parameter α plays an important role because it determines the relative importance of the topology data and the sequence data. Although our objective function is identical to that used in the IsoRank algorithm [9], there are a number of important differences. As mentioned in the chapter on Previous Algorithms, IsoRank performs a random walk on the graph $G = G_X \otimes G_Y$, the tensor product of the two networks, where at each step, the walk is restarted with probability $1 - \alpha$ at a node $v = (x, y)$ in G chosen at random from the distribution proportional to the sequence similarity $s(x, y)$ [22]. On the other hand, our algorithm can be thought of as performing a walk on the set of all matchings in the complete bipartite graph, and this walk is not random but has the property that every step increases the value of the objective function. Another difference from IsoRank is that the output of IsoRank in terms of the pairwise alignment scores R_{ij} changes continuously with α, whereas in our algorithm, the set of possible matchings is discrete. It is clear that the interval $[0, 1]$ can be subdivided into non-overlapping subintervals such that on each one, the resulting matching is the same. Finally, our algorithm is not based on a spectral method, unlike both IsoRank and IsoRankN.

It is conceivable to obtain an "ensemble" or "fuzzy" mapping by using the mappings produced with several different values of α. The common part of these mappings would provide a more reliable set of functional orthologs, while the differences between the mappings could be used to evaluate the confidence of the assignments produced. In order to apply that strategy, it suffices to compute a single maximum-weight bi-

partite matching, which is the more time-intensive part of the algorithm, and save it in order to be able to reuse it with different values of the parameter α.

In order to speed up the algorithm without degrading its performance, it is possible to combine it with preprocessing by a graph-partitioning algorithm. For instance, if the two networks, G_X and G_Y, are each partitioned into two pieces (for instance, via a min-cut algorithm [47]), say, (G_X^1, G_X^2) and (G_Y^1, G_Y^2), one could apply the algorithm on each pair of pieces and then find the one that works best. Because there would be few edges between G_X^1 and G_X^2, as well as between G_Y^1 and G_Y^2, the difference in performance would not be significant. However, since the running time is $O(|M|^3)$, dominated in practice by the maximum-weight bipartite matching, if we assume that the number of vertices in both pieces is roughly equal, this would reduce the overall running time by a factor of 2. This strategy, of course, can also in principle be applied with other kinds of partitioning algorithms (such as community detection algorithms [48]).

Finally, as discussed in the previous section, this algorithm could yield new insights into the evolutionary history of the two species being analyzed.

Part II

Regulatory Networks

"Essentially, all models are wrong, but some are useful."
George E. P. Box, British statistician, 1993 Guy Medalist.

In this part of the thesis, we consider the problem of extracting information about putative drug targets from a regulatory network by combining it with gene expression data. The first chapter introduces regulatory networks and the causal reasoning framework. The second chapter describes the problem of drug target validation and describes our methods to approach it. The third chapter discusses some interesting algorithmic aspects of these methods. The final chapter addresses some of the challenges our approach faced and suggests directions for overcoming them in the future.

Chapter 5

Reasoning on Regulatory Networks

A gene regulatory network describes how the products of different genes interact with each other and with other entities inside a cell. These interactions influence the relative abundance of transcripts of these genes, corresponding messenger RNA molecules and proteins. While these interactions are complex, their net effect can be captured experimentally by gene expression profiling. One natural problem is the inference of possible regulatory mechanisms that created the observed effect. We address this problem here.

In this chapter, we introduce regulatory networks and the causal reasoning framework. We then define the computational problems of target validation and target discovery, and describe a caveat relating to the incompleteness of currently available models. We conclude by mentioning the applications currently underway.

5.1 The Nature of Regulation

The particular representation of regulation adopted in our work was based on the so-called CMAP model [49]. This model can be represented as a directed graph

$G = (V, E)$ whose nodes V are molecules or genes, and where a directed edge from node A to node B means that the abundance of B is positively regulated by A (in which case the edge $e = (A, B)$ has a plus sign) or negatively regulated by A (in which case the edge $e = (A, B)$ has a minus sign). Such a model can readily be constructed from a set of interactions curated from a freely available biomedical literature database such as PubMed [50] or from a commercial source such as Ingenuity [51].

5.2 Causal Reasoning

Causal reasoning is a framework in which specific assumptions are made about the way in which regulation happens. Namely, it is assumed that regulation occurs in discrete timesteps and that the directions of regulation are composed in the natural way irrespectively of the abundances of each of the biological entities involved.

Algorithmically, causal reasoning models the effects of a change in the abundance of A on the abundance of Z by tracing the shortest path from A to Z in G and then evaluating its sign (determined by the product of the signs of the edges along the path). If this overall sign turns out to be a plus sign, it is expected that A upregulates Z, and if it is a minus sign, that A downregulates Z.

5.3 Problem Definition

The problem addressed in our work at Pfizer during the summer of 2009 was that of computational target validation and discovery. Roughly speaking, a compound (drug candidate) is generally believed to act on a biological system (such as a tissue or an organism) by affecting one or more targets (molecules or genes) in that system. However, the effects of the compound go beyond the target, as information about the intervention is propagated in the system via a number of mechanisms (usually

based on genetic transcription or molecular interaction). Those cumulative effects, in particular those related to gene expression, can then be experimentally observed and measured (for instance via gene expression profiling, described above). The problems of computational target validation and discovery can then be stated as follows:

Computational Target Validation: Given a model of a biological system, a compound for which the targets are believed to be known, and an experimental gene expression profile for this compound, determine whether the expression profile can be explained by the compound's effect on those targets, assuming the correctness of the model.

Computational Target Discovery: Given a model of a biological system, a compound for which the targets are not known, and an experimental gene expression profile for this compound, determine a set of one or more targets which best explain the expression profile, assuming the correctness of the model.

5.4 Model Incompleteness

In this work we assume that both the targets and the relevant genes are present in the model of the biological system; this assumption does not hold in practice, however, as only roughly 2/3 of the relevant targets and 1/3 of the relevant genes are found in the models we use. Hence, the remainder of the discussion focuses only on those entities (targets and genes) that are indeed present in the model.

The advantage of our methodology is that it is able to suggest directions of regulation for the missing targets and genes, and thus could be used to suggest experiments that would allow one to incorporate these entities into the existing knowledge base. Conversely, as the knowledge base expands, we expect that the predictive power of our methods will improve considerably.

5.5 Current Applications

Preliminary results on the application of our methods in the context of hepatitis C virus (HCV) infection and cardiac compound toxicity will be presented as a poster [2]. In the HCV case, we analyzed a data set by Wherry et al [52] comparing the gene-expression profiles of lymphocytic choriomeningitis virus (LCMV)-specific $CD8^+$ T cells from chronic infection to functional LCMV-specific effector and memory $CD8^+$ T cells generated after acute infection. In addition to well established genes such as IFNA2, we find novel suggestions for interesting intervention points in the network. In parallel, we investigated possible key drivers for cardiotoxicity by applying our method to a set of rat expression profiles generated in response to compounds with known cardiotoxic liabilities. In a pilot study of 10 compounds, we find known cardiotoxicity mechanisms that have already been flagged as relevant. Further investigation with a larger data set is currently under way.

Chapter 6

Drug Target Validation Methods

We now present the computational methods required for addressing the problem of drug target validation as well as the harder problem of drug target discovery. We begin by describing the graph transformation that allows us to restate the problems in a more standard way. We then discuss the computation of distances in the graph, and our novel scheme for scoring and ranking the targets. Finally, we explain how we calculate the significance of our results.

6.1 Causal Graph and Computational Graph

The graph $G = (V, E)$ described in the problem definition is referred to as the *causal graph.* Although it is possible to adapt a regular shortest-paths algorithm (such as breadth-first search if G is unweighted or Dijkstra's algorithm if G is weighted) to find the shortest path from A to Z of a specified sign (plus or minus), it is advantageous to transform $G = (V, E)$ to the *computational graph* $G_C = (V_C, E_C)$ in order to apply standard algorithms to it to produce the desired information. Informally, G_C is obtained by creating two copies of each node of G, one with a + sign, one with a

– sign, and letting the corresponding edges be duplicated as well, so as to separate positive paths (those between nodes of the same sign) from negative paths (those between nodes of the opposite sign). Formally, denoting by $\sigma(e)$ the sign of e, this transformation is defined as follows:

$$V^+ := \{v^+ | v \in V\}, V^- := \{v^- | v \in V\};$$
$$E^+ := \{(u^+, v^+) | (u,v) \in E, \sigma((u,v)) = +\} \cup \{(u^-, v^-) | (u,v) \in E, \sigma((u,v)) = +\};$$
$$E^- := \{(u^+, v^-) | (u,v) \in E, \sigma((u,v)) = -\} \cup \{(u^-, v^+) | (u,v) \in E, \sigma((u,v)) = -\};$$
$$V_C := V^+ \cup V^-, E_C := E^+ \cup E^-.$$

, This transformation also allows us to identify redundant edges (due to the fact that the same interactions may be described in multiple articles) and remove them from the graph in order to make G_C a *simple graph.* Note that we also remove (positive) self-loops, since those can never be a part of a shortest path between two nodes.

6.2 Distance Matrix Computation

We compute the matrix containing the length of all-pairs shortest paths for G_C since it is usually possible to do so for the moderate-sized graphs (10^4 to 10^5 nodes and 10^5 to 10^6 thousand edges) that we typically deal with. In order to reduce memory and time requirements, we note that $d(A^+, B^+) = d(A^-, B^-)$ in G_C, where $d(\cdot, \cdot)$ denotes the distance in the graph, since the paths from A^+ to B^+ and the paths from A^- to B^- in G_C are precisely the positive paths from A to B in G. Similarly, we have $d(A^+, B^-) = d(A^-, B^+)$, so it suffices to compute one-half of the distance matrix, as only the distances from the nodes in V^+ are sufficient. This computation typically takes between 5 seconds and 5 minutes.

6.3 Scoring Function

For both the target validation and the target discovery problems, it is important to define a way to score the "goodness-of-fit" of a given target to a given set of overexpressed and underexpressed genes. Note that information on whether the target is upregulated or downregulated by the compound is usually available, and furthermore, the gene expression data allows us to determine the subset G^+ of all genes that are significantly overexpressed and the subset G^- of all genes that are significantly overexpressed. We denote $\sigma(g)$ the sign of gene g (+ if it is significantly overexpressed, − if it is significantly underexpressed), and we also denote $\bar{g}$ the gene g taken with the opposite sign. Let $G^{\pm} := G^+ \cup G^-$ denote the set of all significant genes in the gene expression profile. We can think of a target t as "good" if, for most genes $g \in G^{\pm}, d(t,g) < d(t,\bar{g})$.

We assume that each edge represents one unit of time. For example, information takes twice as long to propagate along a path of length 2 than along a single edge. We further assume that only the information that arrives at the earliest possible time is relevant, regardless of what happens at later times. Though necessary in order to make progress, both of these assumptions are certainly problematic. Some possible ways of addressing these problems are described in the Future Work chapter.

Based on our assumptions, a scoring function can be designed as follows. For each gene-target pair (g,t), let $d_{\min}(t,g) := \min\{d(t,g), d(t,\bar{g})\}$. In other words, $d_{\min}$ is the time it takes information from t to reach g (with either sign). If $d(t,g) < d(t,\bar{g})$, then g is said to be *correct* with respect to t. If $d(t,g) > d(t,\bar{g})$, it is said to be *incorrect* with respect to t. If $d(t,g) = d(t,\bar{g}) < \infty$, it is said to be *ambiguous* with respect to t. Finally, if $d(t,g) = d(t,\bar{g}) = \infty$, neither g nor $\bar{g}$ is reachable from t, and g is said to be *unknown* to t.

Let $\beta \in (0,1)$ be a diffusion parameter (which, roughly speaking, corresponds to

the dissipation of information along an edge of G_C). Our scoring function for a target t and a geneset $G^{\pm}$ measures the amount of information supporting t as the correct target for the geneset. The scoring function is defined by

$$\begin{aligned} S(t, G^{\pm}) &:= \sum_{g \in G^{\pm}, g \text{ correct w.r.t. } t} \beta^{d_{\min}(t,g)} - \sum_{g \in G^{\pm}, g \text{ incorrect w.r.t. } t} \beta^{d_{\min}(t,g)} \\ &:= \sum_{g \in G^{\pm}} \beta^{d_{\min}(t,g)} \operatorname{sgn}[d(t, \bar{g}) - d(t, g)]. \end{aligned}$$

Here, sgn denotes the *signum*(x) function, which is 1 for $x > 0, -1$ for $x < 0$ and 0 for $x = 0$.

An alternative scoring function, which has not been tested due to the large number of parameters it requires, is

$$S(t, G^{\pm}) := \sum_{g \in G^{\pm}} w(g)[d(t, \bar{g}) - d(t, g)],$$

where $w(g)$ is a non-negative weight reflecting our confidence level that g actually belongs to the set $G^{\pm}$.

With either scoring function, high positive values correspond to good targets and high negative values, to bad ones. Note that both functions are linear with respect to $G^{\pm}$ and also that $S(t, G^{\pm}) = -S(\bar{t}, G^{\pm})$, in other words, changing the sign of the target changes the sign of the score. Both of these are desirable properties.

6.4 Target Ranking

In order to compare two targets, t_1 and t_2, for a given geneset $G^{\pm}$, one can compare either their scores with $G^{\pm}$ or the ranks of that score among the scores for all possible

targets. It is clear that t_1 is better than t_2 (based on our assumptions) if $S(t_1, G^{\pm}) > S(t_2, G^{\pm})$ in absence of any additional information. However, it could be the case that t_1 scores highly with any geneset, not just $G^{\pm}$, due for instance to its central position in the graph G_C, whereas t_2 scores particularly well with $G^{\pm}$. For this reason, it is important to understand not only how well t_1 scores with $G^{\pm}$ relative to other targets, but also how well it scores with $G^{\pm}$ relative to other genesets with comparable properties. The latter question is addressed in the following subsection. For now, let us just mention that we record both the score and the rank (among all possible targets) for each of the candidate targets (be it a restricted set in the target validation mode or a less restricted set in the target discovery mode).

6.5 Significance Calculation

In order to assess the significance of a particular target-geneset pairing $(t, G^{\pm})$, we perform the following tests.

First, we generate several random genesets of the same size as $G^{\pm}$, say $\{G_1, \ldots, G_N\}$. Note that some earlier work on causal analysis advocated conserving not only the size of the geneset, but also the degrees of each gene [53]; however, due to the incomplete nature of the graph, we have chosen not to use degree information for the random generation of genesets. We do enforce the requirement that the number of genes with $\sigma(g) = +$ (and hence, also the number of genes with $\sigma(g) = -$) be the same in each G_j as it is in $G^{\pm}$. For each $1 \leq j \leq N$, we look at the rank r_j of $S(t, G_j)$ in the ensemble of all scores, $S(t_i, G_j)_{i \in I}$, where I is the indexing set for the candidate targets. The significance p_0 of the rank r_0 of $S(t, G^{\pm})$ in the ensemble of all scores, $S(t_i, G^{\pm})_{i \in I}$, is then estimated as $p_0 := |\{j | r_j > r_0\}|/N$, i.e. the fraction of random genesets that give t a higher rank than $G^{\pm}$.

Second, we compute the fraction of all genesets of the same size as $G^{\pm}$ that

give a higher score with t than $G^{\pm}$. Surprisingly, this computation can actually be performed analytically, as described in the chapter on Algorithmic Aspects. This gives us a value $p_1 := |\{j \in [L] | S(t, G_j) > S(t, G^{\pm})\}|/L$, where L is the total number of possible genesets of the same size as $G^{\pm}$ (with the same sign distribution as $G^{\pm}$), and $[L]$ denotes the set $\{1, 2, \ldots, L\}$. Note that this value is different from the one estimated by p_0 since this computation only takes the single target t into account.

Lastly, we also look at an estimate of how good the score remains under changes to the underlying causal graph. In order to do this, we simply look at the rank of $S(t, G^{\pm})$ in different random graphs sharing some connectivity properties of the causal graph G, and define q_0 to be the fraction of the graphs where this rank is better than that for G. This graph randomization procedure is also described in more detail in the following chapter, Algorithmic Aspects.

Chapter 7

Key Algorithmic Aspects

In this chapter, we discuss some important algorithmic aspects of the methodology described in the previous chapter. Because a causal graph typically contains on the order of $10^4 - 10^5$ nodes and $10^5 - 10^6$ edges, it is critical to use highly scalable algorithms both in terms of running time and in terms of memory, while preserving the correctness of their output. We begin by discussing a procedure for randomizing a causal graph, a problem that has not been addressed in previous work where the focus has been on undirected and/or unsigned edges. We also include a pseudocode for the proposed procedure. Then we discuss two additional aspects - adding weights to the graph to improve its predictive power, and a memory-optimized algorithm for doing convolutions.

7.1 Graph Randomization

In order to randomize a given causal graph G, we try to produce a graph which preserves some vertex connectivity properties as G, but has a very different set of edges. In our work, we settle on a specific definition of graph randomization, which,

however, has not (to our knowledge) been studied in this particular form up to now.

Let us define, for $v \in V$, $d^+_{\text{in}}(v)$ as the number of incoming edges labeled by $+$, which we call the *positive in-degree* of v. Similarly, we define $d^-_{\text{in}}(v)$, the *negative in-degree* of v. In an analogous way, $d^+_{\text{out}}(v)$ and $d^-_{\text{out}}(v)$ are called the *positive* and *negative out-degree* of v, respectively. As an illustration, if $V = \{u, v\}$ and $E = \{(u, v), (v, u)\}$ with $\sigma((u, v)) = +$ and $\sigma((v, u)) = -$, we have

$$d^-_{\text{in}}(u) = d^+_{\text{out}}(u) = 1,\ d^+_{\text{in}}(u) = d^-_{\text{out}}(u) = 0;$$
$$d^+_{\text{in}}(v) = d^-_{\text{out}}(v) = 1,\ d^-_{\text{in}}(v) = d^+_{\text{out}}(v) = 0.$$

The idea for randomizing a given causal graph is that we preserve the positive and negative in-degree and out-degree of each vertex but allow the connections (edges) to vary freely subject to those constraints. More precisely, let G be a causal graph, and let $F(G)$ be the family of all graphs having the same positive and negaitve in-degree and out-degree sequences (four sequences in total). We would like to produce a graph from $F(G)$ uniformly at random.

The problem of random graph generation has been well studied in the context of undirected graphs [32, 54] and less well studied in the context of directed graphs [55, 56]. It has not at all been studied for graphs with signed edges, such as the causal graphs we deal with here. Nevertheless, the techniques for generating random graphs carry over to the signed edge setting, with a few caveats explained below.

The standard - and most commonly used - algorithm for graph randomization is *edge switching.* In edge switching, one step of the algorithm consists of picking two edges, say (a, b) and (c, d), and interchanging their endpoints to produce two new edges, (a, d) and (c, b), provided that these new edges keep the graph connected and simple (no parallel edges and no self-loops). This elementary operation suffices to produce any connected simple graph with the same degree sequence as a given one

in the undirected setting [54]. In the directed setting, however, a further operation needs to be added which we call *triangle flipping*. Given the edges $(a,b),(b,c),(c,a)$, which form a directed cycle of length 3, triangle flipping replaces them with the edges $(a,c),(c,b),(b,a)$. This operation, combined with edge switching, can be shown to suffice in the directed case [55].

In the signed setting, the natural generalization is to only perform edge switches or triangle flips on edges of the same sign (otherwise, the degree sequences may be altered). The main question is whether this is sufficient to cover all of $F(G)$. The general answer is no, but it is yes in most practical situations we worked with. Note that we only require the underlying (undirected, unsigned) graph to be connected.

First, in the signed setting, two additional obstacles arise. (We strongly believe, but have not yet proved, that these are the only two possible obstacles.) The first one is the presence of *strong quadrilaterals*: pairs of edges $(a,b),(c,d)$ of the same sign (say, $+$) such that the edges $(a,d),(b,c)$ of the opposite sign $(-)$ also exist in the graph. In this case, the graph obtained by changing the signs on the edges of a strong quadrilateral has the same degree sequences as the original one, but it cannot be obtained by legal edge switching because an edge switch would destroy simplicity. The second obstacle is the presence of *strong triangles*: triplets of edges $(a,b),(b,c),(c,a)$ of the same sign (say, $+$) such that the edges $(a,c),(c,b),(b,a)$ of the opposite sign $(-)$ also exist in the graph. The graph obtained by changing the signs on all the edges of a strong triangle has the same degree sequence as the original one, but it cannot be obtained by legal edge switching or triangle flipping either.

Now, while these examples show that it is impossible to produce all the graphs in $F(G)$ via local operations in general, the situation is not so bleak for large sparse causal graphs we deal with in practice. We will show how to deal with strong triangles in a large sparse causal graph; the construction for strong quadrilaterals is similar.

Let a, b, c form the vertices of a strong triangle, and assume without loss of generality that $(a, b), (b, c), (c, a)$ are positive edges. Suppose that there are positve edges $(a_1, a_2), (b_1, b_2), (c_1, c_2)$ in the graph such that $\{a_1, a_2, b_1, b_2, c_1, c_2\}$ are all distinct from $\{a, b, c\}$ and from each other. The following procedure, illustrated below in figure 7-1, "flips" both parts of the strong triangle:

1. Opening: Switch (a, b) with $(c_1, c_2), (b, c)$ with $(a_1, a_2), (c, a)$ with (b_1, b_2).

2. Flipping: Flip the triangle $(a, c), (c, b), (b, a)$ (which can now be done).

3. Closing: Switch (a_1, c) with $(a, c_2), (b_1, a)$ with $(b, a_2), (c, b_2)$ with (c_1, b).

4. Restoring: Switch (b_1, a_2) with (c_1, b_2) and then switch (c_1, a_2) with (a_1, c_2).

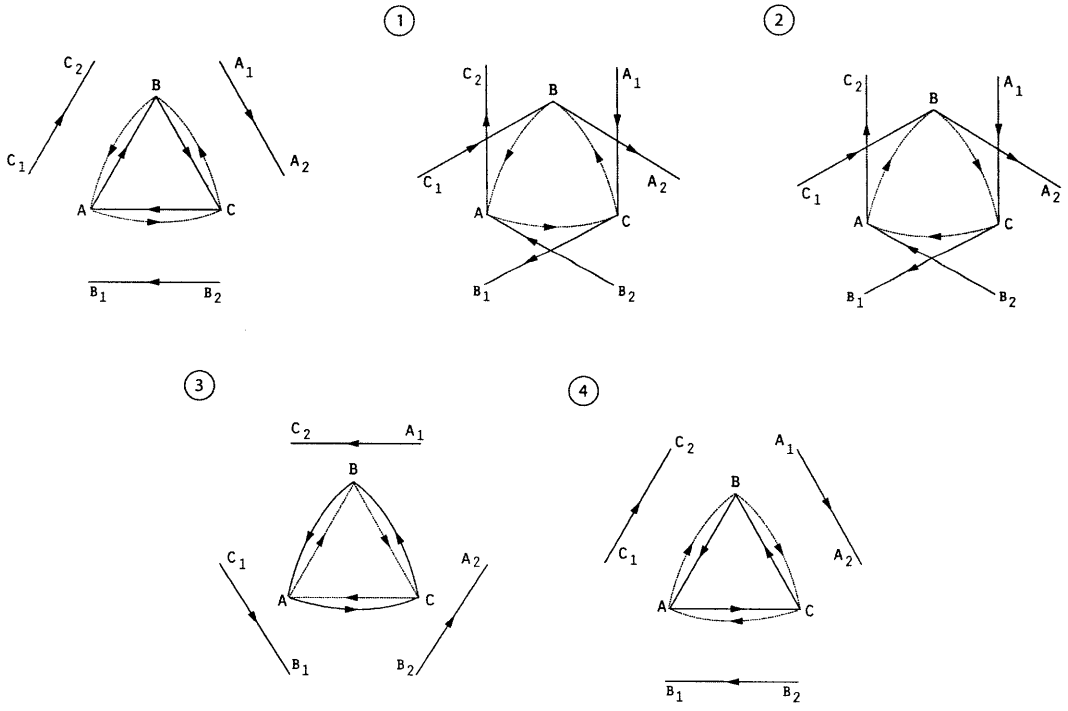

Figure 7-1: The sequence of operations that flips a strong triangle. Dotted lines indicate negative edges, while solid lines indicate positive edges.

Thus, provided that the auxiliary edges $(a_1, a_2), (b_1, b_2), (c_1, c_2)$ exist for which operations 1 and 4 are possible (2 and 3 are always possible as a consequence), a strong triangle can be flipped using the basic operations. In our implementation, we try to bypass the obstacles in the way described above only whenever they are discovered during the randomization process. Its pseudocode is given below.

Input: a causal graph G, a parameter γ for the number of iterations (default: 3).

Output: a random causal graph in $F(G)$; 1 if an obstacle could not be bypassed.

Iteration: for i **from** 1 **to** $\gamma \cdot |E(G)|$ **do**

1. Select two edges (a, b) and (c, d) at random from $E(G)$ and **increment** i
2. **if** $\sigma(a, b) \neq \sigma(c, d)$ **or** $a = c$ **or** $b = d$ **goto** 1
3. **if** $b = c$ **goto** 4 **else goto** 5
4. **if** $(d, a) \in E(G)$ and $\sigma(d, a) = \sigma(a, b)$ **goto** 6 **else goto** 1
5. **if** *switchable*$((a, b), (c, d))$ **goto** 7 **else goto** 9
6. **if** *flippable*$((a, b), (c, d), (d, a))$ **goto** 8 **else goto** 10
7. *switch*$((a, b), (c, d))$, **goto** 1
8. *flip*$((a, b), (c, d), (d, a))$, **goto** 1
9. **if** $(a, d), (d, b), (b, a) \in E(G)$ **and** $\sigma(a, d) = \sigma(d, b) = \sigma(b, a) \neq \sigma(a, b)$ **goto** 11 **else goto** 1
10. **if** $(a, d), (c, b) \in E(G)$ **and** $\sigma(a, d) = \sigma(c, b) \neq \sigma(a, b)$ **goto** 12 **else goto** 1
11. **if** *correctable*$((a, b), (c, d), (d, a), (a, d), (d, b), (b, a))$ **goto** 1 **else return** 1
12. **if** *correctable*$((a, b), (c, d), (a, d), (c, b))$ **goto** 1 **else return** 1

return G

7.2 Distance Computation, Adding Graph Weights

To compute the distances in G_C, we use the *shortest.paths* function as implemented in the igraph package [57]. In order to reduce the multiplicities of each distance (to give a higher differentiation power to our method), the edges can be weighted, and the distances in a weighted version of G_C can be computed. This requires no modification to the code since the *shortest.paths* function uses breadth-first search for unweighted and Dijkstra's algorithm for weighted graphs. One possible weighting scheme is assigning to each edge the in-degree of its head, reflecting a "delay" proportional to the total number of entities trying to interact with a given one.

7.3 Convolutions

In order to perform the analytic computations described in the previous chapter, we need to compute the probability density function of the variable $X_1 + \cdots + X_n$ where the X_i are independent and identically distributed according to a given probability density function ρ . This is given by the n-fold *convolution* of ρ with itself.

Rather than using standard methods based on the Fast Fourier Transform (FFT) [58] to do this, which require a number of equally spaced values for ρ (most of which are 0 in the typical cases we work with), we choose to work with a vector representation u, f of a discrete probability density function taking the value u_i with the frequency f_i. To compute the convolution of two such density functions, u, f and $\bar{u}, \bar{f}$, we simply form all possible sums $u_i + \bar{u}_j$ and assign them the frequency $f_i \cdot \bar{f}_j$. As a postprocessing step, we sum the frequencies corresponding to identical values. This approach is not necessarily speed-optimized, but it is memory-optimized, allowing for the computation of convolutions for fairly large values of n.

Chapter 8

Directions for Future Work

Preliminary experiments on actual causal graphs have shown that, while promising, the methodology developed in our work needs to be sharpened in order to improve its predictive power. This chapter presents some possible ways of doing that. We begin by describing a generalization of the original problem to sets of targets. We then propose additional methodological improvements that are both sound and algorithmically tractable.

8.1 Target Set Discovery

In this work, the focus has been on the discovery of a single target. Alternatively, it may be interesting to find a small set of targets that can explain a gene expression profile, since many compounds affect more than one target. A good starting point is to formulate this problem as a set cover problem [59], which, although NP-hard, admits a number of good heuristics and can be solved exactly in many small-dimensional instances (such as, for instance, when the desired set cover is small).

Specifically, let S_i be the subset of $G^{\pm}$ correctly explained by the i-th target. Our

goal could then be to cover $G^{\pm}$ with the smallest possible number of S_i. This is readily encoded as an integer linear program [60]:

$$\min \sum_i x_i \text{ subject to } \sum_{j \in S_i} x_i \geq 1 \; \forall \; j, \; x_i \in \{0,1\} \; \forall \; i.$$

This approach can be refined by requiring that all correct explanations of each gene be better than all corresponding incorrect explanations (if any) by other targets. Let us construct the directed graph $G = (V, E)$ (different from the original causal graph) with V the set of possible targets and an edge (i, j) if and only if there is a gene g which is correctly explained by i and incorrectly explained by j and the explanation provided by j is better than the one provided by i. The new problem becomes:

$$\min \sum_i x_i \text{ subject to } \sum_{j \in S_i} x_i \geq 1 \; \forall \; j, \; x_i + x_j \leq 1 \; \forall \; (i,j) \in E(G), \; x_i \in \{0,1\} \; \forall \; i.$$

There are other similar refinements that introduce softer constraints into the system, leading to similar formulations. In all cases, solving the fractional relaxation of the integer linear program followed by rounding can lead to good approximate solutions. If only a small number of targets is being sought, commercial software such as CPLEX [61] or even an open-source solver such as lpsolve [62] can solve the integer linear program optimally in a reasonable amount of time.

