Scalable Computation and Analysis of Elementary Flux Modes in Metabolic Networks

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Dimitrije Jevremovic

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Dr. DANIEL BOLEY

May, 2013

Acknowledgements

First and foremost, I want to thank my Ph.D. adviser Dr. Daniel Boley. It has been an immense honor to be his Ph.D. student and have him as a mentor, colleague and collaborator. I am deeply grateful for his guidance, mentoring, and many hours of constructive and exciting discussions about linear algebra, algorithm design, computational biology and research in general. I would also like to express my gratitude to Dr. Friedrich Srienc and Dr. Carlos P. Sosa. Dr. Srienc as a principal investigator and motivator has introduced and approached us to the unanswered problems in the computational cellular reaction network analysis, which ultimately lead to the work and results described in this thesis manuscript. Dr. Sosa, acting in the capacity of an expert for the use of supercomputing resources in life sciences, facilitated the development and testing of the software at the computing platforms at IBM and Minnesota Supercomputing Institute.

I owe a heartfelt gratitude to my late father Radiša Jevremović, a professor of electrical engineering at the time, who at an early age instilled in me the curiosity and passion for mathematical and physical sciences. His love and passion for life will always remain a great inspiration to me. I would unlikely have endured the nearly six years spent in graduate school, without the support from my mother Dragoslava Jevremović who understood and supported my decision to pursue graduate education and for that purpose move to a remote part of the world. Julijana Mirčevski, my aunt and a retired software engineer, is a major “culprit” for my decision to pursue graduate studies at a university in the United States and explore the field of computational biology.

I am immensely grateful to Valerie Erickson and Dr. Charles Kath from St. Paul, Minnesota, who have welcomed me with open hearts and enthusiasm into the Twin Cities. Throughout the years spent in the area, Valerie and Charles have tirelessly supported me in my endeavors and work in graduate school. I owe special thanks to many wonderful Minnesotans who also greeted my at my arrival, eased my move into the area and made wonderful company during many occasions in the previous years, namely Dawn Johnson, Karen Trudeau, Kristen and Daniel Charette and many others. Finally, I am thankful to my friend Dr. Danijela Stojanac, sister-in-law Sanja Miler-Jevremović and her spouse and my brother Vladan Jevremović with my wonderful nephew Maksim Jevremović, and cousin

Dejan Mirčevski for being wonderful hosts during the holiday breaks when I visited them at their homes in Chicago, Littlehampshire, Mountain View and Boston.

For the financial support I would like to acknowledge the Biomedical Informatics and Computational Biology Traineeship Program at the University of Minnesota, Rochester in 2007/08 and 2008/09, National Science Foundation grant in 2009/2010 and the IBM Ph.D. Fellowship Program in 2010/11, 2011/12 and 2012/13. In addition, I would like to thank the Technical Committee on Parallel Processing and British Biochemical Society for the travel grants which allowed me to travel to present my work at the two conferences in United States and England. I would also like to thank Dr. Cong Trinh from the University of Tennessee, Knoxville, Christopher Flynn and Dr. Pornkamol Unrean for the valuable input, contribution and comments to the work described in this manuscript, as well as to Dr. Mihailo Jovanović of the University of Minnesota, Twin Cities. Finally, I would like to thank my office mates for the camaraderie, witty moments and support.

Dedication

In loving memory of my late father Radiša Jevremović and grandfather Jovan Djenić

Abstract

The full *in silico* reconstruction of the genome-scale cellular metabolic networks in the recent years was enabled and supported with the sequencing of genomic and transcriptomic data across numerous organism species, as well as with the growth of various biological databases. Metabolic networks provide a comprehensive overview of the cellular landscape and its phenotype prediction for the given environmental and growth conditions. Imposing the thermodynamic and steady state constraints on the reactions and internal metabolites of the given metabolic network, respectively, one still has a justifiable model which can be used to study the cellular events of interest. Metabolic pathway is introduced as a biological and mathematical concept of a subset of active reactions with a non-zero flux. In particular, elementary flux mode is a metabolic pathway with a physiological property reflected in the requirement for its enzymatic minimality. Elementary flux modes, as building blocks of the metabolic network, have numerous applications in chemical engineering and biochemistry for the study of phenotype of wild type and mutant cells. However, the large number of elementary flux modes for even medium size metabolic networks, as well as the high computational cost of the underlying Nullspace Algorithm, still present a challenge in the computation and analysis.

The problem of enumerating elementary flux modes is equivalent to the one of enumerating the vertices in the degenerate polytope which still remains an open problem. Algorithm used in such enumeration is the Double Description Method, and its adaptation in the metabolic network analysis is the so-called Nullspace Algorithm. The Nullspace Algorithm is studied to identify its major bottlenecks and possible improvements in design and implementation. The reduced algebraic rank test is proposed in the event when the network has reversible reactions, and it significantly reduces the time required to check the elementarity of the candidate elementary flux mode. In the case when the network admits reversible metabolic pathway, a procedure is given to use the Nullspace Algorithm to compute one of many possible minimal generating sets, itself being a minimal subset of elementary flux modes whose linear combination can fully characterize the solution space .

Initially, parallelization of the enumeration of elementary modes was proposed in the form of Combinatorial Parallel Nullspace Algorithm using MPI library routines for the message-passing distributed memory environment. Due to insufficient memory scalability and need to replicate major data structures across all the compute nodes, further steps were undertaken to attain both memory and time scalability and be able to fit larger metabolic networks to the algorithm. Global Arrays, the partitioned global address space library, was used to facilitate the development of the distributed-memory algorithm with a shared-memory view of the global memory. It allowed a merge of the concept of the shared-memory programming within a distributed-memory environment, hence allowing the computation of nearly 70 million elementary flux modes for the 83-reaction metabolic network of the *Sacharomyces cerevisiae*.

In the event of limited memory and processor resources, a splitting of work needed to compute all elementary flux modes was needed. An earlier divide-and-conquer idea was taken and further developed into a functional implementation. It was merged with the Combinatorial Parallel Nullspace Algorithm to yield the Combined Parallel Nullspace Algorithm and demonstrated reduced memory footprint on each of the subproblems.

The full set of elementary flux modes was used to enumerate multiple subsets of reactions as candidates for knockout which would collapse the metabolic network to the subset of desired elementary flux modes. Metabolic network design goals of (1) network flexibility and (2) growth-coupled production of the target chemical were incorporated within the branch-and-bound breadth-first search algorithms. Algorithms were executed on the metabolic networks of *Escherichia coli* and *Sacharomyces cerevisae* and were compared to an earlier exhaustive variation of the Berge's algorithm.

Finally, a collection of ideas based on non-linear optimization methods is proposed to model the cellular phenotype with a proper objective function and infer the efficient subnetwork of the given metabolic network. Proposed methods are used with the goal of inferring reactions targeted for deletion, where the residual network will contain as many efficient elementary flux modes as possible, thus reducing the cost of running one of the previously implemented algorithms. Optimization methods aim to model the metabolic goals of the minimal energy consumption and enzymatic minimality using the L1-regularized quadratic programming framework.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computational sciences hold a great hope to accelerate the pace of discovery in the life sciences and thus lead to answering major questions and solving open problems. This synergy of disciplines is required mostly due to the immense amount of biological data which has constantly been generated in the past several decades. This acquisition of data is facilitated with the use of the continually improving instrumentation. Computational algorithms and robust software implementations are required to analyze this vast amount of data and extract useful and meaningful knowledge. One of the major efforts is the collection and archiving of the tissue data and the dissection of a biological cell. Events in the biological cell can be studied through the network of its biochemical reactions and molecules, which are localized in the cytoplasm, nuclei and organelles, and across the cellular membranes. This network of transformations between the cellular molecules illustrates the *cellular metabolism*. Such a network is formally known as a *metabolic network* and serves as a tool to study the cellular metabolism. The constraint-based analysis of the metabolic network disregards the need for the knowledge of the biochemical reaction kinetic parameters and observes the reaction activities in the so-called *quasi-steady state* under the known thermodynamic constraints. Under these approximations and constraints, the mathematical modeling and the simulations produce results which align well with the experimentally obtained data, hence justifying the use and adoption of the methodology described in this thesis within systems biology. Phenotype of the cellular metabolic networks is often studied using the concept of the metabolic pathway which is a set of reactions converting one substrate metabolite into one or more product metabolites.

One particular class of metabolic pathways are the elementary flux modes which are characterized with their enzymatic minimality. The use of elementary flux modes has many applications in chemical engineering and biochemistry, however their enumeration for even medium-size metabolic networks still presents a significant computational challenge. The so-called Nullspace Algorithm is used to compute the elementary flux modes and is derived from the Double Description Method, which is an algorithm used to enumerate the vertices in degenerate polytopes.

This thesis proposes new and improved algorithms important for the study of the quasi-steady state cellular metabolic networks such as (1) improvements in the serial Nullspace Algorithm, (2) simple algorithmic procedure to compute the minimal generating set in a metabolic network which admits reversible pathways, (3) combinatorial parallelization of Nullspace Algorithm, (4) combined parallelization of the Nullspace Algorithm which incorporates existing divide-and-conquer paradigm, (5) parallelization of the Nullspace Algorithm using partitioned global address space library known as the Global Arrays Toolkit, (6) algorithms for the enumeration of efficient reaction knockout subsets used in efficient cellular strain design and based on using elementary flux modes and (7) algorithms for the enumeration of efficient elementary flux modes and finding of efficient reaction knockout subsets for efficient cellular strain design using non-linear optimization methods.

- Chapter 2 briefly introduces the field of systems biology, further acquainting the reader with the concept of metabolic network and the way it describes the cellular activities from the aspect of the network of biochemical reactions. Metabolic network analysis is then illustrated through the two families of methods, namely, optimization-based and pathway-based methods. On of the major goals in outlining the two families of methods is to show how they can be applied to infer the targets for the genetic modification leading to the efficient design of cellular microorganism strains. Finally, the chapter mentions the problem of integrated biological network analysis and whole-cell modeling, giving a hint how metabolic network analysis may be merged with other biological data to allow more accurate prediction of the cellular phenotype.

- Chapter 3 starts with the a more comprehensive description of the Nullspace Algorithm which is mentioned first in Chapter 2 in the subsection of the pathway-based methods and elaborates on its complexity and possible improvements. A portion of the Nullspace Algorithm which in every iteration runs a collection of algebraic rank tests is proposed with an improvement which significantly reduces its computational cost. Lastly, a simple method to compute the minimal generating set in the metabolic network which admits reversible pathways is proposed.

- In Chapter 4, a distributed-memory parallelization is proposed under the name *Combinatorial Parallel Nullspace Algorithm*. A pseudo-code of the parallelized algorithm, accompanied with the computational complexity analysis and the results of the runs on *Escherichia coli* and *Sacharomyces cerevisiae* networks is provided. The parallel algorithm, while of insufficient memory scalability, reduced communicate overhead using the merge-tree pattern.

- Chapter 5 formalizes the earlier proposed reverse divide-and-conquer idea in the computation of elementary flux modes. This idea is taken to the next level where its implementation is combined with the Combinatorial Parallel Nullspace Algorithm described in Chapter 4 to result in the Combined Parallel Nullspace Algorithm which attains smaller memory footprint and improved performance.

- Chapter 6 outlines the extension of the Combinatorial Parallel Nullspace Algorithm using a partitioned global address space library, known as the Global Arrays Toolkit. An implementation of Nullspace Algorithm using this library attains improved memory scalability which is demonstrated on different *Sacharomyces cerevisiae* network models.

- Computed elementary flux modes are used in Chapter 7 to infer reaction knockout targets leading to an efficient target chemical producing cell. Different algorithmic variations are proposed which enumerate multiple possible solutions, accounting for network flexibility, growth-coupled regime and minimal required yield.

- A set of algorithms used to study large metabolic networks which cannot be fit into any of the algorithms described in preceding chapters are proposed in Chapter 8. Nonlinear optimization methods are used to infer reaction knockout targets and obtain a residual metabolic network which may constitute an efficient target chemical producing network. Alternatively, similar framework is applied to an idea of computing as many efficient elementary flux modes as possible, where efficient modes are those with high target chemical yield.