8.2 Parameter Training

In this work, a variety of values for β have been used. A valuable approach would be to use the set of known targets to learn the best value for β (or any other model parameters) and then use that value for all future computations.

8.3 Incorporating All Paths

In this work, only the shortest path between two nodes has been considered. A possible extension would be to consider all paths between two nodes, weighted by a function exponentially decreasing in the length of the path (in order to converge to a finite value). This approach has been described in an earlier technical report [63].

8.4 Trees vs. Shortest Paths

In this work, all the shortest paths from a target to the relevant genes were selected independently. Alternatively, it may make sense to find the smallest set of edges that connect a target to all the relevant genes. This problem is known as the Steiner Tree problem [64], and although it is NP-hard in the general case, good heuristics exist for finding approximate solutions.

8.5 Removing Indirect Connections

Since the network we are working on is not a "true" regulatory network, but rather one derived by literature curation, it must necessarily contain shortcuts (non-elementary regulatory relationships). As an example, if A upregulates B and B upregulates C, a regulatory network would not also contain an edge from A to C, whereas a human researcher familiar with A, B and C is likely to use the shortcut and state that A upregulates C in a paper.

In the general case, if one defines a connection as indirect if removing it does not disconnect its endpoints in the computational graph, the problem of removing all indirect connections is the algorithmic task of computing the *transitive reduction* of a directed graph. It was shown by Aho, Garey, and Ullman [65] that this problem has

the same complexity as the problem of *transitive closure*, that of connecting by an edge all pairs of nodes that are connected by a directed path. Since this complexity is at least quadratic in the number of nodes [66], a full transitive reduction may be impractical for our purposes.

Fortunately, some of the interactions are actually labeled as "inferred", and those can be removed from the network provided that a path between their endpoints exists in the computational graph (corresponding to a path in the causal graph of the correct sign). Removing such edges alters some of the distances in the graph but preserves the connectivities (in other words, it does not disconnect nodes that had been connected before). It is then possible to decide which of the edges can be removed by removing all of them and recomputing the shortest paths between the vertices incident on them. Those that should be kept are precisely those whose endpoints become disconnected after their removal.

Part III

Metabolic Networks

"There is nothing so practical as a good theory."
Kurt Lewin, founder of modern social psychology.

In this part of the thesis, we investigate the information that can be extracted from a metabolic network. In the first chapter, we introduce constraint-based models, compare them with other models of metabolism and describe some sources of incompleteness. In the second chapter, we introduce a novel concept of duality in a metabolic network and show how it can be harnessed to perform a complete structural analysis. In the third chapter, we describe several well-studied tasks which are grouped under the concept of minimality, and introduce a new algorithm for addressing them. In the fourth chapter, we show how a metabolic network can be refined in light of different kinds of experimental data. In the fifth and final chapter, we describe the details of a Python implementation of all the algorithms discussed in the previous chapters, and justify some of the choices we made.

Chapter 9

Constraint-Based Metabolic Model

Metabolism can be defined as the set of chemical reactions that sustain life. It has classically been subdivided into specific *pathways* centered around a particular molecule (*metabolite*), and these pathways were studied individually. However, with the emergence of whole-genome sequencing capabilities, it became possible to produce an almost complete list of all the reactions available to an organism, a *genome-scale metabolic reconstruction*, and thus, study metabolism on the systemic level. In this chapter, we briefly discuss different approaches to a system-level study of metabolism before settling on a particular model. We specify the components of the model and the assumptions behind it in the rest of the chapter, and conclude with an example.

9.1 Frameworks for Analyzing Metabolism

Approaches based on *kinetics* look at a biochemical system starting away from *equilibrium*, and model the change in *concentrations* of the different metabolites over time using a system of ordinary differential equations. These approaches are very successful in describing the behavior of small and medium-scale *biochemical reactors*,

but require the knowledge of various reaction rates and, in the case of enzymatic reactions described by the Michaelis-Menten equation, an additional constant specific to each reaction. Biochemical Systems Analysis (BSA), pioneered by Savageau, is an application of these approaches to larger systems [67, 68]. However, the large number of parameters required makes it impractical for analyzing genome-scale models of metabolism.

A very interesting line of work was pioneered by Horn and Jackson [69]. They investigated the properties of a biochemical system that could be inferred simply from its *deficiency*, an integer-valued function based only on the *stoichiometric matrix*, and the system's graphical representation. It is concerned with features that are independent of the specific rates of reaction provided that they follow the *law of mass action*. This line of work has been carried on by Feinberg [70, 71] and remains a moderately active area of research. Unfortunately, the low-deficiency conditions under which a biochemical system exhibits invariance under kinetic constants are typically too restrictive for most large-scale metabolic systems.

Metabolic Control Analysis (MCA) is yet another framework for investigating metabolic systems. MCA quantifies how variables, such as fluxes and species concentrations, depend on network parameters. In particular it describes how network dependent properties, called *control coefficients*, depend on local properties called *elasticities* [72]. Several theoretical results have been obtained within this framework [73], but their applications to actual systems appear to be limited, in part due to the amount of parameters.

Finally, several researchers have focused on the study of the graphs representing a metabolic system. There are indications that these graphs, properly analyzed, reveal a lot of relevant information, as evidenced by the gene essentiality predictions by Wunderlich and Mirny [74]. In addition, several *de novo* pathway discovery algorithms

have been proposed that use the graphical representation of a system [75, 76], and these have been favorably compared with the constraint-based approaches [77]. There has also been an interesting discussion of whether metabolic networks are *small-world* and *scale-free* [78, 79, 80]. Ultimately, however, graph-based methods appear to have only limited predicted power because they cannot take the stoichiometry of the reactions into account.

All the approaches discussed so far suffer from one of two limitations. Either they require a large number of parameters (BSA and MCA), which makes them poorly scalable, or they cannot represent the full gamut of metabolic behaviors, either because of their narrow applicability (low-deficiency conditions) or their disregard for stoichiometry (graph-based analyses). Constraint-based models address these limitations because they do not require any knowledge of parameters, yet they apply to a range of biochemical systems and represent a large variety of metabolic behaviors. The rest of the chapter details the assumptions made in a constraint-based model.

9.2 Reactions, Metabolites and Mass Balance

A constraint-based model is centered around reactions. Each reaction is *catalyzed* by one or more enzymes, which is in turn the product of one or more genes. The relation between genes, enzymes and reactions is not always one-to-one; Boolean formulas such as G_1 OR (G_2 AND G_3) indicate that a reaction can be catalyzed by the product of gene 1 or a combination of products of genes 2 and 3. We will assume that all isozymes (enzymes catalyzing the same reaction), say $G_1, \ldots, G_t$, are grouped together, and that the reaction is then catalyzed by G_1 OR ... OR G_t.

Each reaction interconverts different metabolites. The metabolites that are consumed are called *reactants*, while those that are produced are called *products*. In addition, there may be one or more *cofactors* (helper molecules) required to make the

reaction happen. At present, as many as 40% of all reactions lack information about the genes, enzymes or cofactors required for them to occur [81].

An important feature of a constraint-based model is that, unlike a graph-based model, it takes into account the *stoichiometry* of the reactions. In other words, the proportions between the amounts of different reactants and products in a reaction are fixed. Typically, this requires that the number of each type of atoms is the same for reactants and for products; a reaction with this property is called *elementally balanced.*

Metabolites that are external to the cell (or the biochemical system under consideration) are labeled with the symbol $[e]$ to distinguish them from their counterparts inside the cell. All reactions are partitioned into two groups: the exchange reactions $\mathcal{E}$ which contain external metabolites, and the internal reactions $\mathcal{C}$ which do not.

The constraint-based models of some eukaryotic organisms such as *S. cerevisiae* may include additional information about the location of each metabolite in a specific compartment of the cell [82, 83]. These compartments usually correspond to cell organelles, such as the Golgi apparatus, the lysosome and the vacuole. In the context of a compartmentalized model, identical metabolites in different compartments are differentiated by a one-letter compartment abbreviation in round or square brackets (such as [g], [l], [v] for the three organelles listed above), and exchange reactions also include all reactions exchanging metabolites between compartments.

A useful way of representing the set of reactions is via the *stoichiometric matrix* S , where $S_{i,j}$ is the coefficient of the i-th metabolite in the j-th reaction. This coefficient is negative for reactants, positive for products, and 0 for metabolites not involved in the reaction. We will assume that these *stoichiometric coefficients* are rational numbers. The i-th row of S represents metabolite i, and the j-th column of S, reaction j. External metabolites are not explicitly included in the stoichiometric

matrix S, although they are included in the reactions.

In kinetic models, the time behavior of the metabolic network is described by the system of ordinary differential equations [84]

$$\dot{x} = Sv(x), \tag{9.1}$$

where S is the stoichiometric matrix, x is the vector of concentrations and v is the vector of *fluxes* (or reaction ticks). In constraint-based models, the system is assumed to be at a *quasi-steady-state*, so that each internal metabolite is balanced: $\dot{x} = \mathbf{0}$. The state of the system is represented by the vector of fluxes (*mode*) v. The quasi-steady-state assumption implies that v can be a mode of the system only if

$$Sv = \mathbf{0}. \tag{9.2}$$

9.3 Thermodynamic Information

An additional bit of information that is sometimes available about each reaction in a constraint-based model is whether it is *reversible* (able to proceed in both the forward and reverse direction) or *irreversible* (only able to proceed in the forward direction). In general, the reversibility of a reaction depends on the conditions in which the reaction takes place, e.g. the pressure, the temperature and the pH. However, the range of variation of these parameters in the intracellular environment is typically narrow enough to determine the direction of some reactions. When information about the reversibility of a reaction is not available, we assume it to be reversible.

The set of all irreversible reactions is denoted by $\mathcal{I}$. If $\mathcal{I} = \emptyset$, we call the network *fully reversible*. Networks are postulated to be fully reversible in situations when, for instance, one tries to infer the direction of some of the reactions computationally [85].

In the other extreme case, if $\mathcal{I} = \{1, 2, \ldots, n\}$ (where n denotes the total number of reactions), we call the network *fully irreversible*. Fully irreversible networks are often obtained when each reversible reaction is split into its forward and backward reaction by a process known as *reconfiguration* [86].

The second assumption in a constraint-based model is that an irreversible reaction can only have a non-negative flux at steady-state. This can be written as

$$v_i \geq 0 \; \forall \; i \in \mathcal{I}. \tag{9.3}$$

It is possible to further constrain the numerical values of a flux by introducing constraints of the form $\alpha_i \leq v_i \leq \beta_i$. However, these almost always have one of two forms, $0 \leq v_i \leq B$ or $-B \leq v_i \leq B$ for some constant B, and because the model is linear, such constraints are constraints on either the magnitude or the sign of a flux, and can be disregarded in the type of essentially qualitative analysis we will conduct.

In addition to the constraints on irreversible reactions, the *second law of thermodynamics* is sometimes included as an additional constraint on the system, leading to so-called Energy Balance Analysis (EBA) [87]. Applied to a chemical reaction, this law states that the entropy of the reaction is always nondecreasing. It follows that fluxes must flow from reactants of higher *chemical potential* to ones of lower chemical potential [88]. Therefore, the EBA constraint can be formulated as

$$v_i w_i \leq 0 \; \forall \; i \in \mathcal{C} \text{ and } v_i = 0 \implies w_i = 0 \; \forall \; i \in \mathcal{C}, \tag{9.4}$$

where w is the vector of chemical potentials for each reaction.

Furthermore, since the chemical potential of a reaction is the sum of the chemical potentials of the metabolites participating in the reaction weighted by their stoichiometric coefficient, there is an additional constraint on w, namely, that it equals $y^T S$

for a vector y encoding the chemical potential of each metabolite. We can write this constraint as

$$w \in \text{Row}(S_{\mathcal{C}}), \tag{9.5}$$

where $S_{\mathcal{C}}$ is the submatrix of S with columns in $\mathcal{C}$ and $\text{Row}(\cdot)$ denotes the rowspace.

Under conditions of constant temperature and pressure, the chemical potential is directly proportional to the *Gibbs free energy* [89]. The value of the Gibbs free energy of a reaction, ΔG_r^0 is typically difficult to obtain. Nevertheless, it is available for some of the reactions in some well-studied model organisms [90]. Estimates of ΔG_r^0 can also be obtained by computational methods [91]. Using these estimates, further constraints can be added to the constraint-based model which include the concentrations of metabolites and chemical potentials. We defer the discussion of these constraints until the end of the chapter on Model Refinement.

9.4 The Growth or Biomass Reaction

Many constraint-based models (especially those of bacteria) make the additional assumption that the organism maximizes a specific reaction called *growth* or *biomass reaction*, usually denoted by the subscript "biomass", which represents the synthesis of all metabolites it needs in order to replicate itself, taken in definite proportions. As we will see later in the chapter on Model Refinement, this assumption is the weakest link in a constraint-based model, and it must be made with great caution if it is to be made at all. For this reason, we do not make this assumption in most of the work that follows, and only include it here for the sake of completeness.

The biomass composition in terms of metabolites is usually defined experimentally. However, there are a few computational approaches trying to infer the biomass composition based on the fitness between the resulting model and experimental flux

data [92, 93, 94]. Some metabolic reconstructions contain different possible biomass reactions, with the appropriate one selected based on the simulated conditions [95].

If we accept the assumption of biomass maximization, then the particular flux vector v^* that will be selected out of all the ones allowed by the constraints described above will be the one with the maximal value of v_{biomass}. Denoting by F the space of flux vectors satisfying equations (9.2 - 9.5) (some of which we may choose not to include in the model), we can then write

$$v^* \in \underset{v \in F}{\operatorname{argmax}}\, v_{\text{biomass}}. \tag{9.6}$$

The argmax is usually a set and not a single vector because the model is typically *underconstrained* and thus allows multiple optimal solutions [96]. However, the optimal growth rate,

$$\mu := \max_{v \in F} v_{\text{biomass}}, \tag{9.7}$$

is well-defined.

9.5 A Complete Constraint-Based Model

We do not describe the process of creating a genome-scale metabolic reconstruction here. This is a laborious process which is usually only partially automated. A detailed step-wise protocol for creating such a reconstruction was recently proposed by Thiele et al. [97]. Our concern here is only the steps that follow a reconstruction.

Three steps need to be performed to convert a reconstruction to a computable model. First, the cell boundary needs to be defined. Second, the list of reactions needs to be transformed into a stoichiometric matrix. Finally, constraints need to be imposed on the resulting model.

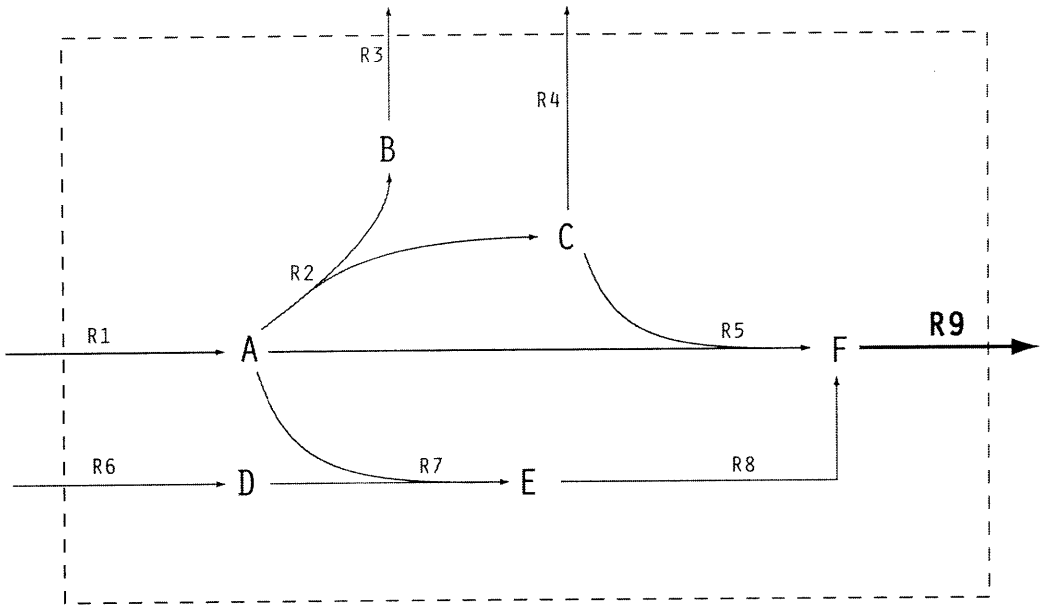

Figure 9-1: An example metabolic network (6 metabolites, 9 reactions).

An example network, adapted from Klamt et al [98], is illustrated in figure 9-1. It contains $m = 6$ internal metabolites and $n = 9$ reactions. The internal metabolites are $\{A, B, C, D, E, F\}$ while the external metabolites are $\{A[e], B[e], C[e], D[e], F[e]\}$. The list of reactions is:

$$R_1 : A[e] \rightarrow A \qquad R_2 : A \rightarrow B + C \qquad R_3 : B \rightarrow B[e]$$

$$R_4 : C[e] \leftrightarrow C \qquad R_5 : A + C \rightarrow F \qquad R_6 : D[e] \rightarrow D$$

$$R_7 : A + D \rightarrow E \qquad R_8 : E \rightarrow F \qquad R_9 : F \rightarrow F[e]$$

The exchange reactions are $\mathcal{E} = \{1, 3, 4, 6, 9\}$, $\mathcal{C} = \{2, 5, 7, 8\}$ are the internal reactions. Reaction 4 is reversible, the irreversible reactions are $\mathcal{I} = \{1, 2, 3, 5, 6, 7, 8, 9\}$.

The stoichiometric matrix is $S :=$

$$\begin{pmatrix} 1 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 \end{pmatrix}$$

We impose (9.2) and (9.3) on the flux vector $v := [v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9]$:

$$Sv = \mathbf{0}, v_i \geq 0 \text{ if } i \neq 4. \tag{9.8}$$

If we were to include the energy balance constraints (9.4) and (9.5), we would add

$$w := [w_2, w_5, w_7, w_8] \in \text{Row}(S_C), w_i \leq 0 \; \forall \; i \text{ and } v_i = 0 \implies w_i = 0 \; \forall \; i. \tag{9.9}$$

Finally, if we were to accept the hypothesis in (9.6) that the flux through the biomass reaction, which in this network is reaction 9, is maximized, we would write

$$v^* \in \underset{v \in F}{\text{argmax}} \, v_9. \tag{9.10}$$

Chapter 10

Structural Analysis of Metabolism

Depending on the particular set of assumptions we select from the ones described in the previous chapter, we can draw different kinds of inferences from a constraint-based model. This chapter describes the structural insights that can be obtained on the basis of only the first two assumptions: mass balance at steady-state and nonnegativity of fluxes through irreversible reactions. We begin by introducing a very important concept that will serve as the underlying theme for much of the analysis that follows: duality in metabolic networks. We then go on to discuss three types of structural features it can help us uncover: blocked reactions (those that cannot have any flux through them at steady-state), effectively unidirectional reactions (reversible reactions that cannot in fact sustain a flux in both directions) and enzyme subsets (groups of reactions whose fluxes are in specific proportions). We end by describing a complete model reduction which converts the model to a unique canonical form, another new concept in the study of metabolic networks.

10.1 Duality in Metabolic Networks

Duality in our work refers to vector space duality, which is different from Boolean duality described in [99]. It is also different from, though closely related to, linear programming (LP) duality used, for instance, to search for a good objective function [92]. We begin by considering fully reversible networks, which make the duality most explicit, and then adapt it to fully irreversible networks.

Let us recall that we are currently only using the constraints described in equations (9.2) and (9.3). The set F of all *modes* (flux vectors) satisfying these equations is called the flux cone. F is a cone since any positive scalar multiple of a mode is still a mode, and F is convex because it is defined by linear equations and inequalities.

The *support* $R(v)$ of a mode v is defined as the set of all reactions that are active in v; in other words, $R(v) := \{i | v_i \neq 0\}$. We also define the *positive support* $R_+(v)$ and the *negative support* $R_-(v)$ as the set of all reactions that have a strictly positive (strictly negative) flux in v, respectively. In other words, $R_+(v) := \{i | v_i > 0\}$ and $R_-(v) := \{i | v_i < 0\}$. We then have $R(v) = R_+(v) \cup R_-(v)$.

The modes that have minimal support are called *elementary flux modes*(EFMs). In other words, v is an elementary flux mode if v is a mode and any other mode w such that $R(w) \subsetneq R(v)$ must be the trivial mode $w = \mathbf{0}$. The number of reactions in a mode v, $|R(v)|$, is called its length.

Given a reaction i, a *cut set* for this reaction is a subset X of reactions not containing reaction i such that constraining these reactions to have flux 0 makes reaction i unable to have a non-zero flux. In other words, X is a cut set for reaction i if any mode v of S with $v_X = \mathbf{0}$ has $v_i = 0$. Here, v_X denotes the part of the vector v whose entries correspond to the reactions in X. A cut set X for reaction i is a *minimal cut set* (MCS) if there is no cut set for reaction i that is a proper subset of X. The number of reactions in a cut set X, $|X|$, is called its size.

The main tool we need is Farkas' lemma [100], which can be paraphrased as the statement that a linear inequality holds for a convex set described by system of linear inequalities if and only if it can be obtained as a linear combination of the linear inequalities describing the set with non-negative coefficients.

It allows us to prove the following lemma (all the proofs are in the Appendix).

Lemma 2 (Cut Sets in Fully Reversible Networks). *Let S be the stoichiometric matrix of a fully reversible network. Then X is a cut set for reaction i if and only if there exists a y such that $R(y^T S) \subseteq X \cup \{i\}$, with $y^T S$ having a positive entry in position i. X is a minimal cut set for reaction i if and only if $X \cup \{i\}$ is a minimal support of vectors with positive i-th component in the rowspace of S.*

Now the duality between EFMs and MCSs becomes clear. While EFMs are minimal supports of nonzero vectors in the nullspace of S, MCSs correspond to minimal supports of nonzero vectors in the rowspace of S, which is the orthogonal complement of the nullspace. In addition, the following property is seen to hold: X is a (minimal) cut set for reaction i if and only if $X \cup \{i\} - \{j\}$ is a (minimal) cut set for $j \in X$.

Furthermore, since a matrix K whose rows form a basis of the nullspace of S can be computed efficiently [23], and the nullspace of K is then precisely the rowspace of S, lemma 2 immediately shows that computing the MCSs of S is the same as computing the EFMs of K and vice versa. This is the first explicit statement of this duality, to our knowledge.

When the network is fully irreversible, the support becomes the negative support.

Lemma 3 (Cut Sets in Fully Irreversible Networks). *Let S be the stoichiometric matrix of a fully irreversible network. Then X is a cut set for reaction i if and only if there exists a y such that $R_-(y^T S) \subseteq X$, with $y^T S$ having a positive entry in position i. X is a minimal cut set for reaction i if and only if X is a minimal negative support of vectors with positive i-th component in the rowspace of S.*

The duality between EFMs and MCSs still holds, but needs to be modified slightly in this case. Now EFMs are minimal supports of vectors with nonnegative components in the nullspace of S, while MCSs for irreversible reactions correspond to minimal negative supports of nonzero vectors in the rowspace of S. In this case, we note the following property: if $v \in \text{Row}(S)$, then $R_-(v)$ is a cut set for every reaction in $R_+(v)$.

General networks (that are neither fully reversible nor fully irreversible) require an approach that treats the reversible and the irreversible reactions differently. Even though a hybrid of lemmas 2 and 3 can be obtained, we choose not to state or formally prove it here.

10.2 Blocked Reactions

Reaction i will be called *blocked* if conditions 9.2 and 9.3 imply that $v_i = 0$. We now identify three distinct reasons why a reaction could be blocked in a network.

The first reason is the *topology*, or connectivity, of the network. If reaction i involves a metabolite that is unique to it, the mass balance condition on this metabolite immediately implies that the flux through that reaction is 0. Additionally, if there is another reaction j that shares a unique metabolite with reaction i, it is also blocked as a consequence. These consequences can propagate through the network, sometimes in a dramatic way. When a reaction is blocked because of network topology in this way, we call it *topologically blocked.* Note that exchange reactions are generally not topologically blocked because any external metabolite they transport is not constrained by mass balance.

All topologically blocked reactions can be identified by the following algorithm.

Algorithm1 [Topologically blocked reactions, general case]

Input: a stoichiometric matrix S.

Output: the set Q of all topologically blocked reactions in S.

Initialization: $Q := \emptyset; d_j := |\{i|S_{i,j} \neq 0\}| \; \forall \; j$.

Iteration: **while** there exists j such that $d_j = 1$ **do**

$B := \{j|d_j = 1\}$

$Q_{\text{new}} := \{i|S_{i,j} \neq 0 \text{ for some } j \in B\}$

for $i \in Q_{\text{new}}$

for j such that $S_{i,j} \neq 0$

$d_j := d_j - 1$

$Q = Q \cup Q_{\text{new}}$

Termination: **return** Q.

The second reason is stoichiometry. Even when a reaction does not contain any unique metabolites, it is still possible that the stoichiometry of the network forces it to be blocked. This happens when $Sv = 0$ by itself (without any non-negativity constraints) implies $v_i = 0$. We call all such reactions *stoichiometrically blocked* because their blockage is caused by stoichiometry. Note that all topologically blocked reactions are stoichiometrically blocked, but not vice versa.

If our network is fully reversible, lemma 2 (with $X = \emptyset$) shows that reaction i is stoichiometrically blocked precisely when $e_i \in \text{Row}(S)$, where e_i is the i-th standard basis vector with 1 in position i and 0 elsewhere. It is possible, by performing a Gauss-Jordan elimination on S, to determine all reactions for which this is the case. The reduced row-echelon form of S will contain e_i as one of its rows if and only if e_i is in the rowspace of S. Thus, it is in principle not necessary to compute the matrix K whose rows form a basis of the nullspace of S, as in [101]. For completeness, we restate this as an algorithm.

Algorithm2 [Stoichiometrically blocked reactions, fully reversible case]

Input: a stoichiometric matrix S.

Output: the set Q of all stoichiometrically blocked reactions in S.

Algorithm: Compute R, the reduced row-echelon form of S; $Q := \{i | e_i \text{ a row of } R\}$.

Termination: **return** Q.

The third reason is thermodynamics. Even though a reaction may not blocked if only the mass-balance conditions are imposed on it, it may become blocked when the additional constraints $v_i \geq 0 \ \forall \ i \in \mathcal{I}$ are imposed. We call all such reactions *thermodynamically blocked* because their blockage is caused by thermodynamics. Note that all stoichiometrically blocked reactions are thermodynamically blocked, but not vice versa.

If our network is fully irreversible, lemma 3 (with $X = \emptyset$) shows that reaction i is thermodynamically blocked precisely when a vector $v \geq \mathbf{0}$ with $v_i > 0$ is in the rowspace of S, which is also the nullspace of the nullspace matrix K. But then $\emptyset$ is also a cut set for all reactions in $R_+(v)$. In this way, one can see that a subset X is thermodynamically blocked if and only if there is a vector in the rowspace of S which has positive support X. If X_1 and X_2 are two such subsets (not necessarily disjoint), then adding the corresponding vectors shows that $X_1 \cup X_2$ is also such a subset. Hence, we need to find the largest positive support of a non-negative vector v in the nullspace of K. This can be done by the following algorithm.

Algorithm3 [Thermodynamically blocked reactions, fully irreversible case]

Input: a stoichiometric matrix S.

Output: the largest positive support Q of a vector $x \geq \mathbf{0}$ in the nullspace of K.

Initialization: $w := \mathbf{1}$ (the vector of all ones) ; $Q := \emptyset$; $z := 1$.

Iteration: **while** $z > 0$ **do**

Solve $z := \max \sum_i w_i x_i$ subject to $Kx = \mathbf{0}, \mathbf{0} \leq x \leq \mathbf{1}$.

$w_i := 0 \ \forall \ i \in R(x)$

$Q := Q \cup R(x)$.

Termination: **return** Q.

The normalization condition $x \leq \mathbf{1}$ only serves the purpose of bounding the linear program. By construction, this algorithm will produce a vector w with 1 in the i-th position if and only if $i \notin Q$. In practice, this algorithm only requires the solution of a handful of linear programs before it converges, making it more efficient than the direct approach requiring one linear program per reaction.

Now, while algorithm 1 is general, algorithms 2 and 3 apply only to the fully reversible and fully irreversible cases, respectively. However, it turns out that they suffice to identify all the blocked reactions in the general case. The following algorithm shows how this can be done.

Algorithm4 [Removing all blocked reactions, general case]

Input: a stoichiometric matrix S, the irreversible subset $\mathcal{I}$.

Output: S with all the blocked reactions removed.

Algorithm: $Q := \mathbf{Algorithm1}(S)$

$S := S_{-Q}$ (S with the columns in Q removed)

$K := \mathbf{NullspaceMatrix}(S)$

$Q := \mathbf{Algorithm3}(K_{\mathcal{I}})$

$S := S_{-Q}$

$Q := \mathbf{Algorithm2}(S)$

$S := S_{-Q}$

Termination: **return** S.

Blocked reactions are sometimes called *strictly detailed balanced* in the literature [86]. A stoichiometric matrix in which no reaction is strictly detailed balanced is called *consistent* [102]. The fact that **Algorithm4** outputs a consistent stoichiometric matrix is proven in lemma 4.

Lemma 4 (Correctness of Sequential Deletions). ***Algorithm4*** *correctly identifies and removes all strictly detailed balanced reactions from a stoichiometric matrix S.*

10.3 Effectively Unidirectional Reactions

Now that we have a consistent matrix, we need to address the question of *effective unidirectionality*, a concept introduced into the field by our work, although it is closely related to the concept of *pseudo-irreversibility* [103]. We call a reversible reaction *effectively unidirectional* if it can sustain a flux of only one sign. If this sign is positive, we call the reaction *effectively forward*, while if it is negative, we call it *effectively backward.*

To decide whether a reversible reaction i is effectively unidirectional, it suffices to test the feasibility of two linear programs, one with $v_i = 1$ and one with $v_i = -1$. Since we assume that S is now a consistent matrix, at least one of those will always be feasible, so in many cases it will not be necessary to test the feasibility of both.

While there seems to be no simpler algorithm to identify all the effectively unidirectional reactions, a short list of candidates can be generated by applying **Algorithm2** to $S_{-\mathcal{I}}$. The fact that $\mathcal{I}$ is a cut set for an effectively unidirectional reaction follows immediately from the proof of lemma 4.

Once all the effectively unidirectional reactions have been identified, we reverse the effectively backward reactions by multiplying all the coefficients in the corresponding column of S by -1, and then add all the effectively unidirectional reactions to $\mathcal{I}$. After this modification, S has the following property: every reaction can sustain a positive flux, and all reversible reactions can sustain both a positive and a negative flux. In this case, we call S *properly directed.* As we will see in the following section on Enzyme Subsets, being properly directed is a desirable property. This gives us

Algorithm5 [Properly Directing a Consistent Stoichiometric Matrix]

Input: a consistent stoichiometric matrix S, the irreversible subset $\mathcal{I}$.

Output: a properly directed version of S.

Algorithm: $Q :=$ **Algorithm2**$(S_{-\mathcal{I}})$

$$P := \{j \in Q | \nexists v \text{ such that } Sv = \mathbf{0}, v_i \geq 0 \; \forall \, i \in \mathcal{I}, v_j = -1\}$$
$$N := \{j \in Q | \nexists v \text{ such that } Sv = \mathbf{0}, v_i \geq 0 \; \forall \, i \in \mathcal{I}, v_j = 1\}$$
$$S_N := -S_N$$
$$\mathcal{I} := \mathcal{I} \cup P \cup N$$

Termination: **return** $S, \mathcal{I}$.

10.4 Enzyme Subsets

An enzyme subset is defined as a set of reactions such that any steady-state fluxes in a set are in a fixed ratio. In particular, two reactions, say i and j, are part of the same enzyme subset if and only if there is a constant $\kappa \neq 0$ such that $v_i = \kappa v_j$ for all flux modes v. It was stated in [86] that *most* enzyme subsets can be identified by analyzing the nullspace matrix K and identifying proportional columns. Using duality, we prove the following

Lemma 5 (Enzyme Subsets). *In a consistent stoichiometric matrix S, i and j are in an enzyme subset with ratio κ if and only if $e_i - \kappa e_j \in Row(S)$.*

Lemma 5 shows that all the enzyme subsets can be identified from the proportional columns of K. Hence, the algorithm given by Pfeiffer et al. [104] correctly identifies *all* such subsets.

Furthermore, it is easy to see that if in addition to being consistent, S is also properly directed, then any enzyme subset will consist either entirely of irreversible reactions or entirely of reversible reactions. For suppose $i \in \mathcal{I}$ and $j \notin \mathcal{I}$ are in the same subset with proportionality constant $\kappa \neq 0$. Since j admits fluxes of either sign, choose an admissible v such that $\kappa v_j < 0$. But then we get $v_i < 0$, contradicting the irreversibility of $i \in \mathcal{I}$. In the same way, the proportionality constants in a subset composed of irreversible reactions are positive for S consistent and properly directed.

Now suppose that an enzyme subset containing reactions $i_1, i_2, \ldots, i_t$ has been identified, and let r_j be the ratio of the flux through i_j to the flux through i_1, for $2 \leq j \leq t$. We can then *lump* the enzyme subset together into a single new reaction:

$$S_{\text{new}} = S_{i_1} + \sum_{j=2}^{t} r_j S_{i_j}. \tag{10.1}$$

Note that if the original reactions are catalyzed by the genes $G_1, \ldots, G_t$, the new reaction is catalyzed by G_1 AND ... AND G_t. The following lemma shows that the resulting network is equivalent to the original one in the following sense: the flux vectors in one network are in a one-to-one correspondence with those in the other.

Lemma 6 (Equivalence after Lumping of Enzyme Subsets). *Let $\tilde{S}$ be the matrix obtained from S by lumping its enzyme subsets. Then S and $\tilde{S}$ are equivalent.*

It follows that no information is lost from the stoichiometric matrix if enzyme subsets are lumped into a single reaction. There is, however, one caveat to be made. It is possible that one of the lumped reactions is $\mathbf{0}$. In this case, we say that the enzyme subset forms a *zero loop*. A zero loop would be able to carry an arbitrarily high flux, and since this is a physically impossible situation, all such loops are deleted.

For convenience, we restate this modification as an algorithm.

Algorithm6 [Identifying and Lumping Together the Enzyme Subsets]

Input: a consistent properly directed stoichiometric matrix S.

Output: S with each enzyme subset lumped into a single reaction.

Algorithm: $K :=$ **NullspaceBasis**(S)

$ES :=$ maximal sets of proportional columns of K with ratios

for each group in ES **do**

combine the group into a new reaction according to (10.1)

remove the newly created zero loops from S

Termination: **return** S.

10.5 Model Reduction to a Canonical Form

If the stoichiometric matrix S does not have independent rows, some of the constraints in equation (9.2) are redundant. Redundant constraints are easily identified and eliminated by a simple Gaussian elimination on S^T, the transpose of S [23]. The columns of S^T containing a leading 1 will then form a basis of the column space of S^T , which is the row space of S, whereas all the other constraints in equation (9.2) can be eliminated. This gives us the final algorithm of this chapter:

Algorithm7 [Eliminating Redundant Constraints]

Input: a stoichiometric matrix S.

Output: a minimal subset of rows spanning the rowspace of S.

Algorithm: $G :=$ **GaussianElimination**(S^T)

$S :=$ rows of S corresponding to the leading ones in G.

Termination: **return** S.

If a non-negative linear combination of the rows yields **0**, the linear combination is referred to as a *conservation relation.* If all reactions in the network are elementally balanced, then there will be at least 5 conservation relations in the initial stoichiometric matrix: one for each of the elements C (carbon), H (hydrogen), O (oxygen), N (nitrogen) and P (phosphorus). However, as the extracellular compounds are deleted from S, these conservation relations will usually not hold for S. However, there may be structural organic groups (such as the hydroxy group) that are conserved, and each of these gives a redundant constraint [84].

It is easy to see that a metabolic network is elementally balanced with respect to some set of elements (using the terminology of [84], it is a "closed system with atomic

representation") if and only if there exists a vector y with positive components such that $y^T S = \mathbf{0}$. Indeed, one can think of y as containing the number of elements (atoms) in each metabolite, and the total number of elements must be conserved in any elementally balanced reaction. The following lemma states that this happens if and only if a condition called the "no free lunch" condition holds.

Lemma 7 (Elemental Balance and the No Free Lunch Condition). *Let S be a stoichiometric matrix of a metabolic network (with the external metabolites included). Then S is elementally balanced if and only if no linear combination of the reactions in S can result in the production of one or more metabolites out of no reagents.*

Note that this lemma settles the question of deciding the *conservativity* of a given stoichiometric matrix posed by Schuster and Höfer [84] using a single linear program. Indeed, one simply needs to verify the "no free lunch" condition by checking whether a non-negative vector in the column space of S (with the external metabolites included) can have a positive sum of elements. To our knowledge, this is the first polynomial-time algorithm for checking conservativity of a given stoichiometric matrix.

However, not all linear combinations of the rows giving $\mathbf{0}$ are non-negative, but they all still result in redundant constraints. In fact, it is possible to count the precise number of redundant constraints in a metabolic network due to each of the algorithms 4 through 6. This is given by the following lemma:

Lemma 8 (Number of Non-Redundant Constraints). *Let S be an $m \times n$ stoichiometric matrix with rank r, and let K be its nullspace matrix. Suppose that **Algorithm4** eliminates $\mathcal{D} \subseteq \mathcal{I}$ as thermodynamically blocked reactions, and let $r_0 = rank(K_{\mathcal{D}})$. Suppose that **Algorithm4, Algorithm5, Algorithm6** eliminate a total of T reactions. Then the final S contains $r - T + r_0$ non-redundant constraints.*

We say that S is in *canonical form* if it is consistent, properly directed, contains no enzyme subsets or zero columns, and has full row rank. We claim that the matrix

S obtained as the output of successively applying the steps of a *reduction cycle* (algorithms 4 through 7) is in canonical form. In other words, no further reduction of S is possible. This is the statement of the following lemma:

Lemma 9 (Canonical Form after a Single Reduction Cycle). *Let S be the stoichiometric matrix of a metabolic network after one reduction cycle. Then S is in canonical form.*

Note that we choose not to eliminate uniquely produced or uniquely consumed metabolites, as described in [86]. This is partly because they may in fact increase the size of the stoichiometric matrix. Indeed, since most reactions in a typical network are irreversible, it is very likely that several of the reactions consuming a uniquely produced metabolite are irreversible. However, when they are lumped together with the unique producing reaction, they need to be made reversible, and each one then yields two reactions during reconfiguration, thus increasing the total number of reactions. The same reasoning applies to uniquely consumed metabolites.

Reconfiguration, i.e. the splitting of each reversible reaction into a forward and a backward reaction, is best performed at the very end of the reduction cycle, if at all. In the following chapters, most of the algorithms will assume that the network is fully irreversible, i.e. it has been reconfigured. However, they could still be made to work for a non-reconfigured network in most cases, and this choice can be made in each individual application. One important result is that the stoichiometric matrix remains in canonical form after reconfiguration.

Lemma 10 (Canonical Form in Reconfigured Networks). *Let S be the stoichiometric matrix of a metabolic network. If S is in canonical form, then so is the stoichiometric matrix of the reconfigured network.*

It is easy to see that the canonical form of a stoichiometric matrix is generally not unique, since the reactions and the metabolites can be permuted while keeping the

matrix in canonical form. Also, scaling any row (metabolite) or column (reaction) by a nonzero scalar multiple (a positive scalar multiple in the case of irreversible reactions to preserve proper directedness) results in another matrix in canonical form. For instance, the scaling of the lumped reaction can vary depending on which reaction is chosen as the first reaction of a subset in **Algorithm6**. In addition, the set of non-redundant rows found by **Algorithm7** is not unique (although its size is, according to Lemma 8). However, these in fact appear to be the only "degrees of freedom" that a canonical form has.

The main property of a matrix in canonical form is that it contains no linear relations of length 1 or 2 between its columns or the columns of its nullspace matrix, and no linear relations of any length between its rows, and it thus seems intuitively plausible that the only degrees of freedom are indeed those described above. However, this fact is difficult to prove, and since none of our results rely on it, we leave a proof as an exercise to the interested reader.

Chapter 11

Minimality in Metabolic Models

Minimality is an important topic in the study of biological systems in general, and metabolism in particular. What is the smallest number of genes necessary for a given function? What is the smallest number of mutations that can lead to a change in the organism's phenotype? How can an objective be achieved with the smallest expenditure of resources? There are some of the questions that this chapter will attempt to answer in the context of cell metabolism. We begin by revisiting elementary flux modes and minimal cut sets, and describe both the complexity of the problem of finding them in a constraint-based model, as well as novel efficient algorithms for it. We also stop to consider the energetic feasibility of elementary flux modes. We then discuss two engineering applications, also involving minimality - the design of minimal media and (qualitative) strain optimization. The latter application will be presented in a way different from what has been done in literature so far.

11.1 Elementary Flux Modes

Let us recall that elementary flux modes (EFMs) are modes that have minimal support. In other words, v is an elementary flux mode if v is a mode and any other mode w such that $R(w) \subsetneq R(v)$ must be the trivial mode $w = \mathbf{0}$. The number of reactions in a mode v, $|R(v)|$, is called its length. In general, the size of the support of a vector v is sometimes called its cardinality or 0-norm (denoted $\|v\|_0$), even though it is not strictly speaking a norm [105]. We will use these notations interchangeably.

The results of Acuña et al [102] establish the NP-completeness of the problem of deciding whether a given metabolic network has an EFM of length at most k. It easily follows that the problem of deciding whether it has an EFM of length at most k involving a given reaction i is also NP-complete, since a polynomial-time algorithm for it would allow one to apply it n times (once per reaction) and return "YES" if and only if one of the applications does. The corresponding optimization problems of finding the shortest EFM and finding the shortest EFM containing a given reaction i are therefore NP-hard.

However, we will establish this NP-completeness result for the special case of a fully reversible network in a different way. This result will be sufficiently general to allow us to prove the NP-completeness of other, related, problems without having to resort to a new reduction every time.

Lemma 11 (NP-Completeness of Sparse Solutions). *Given an $m \times n$ rational matrix S and an integer $k < n$ it is NP-complete to decide whether the system $Sx = \mathbf{0}$ has a nonzero solution with $\|x\|_0 \leq k$.*

As a corollary, we see that it is also NP-complete to decide whether the system $Sx = \mathbf{0}$ has a solution with $\|x\|_0 \leq k$ and $x_i = 1$. By taking A to be the nullspace matrix of S, it also follows that it is NP-complete to decide whether there exists a y such that Ay is nonzero and $\|Ay\|_0 \leq k$. We will use both of these corollaries later.

It has been shown in [86] that elementary flux modes correspond to the extreme rays of the flux cone $F := \{Sv = 0, v \geq 0\}$, where S is the *reconfigured* stoichiometric matrix. Furthermore, every mode can be expressed as a conical combination (linear combination with nonnegative coefficients) of elementary flux modes, though not necessarily uniquely. We will use both of these properties in what follows.

Let us define $F_i := \{Sv = 0, v \geq 0, v_i = 1\}$, the set of all flux modes with a flux of 1 through reaction i. To find a single elementary flux mode involving a reaction i, it is sufficient to solve the linear program

$$\min c^T v \text{ subject to } v \in F_i \tag{11.1}$$

for some nonnegative weight vector c using the simplex algorithm (or any other active-set algorithm for linear programming [106]), since the vertices of F_i are in a one-to-one correspondence with the extreme rays of F involving reaction i. A fairly short elementary flux mode can be found by setting $c = \mathbf{1}$, the vector of all ones.

To find additional short elementary flux modes involving i, we use a novel *runner-up algorithm.* An elementary flux mode v *uniquely* minimizes

$$\sum_{j \notin R(v)} v_j \tag{11.2}$$

with a value of 0 (indeed, if any other mode w gave a value of 0 to this sum, its support $R(w)$ would be contained in $R(v)$, implying that $w = 0$). The idea of the runner-up algorithm is to exclude v from further search, by constraining $\sum_{j \notin R(v)} v_j$ to be strictly positive, while keeping all the other constraints defining F_i.

If we minimize the sum in equation (11.2), the solution to the resulting problem is then a linear combination of v and the *runners-up* $w_1, \ldots, w_k$ which are vertices of F_i. If we use an active set algorithm, only one of the runners-up will be present

in the linear combination. Indeed, since the w_j are minimal with respect to the sum in equation (11.2) when v is excluded, they all achieve the same value $\delta > 0$ for it. Then a solution of the form $\alpha_0 v + \sum_{j=1}^{k} \alpha_j w_j$ (with $\alpha_0 + \sum_{j=1}^{k} \alpha_j = 1$ to ensure it is in F_i) can be written as the convex combination

$$\sum_{i=1}^{k} \left(\frac{\alpha_i}{1-\alpha_0}\right)[\alpha_0 v + (1-\alpha_0)w_i], \tag{11.3}$$

and each term in square brackets achieves the same value $(1-\alpha_0)\delta$ of the objective function. It follows that the solution found by an active set algorithm will contain only one of these terms, say $\alpha_0 v + (1-\alpha_0)w_1$.

This solution can be found by solving the linear program

$$\min \sum_{j \notin R(v)} u_j \text{ subject to } v \in F_i, \sum_{j \notin R(v)} u_j \geq \epsilon \tag{11.4}$$

for some small constant $\epsilon > 0$. Once u is found, we can simply subtract the largest multiple of v that still keeps it nonnegative, and this will be w_1. The same procedure can now be repeated starting from w_1 instead of v. This procedure continues until no new elementary flux mode in F_i is found. By applying this runner-up algorithm for each reaction i and eliminating duplicates, we create a substantial collection of putative metabolic pathways. We summarize this procedure below.

Algorithm8 [Elementary flux modes, fully irreversible case]

Input: a stoichiometric matrix S.

Output: a collection C of elementary flux modes in S.

Initialization: $C := \emptyset$

Iteration: **for** $i \in \{1, \ldots, n\}$ **do**

$v := \mathbf{0}, C_i := \emptyset$

while $v \notin C_i$ **do**

if $v \neq \mathbf{0}$

$C_i := C_i \cup \{v\}$

Solve the problem in (11.4) for u

Compute $w := u - \beta v$ with $\beta := \min_{\{j|v_j \neq 0\}}(u_j/v_j)$

Normalize: $v := (1/w_i)w$

$C := C \cup C_i$

Termination: **return** C. Let us finish this section by describing the process of mapping back a flux vector onto the original network (before the reduction cycle). If the network has been reconfigured, then the flux through the initially reversible reaction is obtained as the difference between the forward and the backward fluxes in the reconfigured network. All strictly detailed balanced reactions get a flux of 0, while the flux through each reaction representing an enzyme subset gets expanded to all the original reactions in the enzyme subset. The flux through the combined reaction is multiplied by the ratio r_j to get the flux through the jth original reaction in the subset, just as described in the proof of lemma 6. If a reaction representing an aggregation of isozymes is in the mode, the flux can be distributed between the isozymes in an arbitrary way. Given two stoichiometric matrices with the same canonical form, we will call two modes *equivalent* if they have the same representation as modes for the common canonical form. This definition will be useful for the following section.

11.2 Energy Balance Analysis

Each of the elementary flux modes we find may or may not be energetically feasible. In order to check energetic feasibility of a mode v, we need to decide if it satisfies equations (9.4) and (9.5), which we restate together as:

$$\exists w \in \text{Row}(S_{\mathcal{C}}), v_i w_i \leq 0 \; \forall \; i \in \mathcal{C} \text{ and } v_i = 0 \implies w_i = 0 \; \forall \; i \in \mathcal{C}. \tag{11.5}$$

For simplicity, we are only going to consider the case of a fully irreversible network. In that case, $w_i < 0$ for each internal reaction i such that $v_i > 0$, and $w_i = 0$ for each internal reaction i such that $v_i = 0$. However, there is an important exception. When a reaction that was initially reversible in the network (before reconfiguration) has a nonzero flux in v (without loss of generality, only one of its directions will have flux, since otherwise the smaller of the fluxes can be subtracted from the larger one), the other direction should not be constrained to a chemical potential of 0 even though the flux through it is 0. Let us denote by $B(v)$ the set of all reactions in v having 0 flux whose reverse reaction has a positive flux.

Replacing w by $-w$, this is equivalent to saying that we are looking for a vector in the rowspace of $S_{\mathcal{C}}$ with the same support as $v_{\mathcal{C}}$ except possibly for reactions in $B(v)$. By scale invariance, we can constrain the minimum nonzero entry of w to be 1. It is thus sufficient to check the feasibility of the following linear program:

$$w \in \text{Row}(S_{\mathcal{C}}), w_i \geq 1 \; \forall \; i \in \mathcal{C} \cap R(v) \text{ and } w_i = 0 \; \forall \; i \in \mathcal{C} - R(v) - B(v). \tag{11.6}$$

If we are only interested in energetically feasible elementary flux modes, we can use the problem in equation (11.6) as a filter for our collection C of EFMs. However, we note that because of the nonlinearity of energy constraints, a linear combination of energetically feasible modes may be energetically infeasible while a linear combination of energetically feasible modes may be energetically infeasible [85]. Therefore, the decision whether or not to filter the elementary flux modes should depend on the particular application.

To conclude this section, we state a fairly straightforward but reassuring result, namely, that the reduction to canonical form does not affect the energetic feasibility of any flux modes.

Lemma 12 (Energetic Feasibility in Canonical Form). *If a mode v is energetically feasible for S, its equivalent mode is energetically feasible for the version of S in canonical form (with or without reconfiguration).*

11.3 Minimal Cut Sets

Let us recall that a *cut set* for reaction i is a subset X of reactions not containing reaction i such that constraining these reactions to have flux 0 makes reaction i unable to have a non-zero flux. X is a *minimal cut set* for reaction i if there is no cut set for reaction i that is a proper subset of X. The number of reactions in a cut set X, $|X|$, is called its size.

Our lemmas 2 and 11 shows that it is NP-complete to determine whether a stoichiometric network has a cut set of size k or less for reaction i. Interestingly enough, for most actual networks that we investigated, there are cut sets of size 3 or less for a large fraction of the reactions. This shows that the problem of finding minimal cut sets may be "easier" than that of finding elementary flux modes in practice. In particular, an efficient algorithm for checking whether a particular subset of reactions is a cut set suffices to identify all minimal cut sets of size 3 or less in genome-scale networks. We present such an algorithm in this section. However, for completeness, we also present a general algorithm for finding small cut sets.

We note that the cardinality of v's negative support (as required by lemma 3) is identical to the 0-norm of the smallest nonnegative vector that needs to be added to v to make it nonnegative. Therefore, in the case of a MCSs in a fully irreversible

network, we introduce an auxiliary vector t , and add the constraints $t \geq \mathbf{0}$ and $t + v \geq \mathbf{0}$. As before, we minimize $\sum_i t_i$ in order to discover small MCSs. (Note that because the t vectors do not form a convex structure, the runner-up algorithm cannot be applied here.) The corresponding linear program can then be written as follows.

$$\min \sum_j t_j \text{ subject to } Kv = \mathbf{0}, t \geq \mathbf{0}, t + v \geq \mathbf{0}, v_i = 1. \tag{11.7}$$

Since, unlike for EFMs, it is possible that the linear program returns a cut set which is not minimal, we need to first verify its minimality. For this we introduce the auxiliary linear program

$$Sv = 0, v \geq 0, v_i = 1 \text{ and } v_X = \mathbf{0} \tag{11.8}$$

Here, i is the target reaction to be blocked. If the linear program is feasible for any X that is a proper subset of the support of t, $R(t)$, then X is not minimal. It is sufficient to check the feasibility of this linear program for $X := R(t) - \{j\}$ for each $j \in R(t)$. If any of these turns out to be infeasible, it is a smaller cutset and the process can be repeated until no further reduction is possible. It is actually sufficient to test each element of $R(t)$ once and remove it if and only if the currently active subset X gives an infeasible linear program. We call this the *take-one-out* procedure. Since $R(t)$ is typically small, this leads to a fast verification in practice.

The complete algorithm is then as follows.

Algorithm9 [Minimal cut sets, fully irreversible case]

Input: a stoichiometric matrix S.

Output: a collection C of minimal cut sets in S, one per reaction.

Initialization: $C := \emptyset$

Iteration: **for** $i \in \{1, \ldots, n\}$ **do**

Solve the problem in (11.7) for u

$X := R(u)$

for $j \in R(u)$ **do**

if the program (11.8) is infeasible for $X - \{j\}$

$X := X - \{j\}$

$C := C \cup X$

Termination: **return** C.

However, of more practical interest than finding small cut sets for each reaction is the problem of finding all the *essential* and all the *synthetic lethal* genes for the organism. A gene is called essential if it is necessary for the organism's survival (in metabolic models, survival is represented by the growth reaction); a set of two or three genes are called synthetic lethal if none of them is essential, but their simultaneous deletion is lethal to the organism. In other words, in the context of a metabolic model, these are precisely the cut sets of size up to 3 for the growth reaction.

In the case of a fully reversible network, this is easily done by applying lemma 2. Indeed, if j is a cut set for reaction i, then by lemma 2 there would be a vector with support of size 2 in the rowspace of S, which means that i and j would be in the same enzyme subset by lemma 5. Furthermore, synthetic lethal pairs would be in an enzyme subset in $S_{-\{i\}}$ because they correspond to the only vectors with support of size 3 that include reaction i, by lemmas 2 and 5 once again. Finally, synthetic lethal triples would have the form $\{j\} \cup \{k, l\}$, where k and l are in an enzyme subset in $S_{-\{i,j\}}$ as they would be the only vectors with support of size 4 that include reaction i, once again by the same pair of lemmas. This gives us the following algorithm:

Algorithm10 [Small cut sets, fully reversible case]

Input: a stoichiometric matrix S in canonical form; target reaction i.

Output: a collection C_k of small cut sets in S for reaction i of size $k \leq 3$.

Initialization: $C_1 :=$ the reactions in an enzyme subset with i (essential genes); $C_2 :=$ pairs of reactions in enzyme subsets of $S_{-\{i\}}$ found by **Algorithm6**; $C_3 = \emptyset$.

Iteration: **for** $j \in \{1, \ldots, n\} - \{i\}$ **do**

Find the enzyme subsets of $S_{-\{i,j\}}$ using **Algorithm6**

$X := \{\{j\} \cup \{k, l\} | k, l \text{ in an enzyme subset }\}$

$C_3 := C_3 \cup X$

Termination: **return** C_1, C_2, C_3.

In the general case, however, things are not so simple. There is no simple approach besides trying all subsets of size $1, 2, 3$ in order and using the feasibility of the linear program in (11.8) to test. There are, however, two significant improvements to be made to this enumeration algorithm. First of all, supersets of cut sets need never be tested (we are only interested in minimal cut sets after all). Second, we know that any cut set for reaction i must disable all flux modes using reaction i. In other words, a cut set is a *set cover* (or *transversal*) of all the elementary flux modes using reaction i (since all other flux modes are linear combinations of these). Although **Algorithm8** does not yield the complete inventory of such EFMs, it is necessary for a cut set to cover any incomplete inventory as well. This gives the following algorithm (which we only describe in words).

Let C be a collection of q EFMs containing i, and let H be the set of all reactions involved in C except for i. We construct a matrix M of size $|H|$ with $M_{j,k}$ being the number of flux vectors that contain both j and k, with the diagonal element $M_{j,j}$ being the number of vectors that contain reaction j. j is essential for the q flux vectors if $M_{j,j} = q$. The pair $\{j, k\}$ is synthetic lethal for the q flux vectors if and only if each one of them contains either j or k, which is to say that $M_{j,j} + M_{k,k} - M_{j,k} = q$. Similarly, the triplet $\{j, k, l\}$ can be synthetic lethal if $M_{j,j} + M_{k,k} + M_{l,l} - M_{j,k} - M_{j,l} - M_{k,l} + \min(M_{j,k}, M_{j,l}, M_{k,l}) \geq q$. Here, we use the inclusion-exclusion principle and

the fact that $\min(M_{j,k}, M_{j,l}, M_{k,l}) \geq |\{v \in C | j,k,l \in R(v)\}|$. We verify the synthetic lethality of subsets which pass this test using the linear program in equation (11.8).

However, because our inventory is incomplete, if H does not contain all the reactions in S we need to see if some of the subsets can be supplemented by reactions not in H. For the singletons covering C that are not cut sets, we check all the possible completions by singletons from H^C , the set complement of H, then by pairs. For the pairs covering C that are not cut sets, we check all the possible completions by singletons from H^C. This gives the complete set of minimal cut sets of size at most 3 for i.

Let us finish this section by describing the process of mapping back a cut set onto the original network (before the reduction cycle). If the network has been reconfigured, the reversible reaction in the original network in the cut set if and only if either its forward or its backward direction is in the cut set. If a reaction representing an aggregation of isozymes is in the cut set, every reaction that has an isozyme in the aggregation is in the cut set. If a reaction representing an enzyme subset is in the cut set, it suffices to take any one of the reactions in the subset. This shows that the duality between EFMs and MCSs extends to a (Boolean) duality between isozymes and enzyme subsets.

11.4 Minimal Media

The method for obtaining EFMs can also be used for finding minimal media, which are the smallest subsets of the exchange reactions $\mathcal{E}$ that can sustain the organism's growth, a problem that has been studied previously [107]. For this, we need to look for flux vectors v such that $v_{\text{biomass}} = 1$ and such that $\|v_{\mathcal{E}}\|_0$ is minimized. This can be achieved by taking the vector $\chi(\mathcal{E})$ instead of $\mathbf{1}$ in equation (11.1), where $\chi(T)$ is the characteristic vector of a set T, with value 1 in components corresponding to the

elements of T and 0 elsewhere. However, the computed medium needs to be checked for minimality by the take-one-out procedure, because there could be two EFMs, v and w, such that $R(v) \not\subseteq R(w)$ but $R(v_\mathcal{E}) \subseteq R(w_\mathcal{E})$. Additional minimal media can be discovered by a variant of the runner-up algorithm where the exclusion constraint is only applied to $v_\mathcal{E}$, not all of v.