- Finally, concluding remarks with contributions and directions for future work are given in Chapter 9.

Chapter 2

Background

2.1 Systems Biology

Systems biology [5] is an inter-disciplinary area which uses mathematical models to describe the cellular behavior and phenomena, as well as the overall properties of the biological systems in general. A wealth of experimentally obtained genomic, transcriptomic, proteomic and metabolomic data can be used to model and simulate either the functioning of the entire biological cell or just one of its segments. Mathematical concepts such as ordinary and partial differential equations, network and graph theory concepts, linear algebraic tools, machine learning algorithms and many others were exploited to study and analyze the cellular systems.

Network representation is an intuitive and first-hand approach used to collect, store and visualize the cellular data [6, 7]. Genetic and transcriptomic data which is acquired using experimental assays and techniques is illustrated using *transcriptional gene regulatory networks*. With the discovery of the cellular proteome and individual protein functional annotation, a concept of *protein interaction networks* is introduced to illustrate the relationships between the intracellular proteins. *Signaling network* illustrates how a cell senses a stimuli in its environment and the way it responds to them [8]. The signals of the stimuli fire a cascade of biochemical reactions which form a signaling pathway through the cell. Finally, cellular biochemical reactions which illustrate various portions of the metabolism are depicted using a *metabolic network*. A formal mathematical model which as accurately as possible represents the cellular metabolic network can be used as a tool in pharmacology and drug discovery, studies of host-pathogen relationship, and analysis of the capabilities of the industrial microorganisms for the production of chemicals and biofuels [9, 10, 11].

2.2 Metabolic Networks

The phenotype of a biological cell can be studied by means of exploring cellular biochemical reactions. Small chemical compounds, known as *metabolites*, can be ingested and secreted across the cellular membrane by means of transport reactions. Within the cellular cytoplasm, nucleus and many organelles, the metabolites are being used in hundreds of interconnected reactions as either their substrates or products. Each single reaction converts one group of substrate metabolites into another group of product metabolites. The amount of metabolite used in a reaction either as the substrate or the product is expressed in the units of moles. This collection of biochemical reactions and their interactions forms a *metabolic network*. Metabolic networks [12, 13, 14] belong to a class of biological networks which allow the representation of biochemical reactions and their relationships within a biological cell or its compartments. A reaction with metabolites lying on the opposite side of a membrane of the cell is considered as an *external reaction*. Otherwise, a reaction with all of its metabolites exclusively lying within the cell boundaries is considered to be an *internal reaction*. Metabolite is considered internal to the network if it is found inside the cell, otherwise it is external in which case it is a substrate/product of an external reaction. Reaction with metabolites on the opposite side of the cellular or organelle membrane is known as the *transport reaction*. It is important to note that transport reactions are the superset of the external reactions. The $m \times q$ stoichiometry matrix, $N_{m\times q}$, is used to quantitatively represent the metabolic network with m internal metabolites and q reactions. The element in i^{th} row and j^{th} column of N represents the amount in moles of the i^{th} metabolite consumed or produced by the j^{th} reaction.

Each biochemical reaction is controlled or catalyzed by one or more enzymes and is characterized by its speed of execution, known as a *reaction flux rate*. Flux rate values for all the q reactions in the metabolic network are collected into a *metabolic flux vector* denoted here as $v_{q\times 1}$. Metabolic network reactions may be reversible or irreversible, where the flux of every irreversible reaction must be non-negative. Hence, an additional thermodynamic constraint must be imposed on the elements of the metabolic flux vector corresponding to the irreversible reactions as $r_i \geq 0$, where $i \in irrev$ are indices of the irreversible reactions. For the given stoichiometry network, the concentration of m metabolites and their change in time can be described using a system of ordinary differential equations as in (2.1).

$$\frac{dC_i}{dt} = \sum_{j=1}^{q} N_{ij} v_j \qquad \text{for} \quad i = 1 \dots m \tag{2.1}$$

where C_i denotes the concentration of the i^{th} metabolite in the network. The construction of the metabolic network was earlier done using the general literature and the existing knowledge in biochemistry and physiology. Cellular enzymes and reactions are available in several databases in the public domain such as KEGG (Kyoto Encyclopedia of Genes and Genomes) [15, 16], EcoCyc [17], MetaCyc [18, 19] and HumanCyc [20]. More recently, full comprehensive *in silico* reconstructions of the genome-scale metabolic networks [21] were performed beginning from the sequenced and annotated cellular genome and complemented with the afore mentioned biochemical and physiological data of the organism. One repository of reconstructed genome-scale metabolic networks is the BiGG (Biochemical Genetic and Genomic) knowledge base [22]. Metabolic network reconstructions[23] are available for both prokaryotic and eukaryotic organisms such as *Escherichia coli* [24], *Sacharomyces cerevisiae* [25, 26], *Haemophilus influenzae* [27], *Helicobacter pylori* [28, 29], *Mycoplasma genitalium* [30], *Staphylococcus aureus* [31], *Homo sapiens* [32, 33] and many others. Data exchange format adopted for the storage and representation of the metabolic networks is SBML (Systems Biology Markup Language) [34, 35] which is a derivative of XML markup language.

2.3 Constraint-Based Metabolic Network Analysis

In many applications and situations of interest, the time resolution and the interval between the observed events are such that it may be assumed that the concentration of the metabolites internal to the metabolic network is constant in time [36, 37, 38]. In other words, this implies that the for every internal metabolite, the amount being produced is equal to the amount being consumed by the participating reactions. Accounting for this and the previously mentioned thermodynamic constraints on irreversible reactions results in a total of two constraints which can be imposed on the metabolic network as given in the following definition.

Definition 1 (Two constraints in steady state metabolic network analysis)**.** Given the metabolic network with its stoichiometry matrix $N_{m \times q}$, following two constraints are defined:

1. *quasi-steady state*: Internal metabolites are not accumulating within the metabolic network. In other words, the amount of metabolite being produced equals the amount being consumed.

$$\sum_{j=1}^{q} N_{ij} v_j = 0 \qquad \text{for} \quad i = 1 \dots m \tag{2.2}$$

2. *thermodynamics*: $v_i \geq 0$ if $i \in irrev$ where $irrev$ is a set of indices of irreversible reactions.

□

These constraints are fundamental and used during the process of the reconstruction of the genome-scale metabolic network. Once the network is reconstructed, the methods for the analysis are again based on the constraints in definition 1. Hence, the two constraints lay grounds for the so-called *COnstraints Based Reconstruction and Analysis* of metabolic networks. In continuation, for convenience we will refer to it as the constraint-based analysis. If the reconstructed metabolic network is available, there may be two possible approaches for its analysis, namely (1) *optimization-based* and (2) *pathway-based*, as will be illustrated in continuation.

2.3.1 Network Redundancies and Inconsistencies

Metabolic network may have redundant metabolites and reactions, whereby eliminating them from the original network may significantly simplify the analysis and alleviate the computational burden, as will be demonstrated in later sections. In the constraint-based analysis of the metabolic networks, most of the known reduction methods are analogous to the methods for the reduction of linear equality and inequality constraints which form part of the linear program [39]. Known reduction techniques are given as [40]:

1. **Metabolite conservation relations** are related to the linear dependencies among rows of the stoichiometry matrix. Considering these metabolite dependency relations the network can be simplified by the removal of redundant metabolites. It suffices to reduce the stoichiometry matrix N to a maximal linearly independent set of rows using simple linear algebra.

2. **Strictly detailed balanced reactions** are those reactions which carry null flux in any quasi-steady state the network is found. They can be identified by solving a simple linear program or using the right nullspace matrix R of the stoichiometry matrix N.

3. **Uniquely produced (consumed) metabolite** is a metabolite M which is produced (consumed) by a single reaction v_{i_0}, and respectively consumed (produced) by k other reactions, namely $v_{i_1} \dots v_{i_k}$. Metabolite M and $k+1$ reactions can be removed and substituted with k new reactions.

4. **Enzyme subset** is defined as a subset of reactions with relative constant flux ratio at any steady state. If the enzyme subset has k reactions $v_{i_1} \dots v_{i_k}$, then $k-1$ of them can be eliminated from the network, as their later recovery is possible if a flux of the one remaining reaction is known. Reactions of enzyme subsets are computed using the right nullspace matrix R of the stoichiometry matrix N.

To obtain the reduced stoichiometry network, one should iteratively apply the listed techniques to the original stoichiometry network [41]. This is an adopted practice prior to performing any analysis of metabolic network capabilities using one of the methods described later in this chapter.

2.3.2 Optimization-based Analysis Methods

Besides the constraints imposed on the metabolic network which account for the quasi-steady state and the reaction thermodynamics, evolutionary nature and goals of the biological cell may be included as well. It is particularly the case for the biological cells of microorganisms, such as various bacteria and fungi, as well in some other cell types in eukaryotes which strive to optimize the cellular growth, or in some cases energy production (ATP). An adaptive evolutionary nature of the biological cell [42] is mathematically modeled with the maximization of the cellular growth and its respective *biomass* reaction. Biomass reaction is an artificial reaction appended to the metabolic network model and it assures that all of the metabolites necessary for the cellular growth are present in an experimentally determined proportions. It is important to mention that while the cellular growth may be adopted as a valid optimization objective, some other cells such as neuronal or hepatic cells in the human tissue may function to optimize its metabolic network flux distribution dictated by some other biological goal. Hence, with an appropriately selected cellular objective and under the given constraints the metabolic network can be analyzed using various linear and non-linear optimization methods [43].

In principle, optimization methods can be used to analyze the metabolic network and the cellular phenotype of either (1) wild-type or (2) mutant cell. In the later, mutant cell is obtained using genetic modification techniques such as knock-out, knock-in or over-expression of the genes which are responsible for the reactions in the metabolic network. This translates into stricter equality and inequality constraints on the modified reactions in the mutant cell. It is assumed that the mutant cell, following the genetic modifications, is evolved throughout several generations striving to attain optimal i.e. maximal cellular growth. In illustration of the methods to follow, it will be assumed that the metabolic network is given with its stoichiometry matrix $N_{m\times q}$, and the corresponding flux vector with $v = v_{q\times 1}$. In the case when wild-type cell flux vector is contrasted with the mutant cell flux vector, the notation $v^{(wt)}$ and $v^{(mut)}$ will be used, respectively.

2.3.2.1 Flux Balance Analysis

The earliest method to explore the cellular behavior and phenotype was *flux balance analysis* [44, 45, 46, 47, 48, 49]. Assuming that an evolutionary goal for the cellular metabolic network is the optimization of growth, the linear program as given in (2.3) can be formulated.

$$
\begin{aligned}
\max \quad & v_{biomass} \\
\text{subject to} \quad & N \cdot v = 0 \\
& v_{irrev} \geq 0 \\
& v_{substrate} = 1
\end{aligned}
\tag{2.3}
$$

Additional constraint to the linear program is a fixed substrate uptake from the environment (e.g. glucose). Flux balance analysis has somewhat limited power, unless more data can be appended to the model in the form of constraints. For example, gene regulatory network information was integrated with the metabolic network to result in the methods such as rFBA (Regulatory FBA) [50, 51, 52], SR-FBA (Steady state Regulatory FBA) [53], GIMME (Gene Inactivity Moderated by Metabolism and Expression) [31] and PROM (Probabilistic Regulation of Metabolism) [54]. On the other side, MD-FBA (Metabolite Dilution Flux Balance Analysis) [55] framework was proposed to allow the biomass optimization, while accounting for the dilution of all the intermediate metabolites.

2.3.2.2 MOMA and ROOM

When a subset of reaction knockouts were performed in a cell, it was assumed that the mutant cell would strive to minimize its overall flux distribution ($v^{(mut)}$) deviation from the flux distribution in the wild-type cell ($v^{(wt)}$). Optimization procedure named MOMA (Minimization Of Metabolic Adjustment) [56] can be applied to minimize the sum of squared differences between the flux distributions in the wild-type and mutant metabolic networks. This is shown in (2.4) in the formulation of the quadratic optimization problem with linear constraints, where KO is a set of indices of deleted reactions.