11.5 Strain Optimization

Generally speaking, strain optimization is the task of genetically modifying an organism by deleting a small number of genes in order to maximize the production (yield) of a certain biochemically relevant compound. Since the organism is usually assumed to be maximizing its own objective, called the biomass [108, 109], this task is commonly stated as a bilevel optimization [110, 111, 112]. The premise of bilevel optimization is that the genetically modified organism will still attempt to maximize its biomass production. However, this premise has been called into question, and alternative hypotheses, such as minimization of metabolic adjustment [113] or regulatory on-off minimization [114] may be closer to the truth. Furthermore, due to the fact that many different alternate solutions may produce biomass at equivalent levels [115], previously published algorithms may produce overly optimistic predictions [110].

Here, we bypass these issues by solving the following version of the problem: find a small genetic modification such that the minimum yield of the desired compound per unit biomass exceeds a specified level λ . In this modified problem, it is irrelevant whether the organism is actually trying to optimize its biomass production; as long as the organism is producing biomass (optimally or suboptimally), it will produce the desired compound at the specified level λ or better.

For simplicity, we only consider the case of an irreversible network. Note that

since any flux vector is a nonnegative linear combination of EFMs, the yield of the desired compound per unit biomass is bounded below by the corresponding yield of any biomass-producing EFM. Some of these EFMs will have a yield below λ, and others, at or above λ. It follows that our problem is equivalent to finding a small cut set for all EFMs with a yield below λ. This can be restated as finding a small subset X of reactions such that the inequality

$$v_{\text{target}} \geq \lambda v_{\text{biomass}} \tag{11.9}$$

holds for any mode v provided that $v_X = 0$. Here, target denotes the export reaction for the target compound and biomass, the biomass-producing reaction.

Lemma 13 (Strain Optimization in Irreversible Networks). *Equation (11.9) holds for any mode v with $v_X = 0$ if and only if there is a vector u in the rowspace of S such that $R_-(u) \subseteq X \cup \{target\}$, with $u_{target} = -1$ and $u_{biomass} = \kappa$ for some $\kappa \geq \lambda$.*

This immediately gives us an algorithm for finding a set of genes whose knockout will achieve our objectives. This algorithm is similar to **Algorithm9** for finding MCSs.

Algorithm11 [Strain optimization via knockout, irreversible case]

Input: a consistent stoichiometric matrix S; a target reaction; a level λ.

Output: a small set X guaranteeing (11.9) holds.

Initialization: $w = \mathbf{1}$ (the vector of all ones).

Algorithm: Solve the following linear program for t:

$\min \sum_j t_j$ s. t. $Kv = \mathbf{0}, t \geq \mathbf{0}, t + v \geq \mathbf{0}, v_{\text{target}} = -1, v_{\text{biomass}} \geq \lambda$.

$X := R(t)$

Reduce X to minimality by the take-one-out procedure.

Termination: return X.

Before ending this chapter, we note that although minimizing the 0-norm of a nonzero vector in a convex set is an NP-hard problem, a variant of this problem is of interest in the field of compressed sensing [116, 117, 118], and a number of techniques have been devised for it [119], some of which are surprisingly successful. We describe one particular technique, proposed in [120], which is based on a weighted 1-norm minimization. Some reasons for the efficacy of this technique are given in [121].

This technique consists in iteratively varying the vector c in equation (11.1) depending on the components of the vector returned by the previous iteration; here, F_i can be any convex set. More specifically, a constant $\tau > 0$ is chosen. $c = \mathbf{1}$ at the first the iteration, and then at each subsequent iteration, c is chosen as $c_i = (|v_i| + \tau)^{-1}$. Convergence typically occurs after only a few iterations.

This technique does not apply to the search for EFMs because a vertex of F_i is discovered at the first iteration if an active set algorithm is used. However, it can apply to the search for MCSs if in addition to $t + v \geq \mathbf{0}$ we impose the constraint $t - v \geq \mathbf{0}$ instead of $t \geq \mathbf{0}$, so that at the minimum, t becomes the component-wise absolute value of v. It can also be adapted to work in the search for minimal media and small cut sets for strain optimization. In fact, it can be used instead of the take-one-out procedure in many cases. However, the resulting improvements in the 0-norm were only marginal in most of our experiments.

Chapter 12

Metabolic Model Refinement

In this chapter, we propose a framework for refining a computational model of metabolism based on the results of growth experiments. Although metabolic networks are arguably the most complete networks available today, they still suffer from incomplete agreement with experimental results and thus need to be refined [122, 123, 124, 125]. Once again, we consider only the fully irreversible case here. We begin by discussing a new approach to correcting false positives and false negatives by respectively deleting reactions and removing constraints from the model. We then offer a critique of the way the biomass reaction is chosen in some current models and propose a novel algorithm for de novo biomass reaction discovery. We conclude by sketching methods for the incorporation of metabolite profiles into a metabolic model.

12.1 Model Refinement with Growth Experiments

Since we are concerned only with qualitative analysis, we will consider only the binary outcomes of growth experiments in different media (and possibly under different gene deletions). Suppose that there are k such experiments. The i-th experiment can be

succinctly described by the triplet $(\mathcal{D}_i, \nu_i, \varsigma_i)$. Here, $\mathcal{D}_i$ is the set of reactions that are disabled (these contain external reactions that provide nutrients not included in the media and any reactions catalyzed by the products of knocked-out genes), $\nu_i \in \{+, -\}$ is the experimental (in vitro) outcome and $\varsigma_i \in \{+, -\}$ is the computational (in silico) outcome. Here, + stands for "growth" and −, for "no growth". We will refer to a set of conditions as positive if $\varsigma_i = +$, and negative otherwise. It will be a true (positive or negative) if $\nu_i = \varsigma_i$, and a false (positive or negative) otherwise.

As pointed out in [125], there are two possible kinds of inconsistencies: experiments with $(\nu_i, \varsigma_i) = (-, +)$ (hence, false positives) and those with $(\nu_i, \varsigma_i) = (+, -)$ (hence, false negatives). Let us now examine the possible ways to refine the model.

First of all, the set of reactions available to the organism can be altered. Deleting reactions is equivalent to constraining the flux through them to 0, and therefore can only restrict the flux cone. Therefore, false positives can be corrected by reaction deletion. On the other hand, adding reactions can only enlarge the flux cone because the original flux cone is equivalent to the modified flux cone with all the newly added reactions constrained to have no flux. Therefore, false negatives can be corrected by reaction insertion. This is precisely the approach most commonly adopted [125].

However, while the set of reactions that can be deleted is typically of moderate size, the set of reactions that can be inserted usually contains an entire database, which leads to a two-fold risk. There may be multiple acceptable solutions of similar size, and furthermore, most of these solutions will be biologically meaningless since the reactions to be added may be entirely foreign to the organism being studied. For these reasons, we propose a different way of dealing with false negatives, which is to alter the set of metabolites.

Inserting metabolites would mean accounting for additional metabolites (such as co-factors, for instance) that were not originally included in the reactions of a network.

This yields additional constraints on the flux cone and therefore restricts it. On the other hand, deleting metabolites would mean relaxing the balance constraints on these metabolites in the model, and thus would enlarge the flux cone and allow for the correction of false negatives. This makes particularly good sense for so-called currency metabolites, such as ions and water molecules, since they participate in a number of non-metabolic reactions inside the cell. Furthermore, discovering metabolites that should not be forced to be balanced in order to agree with experiments can then be used as a guide in the search of missing reactions in a later stage. In addition, the sign of the overall balance for these metabolites can be used as an indicator for whether these reactions consume or produce them.

The foregoing discussion indicates that the false positives and the false negatives can be corrected independently from one another. Typically, algorithms that have been proposed in the past [122, 125] are based on Integer Linear Programming, which limits their applicability to at most a handful of experiments. Since our approach is based only on Linear Programming, it is able to handle much larger sets of experiments all at once. It is based on reaction deletion for the positive experiments and metabolite deletion for the negative ones, and also uses duality. We describe it below.

12.2 Correcting False Positives

A false positive is an experiment in which the organism grows in silico, but not in vitro. Therefore, in order to correct it, one would need to delete a set X of reactions that is a cut set (not necessarily minimal) for the biomass reaction. The stoichiometric matrix we consider for the i-th experiment is not S but $S_i := S_{-\mathcal{D}_i}$, which is S with the columns corresponding to the blocked reactions $\mathcal{D}_i$ removed. Then, by lemma 3, we are looking for negative supports of vectors u in the rowspace of S_i with $u_{\text{biomass}} = 1$.

In the case of multiple false positives, we are looking for a common negative

support of such vectors. While it seems difficult at first to encode that restriction, we note the following key fact: for nonnegative vectors, the union of the supports is equal to the support of the component-wise maximum. In other words, $R(u) \cup R(v) = R(\max(u, v))$. Therefore, it is sufficient to introduce one more vector which will be the component-wise maximum of all the others and then to minimize its support.

While we cannot make sure the true positives remain positive after deleting the set X directly, we begin by ensuring that all the reactions essential in any one of the true positive experiments do not get deleted. In other words, we identify all the cut sets of size 1 in the true positive experiments. For this purpose, we use the version of **Algorithm10** adapted to fully irreversible networks described previously.

This gives the following algorithm.

Algorithm12 [Correcting false positives, fully irreversible case]

Input: a consistent stoichiometric matrix S; a set of experiments $(\mathcal{D}_i, \nu_i, +)$.

Output: a small set X of reactions to be deleted.

Initialization: $U := \emptyset; I_+ := \{i | \nu_i = +\}$ and $I_- := \{i | \nu_i = -\}$

Iteration: **for** $i \in I_+$

Compute the set U_i of essential reactions under condition i

$U := U \cup U_i$.

for $i \in I_-$

$K_i :=$ **NullspaceMatrix**(S_i).

Solve $\min \sum_j v_j$ subject to $v \geq t_i \; \forall \; i \in I_-, v_U = 0,$

where $K_i u_i = 0, t_i + u_i \geq 0, u_{i,\text{biomass}} = 1, t_i \geq 0 \; \forall \; i \in I_-$.

$X := R(v)$

Reduce X to minimality by the take-one-out procedure.

Termination: **return** X.

12.3 Correcting False Negatives

A false negative is an experiment in which the organism grows in vitro, but not in silico. Since we are dealing with false negatives by relaxing the balance restrictions for a subset Y of metabolites, we are looking for vectors v such that $Sv = w$ has a small support. This support corresponds precisely to the metabolites that are not balanced.

In the case of multiple false positives, we use the key fact that the union of supports of nonnegative vectors equals the support of their component-wise maximum.

While we cannot make sure the true negatives remain negative after unbalancing the metabolites in Y directly, we ensure that all the metabolites essential in any one of the true negative experiments do not get deleted. To do this, we identify a y such that $y^T S_i = e_{\text{biomass}}$ for condition i, and then test only those metabolites in $R(y)$.

This gives the following algorithm.

Algorithm13 [Correcting false negatives, fully irreversible case]

Input: a consistent stoichiometric matrix S; a set of experiments $(\mathcal{D}_i, \nu_i, -)$.

Output: a small set Y of metabolites to be unbalanced.

Initialization: $U = \emptyset, I_+ := \{i|\nu_i = -\}$ and $I_- := \{i|\nu_i = +\}$.

Iteration: **for** $i \in I_+$

Compute the set U_i of essential metabolites under condition i

$U := U \cup U_i$.

Solve $\min \sum_j v_j$ subject to $v \geq t_i \; \forall \; i \in I_-, v_U = 0$,

where $S_i v_i = w_i, v_{i,\text{biomass}} = 1, t_i + w_i \geq 0, t_i - w_i \geq 0 \; \forall \; i \in I_-$.

$Y := R(v)$

Reduce Y to minimality by the take-one-out procedure.

Termination: **return** Y.

12.4 De Novo Biomass Reaction Discovery

The composition of the biomass reaction is generally subject to a lot of experimental error. It seems to be a tacit assumption in the field of metabolic network analysis that small changes in the composition cannot significantly affect the growth rate if biomass maximization is assumed. However, this is not the case. As we show below, the biomass composition needs to obey a set of one or more special linear relations in order for the organism to be able to exhibit growth. Furthermore, our computational experiments resulted in the surprising discovery that a number of existing metabolic models listed on the In Silico Organisms website [126] do not exhibit organism growth at all, even in rich media and without any restrictions on the direction of the fluxes. The fact that this had not been discovered previously is due to the effects of round-off error, which we discuss more in detail in the next chapter.

For convenience, we will assume that the biomass reaction is given by S_0 and use $S_{>0}$ to denote the stoichiometric matrix without the biomass reaction. The same convention will apply to flux vectors. We begin by stating the following lemma:

Lemma 14 (Restrictions on the Biomass Composition). *Given a stoichiometric matrix S with biomass reaction S_0, growth is possible only if $S_0 \in Col(S_{>0})$. Furthermore, if this condition is not satisfied, there exists $\delta > 0$ such that it is not satisfied for any S_0' such that $\|S_0 - S_0'\| \leq \delta$.*

One possible way of remedying the situation when an organism cannot exhibit in silico growth because of stoichiometry would be to find a small subset of metabolites to be unconstrained to allow growth. This can be achieved with the special case of **Algorithm13** when there is only one experiment and $\mathcal{D}_1 = \emptyset$ (sometimes called a *rich medium*). Note that the corollary to lemma 11 implies that finding the smallest such subset is NP-hard.

However, it is also possible that the organism cannot exhibit growth due to the irreversibility constraints. In that case, it may be possible to remove a small subset of those constraints to allow for growth. For a fully irreversible network (which we can assume S to be after reconfiguration), this is equivalent to finding a vector v such that $Sv = 0$ with $v_0 = 1$ and $R_-(v)$ is small. However, this is precisely the problem of finding a minimal cut set for reaction 0 in K, the nullspace matrix of S. Hence, this problem can be solved with the special case of **Algorithm12** when there is only one experiment and, as above, $\mathcal{D}_1 = \emptyset$ (but with S and K interchanged). Once again, finding the optimal solution to this problem is NP-hard by lemma 11.

So far, we discussed how to refine the model to allow growth for the specific biomass composition we are given. However, there is an alternative way to refining the model based on growth experiments: instead of removing constraints on metabolites or reactions, we can find a suitable biomass reaction *de novo*, though possibly using some qualitative information from the biomass reaction that was originally given.

It follows from lemma 14 that, for any growth experiment $(\mathcal{D}_i, \nu_i, \varsigma_i)$ with $\nu_i = +$ to be correctly replicated in silico, we need to have $S_0 \in \mathrm{Col}(S_i) = \mathrm{Col}(S_{-\mathcal{D}_i})$, where Col is the column space. Hence S_0 is a vector in $C_+ := \cap_{\{i|\nu_i=+\}}\mathrm{Col}(S_i)$. An important question is whether such a restricted biomass reaction can in principle correctly predict the experiments with $\nu_i = -$ as well? The following result states that this is indeed possible (at least in the fully reversible case) provided that the result of every individual experiment with no in vitro growth can be correctly predicted by some biomass reaction in C_+.

We use the definitions $I_+ := \{i|\nu_i = +\}$ and $I_- := \{i|\nu_i = -\}$ from the previous section. We call an experiment i in I_- *satisfiable with respect to* I_+ if there is a suitable biomass reaction that produces in silico growth for every experiment in I_+ and no in silico growth for i.

Lemma 15 (Existence of a Suitable Biomass Reaction in Fully Reversible Networks). *In a fully reversible network, if every experiment i in I_- is satisfiable with respect to I_+, then all the experiments in I_- are simultaneously satisfiable with respect to I_+. Furthermore, in that case, for any biomass reaction S_0 that produces in silico growth for every experiment in I_+ and any $\delta > 0$ there exists a S_0' such that $\|S_0 - S_0'\| \leq \delta$ and S_0' simultaneously satisfies all the experiments in I_- with respect to I_+.*

Lemma 15 is interesting because it shows negative design (avoiding growth in I_- experiments) can be effectively combined with positive design (allowing growth in I_+ experiments) in the case of metabolic networks. This has been shown to be a difficult problem in the case of RNA and protein design in an earlier paper by the author and his collaborators [4]. We also note that a basis for C_+ can be easily computed by using the fact that

$$T_1 \cap \cdots \cap T_k = (T_1^\perp + \cdots + T_k^\perp)^\perp, \tag{12.1}$$

where $T^\perp$ denotes the orthogonal complement of the vector subspace T and $+$ is the direct sum of vector spaces.

In order to further restrict the de novo biomass reaction, we can use some of the clues provided by the original biomass reaction given for the model. One possible condition is that the biomass reaction to be designed should only contain those metabolites that are considered part of the original biomass composition. Another possible condition is that the deviation from the original biomass reaction be minimized; depending on whether the deviation is measured by the 1-norm $\|\cdot\|_1$ or the Euclidean norm $\|\cdot\|_2$, this leads to a linear program or a least-squares problem, respectively. Once a satisfying biomass reaction has been found, we need to check whether it produces growth for any of the experiments in I_-. If it does, a small perturbation as described in lemma 15 will yield the desired result, provided that a simultaneously satisfying biomass reaction exists. This leads to the following algorithm:

Algorithm14 [Biomass reaction design, fully reversible case]

Input: a stoichiometric matrix $S_{>0}$ with biomass reaction S_0; experiments $(\mathcal{D}_i, \nu_i, \varsigma_i)$.

Output: a biomass reaction S_0^* that agrees with all the experiments, if one exists.

Initialization: $I_+ := \{i|\nu_i = +\}$ and $I_- := \{i|\nu_i = -\}$

Iteration: Compute a basis B for C_+ using equation (12.1) with $T_i := \text{Col}(S_i)$

Restrict B by enforcing the condition $R(x) \subseteq R(S_0)\ \forall\ x \in \text{Col}(B)$

for $i \in I_-$

if $\text{Col}(B) \subseteq \text{Col}(S_i)$

return 1

Solve $\min \|S_0^* - S_0\|$ subject to $S_0^* \in \text{Col}(B)$

for $i \in I_-$

if $S_0 \in \text{Col}(S_i)$

Perturb S_0^* away from $\text{Col}(S_i)$

Termination: **return** S_0^*.

In the case of a fully irreversible metabolic network, however, things are not so simple. As the following example shows, the conclusion of lemma 15 does not hold.

$$S := \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}; (\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3) = (\{1\}, \{2\}, \{3\}); (\nu_1, \nu_2, \nu_3) = (-, -, +). \quad (12.2)$$

Here, even though the individual no-growth experiments are satisfiable with respect to the unique growth experiment, they are not simultaneously satisfiable with respect to it. This is because, unlike proper vector subspaces, proper "subcones" can completely cover a cone. In addition, computing the vertex representation of the intersection of two (polyhedral) cones given by their vertices is NP-hard [131], which makes it difficult to apply equation (12.1), and hence to decide the existence of a

simultaneously satisfying biomass reaction. At this stage, we are forced to leave this as an open question.

12.5 Incorporating Metabolomics Data

This section is somewhat speculative, as it describes work in progress. *Metabolomics*, which is the measurement of the concentrations of different metabolites in a biological system, is a growing field of bioinformatics [127]. It has not yet been successfully integrated into computational and systems biology, despite a couple of initial attempts [128, 129]. Here, we discuss how this could be done in principle.

Because constraint-based models make the fundamental assumption about the system being in a quasi steady-state, equation (9.1), which takes concentrations into account, reduces to equation (9.2), which does not. There are, however, two possible ways of bringing the concentrations back into the model.

The first, and perhaps more experimentally challenging way, is to measure metabolite concentrations at regular time intervals and then try to use equation (9.1) with the derivatives approximated by finite differences. Thus, instead of being in the nullspace of S, feasible flux modes would be in an affine space obtained by translating the nullspace of S by a particular solution of $Sv = d$, where d is the vector of the computed finite differences. The limitations of this approach lie in the resolution of metabolite concentration measurements that are experimentally feasible.

The second, and perhaps more computationally challenging way, is to incorporate the concentrations in a more indirect manner through the energy constraints, as suggested by Hoppe et al. [130]. We recapitulate and modify their approach below.

We assume that the system is at a constant temperature and pressure, allowing the Gibbs free energy of a reaction to be identified with its chemical potential [89], and ignore the proportionality constant. Given a vector c of metabolite concentrations,

the Gibbs energy of a reaction i is given by

$$\Delta G_r := \Delta G_r^0 + RT \sum_j S_{i,j} \log(c_j), \tag{12.3}$$

where the $S_{i,j}$ are simply the stoichiometric coefficients.

The values of ΔG_r^0 for many reactions may be found in the literature. Defining w to be the vector of the ΔG_r divided by RT, and w_0 to be the vector of ΔG_r^0 divided by RT, our new set of constraints now includes

$$w = w_0 + S \log(c), \tag{12.4}$$

along with the constraints (9.2), (9.3), (9.4) and (9.5).

Of course, in order to make (12.4) and (9.5) consistent, we need to make sure that $w_0 \in \text{Row}(S)$. This may not be the case due to experimental error, and the best guess as to what the correct w_0 should be is the projection of w_0 onto $\text{Row}(S)$, which is the least-squares solution of $y^T S = w_0$ and can be found efficiently [23].

If both w_0 and c are completely known, then w is determined by equation (12.4) and will place a constraint on the sign of each flux in v. This may be too constraining for the model, and we will need to account for the possibility that the concentrations in c have an uncertainty associated with them. In [130] soft penalties are associated with each concentration based on its deviation from the measured value.

Instead of imposing penalties, one could efficiently identify a small subset of the values of w that overconstrain the model by an algorithm identical to the one described in the previous section (based on **Algorithm12**), except with weights $|w_i|$ rather than unit weights so as to favor the components of w that are closest to 0. Then, either the corresponding ΔG_r^0 values could be verified, or a small subset of the concentrations could be identified for a repeated measurement.

Chapter 13

Implementation Details

Although the methods described so far provide a complete arsenal or toolkit for metabolic network analysis, it is critically important to ensure that they can be implemented in a way that makes them both reliable and efficient. These features are the focus of this chapter. We begin by discussing existing implementations, as well as their range of applicability. We go on to describe the difficulties that can arise when floating-point arithmetic is used and the advantages and disadvantages of using fractional arithmetic instead. We continue with some remarks about linear optimization and the guarantees it can provide. We conclude the chapter by listing all the functions that have been implemented for *FAMA: Fast and Accurate Metabolic Analysis.*

13.1 Existing Implementations

A number of tools for analyzing constraint-based metabolic models are available to the scientific community [132, 133, 134]. However, these tools have important limitations. Some of them, based on floating-point arithmetic, may yield incorrect qualitative

predictions, while others are too computationally demanding for large genome-scale metabolic models. Few are able to answer all the questions a researcher may ask. Let us briefly discuss some of these tools.

COBRA [132] is an implementation of metabolic network analysis in the MATLAB environment [135]. It provides a number of functionalities for metabolic network analysis, and appears to be the most complete tool available today. It can read models in the Systems Biology Markup Language (SBML) format [136] and is flexible enough to allow the user to change bounds on the fluxes, add or delete reactions, and alter the objective function (biomass reaction). The main objective of the COBRA toolbox is to perform flux-balance analysis (FBA), which involves maximizing the objective function subject to the usual constraints. Since this involves the solution of a linear program, COBRA is configured to use any of the following solvers: lp_solve, GLPK, LINDO, CPLEX and Mosek [137, 138, 139, 140, 141]. Only the first two of these are freely available through the general public license (GPL).

In addition to FBA, COBRA can perform flux variability analysis (FVA), single and double gene deletions, a special type of dynamic simulation (dFBA), flux sampling, and sensitivity analysis. It can also identify modules and represent them using the Cytoscape software [142].

While quick and versatile, COBRA has one important drawback: all the solvers it relies on (as well as the MATLAB environment itself) work exclusively in floating-point arithmetic, which, as we will see in the following sections, is highly problematic.

METATOOL [133] is a collection of tools currently implemented to work under MATLAB [135] or GNU Octave [143]. However, there is also a stand-alone version that works with text files. METATOOL also provides a number of functionalities, but its main purpose is determining the complete set of elementary flux modes of a metabolic network. In the process of finding the elementary flux modes, it also

performs a reduction of the stoichiometric matrix somewhat akin to the reduction to a canonical form described in the Structure chapter.

The main limitation of METATOOL is that it can only compute the elementary flux modes (along with some structural information), and that this computation is still extremely slow for genome-scale models, despite some recent advances [144].

CellNetAnalyzer [134] is a more general tool for network analysis which processes not only metabolic networks, but also signaling networks. It is also implemented to work under MATLAB [135], and can use the GLPK library [138] for linear optimization. It integrates a lot of the functionality of METATOOL while providing additional functionality of its own, such as FBA, sensitivity analysis, analysis of the topological properties of the network, prediction of mutant phenotypes, and the calculation of minimal cut sets.

CellNetAnalyzer in some sense provides a compromise between COBRA and METATOOL in that it combines some (though not all) of the functionality of both. It is probably the best currently available implementation of a general-purpose metabolic network analysis. Its mutant phenotype predictions are based not on FBA, but on elementary flux modes - this makes the predictions more accurate (as explained in the following section) but at the same time unnecessarily slows them down. As we showed in the chapter on Minimality, these predictions can easily be made without a complete knowledge of the elementary flux modes. Ultimately, the most appropriate tool depends on the task at hand.

13.2 Floating-Point Arithmetic

Let us now examine the question of the effect of floating-point calculations. Generally speaking, most linear-algebraic manipulations can be performed in a way that is *backward stable*, i.e. yielding the exact solution to a slightly perturbed version of

the original problem [23]. The algorithms implemented in high-level programming languages such as MATLAB [135] are able to handle *rounding errors* fairly well. However, it turns out that backward stability is not sufficient for the qualitative analysis of metabolic networks.

Recall our assumption that the stoichiometric matrix has only rational coefficients. Most of the time, these coefficients will be small integers. For a metabolic network represented by such a matrix, it is not interesting to know whether a slightly perturbed version of the network can have a non-zero flow through a given reaction i because this says nothing at all about whether this is true for the actual network. In fact, as discussed in the chapter on Network Refinement, a number of metabolic models cannot exhibit any growth, yet these models are routinely used by researches to make predictions about the essentiality of certain genes and the effect of various genetic modifications on the rate of growth.

We originally implemented all our algorithms in MATLAB and then realized that the conclusions of lemmas 8 and 9 did not seem to hold. This was not the case for the small example networks we tested the algorithms on, so the original hypothesis was that the lemmas were incorrect. However, upon further investigation we discovered that the problems were due to numerical errors. We also discovered that some enzyme subsets were missed while others were identified incorrectly and that even Gaussian elimination was not a reliable predictor of which rows were redundant in a stoichiometric matrix. Rounding results to different precisions only shifted the difficulty to the discovery of the appropriate values of the precision. Finally, we made the decision to redo the implementation using fractional arithmetic in a different language (MATLAB supports only floating-point arithmetic, while MAPLE [145] is generally too slow). This implementation is described in the following section.

13.3 Fractional Arithmetic

The language we chose for implementing our algorithms was Python [42] because it is a high-level language that is freely available to the public and because it provides a flexible and versatile set of *modules* as well as capabilities for interfacing with code in other languages.

Python's *fractions* module, available in versions 2.6 and higher, allows all the processing to be done in fractional arithmetic. However, for completeness we also implemented our own version of fractional arithmetic, which can be used with earlier versions of Python, for backward compatibility. Our version represents a fraction with a list of the form [numerator, denominator], automatically reduced to the lowest terms, and contains all the standard arithmetic operations, as well as comparisons and type conversions.

Unfortunately, there was no freely available implementation of the Gauss-Jordan elimination algorithm, a key step in the **NullspaceMatrix** procedure, using fractional arithmetic. After trying to use the BlocDiag.cpp program from an early version of METATOOL [133] and running into a number of difficulties with integer overflow, we implemented our own version of it. Although it is slow, it takes only on the order of 5-10 minutes for matrices of size $2,000 \times 3,000$. Speeding up this implementation, or finding an alternative one in a different language, is high on the priority list for improving this implementation.

Aside from this procedure, however, all the subroutines are comparable in speed with the original MATLAB implementation, while giving a clear edge in terms of correctness. In all our computational experiments, the numerators and denominators did not exceed 10 digits, which means that most of the intermediate results can be stored inside 32 bit words (*long int*).

One disadvantage of working in fractional arithmetic is that the overall running

time slows down significantly (by a factor of roughly 10) compared to floating-point arithmetic. However, the extra time spent in processing is more than made up by the guaranteed correctness of the results. This idea of guaranteed correctness is discussed further in the following section.

13.4 Linear Optimization

Even though some of the algorithms we need use only linear algebra, others make extensive use of linear optimization. However, as mentioned earlier, almost all implementations of linear program solvers use floating-point arithmetic. Luckily, two packages are available which use fractional arithmetic, and are therefore guaranteed to give exact results. These are exlp [146] and QSopt_ex [147]. Out of those, exlp is generally slower because it performs all the operations in exact rational arithmetic, and it is also less flexible as it accepts only input files in the MPS format [148]. On the other hand, QSopt_ex accepts both MPS and LP formats and is faster due to its algorithm, which we briefly discuss now.

QSopt_ex is built on top of the QSopt solver [149], which uses the simplex algorithm [150]. It works as follows. It starts from a baseline precision of $d = 32$ bits and performs the optimization in floating-point arithmetic. However, once the solution is found, its *basis representation* (namely, the subset of constraints that are satisfied with equality) is checked for optimality using a rational LU factorization of the relevant submatrix. If the solution is optimal, it is returned; otherwise, the precision d is increased and a solution is recomputed starting from the previously returned basis (which is usually close to optimal even when it is not optimal). This continues until an optimal solution is found that passes the optimality test in rational arithmetic [151].

Note that, because the returned basis is checked for optimality in exact arithmetic,

we are guaranteed the correctness of our output. This is different from the situation with floating-point based linear program solvers, where a small value for a variable could mean that it should be 0, but could equally well mean that it is actually small but nonzero. However, there is a tolerance (defined by the linear program) that could ensure that we actually get exact results from this floating-point computation. This is the largest determinant of a square submatrix of the constraint matrix A. Indeed, when a vertex is being computed, the coordinates in the basis B are determined by $A_B x = b_B$, while the others are all 0.

This problem of finding the largest determinant of a square submatrix of an integer matrix A turns out to be NP-hard [152]. Khachiyan [153] gives an approximation algorithm that gets it within a factor of $m^{(m/2)}$, where m is the number of rows of A. An alternative algorithm suggested by Michel Goemans in personal communication can approximate its logarithm within a factor of 2/5 using the fact that this logarithm can be extended to a submodular non-monotone function on the columns of A. Unfortunately, neither of these results is practical as these estimates would require a precision of at least 300 bits for the stoichiometric matrices we typically work with.

QSopt_ex is the only external function currently used in our implementation. It is interfaced with using file input/output, because that turns out to be a lot easier to manage while increasing the overall running time by less than 10%. The output files can be destroyed or kept as efficiently checkable certificates that the solution of the linear program is exact.

We end this chapter by listing the functions used in our current implementation with their associated comments.

13.5 A Complete List of Functions

Code Listing 13.1: Headers of all the implemented functions

```
def reduceMatrix(N, Irr, Filename = 'Reduction.txt'):
    # This function computes the reduced form of a given stoichiometric matrix
    # assuming that the specified list of reactions is irreversible.
    # The reduction proceeds in several steps, each of which is verified using
    # exact linear programming whenever possible.
    # The first step identifies and groups together all isozyme subsets.
    # The second step eliminates all thermodynamically blocked reactions.
    # The third step eliminates all stoichiometrically blocked reactions.
    # The fourth step identifies and groups together all enzyme subsets.
    # The fifth step reconfigures the network (splitting reversible reactions).
    # Intermediate elimination of redundant constraints happens between steps.

def dotProduct(List1, List2):
    # This function computes the dot product between 2 vectors in list representation.

def groupIdentical(List):
    # This function returns lists of indices corresponding to identical elements.
    # For instance, on input [1,2,3,2,1,3,4] returns [[0,4],[1,3],[2,5],[6]] in order.

def findIsozymes(N):
    # This function determines all the isozyme sets in a given stoichiometric matrix.
    # Isozymes are defined as two reactions that are identical, not mutual reverses.

def findRedundant(N):
    # This function returns a list of non-redundant rows in a given matrix.
    # These non-redundant rows form a basis for the rowspace of the matrix.

def findTBlocked(N, Irrev, basename = 'TBlocked.lp'):
    # This function finds thermodynamically blocked reactions in a metabolic network
    # given by its stoichiometric matrix and a list of irreversible reactions. See
    # paper for a detailed justification of the algorithm.
    # Note: returned reactions are irreversible!

def findPosSupport(N, support, weight = [1], Filename =
'trial.lp', Min = 0):
    # This function finds the vector of smallest weight in the rowspace of N whose
    # support is restricted to a given set of entries, which must be non-negative!
    # Note: if the weight vector has a single component, it is assumed to be 1!
    # If a nonzero Min value is specified, components in the support are at least Min.

def findSBlocked(N, NB = False):
    # This function finds stoichiometrically blocked reactions in a metabolic network
    # given by its stoichiometric matrix. If NB is TRUE the nullspace basis is output.

def vectorInSpan(N, vec, Filename = 'trial.lp'):
    # This function determines whether a given vector is in the row span of a matrix.
    # This is achieved by solving a linear program in QSOpt_ex and checking its value.

def findUnidirectional(N, Irrev):
    # This function finds all unidirectional (effectively irreversible) reactions.
    # NOTE: It assumes that all the reactions in the network can have nonzero flux;
    # otherwise may incorrectly classify blocked reactions as only negative.

def findSubsets(N, NB = False):
    # This function finds all enzyme subsets, as well as the reactions that are not
    # part of any enzyme subset (in "active").
    # If NB is TRUE, input is assumed to be the nullspace basis; else it is computed.
```

```
def findFeasible(N, special, Irrev = [], pos = True, Filename =
'trial.lp', disable = []):
    # This function finds a feasible vector in the nullspace of N with given set of
    # irreversible reactions. The entry corresponding to special is 1 if pos is TRUE,
    # -1 otherwise.
    # Additional feature: it is possible to specify a subset of reactions to disable!

def findRatio(N, react1, react2, Irrev, Max = True, Filename =
'trial.lp'):
    # This function finds the minimum or maximum ratio of two given entries in Null(N)
    # with given set of irreversible reactions. Maximize if Max = TRUE, else minimize

def processFile(Filename, opt = False, destroyIn = True,
destroyOut = True):
    # processes a .lp file using QSOpt_ex, returning only the value by default
    # if opt = TRUE, returns the optimal vector as well
    # destroyIn = TRUE means that the input file is deleted after processing
    # destroyOut = TRUE means that the output file is deleted after processing

def parseOutput(Filename, opt = False):
    # This function parses a solution file in the QSOpt_ex format
    # and returns a list containing the value of the objective function ([] if
    # the problem is infeasible, [float('Inf')] if it is unbounded, else [num, den]).
    # If opt = TRUE, also returns a dictionary with values of the nonzero variables.

def convertFrac(element):
    # This function converts an integer or a string fraction into integers [num, den]

def mapBackCutset(cutsets, cols, prevRev, Isozymes):
    # Reverses the steps performed in reducing a stoichiometric matrix
    # in order to create cutsets corresponding to the given ones.

def mapBackFlux(fluxes, cols, prevRev, Subsets):
    # Reverses the steps performed in reducing a stoichiometric matrix
    # in order to create flux vectors corresponding to the given ones.

def convertVector(dict, length):
    # Converts a dictionary representation of a vector into a list representation

def convertBackVector(List, variable = 'V'):
    # Converts a list representation of a vector into a dictionary representation

def findEFMs(N, prevRev, rec = True, I = []):
    # This function finds a large set of EFMs in a stoichiometric network N
    # excluding the "trivial" ones of the type e_i + e_j where i, j come from
    # the reconfiguration of a single initially reversible reaction.
    # Note: assumes N has been reconfigured. Otherwise, set rec to be False
    # and specify the set of irreversible reactions in I (ignoring prevRev!)

def findEFM(N, special, zero = [], Filename = 'trial.lp', rec =
True, I = []):
    # This function finds a set of EFMs in a stoichiometric network N that
    # contain a special reaction with coefficient 1 and do not contain any
    # of the reactions listed in zero. Note: assume N has been reconfigured.
    # Otherwise, set rec to be False and specify irreversible reactions in I.

def peelOff(Vector, otherVectors):
    # This function removes multiples of given vectors from a given one,
    # so as to make the result as sparse as possible. The vectors are
```

```
    # assumed to be nonnegative and to have rational entries [num, den].
    # NOTE: In general, the order in which this is done matters; however,
    # if the vector is an FM and the other vectors are EFMs, it does not!

def findUniqueEFMs(EFMs):
    # This function finds a subset of unique EFMs from a given set of EFMs
    # The defining criteria for equality is the same set of nonzero entries
    # Note that this function applies without modification to MCSs as well!

def findMin1Norm(N, special, weight = [1], zero = [], exclude =
[], eps = 1e-5, Filename = 'trial.lp', option = 'null', rec =
True, I = []):
    # This function finds the vector of smallest overall weight in the nullspace of N
    # whose reactions are assumed to be all irreversible unless rec is given as FALSE
    # In that case, the reactions considered irreversible should be specified in I.
    # Note that rec is ignored for option col! The entry corresponding to special = 1
    # If a zero constraint is present the corresponding entries are forced to equal 0
    # If exclusion constraints are given, ensures the support differs from given ones
    # Note: if the weight vector has a single component, it is assumed to be 1!
    # The available options currently are 'null' for nullspace, 'col' for columnspace

def extremeElement(vector, Max = False):
    # This function finds the smallest element of a vector.
    # If Max is True, finds the maximum element instead

def filterList(List):
    # This function removes the zero entries from a list of fractions

def representable(frac):
    # This function decides whether a fraction is representable as an exact decimal

def getnum(frac):
    # Returns the numerator of a fraction

def getden(frac):
    # Returns the denominator of a fraction

def makeFloat(frac):
    # This function returns the floating point number corresponding to a fraction

def negate(frac):
    # This function returns the negative of a fractional number

def absValue(frac):
    # This function returns the absolute value of a fractional number

def compare(frac0, frac1):
    # This function defines the comparison of two fractional numbers, frac0 & frac1.
    # Returns 1 if frac0 is bigger, -1 if frac1 is bigger, and 0 if they are equal.

def multiply(frac0, frac1):
    # This function defines the multiplication of two fractional numbers.

def add(frac0, frac1):
    # This function defines the addition of two fractional numbers.

def divide(frac0, frac1):
    # This function defines the division of two fractional numbers.

def subtract(frac0, frac1):
```

```
    # This function defines the subtraction of two fractional numbers.

def GCD(a,b):
    # This function computes the greatest common divisor of two integers

def LCM(a,b):
    # This function computes the least common multiple of two integers

def integralize(Vector, mult = False):
    # This function takes a rational vector and returns the smallest integer multiple
    # Note: this function works for a dictionary as well as a list representation(!)
    # If you need the mutliplier to be returned as well, change mult to a True value.

def makeIntegral(Matrix, mults = False):
    # This function makes a matrix into an integral one by scaling rows by integers.
    # If you need the mutlipliers to be returned as well change mults to a True value

def findMCSs(N):
    # This function finds a large set of MCSs in a stoichiometric network N,
    # several for each reaction. Note: assume N has been reconfigured.

def findMCS(N, special, Filename = 'trial.lp'):
    # This function finds a set of MCSs in a stoichiometric network N that
    # block a specified target reaction. Note: assume N has been reconfigured.

def testCutSet(N, Cutset, Target, Filename = 'trial.lp', rec =
True, I = []):
    # This function determines whether a given subset of reactions represents a cutset
    # for a given target reaction in a network which is assumed to be irreversible,
    # unless rec is specified to be False. In that case, reactions considered to be
    # irreversible should be specified in I.

def testSet(Set, Function, Args):
    # Function for testing a monotone property and identifying a local minimum for it
    # It determines whether a given set represents a MINIMAL set that returns TRUE
    # when the Function is called with Set as first argument and the other given Args
    # If Function fails returns FALSE; if not returns a subset that is minimal for it

def extractMinimal(Set, Function, Args):
    # This function returns a minimal subset of a given set that returns TRUE when
    # the Function is called with it as the first argument and the other given Args
    # Note that the processing is performed in the reverse order for simplicity.
    # NOTE: This function assumes that the Set itself returns TRUE for the Function.

def findSmallCutsets(Network, Target, skip = []):
    # This function systematically finds cutsets of size <= 3 in a REVERSIBLE network
    # It exploits the property that X is a cutset for i iff X-j+i is a cutset for j.
    # skip is a subset of indices omitted from the triplet testing for debugging only

def testSubsets(Network, Target, fluxes = [], Kmax = 3,
Filename = 'trial.lp', rec = True, I = [], startInd = 0,
startSubsets = []):
    # This function systematically tests all subsets of size at most Kmax to see
    # if they are cutsets of a given network for a specified target reaction.
    # It returns a list of all minimal such subsets in order of increasing size.
    # The network is assumed to be irreversible unless rec is specified to be False.
    # In that case, the reactions considered irreversible should be specified in I.
    # If the processing had already been partly done change startInd to reflect that!
    # In that case the optional argument startSubsets should contain the initial set.
```

```
def generateSubsets(n, k):
    # This function generates all k-subsets of range(n) in lexicographic order
    # http://snipplr.com/view/17859/generate-all-ksubsets-of-an-nset-sequentially/

def findMinAdded(N, special, weight = [1], exclude = [], eps =
1e-5, Filename = 'trial.lp', extra = {}, option = 'row'):
    # This function finds the smallest weight nonnegative vector to add to a vector in
    # the rowspace of N (nullspace if option 'null') to make the resulting vector >=0.
    # The entry corresponding to special is 1. Lower bounds may be supplied as extra.
    # Note: if the weight vector has a single component, it is assumed to be 1!

def getConstraintMatrix(Filename, varname = 'V'):
    # This function extracts the constraint matrix from an LP file

def checkUnblocked(RemoveConst, N, Irr, growth, Filename =
'Unblock.lp'):
    # This function checks whether removing a given subset of constraints
    # produces a metabolic network that allows for positive growth.
    # It returns TRUE if that is the case, and FALSE otherwise.

def minimalUnblock(N, Irr, growth, algo = 'norm', Filename =
'Unblock.lp'):
    # This function uses a heuristic to compute the smallest set of metabolites
    # that need to be unbalanced in order to produce growth in a blocked model.
    # N is the stoichiometric matrix, Irr are the irreversible reactions, growth
    # is the reaction to be unblocked. The returned set is only inclusion-minimal!
    # Algorithms currently implemented (specified by option algo):
    # greedy (removes metabolites in order of decreasing degree in the network);
    # norm (minimizes the 1-norm of the metabolite balance vector defined as Nv);
    # sense (same as norm, but with iterative weight updates until convergence);
    # redund (return a set of "redundant" metabolites plus an extra relevant one);
    # full (exhaustive enumeration of all possible reaction subsets of size <= 5).

def checkThermoUnblocked(RemoveConst, N, Irr, growth, Filename
= 'Unblock.lp'):
    # This function checks whether removing a given subset of irreversibility
    # constraints produces a metabolic network that allows for positive growth.
    # It returns TRUE if that is the case, and FALSE otherwise.

def minimalThermoUnblock(N, Irr, growth, Filename =
'Unblock.lp'):
    # This function uses a heuristic to compute the smallest set of reactions
    # that need to be made reversible in order to allow growth in a given model.
    # N is the stoichiometric matrix, Irr are the irreversible reactions, growth
    # is the reaction to be unblocked. The returned set is only inclusion-minimal!

def getDegrees(N, opt = 'metabs', direct = 0):
    # This function returns the set of degrees associated with the given network.
    # If opt = 'metabs', degrees of metabolites; if opt = 'reacts', of reactions.
    # If direct is set to a positive number count only the positive coefficients;
    # if to a negative number only the negative ones; otherwise count everything.

def checkEnergyBalance(N, EFMs, Internal):
    # Returns a boolean array corresponding to given EFMs, TRUE iff EFM is feasible.
    # See the paper for a justification of the linear program used for the checking.

def findMinimalMedia(N, Exchange, growth):
    # This function attempts to find minimal several minimal media allowing growth
    # for a given stoichiometric matrix and a given subset of "exchange" reactions.
```

```
def strainOptimize(N, target, biomass, Lambda = 1):
    # Finds a small knockout that results in a production of at least Lambda units of
    # flux through the target reaction for each unit of flux through the biomass one.
    # If the desired target flux level cannot be reached, the function returns False.

def GaussJordan(N, pivoting = True, Gauss = False):
    # This function returns the (pivoted) Row-Reduced Echelon Form of N.
    # It was adapted from http://adorio-research.org/wordpress/?p=192.
    # pivoting = True for partial row pivoting, False if no pivoting.
    # Gauss = True if a Row Echelon Form of N is needed, False otherwise.
    # Returns (A, R); A: transformed input matrix; R: indices of pivots.

def NullspaceBasis(N, GJ = False):
    # This function computes a nullspace basis of N
    # If GJ = True, the input is in reduced row-echelon form.

def transpose(Matrix):
    # This function creates the transpose of an input matrix

def cleanup(Matrix, indices = False):
    # This function removes the zero rows from an input matrix
    # It returns the matrix with the zero rows removed from it
    # If indices is True, also returns the indices of zero rows

def ReadMatrix(Filename):
    # reads a sparse matrix file and converts contents into fractions

def WriteASCIIMatrix(Matrix, Filename = 'Matrix.txt'):
    # Creates an ASCII representation of an input matrix in a file with given name.

def ReadASCIIMatrix(Filename, Rows = True, sep = '\t', Blocks =
True):
    # Reads an ASCII matrix from a file with the specified name.
    # If Rows is True, looks for row identifiers at the end of each row.
    # In that case, the rows are reordered before the matrix is returned!
    # To change the separator which is used to split the entries use sep.
    # If Blocks is True, the parser reads and returns a block structure!

def SubspaceIntersection(ListOfBases, NullB = False):
    # Returns the basis of the intersection of a set of vector spaces
    # given by a list of their basis vectors (assumed to be row vectors)
    # If NullB is TRUE, returns a bais for the intersection's nullspace.

def findEssential(Network, growth, Exchange, allowed, Filename
= 'trial.lp', rec = True, I = []):
    # This procedure finds all the reactions essential for growth in a medium defined
    # by the set of exchange reactions and the subset of allowed exchange reactions.
    # The computations are performed using files derived from the specified Filename.
    # The network is assumed to be irreversible unless rec is specified to be False.
    # In that case, the reactions considered irreversible should be specified in I.
```

Part IV

Application to M. Tuberculosis

"I will give you a talisman. Whenever you are in doubt, or when the self becomes too much with you, apply the following test. Recall the face of the poorest and the weakest man [woman] whom you may have seen, and ask yourself, if the step you contemplate is going to be of any use to him [her]. Will he [she] gain anything by it? Will it restore him [her] to a control over his [her] own life and destiny? In other words, will it lead to swaraj [freedom] for the hungry and spiritually starving millions? Then you will find your doubts and your self melt away."

Mohandas Karamchand Gandhi, Indian political and spiritual leader.

In this final part of the thesis, we apply our methodology to study the systems biology of Mycobacterium tuberculosis (M.tb). It is structured as follows. In the first chapter, we describe some features unique to M. tb. and highlight the public health challenge it represents. In the second chapter, we introduce an algorithm for merging two existing metabolic networks into a single one, and apply it to the two networks for M. tb. In the third chapter, we examine the gene essentiality predictions made by the combined model and compare them to those of the individual models. In the fourth and final chapter, we describe our method for producing a short list of putative drug targets that we hope will be useful for antitubercular drug discovery.

Chapter 14

The Elusive Mycobacterium

Mycobacterium tuberculosis (M. tb), then known as the tubercle bacillus, was first described on 24 March 1882 by Robert Koch, who subsequently received the Nobel Prize in physiology or medicine for this discovery in 1905; the bacterium is also known as Koch's bacillus. This chapter describes some of the known facts about M. tb and the challenges inherent in its treatment. We begin by discussing its physiology, then go on to discuss its diagnosis and treatment, and conclude by describing the biggest challenges remaining in the fight against M. tb.

14.1 Physiology

M. tb requires oxygen to grow. It does not retain any bacteriological stain due to high lipid content in its wall, and thus is neither Gram positive nor Gram negative; hence Ziehl-Neelsen staining, or acid-fast staining, is used. While Mycobacteria do not seem to fit the Gram-positive category from an empirical standpoint (i.e., they do not retain the crystal violet stain), they are classified as acid-fast Gram-positive bacteria due to their lack of an outer cell membrane. [154]

M. tb divides every 1520 hours, which is extremely slow compared to other bacteria. It is a small bacillus that can withstand weak disinfectants and can survive in a dry state for weeks. Its unusual cell wall, rich in lipids (e.g., mycolic acid), is likely responsible for this resistance and is a key virulence factor. [154]

When in the lungs, M. tb is taken up by alveolar macrophages, but they are unable to digest the bacterium. Its cell wall prevents the fusion of the phagosome with a lysosome. Consequently, the bacteria multiply unchecked within the macrophage. The bacteria also carry a gene which prevents acidification of the phagosome, and can evade macrophage-killing by neutralizing reactive nitrogen intermediates. [154]

M. tb appears to be genetically diverse. This genetic diversity results in significant phenotypic differences between clinical isolates. M. tb exhibits a biogeographic population structure and different strain lineages are associated with different geographic regions. Phenotypic studies suggest that this strain variation never has implications for the development of new diagnostics and vaccines. Micro-evolutionary variation affects the relative fitness and transmission dynamics of antibiotic-resistant strains. [154]

14.2 Diagnosis

To diagnose M. tb, sputum is taken on three successive mornings as the number of organisms could be low, and the specimen is treated with 3% KOH or NaOH for liquefaction and decontamination. In the most common staining technique, the Ziehl-Neelsen stain, AFB are stained a bright red, which stands out clearly against a blue background. The reason for the acid-fast staining is because of its thick waxy cell wall. The waxy quality of the cell wall is mainly due to the presence of mycolic acids. This waxy cell wall also is responsible for the typical caseous granuloma formation in tuberculosis. [154]

A grading system exists for interpretation of the microscopic findings based on the number of organisms observed in each field. Patients of pulmonary tuberculosis show AFB (acid fast bacillus) in their sputum in only 50% of cases, which means that, even if no organisms are observed, further investigation is still required. Acid-fast bacilli can also be visualized by fluorescent microscopy using auramine-rhodamine stain for screening, which makes them appear somewhat golden in color. [154]

During an advanced stage of tuberculosis, the organism may infect almost any part of the body, which means that a specimen should appropriately be chosen. An immunochromatographic serological essay for the diagnosis of M. tuberculosis has been developed recently. [154]

14.3 Treatment

Treatment for M. tb is usually administered on an outpatient basis and it consists mainly of medication. Usually, the treatment is given for six to nine months according to a therapy regimen consisting of two months of isoniazid, rifampin, and pyrazinamide, four months of isoniazid and rifampin and ethambutol or streptomycin until the drug sensitivity is known. The drug treatment schema may be changed according to the laboratory results. [154]

Antibiotics are usually part of therapy in people who have no symptoms and whose germs are in inactive state. Antibiotics in this case are helpful in preventing the activation of the infection. The antibiotic used for this purpose is called isoniazid. If taken for six to 12 months, it will prevent the tuberculosis from becoming active in the future. Several side effects have been reported to this drug and some of them can be even life-threatening. One of the side effects caused by this drug is peripheral neuropathy, meaning a decreased sensation in the extremities and which is normally prevented or avoided by administering vitamin B6 at the same time with isoniazid.

[154]

Patients who have active bacteria are usually treated with a combination of medications, at the same time with the main antibiotic, isoniazid. The main drugs used in conjunction with isoniazid are rifampin, ethambutol and pyrazinamide. [154]

Streptomycin, a drug that is given by injection, may be used as well, particularly when the disease is extensive and/or the patients do not take their oral medications reliably. [154]

Usually treatment lasts for few months but it can even be administered for years, in some cases. The rate of success of the treatment is closely related to the patient's compliance and ability to take the drugs as prescribed. [154]

14.4 Challenges

Two of the main challenges facing the field of tuberculosis research are the development of less toxic and/or shorter treatments, as well as the treatment of multi-drug-resistant tuberculosis and extensively drug-resistant tuberculosis. We will mainly address the latter question here, and partially address the former question in the very last chapter.

Multi-drug-resistant tuberculosis (MDR-TB) is defined as TB that is resistant at least to isoniazid (INH) and rifampicin (RMP), the two most powerful first-line anti-TB drugs. Isolates that are multiply-resistant to any other combination of anti-TB drugs but not to INH and RMP are not classed as MDR-TB. [155]

MDR-TB develops during treatment of fully-sensitive TB when the course of antibiotics is interrupted and the levels of drug in the body are insufficient to kill 100% of bacteria. This can happen for a number of reasons. Patients may feel better and halt their antibiotic course, drug supplies may run out or become scarce, or patients may forget to take their medication from time to time. MDR-TB is spread from

person to person as readily as drug-sensitive TB and in the same manner. [155]

The destitute patients suffering from multi-drug-resistant tuberculosis face the problem of not receiving proper treatment. This injustice pertains to the issue of human rights. Treatment and medication for chronic infectious diseases are accessible to those able to afford it, whereas others, like those living in impoverished countries, do not have access to this care. For example, areas such as Africa and Haiti, where there is not a strong foundation for healthcare, treatment is unavailable. As a consequence, only a small minority of affected people are treated. [155]

Extensively drug-resistant tuberculosis (XDR-TB) is a form of TB caused by bacteria that are resistant to the most effective anti-TB drugs. Some contend that XDR-TB strains have emerged from the mismanagement of multidrug-resistant TB (MDR-TB) and once created, can spread from one person to another. The exact nature of this mismanagement is not yet known, but the origin of XDR-TB may coincide with the institution of new policies to promote drug compliance, such as DOTS. [156]

XDR-TB raises concerns of a future TB epidemic with restricted treatment options, and jeopardizes the major gains made in TB control and progress on reducing TB deaths among people living with HIV/AIDS. It is therefore vital that TB control is managed properly and new tools developed to prevent, treat and diagnose the disease. [156]

Chapter 15

Merging Metabolic Networks

Even though the 59 metabolic networks currently available [158] only represent 39 distinct organisms, very little effort has been dedicated to the problem of reconciling different reconstructions for the same organism. Instead, researchers are typically forced to consider each of the available reconstructions individually. In this chapter, we describe an algorithm, called MetaMerge, that allows for an automated or semi-automated merging of two existing networks for the same organism. We begin by motivating the algorithm, then go on to describe every step in detail, and conclude with a short discussion of the combined model as compared to the original models.

15.1 Background

There are a number of factors that make comparison between two different reconstructions challenging. Thus, the two existing metabolic networks published for Mycobacterium tuberculosis in 2007 (Beste et al. [95], Jamshidi and Palsson [159]), both prepared based on extensive literature curation, do not cover the same set of publications related to the metabolism of Mycobacterium tuberculosis. This is evidenced

by the fact that out of the 100 articles referenced by Beste et al. [95] and the 141 referenced by Jamshidi and Palsson [159], only 21 are common. We therefore expect that they also cover different parts of the reactome, even though there is a large amount of overlap. An earlier model by Raman et al. [160] only covers the mycolic acid pathway which is included in one of the genome-scale models [95], and hence will not be considered here. Another challenge could be the idiosyncratic nature of each model's annotation. For instance, in the two networks for Mycobacterium tuberculosis the metabolites are named according to different conventions, the reactions are classified into different pathways, and the combinations of genes necessary for each reaction are formatted differently.

To address this problem, we propose the MetaMerge algorithm, which can reconcile two metabolic networks with minimal supervision from a user, who also does not need to be an expert on the particular organism whose metabolism is being studied. In fact, this supervision can be completely eliminated, though results are typically more reliable when such supervision is present. The algorithm produces a network which conserves the functionality of both the original networks, while providing additional functionality whenever the two networks complement one another.

We apply the MetaMerge algorithm to the two existing networks for Mycobacterium tuberculosis in 2007 ([95, 159]). Although each network individually covers a large fraction of the metabolic reactions available to Mycobacterium tuberculosis, we find here that the two networks can be combined in a synergistic fashion, giving rise to new insights about the organism and opening the door to further model refinement. In particular, our results show that the combined model has more than 60% fewer blocked reactions (i.e. reactions that cannot have any flux through them at steady state) than the two individual models and predicts essential genes with a higher positive predictive value than the individual models. We further make use of

the model to generate a short list of metabolic drug target candidates which could be used as a starting point for the discovery of new drugs. These last two features of the combined model are discussed in the following chapters.

15.2 The MetaMerge Algorithm

We describe MetaMerge, a semi-automatic algorithm for merging two metabolic networks. It can be divided into six steps, each of which is explained briefly below and in detail in the Methods section. The entire algorithm is represented in Figure 15-1.

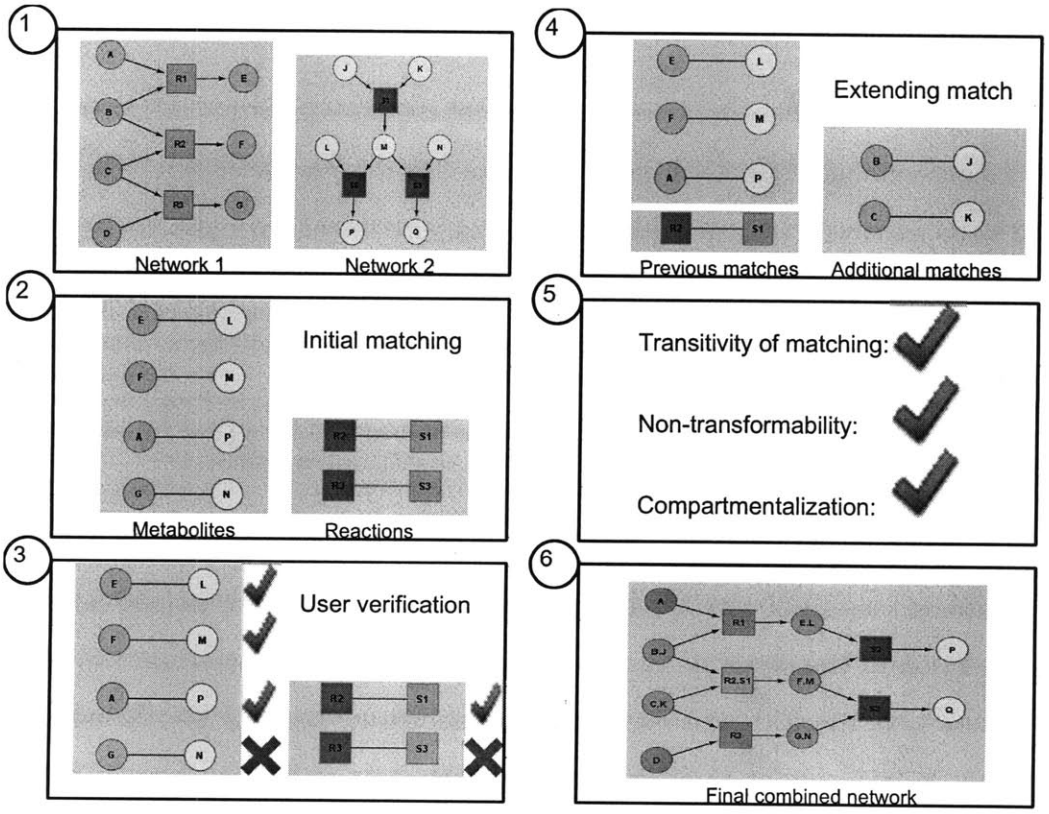

Figure 15-1: The algorithm for merging two metabolic networks.