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{q} (v^{(wt)} - v^{(mut)})^2 \\
\text{subject to} \quad & N \cdot v^{(mut)} = 0 \\
& v_j^{(mut)} = 0, \quad j \in KO
\end{aligned}
\tag{2.4}
$$

Alternatively, ROOM (Regulatory On/Off Minimization)[57] framework minimizes the number of significant changes of the fluxes in the mutant metabolic network with respect to the wild type network. This heuristic is motivated by assumptions that (1) genetic regulatory changes required by flux changes after the reactions are knocked out are minimized by the cell in order to minimize the adaptation cost and (2) regulatory changes can be described using Boolean on/off dynamics which assigns fixed cost to each regulatory change irrespective of its magnitude. This leads to the formulation of mixed-integer linear programming problem as in (2.5).

$$\begin{aligned}
\text{min} \quad & \sum_{i=1}^{q} y_i \\
\text{subject to} \quad & N \cdot v^{(mut)} = 0 \\
& v_{min,i} \leq v^{(mut)} \leq v_{max,i} \\
& v_j^{(mut)} = 0, \quad j \in KO \\
& v_i^{(mut)} - y_i(v_{max,i} - v_{i,u}^{(wt)}) \leq v_{i,u}^{(wt)} \\
& v_i^{(mut)} - y_i(v_{min,i} - v_{i,l}^{(wt)}) \geq v_{i,l}^{(wt)} \\
& y_i \in \{0,1\} \\
& v_{i,u}^{(wt)} = v_i^{(wt)} + \delta|v_i^{(wt)}| \ + \epsilon \\
& v_{i,l}^{(wt)} = v_i^{(wt)} - \delta|v_i^{(wt)}| \ + \epsilon \\
& i \in \{1..q\} \setminus KO
\end{aligned} \tag{2.5}$$

The flux vectors $v^{(wt)}$ and $v^{(mut)}$ are flux distributions of the wild-type and mutant metabolic networks, respectively. Flux interval $[v_l^{(wt)};v_u^{(wt)}]$ determines the local interval around the wild-type point, which is used to establish if the reaction in mutant network significantly deviates from its wild-type flux value. This local interval is determined using user specified parameters δ and ϵ which in original proposal had the values 0.03 and 0.001 for the study of flux prediction. Deleted reactions are indexed by elements in subset KO. Objective function, a sum of binary variables y_i, minimizes the number of unconstrained reaction fluxes which can significantly deviate from their corresponding flux values in the wild-type metabolic network. This is the case if $y_i = 1$ which leads to no additional constraints on the flux $v_i^{(mut)}$. On the other side, if $y_i = 0$ the reaction $v_i^{(mut)}$ can not significantly deviate from its respective wild-type value $v_i^{(wt)}$. One of the disadvantages of this methods is the need to specify the parameters δ and ϵ.

2.3.2.3 Flux Variability Analysis

There may be multiple different flux distributions in the metabolic network which carry the optimal value of the given cellular objective e.g. biomass. If a maximal biomass flux is computed ($v_{biomass}^{(max)}$), it may then be appended as an equality constraint across multiple linear programs which all aim to determine possible flux range of remaining reactions. This procedure is known as FVA (Flux Variability Analysis) [58] and is shown in (2.6).

$$\begin{aligned}
&\text{for } i \in \{1..q\}: \\
&\max/\min \quad v_i \\
&\text{subject to} \quad N \cdot v = 0 \\
&\qquad v_{biomass} = v_{max,biomass} \\
&\qquad v_{min,j} \leq v_j \leq v_{max,j} \quad j \neq i
\end{aligned} \tag{2.6}$$

2.3.2.4 OptKnock

Optimal production of the target chemical in a metabolic network of the wild-type microorganism is usually not coupled to the cellular growth, which results in very low target yield in such wild-type strains. One approach is to perform genetic modification by knocking out or changing the expression range of one or more reactions in the metabolic network, thus enforcing the coupling of the optimal biomass flux to the high target chemical production. Once the cell is genetically modified, the mutant strain evolves throughout several generations, as through the natural selection it will counteract any genetic or environmental perturbations by redirecting its metabolic flux to restore the optimal cellular growth [59]. If the genetic modification was rationally selected, the optimal cellular growth will be accompanied by optimal or sub-optimal chemical production. The earliest method to attain this goal was OptKnock [60, 61], a framework proposed in the form of a bi-level mixed-integer linear programming problem, outlined in (2.7).

$$\begin{aligned}
\max_{y} \quad & v^{(mut)}_{chemical} \\
\text{subject to} \quad & \max_{v} \quad v^{(mut)}_{biomass} \\
& \text{subject to} \quad N \cdot v^{(mut)} = 0 \\
& \qquad v^{(mut)}_{biomass} \geq \gamma v^{(wt)}_{max,biomass} \\
& \qquad v_{min,j} \cdot y_j \leq v^{mut}_j \leq v_{max,j} \cdot y_j \\
& y_j \in \{0,1\} \\
& \sum_{j=1}^{q} (1 - y_j) \leq K
\end{aligned} \tag{2.7}$$

Parameter γ (usually =0.05-0.1) determines the required minimum flux which the biomass reaction should carry as a fraction of the maximum possible value ($v^{(wt)}_{max,biomass}$) in the wild-type strain. The bi-level mixed integer linear programming problem (2.7) can be transformed into a one-level mixed-integer linear program and then solved using appropriate solver. Basically, for the inner maximization problem its dual problem is derived, and using the linear programming duality theory the values of objective functions of the two problems coincide in the optimal point. Hence, this results in a one-level mixed-integer linear programming problem which maximizes the chemical reactions and contains an augmented set of constraints. The number of reaction deletions is constrained with the parameter K, where $y_j = 0$ denotes the inactivated reaction.

2.3.2.5 RobustKnock

A major disadvantage of the OptKnock framework was the existence of competing pathways with uncoupled production of the chemical with biomass. In other words, existence of pathways with maximal biomass, but very low or zero target chemical flux value was possible. This problem was addressed in a modified version of OptKnock, named RobustKnock [62], which aims to maximize the minimum chemical flux value when coupled to the maximal biomass reaction flux.

$$\begin{aligned}
\max_{y} \quad & \min_{v} \quad v^{(mut)}_{chemical} \\
\text{subject to} \quad & \max_{v} \quad v^{(mut)}_{biomass} \\
& \text{subject to} \quad N \cdot v^{(mut)} = 0 \\
& \qquad\qquad v^{(mut)}_{biomass} \geq \gamma v^{(wt)}_{max,biomass} \\
& \qquad\qquad v_{min,j} \cdot y_j \leq v^{mut}_j \leq v_{max,j} \cdot y_j \\
& y_j \in \{0,1\} \\
& \sum_{j=1}^{q} (1 - y_j) \leq K
\end{aligned} \tag{2.8}$$

Similarly as in OptKnock, the outer max-min problem searches for the reaction knockout subset of the size not larger than K, while the inner problem is the flux balance analysis which optimizes biomass flux for the given knockout combination. A two-step transformation procedure is applied, where in the first step the problem in (2.8) is transformed into a one-level max-min problem. Following, in the second step, the one-level max-min problem is transformed into a standard mixed-integer linear programming problem.

2.3.2.6 OptStrain

Aside from performing reaction deletions, genetic modifications may also include addition e.g. knock-in of non-native reactions into the wild-type cell and its genome. An optimization-based framework, *OptStrain* [63], identifies non-native candidate reactions for addition from the precompiled database of elementally balanced reactions. Optimization objective in this framework is to maximize target chemical yield and minimize the usage of non-native reactions.

2.3.2.7 OptReg

To allow knockout, over-expression and under-expression of reactions in a metabolic network, the OptReg [64] framework was proposed. For the reaction j in the network, its allowed flux value range $[v_{min,j}, v_{max,j}]$ can be easily determined. However, the determination of the interval $[v^{(wt)}_{min,j}, v^{(wt)}_{max,j}]$ corresponding to the wild-type flux value range requires experimental measurements which are appended as equality constraints to the min/max linear programming formulation for each reaction. Once the intervals $[v_{min,j}, v_{max,j}]$ and $[v^{(wt)}_{min,j}, v^{(wt)}_{max,j}]$ are determined, OptReg formulation can be given as in (2.9).

$$\begin{aligned}
\max_{y_j^K, y_j^U, y_j^D} \quad & v^{(mut)}_{chemical} \\
\text{subject to} \quad & \max_{v} \quad v^{(mut)}_{biomass} - \epsilon \cdot \sum_j v_j^{(mut)} \\
& \text{subject to} \quad N \cdot v^{(mut)} = 0 \\
& \qquad v^{(mut)}_{biomass} \geq \gamma v^{(wt)}_{max,biomass} \\
& \qquad v_{min,j} \cdot y_j^k \leq v_j^{(mut)} \leq v_{max,j} \cdot y_j^k \\
& \qquad v_j^{(mut)} \leq [v^{(wt)}_{min,j}(1-C) + v_{min,j}C] \cdot (1 - y_j^d) + v_{max,j} \cdot y_j^d \\
& \qquad v_j^{(mut)} \geq [v^{(wt)}_{max,j}(1-C) + v_{max,j}C] \cdot (1 - y_j^u) + v_{min,j} \cdot y_j^u \\
& (1 - y_j^k) + (1 - y_j^u) + (1 - y_j^d) \leq 1, \quad \forall i \in \{1 \dots q\} \\
& y_j^k \in \{0,1\}; y_j^u \in \{0,1\}; y_j^d \in \{0,1\} \\
& \sum_j (1 - y_j^k) + (1 - y_j^u) + (1 - y_j^d) \leq K
\end{aligned} \tag{2.9}$$

Regulation strength parameter C has values in interval $[0;1]$ and determines the fraction of the range to which down-regulated and up-regulated flux belongs. The higher the value of C more will the reaction deviate from its steady state value. Parameter K is the maximal number of reactions which can be modified (deleted, up-regulated, down-regulated). Inner problem which maximizes biomass flux has within its objective the second term which assures that the selected solution has the minimal flux distribution sum among alternative optimal solutions, where ϵ has a value determined through a trial-and-error process.

2.3.2.8 OptGene

Due to the combinatorial nature of the finding of the knockout subset which is inherent to the frameworks such as OptKnock, as well as the NP-hardness of the mixed-integer linear programming problem, alternative search algorithm based on the evolutionary programming was proposed. Such is the OptGene [65] framework which optimizes proposed linear or non-linear objective function. Algorithm can be illustrated as in the steps that follow:

- **Step 1.** *initialization of population*: Genetic algorithm starts off with a predefined number of individuals, each individual having assigned a profile(chromosome) of present and knocked-out reactions. An individual profile has an assigned vector which indicates with binary values which reactions are present or absent.

- **Step 2.** *scoring*: Fitness score is computed for each individual using the flux distribution obtained when one of the frameworks such as FBA, MOMA or ROOM are run. Value of fitness score, which can be linear or non-linear, is used to decide if an individual will be rejected or retained for use in the next generation. Score can be computed as a BPCY (Biomass-Product Coupled Yield) as in (2.10) thus supporting the non-linear objective.

$$BPCY = \frac{v_{chemical}^{(mut)} \cdot v_{biomass}^{(mut)}}{v_{substrate}} \tag{2.10}$$

  where the substrate flux retains fixed value.

- **Step 3.** *crossover of chromosomes*: The highest scored individuals from the previous step are selected for the crossover. The crossover method used can be one-point, two-point or a uniform crossover.

- **Step 4.** *mutation*: Mutation corresponds to the deletion of a reaction. With a specified mutation probability the individuals are propagated to the new population.

- **Step 5.** *new population and termination*: New population of individuals was created from steps 3 and 4. Population may be subject to iterative execution of steps 2-5 until an individual with desired phenotype characteristics is found.

2.3.2.9 Set-Based Evolutionary Algorithm and Simulated Annealing

Major disadvantage of the OptGene framework is that it requires to beforehand specify the number of reaction knockouts. Introducing the set-based representation of a chromosome of an individual allowed finding variable-size reaction knockout solutions. Two algorithms which include this feature are SEA (Set-based Evolutionary Algorithm) and SA (Simulated

Annealing) [66], where the SEA is a successor to the OptGene. Both SEA and SA algorithms have the Biomass Product-Coupled Yield (2.10) as the fitness score function which is used to evaluate individuals for inclusion in the next generation.