In step 1 of the algorithm, the models are parsed and each metabolite and reaction is augmented with several features. These features include the full names

and molecular formulas for the metabolites and enzyme names and pathway names for the reactions. In more richly annotated models it is possible to have additional features, such as standardized identifiers for the metabolites and enzymes. In less richly annotated models, these additional features can be automatically extracted by querying online databases. The advantage of having standardized identifiers for as many entities (metabolites or reactions) as possible is that, while an entity may have different names, they typically all correspond to the same standardized identifier. For instance, sodium bicarbonate, bicarbonate of soda, baking soda and sodium hydrogencarbonate all have the same CAS number, 144-55-8 [170].

During the merging process (steps 2 to 5), entities in the two models (metabolites and reactions) are identified with one another based on the number of features they have in common; the more features they share, the more likely they are to be chosen as a match. Matches are not necessarily one-to-one; in some cases, where one metabolite in one model is subdivided into several metabolites in another model, one-to-many matches can also occur. This happens, for instance, in the case of trehalose dimycolate, which is denoted TREHALOSEDIMYCOLATE in Beste et al.'s model [95], but is subdivided into tdm1 through tdm4 in Jamshidi and Palsson's model [159] depending on the type of mycolates that get attached to the trehalose.

Because not all matches are one-to-one, a key concept in our algorithm is that of a *matching matrix*, to which we are able to apply graph-theoretic algorithms. The matching matrix is a binary matrix with a row for each entity from one model and a column for each entity from the other model, where an entity refers to either a metabolite or a reaction. The matrix contains a 1 in those entries which correspond to a match, and a 0 elsewhere. The matrix format ensures that the merging works symmetrically for the two models, and that the result would not change if the two models were to be considered in the opposite order.

The matching matrices (one for the metabolites and one for the reactions) are initialized in step 2, possibly cleaned up in step 3, and then expanded in step 4. Steps 3 and 4 are then repeated until no further expansion can be proposed, or until every entity in one of the models has been matched (the latter case being unlikely due to the incomplete overlap between any pair of models).

Postprocessing occurs in step 5, in which two conditions are checked. The first condition ensures that the metabolites of the two models can be divided into non-overlapping classes, with no matches occurring between classes and all possible matches occurring inside a class. The second condition ensures that no two metabolites in the same class appear on different sides of a reaction. If the models are compartmentalized (divided into compartments), then an additional conditions needs to be checked that ensures that metabolites in a certain compartment in one of the models are matched only to metabolites in the corresponding compartment in the other model. Violations of these conditions, though rare, have to be repaired manually.

Step 6, the final step of the MetaMerge algorithm, is the preparation of the combined model. To create the combined model from the matching matrix for metabolites (the matching matrix for reactions only serves as a guide in the algorithm), we take the approach of ensuring the combined model preserves the full functionality of each individual model while allowing for new, synergistic functionality not present in the individual models. Because the most common way of analyzing metabolic models is the constraint-based formalism [108], we achieve this as follows. We create a single new metabolite in the new model to represent each class of metabolites from the individual models, according to the matching matrix. We then rewrite each reaction using the new metabolites, and combine identical reactions into a single reaction, treating the enzymes or enzyme combinations necessary to catalyze them as isozymes.

This particular method of combining the models means that the mass-balance

constraints on each metabolite in the combined model are no more stringent than they were in the initial models, and possibly less stringent. This is because identifying two metabolites is equivalent to adding the corresponding constraints, and any flux vector satisfying both constraints satisfies their sum.

15.2.1 Stage 1: Model parsing and feature preparation

Because the models [95, 159] were presented as Excel spreadsheets, we wrote our own script in Python [42] to parse them into an internal computer-readable format. However, we subsequently converted them into files written in version 2 level 4 of SBML (Systems Biology Markup Language) [161], the current standard for metabolic networks. These models, as well as the merged model, have also been uploaded to the BioModels database [165].

The following features were used to compare metabolites: abbreviation, official name, CAS number [170], IUPAC name [171], BioCyc identifier [172], KEGG identifier [157], molecular formula, in-degree in the network, out-degree in the network. We extracted the CAS numbers and IUPAC names semi-automatically because no freely accessible website had a URL structure that could be built explicitly based on the metabolite name. The BioCyc and KEGG identifiers were used to retrieve the CAS number and IUPAC name for metabolites for which they had not been found using the semi-automatic web search. The first six features were each given a weight of 10, while the last three features were each given a weight of 1 to account for non-uniqueness. To match official names, we used fuzzy string matching with the default cutoff of 0.6, implemented in Python [42] as the *get_close_matches* method in the *difflib* module. We considered only perfect matches for the remaining features.

The following features were used to compare reactions: reaction name, pathway name, gene name, protein name, enzyme name, irreversibility, number of reactants,

number of products. Unlike for metabolites, all of the reaction features were available directly from the complete models. However, to retrieve enzyme names from their EC numbers [163] we used an automatic search on the ExPASy Proteomics Server [173], implemented with the *urlopen* method in the *urllib* module in Python [42].

15.2.2 Stage 2: Initial matching of reactions and metabolites

We created two score matrices, M and N , using the features described immediately above. The first five features were each given a score of 10, while the last three features were each given a score of 1 to account for non-uniqueness. The various names were considered to be a match if they contained at least one non-function word in common, and we only considered perfect matches for the remaining features. M_{ij} (respectively N_{ij}) is the total score of the matching features for the pair of metabolites (respectively reactions) i and j.

We also created two binary matching matrices, M^B and N^B , initialized to contain a 0 in every position. The entry M^B_{ij} is 0 if the two metabolites have not been matched and 1 if they have been matched. If the algorithm is being used interactively (with input from the user), these matrices become ternary: an entry of 1 means that the match has been accepted by the user, and -1, rejected, while an entry of 0 means that the match has never been proposed. The matrices M^B and N^B are then used during the iterative stage to keep track of matching decisions in the previous iterations.

To initialize a matching of reactions and metabolites, we used the pairs of reactions (i, j) with a total score N_{ij} above a cutoff of 30. Subsequently the highest-scoring pairing of the metabolites was computed for each of the reaction pairs in the matching using a greedy algorithm. This algorithm identifies the metabolites k in reaction i and l in reaction j with the largest total score M_{kl}, matches these to one another, and then repeats the process with the remaining metabolites until all the metabolites

in one of the reactions have been matched or all remaining pairs of metabolites have a total score of 0.

15.2.3 Stage 3: Optional interactions with the user

The current set of newly matched reactions can be presented to the user for confirmation. The confirmation proceeds in two steps. First, the user confirms that the reactions should be matched to each other. Second, the user checks the matching between the metabolites in these two reactions.

We implemented several features in the user interface. The first set of features are inspection options, by which the user could see not just the score for a given pair of reactions or metabolites, but also their complete sets of features side by side, as well as any decisions about the pair that had been made previously. The second set of features are browsing options, which allow the user to go forward and backward in the list of matched reactions. The third set of features are confirmation options, which allow the user to accept or reject a proposed match of reactions or metabolites, as well as to enter their own match in either text format or using the numbers displayed next to each of the metabolites.

To speed up the confirmation process, the user also has the option of accepting all the proposed matches between the metabolites of two reactions. A metabolite could also be matched to "nothing" if it was present in one reaction, but not in the other. In addition, when two internal metabolites are matched, the user is asked whether the corresponding external metabolites (if both exist) should be matched.

Although these interactions could in principle be bypassed entirely, we found that they were helpful in steering the algorithm away from possible mistakes in the matching process. For instance, several close matches with similar scores were sometimes available for a given reaction, but only one of them was the correct match, and this

could only be detected by using the browsing feature of the interface. Furthermore, the possibility of seeing previous decisions on putative metabolite matches helps to maintain overall consistency in the matching, which prevents violations of transitivity (as discussed below).

15.2.4 Stage 4: Extension of the current matching

During the matching of metabolites in matched reactions, new pairs of matching metabolites are usually found. If that is the case, the extension algorithm finds all the pairs of reactions that have not yet been matched, and whose metabolites could be matched almost perfectly (i.e. all but 1 metabolite in each reaction can either be matched to one another or to nothing, based on previous confirmations). Convergence occurs when no further extension of the current matching is possible.

15.2.5 Stage 5: Transitivity and non-transformability

In the postprocessing phase of the algorithm, the matching of the metabolites is cleaned up. This phase is currently performed in a semi-automatic way, although it might be possible to automate it completely. First, transitivity of the metabolite matching needs to be ensured. A matching matrix is said to be *transitive* if it can be perfectly covered by rectangles. In graph-theoretic terms, this is equivalent to saying that the bipartite graph formed by the matching is a disjoint union of *bicliques* [174]. The transitivity is easily checked by greedily covering the matching matrix with rectangles and seeing if any of them overlap. If so, the overlapping pairs of rectangles are presented to the user to decide how to remedy the problem.

Second, non-transformability of the metabolite matching needs to be ensured as well. A transitive matching matrix is said to be *non-transformable* if there is no rectangle in its covering that contains two metabolites from the same network that

participate on different sides of a reaction. This will lead to the undesirable behavior of metabolites canceling out on either side of a reaction in the merged network.

A third condition which is not strictly necessary, but which makes the merging process a lot easier and cleaner, is *compartmentalization.* In the context of a model with only two compartments (the cell and the extracellular space) this simply means that no internal metabolite in one network is matched to an external metabolite in the other network. In the case of more complex models, such as the 8-compartment yeast model [83], this would mean that any pair of matched metabolites must belong to the same compartment.

15.2.6 Stage 6: Creation of the merged network

In order to create the merged network, we combined each group of metabolites corresponding to a rectangle in the matching matrix (equivalently, a biclique in the matching graph) into a single new metabolite. Optionally, we allow for some of the metabolites that were frequently matched to "nothing" to be deleted from the merged network. In the case of the two Mycobacterium tuberculosis networks considered in this work, we only deleted protons (H) and water molecules (H_2O). Each reaction is then rewritten using the new metabolites. This results in many identical reactions, which are considered to be isozymes of one another. The merged network contains only one reaction from each group of isozymes. To determine the reversibility of a reaction in the merged network, we examine the reactions from the original networks that are represented by it. If at least one of them is reversible or if two of them are oppositely directed, the new reaction is taken to be reversible; otherwise, it is taken to be irreversible.

15.3 Comparing Combined and Original Models

Figure 15-2 shows the combined model for Mycobacterium tuberculosis. The circular nodes represent metabolites, and the polygonal nodes represent reactions. An entity (metabolite or reaction) that is only present in model 1 [95] is represented in blue, one that is only present in model 2 [159] is represented in yellow, and one that is present in both is represented in green.

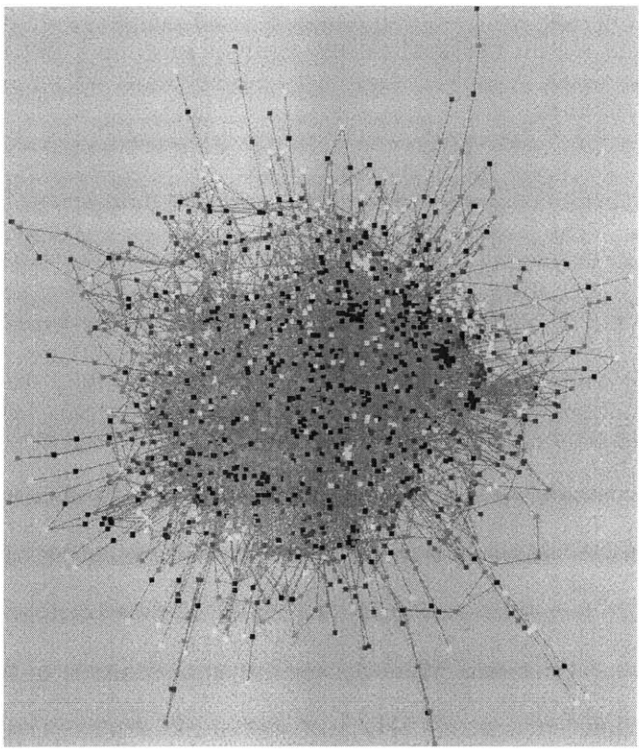

Figure 15-2: The merged network for Mycobacterium tuberculosis.

The statistics for the two initial models and the combined model are given in Table 15.1. It shows that the overlap between the two models is about 70% in the genes and in the metabolites, but only about 45% in the reactions. This is due to the fact that we only considered two reactions to be identical if they agreed completely in their metabolites according to the new classification, except possibly for the currency

metabolites H (protons) and H_2O (water), as well as in their coefficients. While there are a number of pairs of very similar reactions in the combined model, we did not make the choice between one or the other for the following reasons. First, given the different literature sources used in preparing the two reconstructions, it requires special expertise to decide which of the two reactions should be used as the correct one. Second, keeping both reactions was necessary in order to ensure that the functionality of both initial models was preserved in the combined model. Finally, models such as this one should be treated as repositories of knowledge, and may represent the same reaction in conflicting ways. Such conflicts can and should be resolved by a consensus of experts, as was recently done for yeast [166].

Table 15.1: Statistics for the three models

Number of	Model 1 [95]	Model 2 [159]	Model 3 (combined)
Genes	743	661	917
Reactions	873	937	1400
Cytosolic	741	837	1207
Exchange	132	100	193
Reversible	187	254	180
Irreversible	686	683	1020
Metabolites	753	825	1017
Internal	652	740	880
External	101	85	137

Chapter 16

Gene Essentiality Prediction

This chapter looks at the gene essentiality prediction based on the combined model for Mycobacterium tuberculosis. We start by describing the in-vitro experiment that was performed in 2003 to determine the genes essential for M. tb. We go on to describe the computational experiments that we performed, and conclude with a discussion of the results.

16.1 Transposon Site Hybridization Experiment

To construct our gold standard for gene essentiality, we used results of the transposon site hybridization (TraSH) experiment performed by Sassetti et al. [169] and followed the methodology of Beste et al. [95]. Thus, we used both the threshold 0.2 as in the original experiment, as well as the more stringent threshold 0.1, for deciding the essentiality of a gene.

The medium used in the TraSH experiment, in addition to the Middlebrook medium described below, contained OADC, an enrichment composed of oleic acid, bovine albumin, dextrose and catalase. Out of these additional components, only

dextrose (under the name of D-glucose) and catalase (as an enzyme) were present in the original metabolic models. For this reason, the OADC enrichment could not be reproduced in our computational experiments, and may account for part of the discrepancy between the computational and the experimental essentiality predictions.

16.2 Computational Essentiality Experiment

We compared the prediction of our combined model to the results of the transposon site hybridization (TraSH) experiment performed by Sassetti et al. [169], widely considered to be the gold standard for determining in vitro essentiality of genes in Mycobacterium tuberculosis. We simulated the in vitro growth of Mycobacterium tuberculosis on the Middlebrook medium 7H10, which was also used in the TraSH experiment [169]. In order to do so, following the papers describing the original models, we enabled all the reactions corresponding to the import of elements of the Middlebrook medium: ammonium, biotin, calcium, chloride, citrate, ferric iron, L-glutamic acid, phosphate, potassium, sodium, and sulfate. Additionally, the following nutrients were added for growth in model 1 [95]: carbon dioxide, glucose, molybdenum, nitrogen dioxide, and oxygen. For model 2 [159], the added nutrients were: carbonic acid, copper, glucose, magnesium and oxygen.

The in-vitro biomass composition was used for model 1, and the reduced biomass composition (both with and without the cofactors) was used for model 2. Unfortunately, model 2 was not able to produce any biomass on the medium described here, whether cofactors were included or not. For this reason, we only used the biomass composition for model 1 in all our experiments and applied it to the combined model as well.

To classify a gene as essential or non-essential in the experiment, one must establish a threshold ratio of the microarray hybridization signal obtained from labeled

insertion sites in a saturated transposon mutant library compared with a control of labeled genomic DNA. Our results differ somewhat from those reported by Beste et al. [95], because our analysis was performed using the methodology described in chapter 11. For each reaction, the genes required to catalyze it were assembled in conjunctive normal form (OR of ANDs). From this information, a list of all the reactions disabled by the knockout of each gene was compiled. Subsequently, these reactions were constrained to have a flux of 0 and the gene was determined to be essential if the resulting metabolic network was not able to exhibit growth, and non-essential otherwise.

16.3 Results and Discussion

There are fewer genes predicted to be essential for the combined model than for model 1 (89 and 203, respectively). This is to be expected since the combined model contains the full functionality of the initial models plus additional functionality. Furthermore, the combined model gives a higher positive predictive value (63 true positives out of 89, or 71%, vs. 139 true positives out of 203, or 68%, with a threshold of 0.2, and 49 true positives, or 55%, vs. 106 true positives, or 52%, with a threshold of 0.1). The negative predictive value, on the other hand, is somewhat lower (436 true negatives out of 637, or 68%, vs. 285 true negatives out of 381, or 75%, with a threshold of 0.2, and 508 true negatives, or 80%, vs. 325 true negatives, or 85%, with a threshold of 0.1). It is important to note here that many of the genes involved in metabolism may also play a non-metabolic role, and it is this role that may make them essential. On the other hand, genes that are essential to the metabolism of the mycobacterium will be essential regardless of their other roles. Therefore it is more important for a metabolic model to predict the essentiality of a gene with high positive predictive value than with high negative predictive value, and therefore the combined model is

to be preferred over model 1 if the only criterion for choosing a model is its ability to predict gene essentiality. The results of the essentiality experiments are summarized in Table 16.1. The p-value given by the Fisher exact test was less than 10^{-10} in all cases.

Table 16.1: Gene essentiality in models 1 and 3

Predictions in	Model 1 [95]		Model 3	
Threshold ratio	0.1	0.2	0.1	0.2
True positive	106	139	49	63
False positive	97	64	40	26
True negative	325	285	508	436
False negative	56	96	129	201
Positive predictive value	52%	68%	55%	**71%**
Negative predictive value	**85%**	75%	80%	68%
Correct predictions	74%	73%	**77%**	69%

As the results of this computational experiment show, however, the original models as well as the combined model still have a long way to go before they can capture the full range of gene essentiality information. In order to improve the models' predictive power, new knowledge about metabolism of the mycobacterium will need to be incorporated and further refinement will be necessary both in light of new experimental data.

Chapter 17

List of Putative Drug Targets

In this final chapter, we describe our methodology for identifying putative drug target candidates using the combined metabolic model for Mycobacterium tuberculosis. We begin with a detailed description of the method, compare it to previous work, and finally describe the results.

17.1 Methodology

In order to identify putative drug target candidates, we leverage the knowledge of the drug targets for antitubercular drugs currently in use as well as the structure of the combined metabolic network. Specifically, we focus on the targets for two first-line antitubercular drugs: ethambutol (ETH) and isoniazid (INH) [167]. We do not consider the three other first-line antitubercular drugs, pyrazinamide (PZA), rifampicin (RMP) and streptomycin (STM). This is because the targets of pyrazinamide are still the subject of a debate [168], while the targets of the other two drugs, although known, are not part of the combined metabolic network because they do not appear in either of the original models.

As described in chapter 10, all the reactions in a metabolic network can be divided into reaction subsets, such that all reactions in each subset always have pairwise proportional fluxes at steady-state; in particular, if one of them is blocked and has flux 0, all the other ones do as well. We look for targets whose inhibition would result in effects similar to those of each of the three drugs. In order to do so, we find the reactions in the combined network that are disabled by the inhibition of the drug target, and then find all the reactions in a subset with each of those. We propose those genes that are essential to at least one of the reaction in those subsets as drug target candidates.

17.2 Comparison with Previous Work

We note that this we use a similar, although somewhat less stringent, criterion to that used by Jamshidi and Palsson [159], since their hard-coupled reactions (pairs of reactions uniquely producing and uniquely consuming a given metabolite) always belong to the same reaction subset, while reaction subsets cannot always be obtained from hard-coupled reactions. In fact, it is possible to construct an example of a metabolic network where two reactions in an enzyme subset of size 2 do not share any metabolites.

17.3 Results

Our method leads to 33 target candidates whose inhibition would result in effects similar to those of ethambutol, and 3 target candidates, to those of isoniazid. Interestingly, the 3 target candidates based on the effects of isoniazid (Rv1412, Rv1416 and Rv1500) are also target candidates based on the effects of ethambutol. The 33 putative target candidates, classified into 4 categories according to their pathway

or subsystem, appear in Table 17.1. The highlighted genes are known targets for ethambutol.

Table 17.1: Putative drug target candidates by pathway

Membrane metabolism	Rv1662	Rv1663	Rv2051c	Rv2378c	Rv2379c
	Rv2380c	Rv2381c	Rv2382c	Rv2383c	Rv2384
	Rv2611c	Rv2940c	**Rv3793**	**Rv3794**	**Rv3795**
Cofactor metabolism	Rv0542c	Rv0548c	Rv0553	Rv1412	Rv1416
	Rv1940	Rv2217	Rv2218	Rv2678c	Rv2786c
Fatty acid metabolism	Rv0129c	Rv1886c	Rv2524c	Rv3804c	
Miscellaneous pathways	Rv0535	Rv1500	Rv2220	Rv2523c	

Part V

Appendix

Appendix A

Proofs of All the Lemmas

We begin by restating the relevant equations from the thesis.

$$Sv = 0 \text{ and } v_i \geq 0 \; \forall \; i \in \mathcal{I}. \tag{A.1}$$

$$S_{\text{new}} = S_{i_1} + \sum_{j=2}^{t} r_j S_{i_j}. \tag{A.2}$$

$$w \in \text{Row}(S_{\mathcal{C}}), \; v_i w_i \leq 0 \; \forall \; i \in \mathcal{C} \text{ and } v_i = 0 \implies w_i = 0 \; \forall \; i \in \mathcal{C}. \tag{A.3}$$

$$v_{\text{target}} \geq \lambda v_{\text{biomass}}. \tag{A.4}$$

Lemma 2 (Cut Sets in Fully Reversible Networks). *Let S be the stoichiometric matrix of a fully reversible network. Then X is a cut set for reaction i if and only if there exists a y such that $R(y^T S) \subseteq X \cup \{i\}$, with $y^T S$ having a positive entry in position i. X is a minimal cut set for reaction i if and only if $X \cup \{i\}$ is a minimal support of vectors with positive i-th component in the rowspace of S.*

Proof. Let us denote by S_{-X} the matrix obtained from S by removing the reactions

corresponding to X. Then X is a cut set for reaction i if and only if

$$S_{-X}v = \mathbf{0} \implies v_i = 0.$$

By Farkas' lemma, this means that $v_i = 0$ is a linear combination of the equalities in the system $S_{-X}v = \mathbf{0}$. In other words, there exists a vector y such that

$$y^T S_{-X} = e_i,$$

where e_i denotes the vector with 1 in the i-th component and 0 elsewhere. It follows that

$$y^T S = u,$$

where u has a 1 in the i-th component, some arbitrary entries in the components corresponding to the reactions in X and 0 everywhere else. Note that the magnitude of the i-th component of u can be changed by multiplying y by a positive scalar. Furthermore, if u has $u_j = 0$ for some $j \in X$, then $X - \{j\}$ is still a cut set for reaction i, so X is not minimal. This proves the lemma. □

Lemma 3 (Cut Sets in Fully Irreversible Networks). *Let S be the stoichiometric matrix of a fully irreversible network. Then X is a cut set for reaction i if and only if there exists a y such that $R_-(y^T S) \subseteq X$, with $y^T S$ having a positive entry in position i. X is a minimal cut set for reaction i if and only if X is a minimal negative support of vectors with positive i-th component in the rowspace of S.*

Proof. Let X be a cut set for reaction i. The condition for reaction i being blocked now reads

$$S_{-X}v = \mathbf{0} \text{ and } v_j \geq 0 \; \forall \; j \in (X \cup \{i\})^C \implies v_i = 0.$$

Since $v_i \geq 0$ is one of our constraints, we only need to be able to derive the inequality

$v_i \leq 0$. Applying Farkas' lemma to this situation then yields the existence of a vector y such that

$$y^T S_{-X} = e_i + \sum_{j \in (X \cup \{i\})^C} \lambda_j e_j,\ \lambda_j \geq 0 \ \forall\, j \in (X \cup \{i\})^C.$$

It follows that

$$y^T S = u,$$

where u has a 1 in the i-th component, some arbitrary entries in the components corresponding to the reactions in X, and nonnegative entries in the components corresponding to $(X \cup \{i\})^C$. In this case, we see that if u has $u_j \geq 0$ for some $j \in X$, then $X - \{j\}$ is still a cut set for reaction i. It follows that, for a minimal cut set, the components corresponding to X need to be strictly negative. This proves the lemma. □

Lemma 4 (Correctness of Sequential Deletions). ***Algorithm4*** *correctly identifies and removes all strictly detailed balanced reactions from a stoichiometric matrix S.*

Proof. Since **Algorithm1** removes topologically blocked reactions which are a fortiori stoichiometrically blocked, the output of **Algorithm4** is the same with or without it. Since any stoichiometrically blocked reaction is a fortiori thermodynamically blocked, it is sufficient to prove that there are no thermodynamically blocked reactions in this output. We will establish this by proving the following two statements:

1. **Algorithm3**($K_{\mathcal{I}}$) correctly identifies and removes all the thermodynamically blocked irreversible reactions from S.

2. After all the thermodynamically blocked irreversible reactions are removed from S, all remaining thermodynamically blocked reactions are stoichiometrically blocked.

For the first statement, we note that finding a (maximal) non-negative vector in the nullspace of $K_{\mathcal{I}}$ is equivalent to finding a (maximal) non-negative vector in the nullspace of K with the components corresponding to $\mathcal{I}^C$ set to 0. This, in turn, is equivalent to finding a (maximal) non-negative vector in the rowspace of S with support in $\mathcal{I}$. By Farkas' lemma, this is equivalent to finding all the thermodynamically blocked reactions in S contained in $\mathcal{I}$.

For the second statement, suppose that $i \notin \mathcal{I}$ is a thermodynamically blocked reaction in S. Then $Sv = \mathbf{0}$ and $v_j \geq 0 \; \forall \; j \in I$ imply both $v_i \leq 0$ and $v_i \geq 0$. By Farkas' lemma, this implies that

$$\exists y_- \text{ such that } y_-^T S = e_i + \sum_{j \in \mathcal{I}} \lambda_j e_j, \qquad \lambda_j \geq 0 \; \forall \; j \in \mathcal{I}$$

$$\exists y_+ \text{ such that } y_+^T S = -e_i + \sum_{j \in \mathcal{I}} \mu_j e_j, \qquad \mu_j \geq 0 \; \forall \; j \in \mathcal{I}$$

By adding these together, we conclude that

$$(y_- + y_+)^T S = \sum_{j \in \mathcal{I}} (\lambda_j + \mu_j) e_j.$$

Since we assumed that no irreversible reactions are thermodynamically blocked in S, all the coefficients must in fact be 0, so $\lambda_j = 0 = \mu_j \; \forall \; j \in I$. But then $y_-^T S = e_i$ implies that reaction i is in fact stoichiometrically blocked in S, completing the proof. □

Lemma 5 (Enzyme Subsets). *In a consistent stoichiometric matrix S, i and j are in an enzyme subset with ratio κ if and only if $e_i - \kappa e_j \in Row(S)$.*

Proof. Suppose that $e_i - \kappa e_j \in \text{Row}(S)$. Since every flux mode is orthogonal to every vector in the rowspace of S, it follows that every flux mode v satisfies $v_i - \kappa v_j = 0$.

Conversely, suppose that i and j are in an enzyme subset with ratio κ. Then the equality $v_i - \kappa v_j = 0$ holds for S. This means that the two inequalities, $v_i - \kappa v_j \leq 0$ and $\kappa v_j - v_i \leq 0$ both hold for S. By Farkas' lemma, this implies that there exist scalars $\lambda_k, \mu_k \geq 0$ for $k \in \mathcal{I} - \{i, j\}$ such that

$$x := e_i - \kappa e_j + \sum_{k \in \mathcal{I} - \{i,j\}} \lambda_k e_k \in \text{Row}(S) \text{ and } y := \kappa e_j - e_i + \sum_{k \in \mathcal{I} - \{i,j\}} \mu_k e_k \in \text{Row}(S)$$

Let us consider $z := x + y$. If $z \neq \mathbf{0}$, then we have $\sum_{k \in \mathcal{I} - \{i,j\}} (\lambda_k + \mu_k) e_k \in \text{Row}(S)$, which by lemma 3 means that some of the irreversible reactions are thermodynamically blocked, contradicting the consistency of S. Hence, $z = \mathbf{0}$, so $\lambda_k = \mu_k = 0$ for each k and $x = e_i - \kappa e_j \in \text{Row}(S)$, completing the proof. □

Lemma 6 (Equivalence after Lumping of Enzyme Subsets). *Let $\tilde{S}$ be the matrix obtained from S by lumping its enzyme subsets. Then S and $\tilde{S}$ are equivalent.*

Proof. We lump the enzyme subset together into a single new reaction given by equation (A.2), where the linear combination of the reactions is understood as the reaction with the corresponding linear combination of the reagents and products. Let us renumber the reactions in S in such a way that $i_j = j$ for $1 \leq j \leq t$, so that the first t reactions form the enzyme subset. Let $\tilde{S} = [S_{\text{new}}\ S_{>t}]$, where $S_{>t}$ is formed from S by removing the first t columns. We claim that $\tilde{S}$ is equivalent to S.

Indeed, suppose that v is a flux vector of S. We know that $v_j = r_j v_1$ for $2 \leq j \leq t$. Let $\tilde{v} := [v_1\ v_{>t}]$. Then $\tilde{v}$ is a flux vector of $\tilde{S}$ because

$$\tilde{S}\tilde{v} = v_1 S_{\text{new}} + S_{>t} v_{>t} = v_1 (S_1 + \sum_{j=2}^{t} r_j S_j) + S_{>t} v_{>t} =$$

$$= v_1 S_1 + \sum_{j=2}^{t} (v_1 r_j) S_j + S_{>t} v_{>t} = v_1 S_1 + \sum_{j=2}^{t} v_j S_j + S_{>t} v_{>t} = Sv.$$

Conversely, any flux vector $\tilde{v}$ of $\tilde{S}$ can be expanded into a flux vector v of S by setting $v_1 = \tilde{v}_1$ and $v_j = \tilde{v}_1 r_j$ for $2 \leq j \leq t$. That v is indeed a flux vector of S is proven by the same string of equalities as above, but in the reverse order. □

Lemma 7 (Elemental Balance and the No Free Lunch Condition). *Let S be a stoichiometric matrix of a metabolic network (with the external metabolites included). Then S is elementally balanced if and only if no linear combination of the reactions in S can result in the production of one or more metabolites out of no reagents.*

Proof. We begin with a simpler statement, which is a direct consequence of Farkas' lemma. Let $1 \leq i \leq n$ be fixed, where n is the number of columns (reactions) in S. Then either there exists $w \geq \mathbf{0}$ such that $Sv = w$ for some v and $w_i = 1$, or there exists $y \geq \mathbf{0}$ such that $y^T S = \mathbf{0}$ and $y_i = 1$, but not both.

Indeed, let L be the left nullspace matrix of S. If a w satisfying the first condition does not exist, then $Lw = \mathbf{0}$ (equivalent to $Sv = w$) and $w \geq \mathbf{0}$ implies $w_i = 0$. By lemma 3 with S replaced by L and $X = \emptyset$, this means that there exists a vector $y \geq \mathbf{0}$ with $y_i = 1$ in the rowspace of K. However, the rowspace of K is precisely the left nullspace of S, so that $y^T S = \mathbf{0}$. The other direction is proven analogously.

Now we are ready to prove the more general statement. Suppose that the matrix S is elementally balanced. Then there exists a positive vector y with $y^T S = \mathbf{0}$. By the previous statement, $Sv = w$ and $w \geq \mathbf{0}$ implies $w_i = 0$ for each i, so the "no free lunch" condition holds. Conversely, suppose the "no free lunch" condition holds. Then $Sv = w$ and $w \geq \mathbf{0}$ implies $w = \mathbf{0}$, so that there exists $y \geq \mathbf{0}$ such that $y^T S = \mathbf{0}$ and $y_i = 1$ for each i. Adding these vectors over all possible i gives a positive vector y such that $y^T S = \mathbf{0}$, so the matrix S is elementally balanced. □

Lemma 8 (Number of Non-Redundant Constraints). *Let S be an $m \times n$ stoichiometric matrix with rank r, and let K be its nullspace matrix. Suppose that **Algorithm4** eliminates $\mathcal{D} \subseteq \mathcal{I}$ as thermodynamically blocked reactions, and let $r_0 = rank(K_{\mathcal{D}})$.*

Suppose that ***Algorithm4, Algorithm5, Algorithm6*** *eliminate a total of T reactions. Then the final S contains $r - T + r_0$ non-redundant constraints.*

Proof. Suppose that n_1 constraints are eliminated by **Algorithm1**, n_2 constraints are eliminated by **Algorithm2**, $n_3 := |\mathcal{D}|$ and n_4 reactions are lumped into r_4 enzyme subsets by **Algorithm6**. Then $T = n_1 + n_2 + n_3 + n_4 - r_4$, and the statement is equivalent to saying that S will contain $(m - r) + n_1 + n_2 + (n_3 - r_0) + (n_4 - r_4)$ redundant constraints, where $r_0 = \text{rank}(K_{\mathcal{D}})$.

First of all, note that the redundant constraints can be removed all at once as the last step of the reduction, or they can be removed after each step of the reduction to keep the intermediate matrix of full row rank. Clearly, this does not affect the rank of the final matrix, so the number of redundant constraints can be added to the total after each of the steps.

The initial matrix has m constraints and rank r, so $m - r$ constraints are redundant. This accounts for the first term $m - r$ in the sum, and the resulting matrix is of full row rank.

Reaction i is removed by **Algorithm1** or **Algorithm2** as strictly detailed balanced if and only if e_i is in the rowspace of the matrix. Let us choose a basis of the rowspace which contains all the vectors of type e_i that are in the rowspace (this can be achieved by a Gauss-Jordan elimination of the stoichiometric matrix). When reaction i is deleted, the corresponding row becomes 0, so it corresponds to a redundant constraint and can be deleted as well. Since the matrix from the previous step had full row rank, these are the only rows that become redundant. This accounts for the terms n_1 and n_2.

Similarly, consider the subset $\mathcal{D}$ of irreversible reactions that are eliminated by **Algorithm3**. We know that there is a vector $v > 0$ such that $K_{\mathcal{D}}v = 0$. Let B be a basis of the (nonzero) nullspace of $K_{\mathcal{D}}$. The size of B is precisely $n_3 - r_0$. By

extending each element of B with 0 in positions not corresponding to elements of $\mathcal{D}$, we get an independent set $\tilde{B}$ in the nullspace of K. It can then be extended to a basis of the nullspace of K, which is also the rowspace of S. By the same argument as above, we see that each of those extended vectors leads to a row of zeros after $\mathcal{D}$ is deleted, which corresponds to a redundant constraint. Since the matrix from the previous step had full row rank, these are the only rows that become redundant. This accounts for the term $n_3 - r_0$ in the sum.

Finally, consider the enzyme subsets. For an enzyme subset of size t containing reaction i, there are $t-1$ linearly independent vectors of the type $e_i - \kappa e_j$ in the rowspace of S (one for each $j \neq i$ in the enzyme subset). Furthermore, since enzyme subsets are disjoint, the vectors corresponding to different ones are linearly independent. To complete the proof, it therefore suffices to show that each such vector corresponds to one redundant constraint after the enzyme subsets are combined into a single lumped reaction.

For simplicity, let us renumber the reactions in such a way that reactions 1 and 2 are in the same enzyme subset, so that $Sv = 0$ implies $v_2 = \kappa v_1$ for some $\kappa \neq 0$. We can partition S as $[S_1\ S_2\ S_{>2}]$, and the nullspace matrix K as $[K_1\ K_2\ K_{>2}]$, where $S_{>2}$ denotes the submatrix of S formed by removing the first 2 columns. By lemma 5, $K_1 = \kappa K_2$. Furthermore, S and K have full row rank. Lumping 1 and 2 transforms S into $\tilde{S} := [S_1 + \kappa S_2\ S_{3:N}]$. We claim that the corresponding nullspace matrix becomes $\tilde{K} := [K_1\ K_{3:N}]$. This is easy to check by direct multiplication. However, $\tilde{K}$ still has full row rank because K did and we deleted a multiple of K's first row. Since the ranks of $\tilde{S}$ and $\tilde{K}$ add up to $N-1$, the rank of $\tilde{S}$ has indeed decreased by 1 from that of S. □

Lemma 9 (Canonical Form after a Single Reduction Cycle). *Let S be the stoichiometric matrix of a metabolic network after one reduction cycle. Then S is in canonical*

form.

Proof. By lemma 4, all the reactions remaining after the application of **Algorithm4** can have a non-zero flux at steady-state. **Algorithm5** further transforms the matrix in such a way that every reaction can have a positive flux at steady-state. In other words, if the first two steps of the reduction cycle transform S into S', then $S'u = \mathbf{0}$ and $u_i = 1$ is feasible for each i.

The third step of the reduction cycle, **Algorithm6**, transforms S' into S'', where S'' contains a single combined reaction for each of the enzyme subsets in S'. By the proof of lemma 6 and the conclusion of the previous paragraph, we know that $S''u = \mathbf{0}$ and $u_i = 1$ is feasible for any combined reaction i, as well as for any reaction originally in S'. Furthermore, S'' does not admit any further enzyme subsets, which, by lemma 5, would correspond to vectors of the form $e_i - \kappa e_j$ with $\kappa \neq 0$ (i.e. with a support of size exactly 2) in its rowspace. This is clearly the case if both i and j were originally present in S'. If i is a combined reaction and j is an original reaction from S', then the equivalence between S' and S'' from lemma 6 shows that j must have been part of the same enzyme subset as one of the reactions in the linear combination that gave rise to reaction i. Finally, if both i and j are combined reactions, then the equivalence between S' and S'' shows that one of the reactions in each of the subsets represented by i and j, respectively were part of the same enzyme subset, and so, by transitivity, the two enzyme subsets should have been combined. In addition, S'' contains no zero columns by construction, as those could only have come from lumping together an enzyme subset.

The fourth and final step of the reduction cycle, **Algorithm7**, transforms S'' into S''', where S''' contains a maximal linearly independent subset of the rows of S''. It follows that $\text{Row}(S'') = \text{Row}(S''')$, and therefore, also $\text{Null}(S'') = \text{Null}(S''')$. By combining this with the conclusion of the previous paragraph, $S'''u = \mathbf{0}$ and $u_i = 1$

is feasible for each i, so S''' does not admit any strictly detailed balanced reactions. Furthermore, S''' does not admit any enzyme subsets because its rowspace is the same as that of S''. If S''' had a zero column, say column i, then so would S'' because the rows eliminated from S'' were linear combinations of the rows of S''' and would all have 0 in position i as well, contradicting the fact that S'' has no zero columns. Finally, S''' has linearly independent rows, so there are no redundant constraints to be removed. It follows that S''' is indeed in canonical form after a single reduction cycle. □

Lemma 10 (Canonical Form in Reconfigured Networks). *Let S be the stoichiometric matrix of a metabolic network. If S is in canonical form, then so is the stoichiometric matrix of the reconfigured network.*

Proof. Suppose that S is the stoichiometric matrix of a metabolic network with n reactions, and let $\mathcal{I} \subseteq \{1, 2, \ldots, n\}$ be the set of irreversible reactions in S. Let $\mathcal{R} := \{1, 2, \ldots, n\} - \mathcal{I}$ denote the set of reversible reactions, and let us renumber the reactions so that the matrix S can be written in block form as $S = [S_\mathcal{I}\ S_\mathcal{R}]$ (this, of course, does not affect its being in canonical form). The reconfigured network then has the stoichiometric matrix $\tilde{S} = [S_\mathcal{I}\ S_\mathcal{R} - S_\mathcal{R}]$. Assume that S is in canonical form.

Suppose that $\tilde{y}^T\tilde{S} = 0$ for some vector $\tilde{y} \neq 0$. Partitioning $\tilde{y}$ according to the block form of $\tilde{S}$ gives $[y_1\ y_2\ y_3]^T[S_\mathcal{I}\ S_\mathcal{R}\ -S_\mathcal{R}] = \mathbf{0}$, and hence $y_1^T S_\mathcal{I} = \mathbf{0}$ and $y_2^T S_\mathcal{R} = \mathbf{0} = y_3^T S_\mathcal{R}$, so that $[y_1\ y_2]^T S = \mathbf{0} = [y_1\ y_3]^T S$. Since S is in canonical form, we have $y_1 = \mathbf{0}$ and $y_2 = \mathbf{0} = y_3$, so that $\tilde{y} = \mathbf{0}$, contradicting the original assumption.

Let k be a reversible reaction in the original network, and let k and $\tilde{k}$ be its two directions in the reconfigured network. Note that $e_k + e_{\tilde{k}}$ is a flux vector for $\tilde{S}$, so neither k nor $\tilde{k}$ can be a strictly detailed balanced reaction in $\tilde{S}$. Hence only irreversible reactions from the original network can be strictly detailed balanced in $\tilde{S}$, but that would contradict the fact that S is in canonical form.

By a similar argument, we note that no enzyme subset of $\tilde{S}$ can contain exactly one of k or $\tilde{k}$, so every enzyme subset of $\tilde{S}$ must contain neither or both of them. If an enzyme subset contains both, the fluxes through k and $\tilde{k}$ must be equal for all flux vectors of $\tilde{S}$, so that k would be strictly detailed balanced in S, contradicting the fact that it is in canonical form. If there is an enzyme subset that contains no reversible reactions from S (in either direction), it would also be an enzyme subset of S, once again contradicting the fact that it is in canonical form.

The reconfigured network is trivially properly directed because all of its reactions are irreversible. Finally, no zero column can arise from reconfiguration because it would have to have been present in the original matrix, contradicting the fact that it was in canonical form. This completes the proof. □

Lemma 11 (NP-Completeness of Sparse Solutions). *Given an $m \times n$ rational matrix S and an integer $k < n$ it is NP-complete to decide whether the system $Sx = \mathbf{0}$ has a nonzero solution with $\|x\|_0 \leq k$.*

Proof. The proof in [102] (a reduction from the 3-dimensional matching problem) holds even when the flux vector is not restricted to have non-negative coordinates. Note that this result is proven in [176] for the field of two elements rather than the field of rational numbers using a similar reduction along the way. □

Lemma 12 (Energetic Feasibility in Canonical Form). *If a mode v is energetically feasible for S, its equivalent mode is energetically feasible for the version of S in canonical form (with or without reconfiguration).*

Proof. Suppose that S is the initial stoichiometric matrix, and v is an energetically feasible mode for S. Let w be a vector of chemical potentials for each internal reaction that satisfies equation (A.3) together with v. We will show that every step of the reduction of S to canonical form keeps the vector equivalent to v energetically feasible. We only need to worry about the internal reactions in this proof.

If reaction i is blocked, then $v_i = 0$ and $w_i = 0$ as well, so deleting it can be compensated by deleting the corresponding components from both v and w.

If reaction i is effectively forward, then the sign of w_i had to be determined correctly for the original w in order to ensure energetic feasibility, and that will not change. If it is effectively backward, then multiplying v_i and w_i by -1 simultaneously ensures that v and w satisfy all the constraints.

Finally, consider the lumping together of enzyme subsets. It is sufficient to consider a single enzyme subset because they are non-overlapping, and since the lumping procedure can be done a pair of reactions at a time (as shown in the proof of lemma 9). Therefore, suppose reactions i and j are in an enzyme subset, and let $\kappa \neq 0$ be their ratio, so that $v_j = \kappa v_i$. By lemma 5, the vector $e_j - \kappa e_i$ is then in the rowspace of S. In the modified S, the new reaction is $S_{\text{new}} = S_i + \kappa S_j$.

Suppose first that i and j are both internal reactions. Since w is in the rowspace of S, we had $w = y^T S$ for some y, so that $w_i = y^T S_i$ and $w_j = y^T S_j$. By replacing the i-th and j-th components of w with the new component $w_{\text{new}} = w_i + \kappa w_j$, we ensure that $w_{\text{new}} = y^T S_i + \kappa y^T S_j = y^T (S_i + \kappa S_j) = y^T S_{\text{new}}$, so that the property of being in the rowspace is conserved. We also have $v_{\text{new}} = v_i$ in this case, so that $v_{\text{new}} w_{\text{new}} = v_i(w_i + \kappa w_j) = v_i w_i + \kappa v_i w_j = v_i w_i + v_j w_j \leq 0$, with equality if and only if both terms are 0, so the energetic feasibility of the modified v is preserved. If one of i and j is an exchange reaction, however, then the combined reaction will be an exchange reaction as well, and therefore will not impose any restrictions on w.

If a zero column in S results from lumping together an enzyme subset, the corresponding entry of w will be 0 because w is in the rowspace of S. Since the corresponding entry of v may not be 0, this would violate equation (A.3). This gives us a formal justification for deleting zero columns.

Finally, when a redundant row is deleted from S, it is a linear combination of the

remaining rows of S. Therefore, the corresponding row of $S_{\mathcal{C}}$ is a linear combination of the remaining rows of $S_{\mathcal{C}}$ with the same coefficients, so the rowspace of $S_{\mathcal{C}}$ does not change when the redundant rows of S are deleted, and w remains a vector in the rowspace of $S_{\mathcal{C}}$.

For the reconfiguration procedure, it was mentioned in the main body of the thesis that the restrictions on w do not apply to reactions that represent the other direction of a reaction having nonzero flux. Indeed, because w is in the rowspace of S, its components corresponding to S_i and $-S_i$ will be negatives of each other, which would violate equation (A.3) if v is zero in one and non-zero in the other. □

Lemma 13 (Strain Optimization in Irreversible Networks). *Equation (11.9) holds for any mode v with $v_X = 0$ if and only if there is a vector u in the rowspace of S such that $R_-(u) \subseteq X \cup \{target\}$, with $u_{target} = -1$ and $u_{biomass} = \kappa$ for some $\kappa \geq \lambda$.*

Proof. Suppose that equation (A.4) holds for some X. By Farkas' lemma, the condition on v, which can be rewritten as $\lambda v_{\text{biomass}} \leq v_{\text{target}}$, must be obtainable as a nonnegative linear combination of the equality constraints $S_{-X}v = \mathbf{0}$ and the lower bounds $v_i \geq 0 \; \forall \; i$. Therefore, there exists a vector y such that

$$y^T S_{-X} = \lambda e_{\text{biomass}} - e_{\text{target}} + \sum_{j \in X^C} \lambda_j e_j, \; \lambda_j \geq 0 \; \forall \; j \in X^C.$$

Setting $\kappa = \lambda + \lambda_{\text{biomass}}$ and noting that the only negative components of the resulting vector can come from $y^T S_X$ proves the result in the forward direction. Conversely, suppose that such a u exists. Since every mode v is orthogonal to u, we get $v_X = \mathbf{0} \implies -v_{\text{target}} + \kappa v_{\text{biomass}} \leq 0 \implies v_{\text{target}} \geq \kappa v_{\text{biomass}}$. □

Lemma 14 (Restrictions on the Biomass Composition). *Given a stoichiometric matrix S with biomass reaction S_0, growth is possible only if $S_0 \in Col(S_{>0})$. Furthermore,*

if this condition is not satisfied, there exists $\delta > 0$ such that it is not satisfied for any S_0' such that $\|S_0 - S_0'\| \leq \delta$.

Proof. Consider the condition $[S_0\ S_{>0}]v = \mathbf{0}$. It has a solution with $v_0 = 1$ if and only if $S_0 + S_{>0}v_{>0} = \mathbf{0}$, which means that $S_0 = S_{>0}(-v_{>0})$. Here, $v_{>0}$ denotes the vector obtained from v by removing the first entry. Therefore, S_0 must be a linear combination of the columns of $S_{>0}$, as required.

Suppose that this is not the case. Then we can write $S_0 = S_0^{\|} + S_0^{\perp}$, where $S_0^{\|}$ is the unique projection of S_0 onto the column space of $S_{>0}$, and $S_0^{\perp} \neq \mathbf{0}$. Let $\delta := \|S_0^{\perp}\| > 0$. Then, by the properties of projections [23], any S_0' such that $\|S_0 - S_0'\| \leq \delta$ cannot be in the column space of $S_{>0}$, completing the proof. □

Lemma 15 (Existence of a Suitable Biomass Reaction in Fully Reversible Networks). *In a fully reversible network, if every experiment i in I_- is satisfiable with respect to I_+, then all the experiments in I_- are simultaneously satisfiable with respect to I_+. Furthermore, in that case, for any biomass reaction S_0 that produces in silico growth for every experiment in I_+ and any $\delta > 0$ there exists a S_0' such that $\|S_0 - S_0'\| \leq \delta$ and S_0' simultaneously satisfies all the experiments in I_- with respect to I_+.*

Proof. Suppose that $C_+ := \cap_{\{i \in I_+\}}\mathrm{Col}(S_i)$ is a non-zero subspace of $\mathbb{R}^m$ (otherwise the statement is trivial). Then the condition that every experiment j in I_- is individually satisfiable means that the intersection of C_+ with $\mathrm{Col}(S_j)$ is a proper subspace of C_+. However, no finite union of proper subspaces can cover a vector space (because, once a basis for the vector space is chosen, finitely many linear conditions on the components can always be avoided). Therefore, there exists a biomass reaction in C_+ which simultaneously avoids all the subspaces $\mathrm{Col}(S_j)$ with j in I_-.

For the second part of the statement, let S_0 be a particular vector in C_+. If $S_0 \notin \mathrm{Col}(S_j)$ for any $j \in I_-$, then we can take $S_0' = S_0$. Otherwise, $S_0 \in \mathrm{Col}(S_j)$ for some $j \in I_-$. Let $\delta > 0$ be given, and consider the set $Q := \mathcal{B}(S_0, \delta) \cap C_+$, where

$\mathcal{B}(x,r)$ is the ball of radius r around x. Since $\mathrm{Col}(S_j)$ is a proper subspace of C_+ for every $j \in I_-$, there exists a point in Q such that Q avoids the finitely many linear conditions imposed by the subspaces $\mathrm{Col}(S_j)$ at S_0. This completes the proof. $\square$

Appendix B

List of Notations

The notations are in rough alphabetical order; Latin letters appear before Greek ones.

$\mathcal{C}$	the set of internal reactions in a network	66
c	the number of highest-weighted neighbors considered	24
c	the vector of concentrations, one for each metabolite	110
C_+	intersection of the column spaces of S_i with $\nu_i = +$	107
$\mathrm{Col}(\cdot)$	the column space of a given matrix	107
d	the number of bits of precision used by QSopt	117
$d(\cdot,\cdot)$	distance between two nodes in a graph	46
$\mathcal{D}_i$	the set of reactions disabled in the i-th experiment	102
d^+_{in}	positive in-degree of a node	52
d^-_{in}	negative in-degree of a node	52
$d_{\mathrm{min}}(\cdot,\cdot)$	distance between given nodes irrespective of sign	47
d^+_{out}	positive out-degree of a node	52
d^-_{out}	negative out-degree of a node	52
E	normalized vector of pairwise BLAST scores	19
$\mathcal{E}$	the set of exchange reactions in a network	66
e_i	the i-th standard basis vector	77
F	the set of all admissible modes	74
$F(G)$	family of causal graphs with the same degree sequences as G	52
F_i	the set of all flux modes v with $v_i = 1$	89
G	a graph	23
$\bar{g}$	a gene taken with the opposite sign	47
G^+	set of upregulated genes	47
G^-	set of downregulated genes	47
$G^\pm$	set of all significant genes	47
G_C	a computational graph	45
G_j	a randomly generated geneset in a sample	49

R	rank of a given node	20
$R(v)$	the support (set of active reactions) of a mode v	74
r_j	the ratio of the fluxes through two reactions	82
$R_+(v)$	the positive support of a mode v	74
$R_-(v)$	the negative support of a mode v	74
$\mathrm{Row}(\cdot)$	the rowspace of a given matrix	69
S	a stoichiometric matrix	66
$s(\cdot)$	the sequence similarity, BLAST score	23
$S(\cdot,\cdot)$	score of a given target with a given significant geneset	48
sgn	the signum function	48
$swap(\cdot,\cdot)$	change in objective function after swapping two edges	28
S^T	the transpose of a matrix S	83
S_0	the biomass reaction	106
S_{-Q}	the matrix S with the columns in Q removed	79
$S_{>t}$	the matrix S with the first t columns removed	106
$S_{i,j}$	the entry in row i and column j of a matrix S	66
$S_{\mathcal{C}}$	the submatrix of S with columns in $\mathcal{C}$	69
T	a set of candidate "swappable" pairs	27
t	an auxiliary vector	94
$t(\cdot)$	the topological similarity, number of conserved edges	23
$T^\perp$	the orthogonal complement of a vector space T	108
u, f	values and frequencies of a discrete probability density function	56
v	a vector of fluxes, also called a mode	67
v^*	a mode achieving the maximum growth rate	70
w	vector of chemical potentials for each reaction	68
$w(\cdot)$	weight of a given edge	19

Part VI

Bibliography

Bibliography

[1] Chindelevitch, L., Liao, C.-S. and Berger, B. Local Optimization for Global Alignment of Protein Interaction Networks. Proceedings of the Pacific Symposium on Biocomputing (2010), pp. 123-132.

[2] Ziemek, D., Chindelevitch, L., Enayetallah, A., Jaeger, S., Randhawa, R., Sidders, B., Brockel, C. and Huang, E. Interpreting expression data based on large-scale, high-quality literature networks: the Causal Reasoning Engine. Poster to be presented at ISMB 2010.

[3] Chindelevitch, L., Regev, A. and Berger, B. FAMA: Fast and Accurate Metabolic Analysis. In preparation (2010).

[4] Schnall-Levin, M., Chindelevitch, L. and Berger, B. An Abstract Framework for Structure Design: Inverting the Viterbi Algorithm. Proceedings of the 25th International Conference on Machine Learning (2008), pp. 904-911.

[5] Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M. and Sakaki, Y. A comprehensive two-hybrid analysis to explore the yeast protein interactome. Proceedings of the National Academy of Sciences USA (2001), Volume 98, pp. 4569-4574.

[6] Ho, Y., Gruhler, A., Heilbut, A., Bader, G., Moore, L., Adams, S., Millar, A., Taylor, P., Bennett, K., Boutilier, K., Yang, L., Wolting, C., Donaldson, I.,

Schandorff, S., Shewnarane, J., Vo, M., Taggart, J., Goudreault, M., Muskat, B., Alfarano, C., Dewar, D., Lin, Z., Michalickova, K., Willems, A., Sassi, H., Nielsen, P., Rasmussen, K., Andersen, J., Johansen, L., Hansen, L., Jespersen, H., Podtelejnikov, A., Nielsen, E., Crawford, J., Poulsen, V., Sørensen, B., Matthiesen, J., Hendrickson, R., Gleeson, F., Pawson, T., Moran, M., Durocher, D., Mann, M., Hogue, C., Figeys, D. and Tyers, M. Systematic identification of protein complexes in Saccharomyces cerevisae by mass spectrometry. Nature (2002), Volume 415, pp. 180-183.

[7] Gavin, A., Bösche, M., Krause, R., Grandi, P., Marzioch, M., Bauer, A., Schultz, J., Rick, J., Michon, A., Cruciat, C., Remor, M., Höfert, C., Schelder, M., Brajenovic, M., Ruffner, H., Merino, A., Klein, K., Hudak, M., Dickson, D., Rudi, T., Gnau, V., Bauch, A., Bastuck, S., Huhse, B., Leutwein, C., Heurtier, M., Copley, R., Edelmann, A., Querfurth, E., Rybin, V., Drewes, G., Raida, M., Bouwmeester, T., Bork, P., Seraphin, B., Kuster, B., Neubauer, G. and Superti-Furga, G. Functional organization of the yeast proteome by systematic analysis of protein complexes. Nature (2002), Volume 415, pp. 141-147.

[8] Sharan, R., Suthram, S., Kelley, R., Kuhn, T., McCuine, S., Uetz, P., Sittler, T., Karp, R. and Ideker, T. Conserved patterns of protein interaction in multiple species. Proceedings of the National Academy of Sciences USA (2005), Volume 102, pp. 1974-1979.

[9] Singh, R., Xu, J. and Berger, B. Global alignment of multiple protein interaction networks with application to functional orthology detection. Proceedings of the National Academy of Sciences USA (2008), Volume 105, pp. 12763-12768.

[10] Kalaev, M., Bafna, V. Sharan, R. Fast and accurate alignment of multiple protein networks. Research in Computational Molecular Biology. Volume 4955, Lecture

Notes in Computer Science, Springer, Berlin/Heidelberg (2008) pp. 246-256.

[11] Flannick, J., Novak, A., Do, C., Srinivasan, B. and Batzoglou, S. Automatic parameter learning for multiple network alignment. Research in Computational Molecular Biology. Volume 4955, Lecture Notes in Computer Science, Springer, Berlin/Heidelberg (2008) pp. 214-231.

[12] Liao, C.-S., Lu, K., Baym, M., Singh, R. and Berger, B. IsoRankN: spectral methods for global alignment of multiple protein networks. Proceedings of the International Conference on Intelligent Systems in Molecular Biology (2009), pp. 253-258.