Set-Based Evolutionary Algorithm is illustrated in the following itemized steps:

- **Step 1.** *initialization of population*: The population is initialized similarly as in OptGene algorithm. Specify minimum and maximum reaction knockout set size.

- **Step 2.** *scoring*: Fitness score is assigned to each individual using the value of the BPCY objective function. Objective function value is calculated after running FBA, MOMA or ROOM algorithm, which are outlined in earlier subsections.

- **Step 3.** *crossover of chromosomes*: Crossover of chromosomes on selected individuals is performed as in OptGene. In addition to using crossover and mutation on the selected individuals, in order to allow the variable-sized sets of gene deletions to compete between each other, two operators *grow* and *shrink* are introduced, which with the probability of 5% each are being selected for the run.

- **Step 4.** *mutation*: Similarly as in OptGene, mutations (grow and shrink operators here) are introduced in individuals with some given probability. For example, with the probability 5% for the run of either grow or shrink mutation each, an individual is mutated which allows the exploration of knockout sets of varying size. Grow and shrink mutation operators randomly add or remove elements from the working reaction knockout set.

- **Step 5.** *new population and termination*: New population of individuals was created from Steps 3 and 4. Population may be subject to iterative execution of steps 2-5 until an individual with desired phenotype characteristics is found.

Simulated Annealing algorithm is based on the idea of exploring the solution space and navigating towards the global optimum. Single individual chromosome is observed with the variable-sized set of reaction knockouts. During exploration, if a solution is better it is accepted, while worse solutions are accepted with some small probability in order to avoid ending in a local optima. Mutation operators are defined as in Set-Based Evolutionary Algorithm with the probability of 25% of applying grow and shrink operators each. Simulated Annealing differs from Set-Based Evolutionary Algorithm in the use of fitness score function which here corresponds to the system's energy.

- **Step 1.** *initialization*: Input parameters such as initial (T_0) and final (T_f) temperature, number of iterations performed at each temperature value ($trial$), and cooling schedule are specified. Cooling schedule specifies how the temperature is decreased from T_0 to T_f. After each temperature decrease step, *trial* number of iterations are performed. After specifying the input parameters, an initial individual with its chromosome is specified.

- **Step 2.** *mutation*: Individual's chromosome is mutated using grow and shrink operators as in the Set-Based Evolutionary Algorithm.
- **Step 3.** *cooling*: Decrease the current temperature T_n using the schedule $T_{n+1} = \alpha T_n$ using prespecified parameter α. For the new temperature value perform a given number of evaluations, while the difference between fitness values of the new and old individual is represented as ΔE, change of system's energy. Always accept a better solution or a worse solution with the probability $p_{accept} = e^{-\frac{\Delta E}{T}}$.
- **Step 4.** Repeat steps 2-3 until final temperature T_f is reached.

2.3.2.10 Genetic Design through Local Search

One workaround to attenuate the high cost of the bi-level mixed-integer linear programming methods in the case of the increasing number of allowed genetic interventions was proposed in the GDLS (Genetic Design through Local Search) [67] framework. It is assumed that the algorithm starts with an initial reaction knockout set having not more than K elements. At every algorithm iteration, each solution is modified M times by adding or removing reactions as candidates for knockout, so that the new solution does not differ from the previous by more than k deleted reactions. At the end of every iteration, top M solutions are selected to be used in the next iteration.

- **Step 1.** Start with an initial q-vector $y^{(0)}$ with indicators of reaction knockouts, where q is the number of reactions in the network. Let the initial solutions set be $Y^{(i)} = \{y^{(i)}\}$ where $i = 0$.
- **Step 2.** For every knockout vector $\tilde{y} \in Y^{(i)}$, solve the bi-level optimization problem (equation 2.11) with additional constraint given in (2.12) which denotes that sets $\tilde{y}$ and y^* should not differ by more than k knocked-out reactions, where y^* denotes the optimal solution found for this MILP program. The solution set of for the next iteration may be initialized to $Y^{(i+1)} = \{y^*\}$

$$\begin{aligned}
\max_{y_j} \quad & v^{(mut)}_{chemical} \\
\text{subject to} \quad & \max_{v} \quad v^{(mut)}_{biomass} \\
& \text{subject to} \quad N \cdot v^{(mut)} = 0 \\
& \qquad\qquad v^{(mut)}_{biomass} \geq \gamma v^{(wt)}_{max,biomass} \\
& \qquad\qquad (1-y_j)v_{min,j} \leq v^{(mut)}_j \leq (1-y_j)v_{max,j}, \quad j \in \{1..q\} \\
& \sum_{j=1}^{q} y_j \leq K, \quad y_j \in \{0,1\}
\end{aligned} \tag{2.11}$$

- **Step 3.** Now take the last appended element in $Y^{(i+1)}$ and use it as $\tilde{y}$ to solve the MILP problem in (2.11) with both additional constraints (2.12) and (2.13). Newly obtained optimal solution is y^* and it can be added to the set $Y^{(i+1)}$. This will assure that a new distinct solution is looked up which does not differ from the current one by more than k knocked-out reactions. After running the MILP problem $M-1$ times, set $Y^{(i+1)}$ will have not more than $Y^{(i)} \cdot M$ elements. Top M solutions from $Y^{(i+1)}$ should be retained for the next iteration.

$$\sum_{j:\tilde{y}_j=0} y_j + \sum_{j:\tilde{y}_j=1} (1-y_j) \leq k \tag{2.12}$$

$$\sum_{j:\tilde{y}_j=0} y_j + \sum_{j:\tilde{y}_j=1} (1-y_j) \geq 1 \tag{2.13}$$

- **Step 4.** Repeat the steps 2 and 3 until $Y^{(i)} = Y^{(i+1)}$ or until maximal number of iterations is reached.

2.3.2.11 Flux Scanning based on Enforced Objective Flux (FSEOF)

One strategy which rather than limit itself only to reaction deletions, also identifies the reaction over-expression targets in the metabolic network, is proposed in the FSEOF (Flux Scanning based on Enforced Objective Flux) [68]. Let $v^{(init)}$ be the flux vector obtained when a simple FBA is solved as in (2.3) maximizing the biomass reaction. The value of the flux of the target chemical reaction at this point is $v^{(init)}_{chemical}$, while maximum possible chemical flux value is $v_{max,chemical}$ as found solving an appropriate FBA problem. The reaction amplification targets are being selected as those reactions which increase their flux value v_i without changing their direction during at least one iteration period, as the $v_{chemical}$ value goes from $v^{init}_{chemical}$ to $v^{max}_{chemical}$ in uniformly increasing steps.

2.3.2.12 OptORF

Up until this point and for the sake of simplicity, all of the earlier outlined methods abstracted the transcriptional gene regulatory network and assumed that deletion, over- and under- expression are applied on reactions, rather than on underlying genes. However, if the gene regulatory network information is known and analyzed together with the metabolic network, one may be able to pinpoint which genes in the genome should be modified to influence the enzyme proteins which control the rate of reactions. OptORF is a bi-level optimization framework [69] which extends OptKnock to incorporate the GER (gene-enzyme-reaction) associations. Let GER be a three-dimensional array where a triplet (d_j, b_n, y_g) consists of binary variables representing the reaction (d_j), enzyme (b_n) and gene (y_g). These binary variables denote if gene g is expressed, so that the enzyme n is active inducing a non-zero flux in the reaction j. Denote with J_{GER} a set of indices j of reactions which appear as triplets in GER, $N(j)$ be a set of indices n of enzymes which are associated with reaction j in GER, and $G(n)$ be a set of indices of genes g which are associated with the enzyme n in GER.

While accounting for GER associations, one may step in further to look at the metabolic and transcription factor genes with respective sets G_{MET} and G_{TF}. Metabolic genes are directly responsible for the GER associations, while the transcription factor genes influence the gene expression of the metabolic genes. Further, let there be conditions and effectors (activators and repressors) which influence the expression of both metabolic and transcription factor genes. A condition for the expression of the gene m, here indicated using the binary variable a_m, can be influenced by activator or repressor r, denoted here using the binary variable x_r. Sets of activators and repressors which may influence condition m for the expression of one or more genes are denoted as $R^{Act}(m)$ and $R^{Rep}(m)$, respectively. Binary variables z_g and w_g denote if the gene g was deleted or over-expressed, allowing not more than K_1 deletions and K_2 over-expressions in the cell. A single gene can not be both deleted and over-expressed at the same time. OptORF framework is outlined in (2.14).

$$
\begin{aligned}
\max \quad & v_{chemical}^{(mut)} - \alpha \sum_{g} z_g - \beta \sum_{g} w_g \\
\text{subject to} \quad & d_j \geq b_n \qquad \forall j \in J_{GER}, n \in N(j) \\
& d_j \leq \sum_{n \in N(j)} b_n, \qquad \forall j \in J_{GER} \\
& y_g \geq a_m, \qquad \forall g \in G_{MET} \cup G_{TF} \\
& y_g \leq \sum_{m \in M(g)} a_m, \qquad \forall g \in G_{MET} \cup G_{TF} \\
& (1 - a_m) \leq \sum_{r \in R^{Act}(m)} (1 - x_r) + \sum_{r \in R^{Rep}(m)} x_r, \forall m \in M \\
& a_m \leq x_r, \qquad \forall m \in M, r \in R^{Act}(m) \\
& a_m \leq (1 - x_r), \qquad \forall m \in M, r \in R^{Rep}(m) \\
& z_g - w_g = y_g - y'_g, \qquad \forall g \in G_{MET} \\
& z_g = y_g - y', \qquad \forall g \in G_{TF} \\
& z_g + w_g \leq 1, \qquad \forall g \in G_{MET} \\
& \sum_{g} z_g \leq K_1 \quad \text{and} \quad \sum_{g} w_g \leq K_2 \\
& (b_n - 1) \geq \sum_{g \in G(n)} (y'_g - 1), \qquad \forall n \in N \\
& b_n \leq y'_g, \qquad \forall m \in M, g \in G(n) \\
& x_g = y'_g, \qquad \forall g \in G_{TF} \\
& \max \quad v_{biomass}^{(mut)} \\
& \text{subject to} \quad N \cdot v^{(mut)} = 0 \\
& \qquad v_{min,j} \leq v_j^{(mut)} \leq v_{max,j} \\
& \qquad v_j^{(mut)} = 0, \qquad \text{if} \quad d_j = 0, \forall j \in J_{GER} \\
& d_j, b_n, w_g, z_g, y_g, y'_g, a_m, x_r \in \{0, 1\}
\end{aligned}
\tag{2.14}
$$

OptORF framework was applied on the integrated metabolic [70] and regulatory [71] networks of the *Escherichia coli*.

2.3.2.13 OptForce

OptForce [72] is a framework which identifies reaction deletions, over- and under- expressions in the given metabolic network (wild-type network) to obtain the mutant metabolic network (over-producing network) which produces the target chemical metabolite with the specified minimal amount. As this framework aims to allow not just reaction deletions, but also over- and under- expressions it requires that experimental flux measurements are available for some of the network reactions which determine the wild-type reaction flux distribution. Main idea is to determine minimal set of reaction shifts within their flux range either in decreasing or increasing direction, so that the flux distribution of the over-producing network is attained. For example, if the flux range of reaction j in wild-type and over-producing network do not overlap, a shift will be required. Sets $MUST^U$, $MUST^L$ and $MUST^X$ contain indices of reactions which require respective interventions of over-expression, under-expression or deletion. Due to intensive coupling among reactions, modification of flux in one reaction may induce a change of flux in a set of other reactions. Hence, an idea of observing pairwise sums and differences across reactions in the network lead to the formation of $MUST^{UL}$, $MUST^{UU}$ and $MUST^{LL}$ sets. For example, $MUST^{UL}$ contains a pair of reactions (v_i, v_j) which requires either the over-expression of v_i or the under-expression of v_j. Once the sets $MUST^U$, $MUST^L$, $MUST^X$, $MUST^{UL}$, $MUST^{UU}$ and $MUST^{LL}$ are determined, the next step is to enumerate $FORCE$ sets where a single such set contains minimal number of reactions selected across $MUST$ sets which meet the over-production goal. It is possible to identify a subset of reactions in $FORCE$ sets which will when modified induce modifications in the remaining reactions as well. Following steps illustrate the OptForce framework.

- **Step 1.** *Identifying flux range of reactions for the wild-type strain.* Solving $2q$ linear problems given in (2.15), a range of reaction fluxes in the wild-type network can be determined. It is assumed that it was possible to obtain experimental flux range for the subset of reactions indexed by elements of the subset in E.

$$\begin{aligned}
&\text{for} \quad i \in \{1..q\}: \\
&\quad \text{max/min} \qquad v_i \\
&\quad \text{subject to} \qquad N \cdot v = 0, \\
&\qquad\qquad\qquad v_{biomass} \geq \gamma v_{max,biomass} \\
&\qquad\qquad\qquad v_{min,j} \leq v_j \leq v_{max,j} \quad \forall j \in \{1 \ldots q\} \setminus biomass \\
&\qquad\qquad\qquad v^{(exp)}_{min,j} \leq v_j \leq v^{(exp)}_{max,j} \quad \forall j \in E
\end{aligned} \tag{2.15}$$

  The range of fluxes obtained solving this collection of linear programs corresponds to the wild-type strain and is used in the following step.

- **Step 2.** *identifying flux range of reactions for the over-producing (mutant) strain.*. Similarly as in previous step, $2q$ linear problems in (2.16) are solved to determine flux ranges in the over-producing network for the fixed lower bound of the target chemical.

$$
\begin{aligned}
&\text{for} \quad i \in \{1..q\}: \\
&\quad \max/\min \quad && v_i \\
&\quad \text{subject to} \quad && N \cdot v = 0 \\
& && v_{biomass} \geq \gamma v_{biomass}^{(max)} \\
& && v_{chem} \geq v_{chem}^{(target)} \\
& && v_{min,j} \leq v_j \leq v_{max,j} \quad \forall j \in \{1 \dots q\}
\end{aligned}
\tag{2.16}
$$

- **Step 3.** *Identifying $MUST^U$, $MUST^L$ and $MUST^X$ sets of reactions.* Determine sets of reactions which have to be respectively over-expressed, under-expressed or deleted to get closer to the flux distribution of the over-producing network.

- **Step 4.** *Eliminate reactions in $MUST^U$, $MUST^L$ and $MUST^X$ from consideration in higher order $MUST$ sets.* Reactions to be considered for higher order $MUST$ sets will exclude the reactions in the first-order sets.

- **Step 5.** *Identifying $MUST^{UU}$, $MUST^{UL}$ and $MUST^{LL}$ sets of reactions.* Three distinct bi-level mixed-integer linear programming problems are proposed for each of the three second-order $MUST$ sets.

- **Step 6.** *identifying the $FORCE$ set of engineering interventions.* Using appropriate bi-level mixed-integer linear programming problem which minimizes the total number of reaction interventions while minimizing the chemical production, under the earlier constraint that chemical production is already pushed to the desired flux range, the FORCE set is enumerated.

Finally, OptForce framework uses integer cut enumeration methods earlier exploited in the multiple-gene lethality analysis [73] to enumerate multiple combinations of reaction interventions.

2.3.2.14 SimOptStrain and BiMOMA

The proposal of the original OptStrain (subsection 2.3.2.6) framework required a prior identification of the non-native pathway and reactions which over-produce the target chemical of interest as candidates for the knock-in into the metabolic network, followed by the optimization of the target chemical and biomass reactions in a bi-level mixed-integer linear programming problem analogous to OptKnock. This decoupling was resolved in the SimOptStrain framework [74] where a simultaneous search for both non-native reactions for knock-in and the native reactions for knock-out is performed. Again, as in OptStrain a universal database of reactions is compiled from KEGG and MetaCyc data repositories and is used for the selection of knock-in candidate reactions.

$$\begin{aligned}
\max_{y_i, z_l} \quad & v_{chemical} - \alpha \sum_{i=1}^{q} (1 - y_i) - \beta \sum_{l=q}^{s} z_l \\
\text{subject to} \quad & \max_{v} \qquad v_{biomass} \\
& \text{subject to} \qquad N \cdot v + T \cdot w = 0 \\
& \qquad\qquad\qquad v_j = 0, \qquad \text{if} \quad y_j = 0 \\
& \sum_{i=1}^{q} (1 - y_i) \leq K \quad \text{and} \quad \sum_{l=1}^{s} z_l \leq K'
\end{aligned} \tag{2.17}$$

SimOptStrain framework is illustrated in (2.17), where binary variables y_i and z_l denote if the reaction i is knocked out ($y_i = 0$) or if non-native reaction l is added to the network ($z_l = 1$), respectively. Values K and K' denote the maximal allowed number of knock-out and knock-in reactions in the metabolic network, while s denotes the size of the knock-in candidate database.

One more framework was resurrected and incorporated into the bi-level mixed-integer linear programming problem. Aimed for the reaction flux prediction in the non-evolved mutant strain, MOMA was incorporated into the framework named BiMOMA [74]. BiMOMA sets as an outer problem the optimization of the target chemical reaction, while optimizing the inner quadratic programming problem as shown in (2.18).

$$\begin{aligned}
\max_{y_i} \quad & v_{chemical}^{(mut)} - \alpha \sum_{i} (1 - y_i) \\
\text{subject to} \quad & \min \quad \sum_{i=1}^{q} (v^{(wt)} - v^{(mut)})^2 \\
& \text{subject to} \qquad N \cdot v^{(mut)} = 0 \\
& \qquad\qquad\qquad v_j^{(mut)} = 0, \qquad \text{if} \quad y_j = 0 \\
& \sum_{i=1}^{q} (1 - y_i) \leq K
\end{aligned} \tag{2.18}$$

An almost identical framework MOMAKnock [75] was proposed which converts the bi-level mixed-integer quadratic programming problem into a single-level problem using adaptive piecewise linearization strategy.

2.3.3 Pathway-based Analysis Methods

If one considers only the two constraints given in the definition 1, without the specifying the cellular objective the optimization-based methods yield place to the so-called pathway-based analysis. Pathway-based methods are capable of characterizing the entire solution space of the possible metabolic network states without imposing the cellular objective bias. Contrary to this, optimization based methods are guided by the cellular objective, and capable of exploring only a portion of the entire solution space. Metabolic pathway solution space can be confined using an algorithm for the *enumeration of extreme rays in the bounded polyhedron* where the bounded polyhedron corresponds to the degenarate polytope as will be illustrated in more details in the upcoming subsection.

2.3.3.1 Extreme Ray Enumeration and Double Description Method

The theory used to enumerate the extreme rays in the bounded polyhedral cone can be applied to find all the metabolic network states. First, a double description pair as a fundamental building block for the Double Description Method will be defined. The stoichiometry problem will then be mapped to the Double Description Method in order to demonstrate the equivalence between the geometric concept of extreme rays and the metabolic pathways.

Definition 2 (Double Description Pair [76])**.** Given a $d \times q$ matrix A with full column rank q, and a $q \times n$ matrix R, A and R form a Double Description Pair (DD pair) if and only if

$$\{\mathbf{r} : A\mathbf{r} \geq 0\} = \{\mathbf{r} : \mathbf{r} = R\boldsymbol{\lambda}, \boldsymbol{\lambda} \geq 0\} \tag{2.19}$$

The DD pair is called minimal if there is no other $\widetilde{R}$ forming a DD pair with A having fewer than n columns. □

Here, the columns of R form a set of generators for all the $\mathbf{r}$ that satisfy $A\mathbf{r} \geq 0$, where all such $\mathbf{r}$ are to be expressed as convex combinations of the columns of R.

Definition 3 (Admissible Ray of a Polyhedral Cone [76])**.** For a polyhedral cone given by a matrix $A_{d\times q}$ the $q \times 1$ vector $\mathbf{r}$ such that $A\mathbf{r} \geq 0$ is called an "admissible ray". □

Definition 4 (Support Set of Admissible Ray)**.** Elements of admissible ray r vector with non-zero values are known as "support set". □

A subset of admissible rays which coincide with the edges of an infinite polyhedral cone in $\mathbb{R}^q$ forms "extreme rays" as given in Definition 5.

Definition 5 (Extreme Ray of a Polyhedral Cone [76])**.** Let a polyhedral cone be given by a matrix $A_{d\times q}$ with full column rank q. An admissible ray given as vector $r_{q\times 1}$ is called an "extreme ray" if it cannot be represented as a linear combination of other admissible rays. □

Following from the previous definition, given that generating matrix R is minimal, the columns of R coincide with the extreme rays, because these are exactly the rays that cannot be expressed as a convex combination of any other admissible rays. As a result, the R forming a DD pair with A is unique up to ordering and scaling of the columns.

If A has rank less than q, then one can still define a minimal DD pair, but in this case, not all of the columns of R are extreme rays and the minimal R is no longer unique. In this case, there exists at least one ray $\mathbf{r}$ such that $-\mathbf{r}$ is also a valid ray within the cone, namely any nonzero vector in the right nullspace of A. This corresponds to a non-pointed cone.

Theorem 1 (Extreme Ray Theorem [76])**.** *Let $A_{d\times q}$ be a q-rank matrix. A q-vector $\mathbf{r}$ is an extreme ray for A if and only if the rank of $A_{\mathtt{Z}(A\mathbf{r}),*}$ is $q-1$, where $\mathtt{Z}(A\mathbf{r})$ is the set of indices of the zero entries in the vector $A\mathbf{r}$, or equivalently, the nullity of $A_{\mathtt{Z}(A\mathbf{r}),*}$ is 1 [the nullity is the dimension of the right nullspace]. Here, the notation $A_{\mathtt{Z}(\mathbf{v}),*}$ means the matrix resulting from selecting the* rows *corresponding to the indices $\mathtt{Z}(\mathbf{v})$. Hence the set of extreme rays are uniquely defined up to scale factors.*

Standard Stoichiometry Problem as a Double Description Pair. The rank test for the property of a vector being an extreme pathway depends on the close connection between the stoichiometry problem equation (2.2) and the double description pair (Def. 2), which is sketched here. Given an $m \times q$ stoichiometry matrix N, let

$$A = \begin{pmatrix} N \\ -N \\ E \end{pmatrix} \tag{2.20}$$

where $E = I_{1,\ldots,q_i,*}$ consists of the first q_i rows of a $q \times q$ identity matrix, corresponding to the irreversible reactions $(\rho_1, \ldots, \rho_{q_i})$, where q_i is the number of irreversible reactions. Then the stoichiometry problem is equivalent to finding a matrix R with minimal number of columns such that

$$\begin{aligned} \{\mathbf{r} : A\mathbf{r} \geq 0\} &= \{\mathbf{r} : N\mathbf{r} = 0, \mathbf{r}_{1,\ldots,q_i} \geq 0\} \\ &= \{\mathbf{r} : \mathbf{r} = R\boldsymbol{\lambda}, \boldsymbol{\lambda} \geq 0\}. \end{aligned} \tag{2.21}$$

In the case that A has full column rank q, R should consist of all the possible extreme rays with respect to matrix A. In the case that all reactions are irreversible, then the set of extreme rays corresponds to the metabolic pathways in the stoichiometric network and is also known as the elementary flux modes as will be illustrated later.

However, if some of the reactions are reversible then the system is expanded into higher dimensionality by splitting reversible reactions into two irreversible reactions each. In this case the matrix E is a full identity matrix, and such expanded system can be run through the Double Description Method to compute the set of extreme rays of the polyhedral cone. The obtained extreme rays are mapped back to the original dimensionality and there they correspond to the so-called elementary flux modes of the stoichiometric problem which will be studied in continuation. It is important to note that in this case, in the presence of one or more reversible reactions, the rays obtained in the elementary flux modes are not all extreme with respect to the original polyhedral cone, and a subset of them forms minimal generating rays i.e. extreme rays. The extreme rays in the polyhedral cone is known as the minimal generating set in the domain of the stoichiometric problem. In section 3.4 we will illustrate that in the event when there are as many reversible reactions to allow the existence of the non-pointed polyhedral cone the set of extreme rays is not unique and hence the same stands for the minimal generating set.