[13] Zaslavskiy, M., Bach, F. and Vert, J.-P. Global alignment of protein-protein interaction networks by graph matching methods. Proceedings of the International Conference on Intelligent Systems in Molecular Biology (2009), pp. 259-267.

[14] Berg, J. and Lässig, M. Cross-species analysis of biological networks by Bayesian alignment. Proceedings of the National Academy of Sciences USA (2006), Volume 103, pp. 10967-10972.

[15] Dutkowski, J. and Tiuryn, J. Identification of functional modules from conserved ancestral protein-protein interactions. Bioinformatics (2007), Volume 23, pp. 149-158.

[16] Kelley, B., Sharan, R., Karp, R., Sittler, T., Root, D., Stockwell, B. and Ideker, T. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. Proceedings of the National Academy of Sciences USA (2003), Volume 100, pp. 11394-11399.

[17] Kelley, B., Yuan, B., Lewitter, F., Sharan, R., Stockwell, B. and Ideker, T. Pathblast: a tool for alignment of protein interaction networks. Nucleic Acids Research (2004), Volume 32, pp. 83-88.

[18] Koyutürk, M., Kim, Y., Topkara, U., Subramaniam, S., Szpankowski, W. and Grama, A. Pairwise Alignment of Protein Interaction Networks. Journal of Computational Biology (2006), Volume 13, pp. 182-199.

[19] Srinivasan, B., Novak, A., Flannick, J., Batzoglou, S. and Mcadams, H. Integrated protein interaction networks for 11 microbes. Research in Computational Molecular Biology. Volume 3909, Lecture Notes in Computer Science, Springer, Berlin/Heidelberg (2006) pp. 1-14.

[20] Page, L., Brin, S., Motwani, R. and Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report (1999). Stanford InfoLab.

[21] Altschul, S., Gish, W., Miller, W., Myers, E. and Lipman, D. Basic Local Alignment Search Tool. J. Mol. Biol. (1990), Volume 215, pp. 403-410.

[22] Chindelevitch, L. Thoughts on the IsoRank algorithm. Unpublished manuscript.

[23] Trefethen, L. and Bau, D. Numerical Linear Algebra. Society for Industrial and Applied Mathematics (1997).

[24] Kronecker Product. *http://en.wikipedia.org/wiki/Kronecker_product.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[25] Rearrangement inequality. *http://en.wikipedia.org/wiki/Rearrangement_inequality.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[26] Gene Ontology. *http://www.geneontology.org/.* Last accessed on August 9, 2010.

[27] Sahni, S. and Gonzales, T. P-complete approximation problems. Journal of the Association for Computing Machinery (1976), Volume 23, pp. 555-565.

[28] Chandra, B., Karloff, H. and Tovey, C. New results on the old k-opt algorithm for the TSP. Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms (1994), pp. 150-159.

[29] Croes, G. A method for solving traveling salesman problems. Operations Research (1958), Volume 6, pp. 791-812.

[30] Hubbard, T., Aken, B., Beal, K., Ballester, B., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cunningham, F., Cutts, T., Down, T., Dyer, S., Fitzgerald, S., Fernandez-Banet, J., Graf, S., Haider, S., Hammond, M., Herrero, J., Holland, R., Howe, K., Howe, K., Johnson, N., Kahari, A., Keefe, D., Kokocinski, F., Kulesha, E., Lawson, D., Longden, I., Melsopp, C., Megy, K., Meidl, P., Overduin, B., Parker, A., Prlic, A., Rice, S., Rios, D., Schuster, M., Sealy, I., Severin, J., Slater, G., Smedley, D., Spudich, G., Trevanion, S., Vilella, A., Vogel, J., White, S., Wood, M., Cox, T., Curwen, V., Durbin, R., Fernandez-Suarez, X., Flicek, P., Kasprzyk, A., Proctor, G., Searle, S., Smith, J., Ureta-Vidal, A. and Birney, A. Ensembl 2007. Nucleic Acids Research (2007), Volume 35, pp. 610-617.

[31] Johnson, D. and McGeoch, L. The traveling salsman problem: a case study in local optimization. Local Search in Combinatorial Optimization, John Wiley & Sons, London (1997), pp. 215-310.

[32] Viger, F. and Latapy, M. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. Proceedings of the International Computing and Combinatorics Conference (2005), pp. 440-449.

[33] Taylor, R. Constrained switchings in graphs. Combinatorial Mathematics (1980), Volume 8, pp. 314-336.

[34] Kao, M., Lam, T., Sung, W. and Ting, H. A decomposition theorem for maximum weight bipartite matchings. SIAM Journal of Computing (2001), Volume 31, pp. 18-26.

[35] Kuhn, H. The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly (1955), Volume 2, pp. 83-97.

[36] Lawler, E., Lenstra, J., Rinnooy Kan, A. and Shmoys, D. The Traveling Salesman Problem, John Wiley & Sons, Chichester (1985).

[37] Lueker, G. Manuscript, Princeton University, (1976).

[38] Papadimitriou, C. and Steiglitz, K. On the complexity of local search for the traveling salesman problem. SIAM Journal of Computing (1977), Volume 6, pp. 76-83.

[39] Jordan, M. (editor). Learning in Graphical Models. The MIT Press, Cambridge, (2001).

[40] Bandyopadhyay, S., Sharan, R. and Ideker, T. Systematic identification of functional orthologs based on protein network comparison. Genome Research (2006), Volume 16, pp. 428-435.

[41] Zaslavskiy, M., Bach, F. and Vert, J.-P. A path following algorithm for graph matching. Image and Signal Processing, Proceedings of the 3rd International Conference (2008), pp. 329-337.

[42] Python Software Foundation. *http://www.python.org/psf/*

[43] Hagberg, A., Schult, A. and Swart, P. Exploring network structure, dynamics, and function using NetworkX. Proceedings of the 7th Python in Science Conference (2008), pp. 11-15.

[44] Galil, Z. Efficient Algorithms for Finding Maximum Matchings in Graphs. ACM Computing Surveys (1986), Volume 18, Article 1, pp. 23-38.

[45] IsoRank and IsoRankN. *http://isorank.csail.mit.edu.* Last accessed on August 9, 2010.

[46] HomoloGene. *http://www.ncbi.nlm.nih.gov/homologene.* Last accessed on August 9, 2010.

[47] Tsay, A., Lovejoy, W. and Karger, D. Random Sampling in Cut, Flow, and Network Design Problems. Mathematics of Operations Research (1999), Volume 24, Article 2, pp. 383-413.

[48] Newman, M. Modularity and community structure in networks. Proceedings of the National Academy of Sciences USA (2006), Volume 103, Article 23, pp. 8577-8582.

[49] Lamb, J., Crawford, E., Peck, D., Modell, J., Blat, I., Wrobel, M., Lerner, J., Brunet, J.-P., Subramanian, A., Ross, K., Reich, M., Hieronymus, H., Wei, G., Armstrong, S., Haggarty, S., Clemons, P., Wei, R., Carr, S., Lander, E. and Golub, T. The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease. Science (2006), Volume 313, pp. 1929-1935.

[50] PubMed *http://www.ncbi.nlm.nih.gov/pubmed/* Last accessed on August 9, 2010.

[51] Ingenuity Systems *http://www.ingenuity.com/* Last accessed on August 9, 2010.

[52] Wherry, E. J., Ha, S.-J., Kaech, S., Haining, N., Sarkar, S., Kalia, V., Subramaniam, S., Blattman, J., Barber, D. and Ahmed, R. Molecular Signature of $CD8^+$ T Cell Exhaustion during Chronic Viral Infection. Immunity (2007), Volume 27, pp. 670-684.

[53] Pollard, J., Butte, A., Hoberman, S., Joshi, M., Levy, J. and Pappo, J. A Computational Model to Define the Molecular Causes of Type 2 Diabetes Mellitus. Diabetes Technology and Therapeutics (2005), Volume 7, pp. 323-336.

[54] Stauffer, A. and Barbosa, V. A Study of the Edge-Switching Markov-Chain Method for the Generation of Random Graphs. arXiv:cs/0512105v1 [cs.DM] 29 Dec 2005.

[55] Rao, A. R., Jana, R. and Bandyopadhyay, S. A Markov Chain Monte Carlo Method for Generating Random $(0, 1)$-Matrices with Given Marginals. The Indian Journal of Statistics (1996), Volume 58, pp. 225-242.

[56] Milo, R., Kashtan, N., Itzkovitz, S., Newman, M. and Alon, U. On the uniform generation of random graphs with prescribed degree sequences. arXiv:cond-mat/0312028v2 [cond-mat.stat-mech] 30 May 2004.

[57] Csárdi, G. Package igraph. Available at *http://cneurocvs.rmki.kfki.hu/igraph.*

[58] Fast Fourier transform. *http://en.wikipedia.org/wiki/Fast_Fourier_transform.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[59] Set Cover Problem. *http://en.wikipedia.org/wiki/Set_cover_problem.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[60] Integer Linear Programming. *http://en.wikipedia.org/wiki/Linear_programming* (section on Integer Unknowns). Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[61] CPLEX. *http://www.ilog.com/products/cplex/.* Last accessed on August 9, 2010.

[62] lpsolve. *http://lpsolve.sourceforge.net/5.5/.* Last accessed on August 9, 2010.

[63] Chindelevitch, L. Technical Report on the Internship at CombinatoRx. Available upon request.

[64] Steiner Tree. *http://en.wikipedia.org/wiki/Steiner_tree.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[65] Aho, A., Garey, M., Ullman, J. The Transitive Reduction of a Directed Graph. SIAM Journal on Computing (1972), Volume 1, pp. 131-137.

[66] Grant, K., Abdelguerft, M., Thomas, B., and Dennis, M. A Comparison of Transitive Closure Algorithms for Use in Intelligent Database Systems. Midwest Symposium on Circuits and Systems (1994), Volume 37, pp. 1366-1372.

[67] Savageau, M. Biochemical Systems Analysis. I. Some Mathematical Properties of the Rate Law for the Component Enzymatic Reactions. J. Theoret. Biol. (1969), Volume 25, pp. 365-369.

[68] Savageau, M. Biochemical Systems Analysis. II. The Steady-state Solutions for an n-pool System using a Power-law Approximation. J. Theoret. Biol. (1969), Volume 25, pp. 370-379.

[69] Horn, F. and Jackson, R. General mass action kinetics. Arch. Rational Mech. Anal. (1972), Volume 47, pp. 81-116.

[70] Feinberg, M. The Existence and Uniqueness of Steady States for a Class of Chemical Reaction Networks. Arch. Rational Mech. Anal. (1995), Volume 132, pp. 311-370.

[71] Shinar, G. and Feinberg, M. Structural Sources of Robustness in Biochemical Reaction Networks. Science (2010), Volume 327, pp. 1389-1391.

[72] Metabolic Control Analysis. *http://en.wikipedia.org/wiki/Metabolic_control_analysis.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[73] Reder, C. Metabolic Control Theory: A Structural Approach. J. theor. Biol. (1988), Volume 135, pp. 175-201.

[74] Wunderlich, Z. and Mirny, A. Using the Topology of Metabolic Networks to Predict Viability of Mutant Strains. Biophysical Journal (2006), Volume 91, pp. 2304-2311.

[75] Van Helden, J., Wernisch, L., Gilbert, D. and Wodak, S. Graph-based analysis of metabolic networks. Ernst Schering Research Foundation Workshop (2002), pp. 245-274.

[76] Croes, D., Couche, F., Wodak, S. and Van Helden, J. Inferring meaningful pathways in weighted metabolic networks. J. Mol. Biol. (2006), Volume 356, pp.222-236.

[77] Planes, F. and Beasley, J. A critical examination of stoichiometric and path-finding approaches to metabolic pathways. Briefings in Bioinformatics (2008), Volume 9, pp. 422-436.

[78] Jeong, H., Tombor, B., Albert, R., Oltvai, Z. and Barabási, A.-L. The large-scale organization of metabolic networks. Nature (2000), Volume 407, pp. 651-654.

[79] Wagner, A. and Fell, D. The small world inside large metabolic networks. Proc. R. Soc. Lond. B (2001), Volume 268, pp. 1803-1810.

[80] Arita, M. The metabolic world of Escherichia coli is not small. Proc Natl Acad Sci USA (2004), Volume 101, pp. 1543-1547.

[81] Chen, L. and Vitkup, D. Distribution of orphan metabolic activities. Trends in Biotechnology (2007), Volume 25, pp. 343-348.

[82] Duarte, N., Herrgård, M. Palsson, B. Reconstruction and Validation of Saccharomyces cerevisiae iND750, a Fully Compartmentalized Genome-Scale Metabolic Model. Genome Res. (2004), Volume 14, pp. 1298-1309.

[83] Mo, M., Palsson, B. and Herrgård, M. Connecting extracellular metabolomic measurements to intracellular flux states in yeast. BMC Systems Biology (2009), pp. 3-37.

[84] Schuster, S. and Höfer, T. Determining All Extreme Semi-positive Conservation Relations in Chemical Reaction Systems : A Test Criterion for Conservativity. Journal of the Chemical Society, Faraday Transactions (1991), Volume 87, pp. 2561-2566.

[85] Yang, F., Qian, H. and Beard, D. Ab initio prediction of thermodynamically feasible reaction directions from biochemical network stoichiometry. Metabolic Engineering (2005), pp. 251-259.

[86] Gagneur, J. and Klamt, S. Computation of elementary modes: a unifying framework and the new binary approach. BMC Bioinformatics (2004), Volume 5, Article 175.

[87] Beard, D., Liang, S. and Qian, H. Energy Balance for Analysis of Complex Metabolic Networks. Biophys. J. (2002), Volume 83, pp. 79-83.

[88] Nigam, R. and Liang, S. A Pivoting Algorithm for Metabolic Networks in the Presence of Thermodynamic Constraints. Proceedings of the IEEE Computational Systems Bioinformatics Conference (2005).

[89] Chemical potential. *http://en.wikipedia.org/wiki/Chemical_potential.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[90] Feist, A., Henry, C., Reed, J., Krummenacker, M., Joyce, A., Karp, P., Broadbelt, L., Hatzimanikatis, V. and Palsson, B. A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. Molecular Systems Biology (2007), Volume 3, Article 121.

[91] Henry, C., Jankowski, M., Broadbelt, L. and Hatzimanikatis, V. Genome-Scale Thermodynamic Analysis of Escherichia coli Metabolism. Biophysical Journal (2006), Volume 90, pp. 1453-1461.

[92] Burgard, A. and Maranas, C. Optimization-Based Framework for Inferring and Testing Hypothesized Metabolic Objective Functions. Biotechnology and Bioengineering (2003), Volume 82, pp. 670-677.

[93] Schuetz, R., Kuepfer, L. and Sauer, U. Systematic evaluation of objective functions for predicting intracellular fluxes in Escherichia coli. Molecular Systems Biology (2007), Volume 3, Article 119.

[94] Gianchandani, E., Oberhardt, M., Burgard, A., Maranas, C. and Papin, J. Predicting biological system objectives de novo from internal state measurements. BMC Bioinformatics (2008), Volume 9, Article 43.

[95] Beste, D., Hooper, T., Stewart, G., Bonde, B., Avignone-Rossa, C., Bushell, M., Wheeler, P., Klamt, S., Kierzek, A. and McFadden, J. GSMN-TB: a web-

based genome-scale network model of Mycobacterium tuberculosis metabolism. Genome Biology (2007), Volume 8, Article R89.

[96] Reed, J. and Palsson, B. Genome-Scale In Silico Models of E. coli Have Multiple Equivalent Phenotypic States: Assessment of Correlated Reaction Subsets That Comprise Network States. Genome Res. (2004), Volume 14, pp. 1797-1805.

[97] Thiele, I. and Palsson, B. A protocol for generating a high-quality genome-scale metabolic reconstruction. Nature protocols (2010), Volume 5, pp. 93-121.

[98] Klamt, S. and Gilles, E. Minimal cut sets in biochemical reaction networks. Bioinformatics (2004), Volume 20, pp. 226-234.

[99] Klamt, S. Generalized concept of minimal cut sets in biochemical networks. BioSystems 83 (2006) pp. 233-247.

[100] Boyd, S. and Vandenberghe, L. Convex Optimization. Cambridge University Press (2004).

[101] Nigam, R. Consistency, Feasibility and Multiplicity of Fluxes in Metabolic Pathways. In Proceedings of the 18th International Conference on Systems Engineering (2005) pp. 307-312.

[102] Acuña, V., Chierichetti, F., Lacroix, V., Marchetti-Spaccamela, A., Sagot, M. and Stougie, L. Modes and cuts in metabolic networks: Complexity and algorithms. BioSystems (2009), Volume 95, pp. 51-60.

[103] Larhlimi, A. and Bockmayr, A. A new constraint-based description of the steady-state flux cone of metabolic networks. Discrete Applied Mathematics (2009), Volume 157, pp. 2257-2266.

[104] Pfeiffer, T., Sanchez-Valdenebro, I., Nuño, J., Montero, F., Schuster, S. METATOOL: for studying metabolic networks. Bioinformatics (1999), Volume 15, pp. 251-257.

[105] Norm (mathematics). *http://en.wikipedia.org/wiki/Norm_(mathematics)* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[106] Active Set - *http://en.wikipedia.org/wiki/Active_set.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[107] Suthers, P., Dasika, M., Kumar, V., Denisov, G., Glass, J., Maranas, C. A genome-scale metabolic reconstruction of Mycoplasma genitalium, iPS189. PLoS Computational Biology (2009), Volume 5, Issue 2.

[108] Varma, A. and Palsson, B. Metabolic Flux Balancing: Basic Concepts, Scientific and Practical Use. Nature Biotechnology (1994), Volume 12, pp. 994-998.

[109] Covert, M., Schilling, C., Famili, I., Edwards, J., Goryanin, I., Selkov, E. and Palsson, B. Metabolic modeling of microbial strains in silico. Trends in Biochemical Sciences (2001), Volume 26, pp. 179-186.

[110] Burgard, A., Pharkya, P. and Maranas, C. OptKnock: A Bilevel Programming Framework for Identifying Gene Knockout Strategies for Microbial Strain Optimization. Biotechnology and Bioengineering (2003), Volume 84, pp. 647-657.

[111] Rocha, M., Maia, P., Mendes, R., Pinto, J., Ferreira, E., Nielsen, J., Patil, K. and Rocha, I. Natural computation meta-heuristics for the in silico optimization of microbial strains. BMC Bioinformatics (2008), Volume 9, Article 499.

[112] Lun, D., Rockwell, G., Guido, N., Baym, M., Kelner, J., Berger, B., Galagan, J. and Church, G. Large-scale identification of genetic design strategies using local search. Molecular Systems Biology (2009), Volume 5, Article 296.

[113] Segrè, D., Vitkup, D. and Church, G. Analysis of optimality in natural and perturbed metabolic networks. Proceedings of the National Academy of Science (2002), Volume 99, pp. 15112-15117.

[114] Shlomi, T., Berkman, O., and Ruppin, E. Regulatory on-off minimization of metabolic flux changes after genetic perturbations. Proceedings of the National Academy of Sciences (2005), Volume 102, pp. 7695-7700.

[115] Mahadevan, R. and Schilling, C. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. Metabolic Engineering (2003), Volume 5, pp. 264-276.

[116] Donoho, D. Compressed Sensing (2004).

[117] Candès, E., Romberg, J. and Tao, T. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inform. Theory (2004), Volume 52, pp. 489-509.

[118] Mohimani, G., Babaie-Zadeh, M., Jutten, C. Fast Sparse Representation based on Smoothed L0 Norm. In Independent Component Analysis and Signal Separation, 7th International Conference (2007), pp. 389-396.

[119] Lai, M. On Sparse Solutions of Underdetermined Linear Systems (2009).

[120] Palomar, D. L1-Norm Minimization for Convex-Cardinality Problems. Lecture notes from ELEC547 - Convex Optimization (2009), Hong Kong University of Science and Technology.

[121] Candès, E., Watkin, M. and Boyd, S. Enhancing Sparsity by Reweighted L1 Minimization. Journal of Fourier Analysis and Applications (2008), Volume 14, pp. 877-905.

[122] Reed, J., Patel, T., Chen, K., Joyce, A., Applebee, M., Herring, C., Bui, O., Knight, E., Fong, S. and Palsson, B. Systems approach to refining genome annotation. Proceedings of the National Academy of Sciences (2006), Volume 103, pp. 17480-17484.

[123] Durot, M., Le Fèvre, F., de Berardinis, V., Kreimeyer, A., Vallenet, D., Combe, C., Smidtas, S., Salanoubat, M., Weissenbach, J. and Schachter, V. Iterative reconstruction of a global metabolic model of Acinetobacter baylyi ADP1 using high-throughput growth phenotype and gene essentiality data. BMC Systems Biology (2008), Volume 2, Article 85.

[124] Snitkin, E., Dudley, A., Janse, D., Wong, K., Church, G. and Segrè, D. Model-driven analysis of experimentally determined growth phenotypes for 465 yeast gene deletion mutants under 16 different conditions. Genome Biology (2008), Volume 9, Article R140.

[125] Kumar, V. and Maranas, C. GrowMatch: An Automated Method for Reconciling In Silico/In Vivo Growth Predictions. PLoS Computational Biology (2009), Volume 5, Article 3.

[126] In Silico Organisms. *http://gcrg.ucsd.edu/In_Silico_Organisms/Other_Organisms* UCSD Systems Biology Group. Last accessed on August 9, 2010.

[127] Wishart, D. Computational Approaches to Metabolomics. Chapter 14 in Bioinformatics Methods in Clinical Research, Methods in Molecular Biology, 593. R. Matthiesen (ed.).

[128] Lee, J. M., Gianchandani, E. and Papin, J. Flux balance analysis in the era of metabolomics. Briefings in Bioinformatics (2006), Volume 7, pp. 140-150.

[129] Çakır, T., Patil, K., Önsan, Z., Ülgen, K., Kırdar, B. and Nielsen, J. Integration of metabolome data with metabolic networks reveals reporter reactions. Molecular Systems Biology (2006), Article 50.

[130] Hoppe, A., Hoffmann, S. and Holzhütter, H.-G. Including metabolite concentrations into flux balance analysis: thermodynamic realizability as a constraint on flux distributions in metabolic networks. BMC Systems Biology (2007), Volume 1, Article 23.

[131] Tiwary, H. On the Hardness of Computing Intersection, Union and Minkowski Sum of Polytopes. Discrete Comput. Geom. (2008), Volume 40, pp. 469-479.

[132] Becker, S., Feist, A., Mo, M., Hannum, G., Palsson, B. and Herrgård, M. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox. Nature Protocols (2007), Volume 2, pp. 727-738.

[133] von Kamp, A. and Schuster, S. Metatool 5.0: fast and flexible elementary modes analysis. Bioinformatics (2006), Volume 22, pp. 1930-1931.

[134] Klamt, S., Saez-Rodriguez, J. and Gilles, E. Structural and functional analysis of cellular networks with CellNetAnalyzer. BMC Systems Biology (2007), Volume 1, Article 2.

[135] MATLAB. *http://www.mathworks.com.* Last accessed on August 9, 2010.

[136] Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J., Kitano, H., Arkin, A., Bornstein, B., Bray, D., Cornish-Bowden, A., Cuellar, A., Dronov, S., Gilles, E., Ginkel, M., Gor, V., Goryanin, I., Hedley, W., Hodgman, T., Hofmeyr, J., Hunter, P., Juty, N., Kasberger, J., Kremling, A., Kummer, U., Le Novère, N., Loew, L., Lucio, D., Mendes, P., Minch, E., Mjolsness, E., Nakayama, Y., Nelson, M., Nielsen, P., Sakurada, T., Schaff, J., Shapiro, B., Shimizu, T., Spence, H.,

Stelling, J., Takahashi, K., Tomita, M., Wagner, J., Wang, J. and SBML Forum. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics (2003), Volume 19, pp. 524-531.

[137] lp_solve. *https://sourceforge.net/projects/lpsolve.* Last accessed on August 9, 2010.

[138] GLPK. *http://www.gnu.org/software/glpk/.* Last accessed on August 9, 2010.

[139] LINDO. *http://www.lindo.com.* Last accessed on August 9, 2010.

[140] CPLEX. textithttp://www-01.ibm.com/software/integration/optimization/cplex/. Last accessed on August 9, 2010.

[141] Mosek. *http://www.mosek.com.* Last accessed on August 9, 2010.

[142] Cytoscape. *http://www.cytoscape.org.* Last accessed on August 9, 2010.

[143] Octave. *http://www.gnu.org/software/octave/.* Last accessed on August 9, 2010.

[144] Terzer, M. and Stelling, J. Large-scale computation of elementary flux modes with bit pattern trees. Bioinformatics (2008) Volume 24, pp. 2229-2235.

[145] MAPLE. *http://www.maplesoft.com/products/maple/.* Last accessed on August 9, 2010.

[146] exlp. *http://members.jcom.home.ne.jp/masashi777/exlp.html.* Last accessed on August 9, 2010.

[147] QSopt_ex. *http://www.dii.uchile.cl/~daespino/ESolver_doc/main.html.* Last accessed on August 9, 2010.

[148] MPS format. *http://en.wikipedia.org/wiki/MPS_(format).* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010..

[149] QSopt. *http://www2.isye.gatech.edu/~wcook/qsopt/.* Last accessed on August 9, 2010.

[150] Simplex algorihtm. *http://en.wikipedia.org/wiki/Simplex_algorithm.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[151] Applegate, D., Cook, W., Dash, S., Espinoza, D. Exact solutions to linear programming problems. Operations Research Letters (2007), Volume 35, pp. 693-699.

[152] Papadimitriou, S. The largest subdeterminant in a matrix. Bull. Math. Soc. Greece (1984), Volume 15, pp. 96-105.

[153] Khachiyan, L. On the complexity of approximating extremal determinants in matrices. Journal of Complexity (1994), Volume 11, pp. 138-153.

[154] Mycobacterium tuberculosis *http://en.wikipedia.org/wiki/Mycobacterium_tuberculosis.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[155] Multi-drug-resistant tuberculosis. *http://en.wikipedia.org/wiki/Multi-drug-resistant_tuberculosis.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[156] Extensively drug-resistant tuberculosis. *http://en.wikipedia.org/wiki/Extensively_drug-resistant_tuberculosis.* Wikipedia, the Free Encyclopedia. Last accessed on August 9, 2010.

[157] Kyoto Encyclopedia of Genes and Genomes. *http://www.genome.jp/kegg/.* Last accessed on August 9, 2010.

[158] In Silico Organisms. *http://gcrg.ucsd.edu/In_Silico_Organisms/Other_Organisms.* Last accessed on August 9, 2010.

[159] Jamshidi, N. and Palsson, B. Investigating the metabolic capabilities of Mycobacterium tuberculosis H37Rv using the in silico strain iNJ661 and proposing alternative drug targets. BMC Systems Biology (2007), Volume 1, p. 26.

[160] Raman, K., Rajagopalan, P. and Chandra, N. Flux Balance Analysis of Mycolic Acid Pathway: Targets for Anti-Tubercular Drugs. PLoS Computational Biology (2005), volume 1, p. e46.

[161] The Systems Biology Markup Language. *http://sbml.org/.* Last accessed on August 9, 2010.

[162] Chemical Entities of Biological Interest. *http://www.ebi.ac.uk/chebi/.* Last accessed on August 9, 2010.

[163] E. Webb. Enzyme nomenclature 1992: recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes. Academic Press (1992), San Diego.

[164] NEOS Optimization Server. *http://www-neos.mcs.anl.gov/neos/solvers/index.html.* Last accessed on August 9, 2010.

[165] BioModels Database. *http://www.ebi.ac.uk/biomodels-main/.* Last accessed on August 9, 2010.

[166] Yeast Consensus Model. *http://www.comp-sys-bio.org/yeastnet/.* Last accessed on August 9, 2010.

[167] TB Drug Target Database. *http://www.bioinformatics.org/tbdtdb/druglist.php.* Last accessed on August 9, 2010.

[168] Boshoff, H., Mizrahi, V. and Barry, C. Effects of Pyrazinamide on Fatty Acid Synthesis by Whole Mycobacterial Cells and Purified Fatty Acid Synthase I. Journal of Bacteriology (2002), volume 184(8), pp. 2167-2172.

[169] Sassetti, C., Boyd, D. and Rubin, E. Genes required for mycobacterial growth defined by high density mutagenesis. Molecular Microbiology (2003), volume 48, pp. 77-84.

[170] Chemical Abstracts Service. *http://www.cas.org/.* Last accessed on August 9, 2010.

[171] International Union of Pure and Applied Chemistry. *http://www.iupac.org/.* Last accessed on August 9, 2010.

[172] BioCyc database collection. *http://biocyc.org/.* Last accessed on August 9, 2010.

[173] The Expert Protein Analysis System proteomics server. *http://expasy.org/.* Last accessed on August 9, 2010.

[174] Bein, D., Morales, L., Bein, W., Shields, C., Meng, Z. and Sudborough, I. Clustering and the Biclique Partition Problem. In Proceedings of the Hawaii International Conference on System Sciences: 7-10 January 2008; Waikoloa, edited by R. Sprague, p. 475.

[175] Cytoscape, a network visualization software. *http://www.cytoscape.org/.* Last accessed on August 9, 2010.

[176] Vardy, A. The Intractability of Computing the Minimum Distance of a Code. IEEE Transactions on Information Theory (1997), Volume 43, pp. 1757-1766.