General Double Description Method. The algorithm to compute an R forming a Double Description pair [DD pair] (2.19), starting with the matrix A, proceeds in a recursive manner. Let A_k denote the matrix consisting of the first k rows of A, where the ordering of the rows is arbitrary. Suppose A_k, R_k form a DD pair. The recursive process then proceeds to construct a DD pair A_{k+1}, R_{k+1}, where A_{k+1} is formed by appending the $(k+1)^{st}$ row of A to A_k, and R_{k+1} is formed by taking all possible valid non-negative combinations of columns of R_k. The heart of the recursive algorithm consists of specifying the details of how R_{k+1} is constructed from R_k, proving that the result indeed forms a DD pair with A_{k+1}, and finding a proper way to initialize the algorithm.

Lemma 1 (DD Lemma [76])**.** Let A_k denote the matrix consisting of the first k rows of the $d \times q$ matrix A. Any extreme vector with respect to A_{k+1} is a non-negative combination of at most two extreme vectors with respect to A_k. ∎

Naive DD Algorithm. Suppose we have an initial DD pair (A_{k_0}, R_{k_0}), for some initial value of k_0, the DD Lemma gives a way to compute a DD pair for the entire matrix A.

For $k = k_0, \ldots, d-1$,

1. Form A_{k+1} appending the $(k+1)^{st}$ row of A to A_k.

2. For every pair of columns $\mathbf{r}_1$, $\mathbf{r}_2$ of R_k (extreme vectors *wrt* A_k), form a non-negative combination $\mathbf{s} = \alpha_1\mathbf{r}_1 + \alpha_2\mathbf{r}_2$ such that $[A \cdot \mathbf{s}]_{k+1} = 0$. This is possible exactly when $[A \cdot \mathbf{r}_1]_{k+1}$ and $[A \cdot \mathbf{r}_2]_{k+1}$ are both nonzero and have opposite signs (see remark at the end of the proof of the DD Lemma).

3. Check that $\mathbf{s}$ is extreme *wrt* A_{k+1}, either by checking that $\mathsf{rank}(A_{k+1,\,\mathsf{Z}(\mathbf{s})}) = q-1$, or by checking that the set of indices $\mathsf{Z}(\mathbf{s})$ (set of indices of the zero entries in $A_{k+1} \cdot \mathbf{s}$) is not a subset of the corresponding set of zero indices for any existing column of R_k. Discard any vectors $\mathbf{s}$ failing this test.

4. Every column $\mathbf{r}$ of R_k such that $A_{k+1}\mathbf{r} \geq 0$ is already an extreme vector *wrt* A_{k+1}. So collect all columns $\mathbf{r}$ satisfying this condition, together with all vectors $\mathbf{s}$ found to be extreme *wrt* A_{k+1} in the previous step, to form R_{k+1}. The resulting R_{k+1} forms a DD pair with A_{k+1}. (Actually, if any q rows of A are linearly independent, then $A_{k+1}\mathbf{r} = 0$ cannot occur.)

At the end of this iteration, R_d will form a DD pair *wrt* A. It remains to figure out how to initialize the iteration. In the special case of (2.20), it is easy to construct an initial DD pair in two possible ways.

1. In the case that E block of (2.20) is a complete $q \times q$ identity matrix (i.e., all reactions are irreversible), then we can select the rows corresponding to the E in (2.20) by using the $R = I_{q\times q}$. The initial DD pair is $(E, I_{q\times q})$, and the recursive steps are used to enforce the remaining conditions $N\mathbf{r} \geq 0$, $-N\mathbf{r} \geq 0$. During the recursive algorithm, we add each row of N and the same row from $-N$ together to enforce the condition $N\mathbf{r} = 0$ directly. The resulting algorithm is the *Canonical Basis Algorithm* [77].

2. We can let the initial R be a basis for the nullspace of N, such that the first $q-m$ rows of R form an identity matrix. This is equivalent to setting the initial A_{k_0} to consist of all of the N and $-N$ parts of (2.20) plus the first $q-m$ rows of the E part. The recursive steps are used to enforce the non-negativity conditions *wrt* the remaining rows of E. This initialization works even when E is not a complete identity matrix, as long as E contains at least $q-m$ rows from the complete identity matrix, and the initial rank condition is still satisfied:
$$\mathsf{rank}(A_{k_0}) = \mathsf{rank}\begin{pmatrix} N \\ -N \\ I_{q-m\times q-m},\ 0 \end{pmatrix} = \mathsf{rank}\begin{pmatrix} N_1 & N_2 \\ -N_1 & -N_2 \\ I_{q-m\times q-m} & 0 \end{pmatrix} = q.$$
This is equivalent to the condition that the $m \times m$ submatrix N_2 is non-singular. The reactions (columns of N) may need to be permuted to meet this conditions. In terms of the stoichiometry, this works even if there are some reversible reactions, as long as there are at most m reversible reactions, and one can find m reactions, which must include all the reversible reactions, such that the stoichiometry *wrt* those reactions is "linearly independent." This leads to the *Nullspace Algorithm* [78, 79].

As we transition between the convex analysis and the stoichiometry domain, the two cones are distinguished as a *polyhedral cone* and a *flux cone*, respectively [80].

2.3.3.2 Nullspace Algorithm

Nullspace Algorithm [78, 40, 79] was proposed as an improvement over the less efficient Canonical Basis Algorithm [77], both being derived from the Double Description method as illustrated previously. Since the algorithm corresponds to the enumeration of vertices in the bounded polyhedron its computational complexity [81, 82] still remains an open problem. Regardless of the lack of proof of complexity, there were earlier efforts to estimate the number of elementary flux modes [83] and extreme pathways [84]. In addition, related problems to the one of computing all elementary modes were studied and establish to be NP-hard, such as the one of enumerating elementary flux modes for which two or more specified reactions have non-zero flux.

2.3.3.3 Minimal Generating Set

In the stoichiometry network analysis, the reversibility of reaction decides if the polyhedral cone corresponding to the metabolic network is pointed or not. In the case of pointed polyhedral cone, the extreme rays of the cone are known as the *minimal generating set*. However, if the polyhedral cone is not pointed, the extreme rays and minimal generating set do not coincide, and the vectors of the minimal generating set are not linearly independent. The determination of minimal generating set for the case when the flux cone is not pointed and admits reversible pathways is outlined in the section 3.4.

2.3.3.4 Extreme Pathways

If in the metabolic network, the internal reversible reactions are decomposed into pairs of opposite irreversible reactions, the corresponding flux cone is augmented and the run of the algorithm for the enumeration of extreme rays produces minimal generating set which in the augmented space is known as *extreme pathways*[85]. Extreme pathways in the original flux cone are not conically independent, unless there were no internal reversible reactions. The augmented cone obtained from the original cone is known as EP cone (extreme pathways cone). Extreme pathways were used in the analysis of the metabolic capabilities of the red blood cell metabolism [86], *Haemophilus influenzae* and *Helicobacter pylori* [87, 88].

2.3.3.5 Elementary Flux Modes

Finally, elementary flux modes are computed when the original flux cone is converted into an EM cone whereby all reversible reactions are split into pairs of opposite irreversible reactions. The run of the Double Description Method in the EM flux cone will produce the minimal generating set which coincides with the set of pathways known as *elementary flux modes* in the original flux cone. In the original flux cone, elementary flux modes will be a superset of extreme pathways, them being a superset of the minimal generating set. Set of elementary flux modes is a complete set of pathways which satisfy the properties defined in Def. 6.

Definition 6 (Elementary modes and minimal generating set)**.** Let the $N_{m\times q}$ stoichiometry matrix be representing m internal metabolites and q reactions connecting these metabolites. A metabolic flux vector is a vector $\mathbf{r}_{qx1}$ of reaction rates. The vector $\mathbf{r}$ is said to be *admissible* if it satisfies the quasi-steady state and thermodynamics constraints as given in definition 1 An admissible metabolic flux vector is said to be an *elementary mode*, *elementary flux mode*, or *elementary pathway* if it in addition satisfy the condition of the genetic independence [89, 90, 77, 91]:

1. *genetic independence*: there is no other admissible vector $\mathbf{r}'$ ($\mathbf{r}' \neq \mathbf{r}$ and $\mathbf{r}' \neq 0$) such that the set of indices of the non-zero elements in $\mathbf{r}'$ is a strictly proper subset of set of indices of the non-zero elements in $\mathbf{r}$.

A minimal generating set is a subset of elementary flux modes which satisfies the following:

2. *generating property*: minimal generating set is a minimal subset of elementary flux modes which generates the polyhedral cone. In other words, any admissible path can be written as a convex combination of vectors in the minimal generating set. In the special case when the network admits a reversible pathway, the decomposition into minimal generating vectors may have negative coefficients thus implying the linear combination (Section 3.4).

□

2.3.3.6 Minimal Cut Sets

Using the elementary flux modes, a dual concept of the *minimal cut set* is given in Definition 7.

Definition 7 (Minimal Cut Set [92])**.** Let $ObjReac$ be the target objective reaction in a metabolic network. A subset of reactions is a *minimal cut set* if after its removal from the network there is no feasible flux distribution involving the reaction $ObjReac$ □

Minimal cut sets can be used to study the network flexibility and enumerate possible reaction knock-outs for the purpose of efficient strain design. The concept itself was later generalized [93] whereby a deletion task was defined through the set of elementary modes which should be collapsed. In addition, the minimal cut set in the metabolic networks corresponds to the concept of the minimal hitting set from graph theory. The algorithm for the computation of the minimal cut sets from the elementary flux modes is equivalent to the problem of hypergraph transversal Direct computation of minimal cut sets [94]. Alternatively, recently an approach was proposed where minimal cut sets are equivalent to the elementary flux modes computed for the network which is dual to the original metabolic network [95].

2.3.3.7 Elementary Flux Patterns

A less restrictive concept than the one of genetic independence in the elementary flux modes was introduced in the *elementary flux patterns* (EFP)[96]. Elementary flux patterns are defined for the subsystem of k reactions, with the respective columns being put to the front of the stoichiometry matrix $N_{m \times q}$, to facilitate the understanding of the concept. For a set S of all elementary flux patterns, a set of indices s represents an elementary flux pattern if a following condition is satisfied:

$$\nexists\, s_{i_1}, \ldots, s_{i_t} : \quad \bigcup_{1 \leq k \leq t} s_{i_k} = s, \text{ where } \quad i_1, \ldots, i_t \in \{1, \ldots, |S|\} \tag{2.22}$$

For the case when the considered subnetwork coincides with the entire network ($k = n$) the elementary flux patterns correspond to the elementary flux modes, otherwise an elementary flux pattern can be contained in multiple elementary flux modes. The computation of EFPs is accomplished by iteratively solving a mixed-integer linear program each returning a distinct elementary flux pattern. In every iteration, an additional constraint is added to assure that a previously unseen elementary flux pattern will be returned.

2.3.3.8 CEF (Control Effective Fluxes)

A measure of the individual reaction importance in the metabolic network, which accounts for the network flexibility, can be derived using elementary flux modes. Control effective flux $CEF_l(S_k)$ [97, 98] is calculated for every reaction $l = 1 \ldots q$ in the metabolic network for the specified substrate S_k, and product reactions (e.g. biomass and target chemical). First, efficiency for every elementary mode e_i with respect to the specified substrate and product reactions is given in equations in (2.23).

$$\begin{aligned} \epsilon_i(S_k, BIO) &= \frac{e_i^{BIO}}{\sum_{l=1}^{q} |e_i^l|} \\ \epsilon_i(S_k, CHEM) &= \frac{e_i^{CHEM}}{\sum_{l=1}^{q} |e_i^l|} \end{aligned} \tag{2.23}$$

In the above expressions e_i^l is used to denote the l^{th} reaction flux value in the i^{th} elementary mode. Further, efficiency can be used to compute the control effective flux for the reaction l as in equation (2.24).

$$CEF_l(S_k) = \frac{1}{Y_{BIO/S_k}^{max}} \cdot \frac{\sum_i \epsilon_i(S_k, BIO)|e_i^l|}{\sum_i \epsilon_i(S_k, BIO)} + \frac{1}{Y_{CHEM/S_k}^{max}} \cdot \frac{\sum_i \epsilon_i(S_k, CHEM)|e_i^l|}{\sum_i \epsilon_i(S_k, CHEM)} \tag{2.24}$$

Here Y^{max}_{BIO/S_k} and Y^{max}_{CHEM/S_k} are optimal yields for the biomass and target chemical reactions, where the yield for the reaction l with respect to the substrate for the i^{th} elementary mode is computed as $Y_{i,l/S_k} = \frac{e^l_i}{e^{S_k}_i}$, $(l = \{BIO, CHEM\})$. To assess the validity of the control effective flux metrics, the transcript ratio $\theta_l(S_1, S_2) = \frac{CEF_l(S_1)}{CEF_l(S_2)}$ was computed for the reaction l for the case of cellular growth on two different substrates S_1 and S_2. The *in silico* obtained ratios were compared to the results of microarray analysis, as it was assumed that the control effective flux will correlate with the level of mRNA for the given reaction [98].

2.3.3.9 mCEF (Modified algorithm of Control Effective Fluxes)

In [99] control effective fluxes are utilized to compute the reaction of transcripts in wild-type and mutant metabolic networks, in the event when reactions can be deleted, under- or over-expressed.

$$\begin{aligned} \epsilon_i^{(mut)}(S_k, BIO) &= \frac{e_i^{BIO} \cdot EA_i}{\sum_{l=1}^{q} |e_i^l| \cdot \eta_l} \\ \epsilon_i^{(mut)}(S_k, CHEM) &= \frac{e_i^{CHEM} \cdot EA_i}{\sum_{l=1}^{q} |e_i^l| \cdot \eta_l} \end{aligned} \tag{2.25}$$

where $\eta_l = EAP_l$ if l^{th} reaction is modified, otherwise if there was no change $\eta_l = 1$. Here EAP_l is the enzyme activity parameter which denotes the relative gene expression responsible for the l^{th} reaction of a mutant to wild-type. The term in the nominator $EA_i = \prod_{j=1}^{q} ge_{j,i}$ where $ge_{j,i} = EAP_j$ if j-th reaction is involved in i-th EM, otherwise $ge_{j,i} = 1$. Using the expression for the efficiency of elementary modes for both wild-type and mutant networks in (2.23) and (2.25) control effective fluxes can be derived and used to compute the transcript ratio.

$$\begin{aligned} mCEF_l(mut) = & \frac{1}{Y^{max}_{BIO/S_k}} \cdot \frac{\sum_i \epsilon_i^{(mut)}(S_k,BIO)|e_i^l| \cdot \eta_l}{\sum_i \epsilon_i^{(mut)}(S_k,BIO)} \\ & + \frac{1}{Y^{max}_{CHEM/S_k}} \cdot \frac{\sum_i \epsilon_i^{(mut)}(S_k,CHEM)|e_i^l| \cdot \eta_l}{\sum_i \epsilon_i^{(mut)}(S_k,CHEM)} \end{aligned} \tag{2.26}$$

$$\begin{aligned} mCEF_l(wt) = & \frac{1}{Y^{max}_{BIO/S_k}} \cdot \frac{\sum_i \epsilon_i^{(wt)}(S_k,BIO)|e_i^l|}{\sum_i \epsilon_i^{(wt)}(S_k,BIO)} \\ & + \frac{1}{Y^{max}_{CHEM/S_k}} \cdot \frac{\sum_i \epsilon_i^{(wt)}(S_k,CHEM)|e_i^l|}{\sum_i \epsilon_i^{(wt)}(S_k,CHEM)} \end{aligned} \tag{2.27}$$

Using the expressions for the control effective flux of the i^{th} reaction for the wild-type and mutant networks, the transcript ratio of the two can be computed as in (2.28).

$$\Theta_i(wt, mut) = \frac{mCEF_i(mut)}{mCEF_i(wt)} \tag{2.28}$$

The transcript ratio corresponds to the ratio in enzyme activity in mutant to wild-type networks, which is exploited in the problem of estimating the flux distribution in mutant networks when some of the reactions are deleted, over- or under- expressed, as will be shown in the continuation.

2.3.3.10 Estimation of the Flux Distribution

Once elementary flux modes are computed, they can be used to estimate the most likely overall reaction flux distribution. Having in mind that the overall reaction flux distribution vector can be represented as a linear combination of the elementary flux modes, one may obtain the weights of each elementary mode in this decomposition [100].

1. *Alpha-spectrum*[101] was an early effort to estimate the participation of different pathways in the given metabolic flux vector. For the given matrix of extreme pathways EP and the metabolic flux vector v, the goal is to find the vector of α-weights, so that $v = \alpha \cdot EP$. One may estimate the range of values for each α_i element of the vector, by means of solving the $\min / \max$ problem as in (2.29).

$$\begin{aligned} \text{min/max} \quad & \alpha_i \\ \text{subject to} \quad & \alpha \cdot EP = v \\ & 0 \leq \alpha_i \leq 1 \end{aligned} \tag{2.29}$$

   The constraining of values of α-weights can be further done by imposing the requirement for the minimal number of active extreme pathways in the metabolic flux vector. It is important to mention that the concept of weight estimation can be analogously applied to the minimal generating set or elementary flux modes.

2. Using a different paradigm, a quadratic program was proposed with an objective function which minimizes the square of the elementary mode weights, represented by vector w, in the overall flux distribution vector[102, 103, 104].

$$\begin{aligned} \text{min} \quad & \sum_{j=1}^{|E|} w_j^2 \\ \text{subject to} \quad & \sum_{j=1}^{|E|} w_j \cdot e^{(j)} = v \\ & w_j \geq 0, \text{ if } e^{(j)} \text{ elementary mode is irreversible} \end{aligned} \tag{2.30}$$

3. An approach based on the maximization of the Shannon's entropy [105, 106] was proposed along the fractions of each elementary mode's contribution in the uptake flux of the given metabolic flux vector v and is illustrated in (2.31).

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{|E|} s_j \ln s_j \\
\text{subject to} \quad & \sum_{j=1}^{|E|} w_j \cdot e^{(j)} = v \\
& w_j > 0, \text{ if } e^{(j)} \text{ mode is irreversible} \\
& s_j = \frac{w_j \cdot e^{(j)}_{substrate}}{v_{substrate}}
\end{aligned}
\tag{2.31}
$$

In addition, it was earlier demonstrated that the weights are inversely correlated with the entropy of the overall reaction corresponding to the elementary flux modes, thereby favoring pathways with low entropy generation [107].

4. (ECF) Enzyme Control Flux [108] is used to predict the flux distribution in a mutant network obtained after deletion, over- and under- expression of the specified reactions. It relies on the use of enzyme activity profile and the power-law which uses the change of enzyme activity to estimate the weights w in the elementary flux mode representation of the overall flux distribution in the mutant network. Given the coefficients $w^{(wt)}$, the respective coefficients of the mutant flux distribution are found as in (2.32):

$$w_j^{(mut)} = \gamma \cdot w_j^{(wt)} \prod_{i=1}^{q} a_{ij} \tag{2.32}$$

The parameter a_{ij} is the relative enzyme activity of a mutant to wild-type for the i^{th} reaction in the j^{th} elementary mode. If the i^{th} reaction is involved in j^{th} elementary mode, then $a_{ij} = a_i$, otherwise $a_{ij} = 1$, where a_i is the enzyme activity ratio of the mutant to wild-type for the i^{th} reaction. Normalization factor for $w^{(mut)}$ is given by γ.

If mCEF is used to compute the relative enzyme activity of a mutant to wild type for the i^{th} reaction [99] then the expression for $w_j^{(mut)}$ may be written as:

$$w_j^{(mut)} = \gamma \cdot w_j^{(wt)} \prod_{i=1}^{q} \theta_i(wt, mut) \tag{2.33}$$

The flux distribution in the mutant network, where E denotes the matrix of elementary flux modes, is given in (2.34).

$$v^{(mut)} = w^{(wt)} \cdot E \tag{2.34}$$

5. One method to provide a valid decomposition of a given metabolic flux vector v into as few elementary modes as possible, without having to compute the complete set of elementary modes was proposed in [109]. A series of mixed-integer linear programs is solved iteratively in a finite number of steps K. Algorithm starts for $k = 1$ by solving the problem given in (2.35) for the given flux vector $v^{(k)} = v$.

$$\begin{aligned}
\min \quad & \sum_{j=1}^{q} z_j \\
\text{subject to} \quad & N \cdot e = 0 \\
& e_{j_k} = \operatorname{sgn}(v_{j_k}^{(k)}) \\
& -Mz_j \le e_j \le Mz_j,\ j = 1, \dots, q \\
& e_j = 0, \qquad j \in \{j | v_j = 0\} \\
& e_j \le 0, \qquad j \in \{j | v_j < 0\} \\
& e_j \ge 0, \qquad j \in \{j | v_j > 0\} \\
& z_j \in \{0, 1\}, \qquad j = 1, \dots, q
\end{aligned} \tag{2.35}$$

where M is a large constant positive number. Once the mode e^* is found as an optimal solution, we set $e^k = e^*$ and compute the weight w_k, to update the flux vector as $v^{(k+1)} = v^{(k)} - w_k e^{(k)}$. If the vector $v^{(k+1)}$ is zero then the algorithm is complete and the decomposition is found.

2.3.3.11 Network Robustness and Pathway Redundancy

Elementary flux modes and extreme pathways were used to study the network flexibility and pathway redundancy [87]. One measure of the network flexibility [110] as a function of the number of knockouts d in the metabolic network having q reactions is given in expression (2.36).

$$R(d) = \sum_{i=1}^{\binom{q}{d}} \frac{z^{(i)}}{\binom{q}{d} z} \tag{2.36}$$

In the above expression z is the total number of elementary flux modes, while $z^{(i)}$ denotes the number of remaining elementary flux modes after deleting d reactions for the i^{th} deletion combination out of all possible $\binom{q}{d}$ choices. General network flexibility can be averaged across all values of the metric $R(d)$ for all possible values of d.

2.3.3.12 Minimal Metabolic Functionality

Minimal Metabolic Functionality [111, 1] was proposed as a metabolic network design concept where the goal is to retain as few high-yielding target chemical elementary modes which are growth-coupled by means of using as few reaction knockouts as possible. In *Escherichia coli* this concept was used manually to find a reaction knockout subset for the optimization of the production of ethanol using hexose and pentose [1], and glycerol substrates [112].

2.3.3.13 Flux Design

Flux Design [113] examines as possible over-expression or under-expression/deletion targets those reactions which positively or negatively correlate with the chosen objective reaction, respectively.

2.3.3.14 CASOP (Computational Approach for Strain Optimization aiming at high Productivity)

CASOP [114, 115] is a framework based on using elementary flux modes to compute the importance measures for reactions in the metabolic network. Importance measure of a reaction accounts for both network flexibility and target metabolite yield, and is used to derive reaction ratings which indicate if the observed reaction is a candidate for a knockout or over-expression. The high network flexibility assures a metabolic network robustness and is reflected in the number of the preserved elementary modes as alternative routes to convert the substrate to the target metabolite.

2.3.3.15 Weighting of elementary modes using thermodynamics

An approach [116] which combines elementary flux modes, pathways thermodynamics and genetic algorithms to elucidate a favorable reaction knockout subset for the target chemical production coupled with growth was proposed. Thermodynamics plays a role where the elementary flux modes with more negative Gibbs free energy are more likely to have higher weight in the overall flux distribution. Iteratively, the population evolves using crossover and mutation operations and for each individual a fitness function is used to evaluate the quality of solution and if it will be retained in the next generation.

2.3.3.16 cMCS (Constrained Minimal Cut Sets)

Constrained minimal cut sets [117] are enumerated as reaction knockout subsets where some specified minimum of the desired modes is preserved, while all the undesired i.e. inefficient modes are collapsed. This algorithm uses a variation of an exhaustive Berge's algorithm originally proposed in graph theory.

2.3.3.17 Implementation of Nullspace Algorithm

Several implementations of the Nullspace Algorithm and its variations are available. Early versions of METATOOL [90] initially implemented the Canonical Basis Algorithm [77]. It was later upgraded to implement the improved Nullspace Algorithm [118] thereby reducing the computation time. Exploiting the features of the bit-pattern trees in the generation of candidate elementary modes and multi-core shared-memory systems Nullspace Algorithm also has its parallel versions for the shared-memory system which is implemented in EFM-Tool [41]. Both METATOOL and EFMTool were added as modules into GUI-based software for the analysis of metabolic and signaling networks known as CellNetAnalyzer [119], as well as in COPASI [120]. Finally, ExPA implements the computation of extreme pathways [121].

2.4 Integrated Biological Network Analysis

Some of the previously cited and described methods aim to incorporate heterogeneous biological networks, such as those which augment metabolic network with the gene regulatory network or enzyme activity profile. This section will present in more detail and outline all recent efforts in integrated biological network analysis aiming for more accurate prediction of the cellular phenotype for the given environmental and growth conditions.

.

IOMA (Integrative Omics-Metabolic Analysis)

A framework which allows integration of proteomic and metabolomic data with genome-scale metabolic models for the prediction of flux distribution was proposed in [122]. The framework is laid out as a quadratic programming problem which minimizes the inconsistency between the flux predicted by the metabolic network model and the one obtained form the metabolomic and proteomic data using Michaelis-Menten kinetics. IOMA was compared to less comprehensive method such as FBA or earlier described MOMA, demonstrating its superior predictive power.

iMAT (Integrative Metabolic Analysis Tool)

One effort to integrate the tissue-specific gene and protein expression data with the genome-scale metabolic network under the assumed steady state constraints was presented in [123]. For the given tissue type, the available expression data is given in two subsets R_L and R_H which contain indices of lowly expressed and highly expressed reactions. Highly expressed reactions in R_H subsume those reactions which are over- or under-expressed, while lowly expressed are those reactions which have flux close to zero. Given the subset R_L the binary variable y_i^+ is used denote if the i^{th} reaction is truly lowly expressed in the problem formulation. Similarly, for subset R_H the binary variables y_i^+ and y_i^- denote if the reaction is highly expressed in a way of being over- or under- expressed. Since a single highly expressed reaction can be either over- or under- expressed, the inequality $y_i^+ + y_i^- \leq 1$ is imposed. Hence, the mixed-integer linear programming problem, given the tissue-specific subsets of indices of lowly and highly expressed reactions R_L and R_H is given in (2.37).

$$\begin{aligned}
\min \quad & \sum_{i \in R_H} y_i^+ + y_i^- + \sum_{i \in R_L} y_i^+ \\
\text{subject to} \quad & N \cdot v = 0 \\
& v_{min,i} \leq \leq v_i^{max}, \quad \forall i \in \{1 \dots q\} \\
& v_i + y_i^+ (v_{min,i} - \epsilon) \geq v_{min,i}, \quad \forall i \in R_H \\
& v_i + y_i^- (v_{max,i} + \epsilon) \leq v_{max,i}, \quad \forall i \in R_H \\
& v_{min,i}(1 - y_i^+) \leq v_i \leq v_{max,i}(1 - y_i^+), \quad \forall i \in R_L \quad and \quad y_i^+ \in \{0,1\} \\
& y_i^+ + y_i^- \leq 1 \quad \forall i \in R_H \quad and \quad y_i^+, y_i^- \in \{0,1\}
\end{aligned} \tag{2.37}$$

This integration of heterogeneous omics data is implemented in the iMAT (Integrative Metabolic Analysis Tool) tool as described in [124].

Whole cell modeling

A separate pioneering effort to integrate the regulatory FBA and the system of ordinary differential equations was proposed in [125]. It was later extended to a more comprehensive cell model [126, 127, 128] using 16 cell variables as input and output to the 28 distinct processes. Each process is described using a submodel which is itself modeled using the most appropriate method (e.g. constraint-based modeling, ordinary differential equations). Simulation is performed on a 1-second time scale, where at the beginning of every 1-second interval the processes accept the values in 16 cell variables, and independently compute their output. Following, at the end of simulation step, the computed output across all processes is used to update the 16 cell variables. The proposed and implemented model can predict gene-expression levels, metabolism rates, metabolite concentrations, protein levels, and cell replication times in both wild type and single gene deletion circumstances.

Chapter 3

Nullspace Algorithm and its Development

3.1 Introduction

Focusing on the pathways-based analysis of the metabolic networks, this chapter illustrates the Nullspace Algorithm for the computation of elementary flux modes and its particuliarities. Nullspace Algorithm is outlined in a modular way and respective routines are detailed for the purpose of analyzing the algorithmic complexity and later parallelization as shown in the subsequent chapters. A reduced algebraic rank test is proposed for the case when reversible reactions are not split prior to running the algorithm, which can significantly reduce the computation time. Finally, the chapter demonstrates the non-uniqueness of the minimal generating set and its computation in the metabolic networks which admit one or more reversible pathways.

3.2 Nullspace Algorithm

The two algorithms typically used for the computation of elementary modes are the Canonical Basis Algorithm [77] and the subsequent Nullspace Algorithm [118, 78, 79, 129, 40, 119, 41]. Both algorithms are based on convex analysis and computation of the extreme rays of a convex polyhedral cone as demonstrated earlier in subsubsection 2.3.3.1. The Nullspace Algorithm, being the successor of the less efficient and now abandoned Canonical Basis Algorithm, is the one presently used in the computation of elementary modes.

The Nullspace Algorithm begins by computing an initial basis for the right nullspace of the $m \times q$ stoichiometry matrix such that the sign constraints are automatically satisfied for the first $q - m$ reactions. It then proceeds to form convex combination of these vectors to impose the sign and elementarity constraints on the remaining reactions one-by-one, until a complete set of elementary flux vectors are computed. In the following, we state some of the basic properties of the Nullspace Algorithm.

Theorem 1. *If $N_{m\times q}$ is a stoichiometry matrix with full row rank m, then the columns may be permuted such that a basis for the right nullspace of N has the form*

$$K_{q\times(q-m)} = \begin{bmatrix} I_{(q-m)\times(q-m)} \\ R_{m\times(q-m)} \end{bmatrix} \tag{3.1}$$

Proof. Apply elementary row operations (represented by the nonsingular matrix X) to the matrix N to obtain the reduced row echelon form

$$\tilde{N}_{m\times q} = X_{m\times m}N_{m\times q} = \begin{bmatrix} -R_{m\times(q-m)} & I_{m\times m} \end{bmatrix}. \tag{3.2}$$

The new matrix has the same nullspace, which has the form (3.1) by inspection. □

Prior to finding the right nullspace matrix as in Theorem 1 the original stoichiometry matrix is reduced using the techniques described in subsection 2.3.1 to obtain an equivalent reduced stoichiometry matrix having a full row rank. The reduced full row rank stoichiometry matrix, being in the reduced row echelon form, is used to obtain the initial basis for the right nullspace. Therefore, we shall henceforth assume that the stoichiometry matrix N has been compressed, reduced to row echelon form, and that the columns (i.e. reactions) have been permuted so that the row echelon form has the form (3.2). This is equivalent to finding $q-m$ columns which form a $(q-m)\times(q-m)$ non-singular matrix and putting them first. We further assume that the corresponding $q-m$ reactions are all irreversible, otherwise we must split sufficiently many reversible reactions into pairs of irreversible reactions to make this possible.

If $\overline{Z}(x)$ denotes the set of indices corresponding to the nonzero entries of a given vector $\mathbf{x}$, then $N_{*,\overline{Z}(x)}$ will denote the submatrix of N formed by extracting the columns corresponding to those non-zero entries. It has been shown in [79, 130] that $\mathsf{nullity}(N_{*,\overline{Z}(x)}) = 1$ if and only if $\mathbf{x}$ is elementary mode. Here $\mathsf{nullity}(A)$ denotes the dimension of the right nullspace of a matrix A. During the course of the Nullspace Algorithm, we enforce the following property on each prospective elementary vector $\mathbf{x}$ at each iteration k so that at the end, this property implies that $\mathbf{x}$ is elementary according to Definition 6.

Theorem 2. *Let the Nullspace Algorithm be in its k^{th} iteration of execution where $k = q-m+1,\ldots,q$. A vector $\mathbf{x}$ is an elementary flux mode with respect to reactions $1,\ldots,k$ corresponding to first k columns of matrix N iff*

$$\mathsf{nullity}(N_{*,\overline{Z}_k}) = 1. \tag{3.3}$$

where $\overline{Z}_k$ is the union of the set of indices of non-zero values among first k entries of vector $\mathbf{x}$ together with all indices $(k+1),\ldots,q$.

The property in equation (3.3) enforces the elementarity over the first k reactions. It will be observed that each column of the initial basis R from equation (3.1) satisfies the partial elementary property above for $k = q - m$. As a simple consequence of the above property, a vector satisfying this condition cannot have more non-zero entries than one plus the number of rows in N, leading to the following.

Theorem 3. *Let* $\mathbf{x}$ *be a column-vector which is an elementary flux mode to the stoichiometry matrix* $N_{m\times q}$ *i.e.* $N\mathbf{x} = 0$. *An upper bound on the number of non-zero elements in the vector* $\mathbf{x}$ *is given by*

$$|\overline{Z}| \leq m + 1, \tag{3.4}$$

where $|\overline{Z}|$ *denotes the cardinality of* $\overline{Z}$.

The upper bound stated in Theorem 3 is given for the full elementary property of Def. 6. At the k^{th} iteration, since the entries of a prospective vector $\mathbf{x}$ corresponding to indices $(k+1), \ldots, q$ are all considered implicitly nonzero, the number of nonzeros among the first k entries is reduced from $1+m$ to $1+m-(q-k)$. The result leads to the following necessary condition for elementarity that can be applied very fast.

Theorem 4. *Let* $\mathbf{x}$ *be a column-vector in the right nullspace matrix* R *of the stoichiometry matrix* $N_{m\times q}$ *i.e.* $N\mathbf{x} = 0$. *Let the first* k *elements of the vector* $\mathbf{x}$ *have non-negative values in the positions corresponding to irreversible reactions, and condition (3.3) is satisfied. Denote by* $\overline{Z}_{1,\ldots,k}$ *the set of indices of nonzero elements in the subvector* $x_{1,\ldots,k}$. *If the matrices are in reduced row echelon form as in Theorem 1, then*

$$|\overline{Z}_{1,\ldots,k}| \leq k - q + m + 1 \tag{3.5}$$

Proof. Follows from Theorem 3. □

In brief, the Nullspace algorithm is an iterative procedure which starts with a nullspace basis as in Theorem 1. At each iteration it forms new prospective elementary modes by pair-wise convex combinations of the partial elementary modes it has accumulated so far. Each prospective elementary mode is tested to be elementary, first by testing the condition of Theorem 4 and then by that of Theorem 2. The steps to execute the Nullspace algorithm are sketched in Algorithm 1, and the way the computation is split into its essential parts is shown in Algorithm 2.

The sketch of the Nullspace Algorithm presented omitted several improvements to the efficiency for clarity. First, during every iteration, each new column is normalized with respect to the 1-norm. Second, we are able to keep the matrix $R^{(1)}$ as a bit-valued matrix and compress it into a matrix scaled down by a factor equal to the length of the machine

Algorithm 1 Nullspace Algorithm (sketch) [130].

Assume we have a stoichiometry matrix $N_{m\times q}$ that has full row rank m and in the form as given in Theorem 1, compressed if needed using the methods of subsection 2.3.1. Further, let q_{irrev} and $q - q_{irrev}$ be the number of irreversible and reversible reactions, respectively. The Nullspace Algorithm may be briefly sketched as follows:

1. Denote the initial right nullspace $R_{q\times(q-m)}$ (equation (3.1)) of the stoichiometry matrix $N_{m\times q}$ as:

$$K = \begin{array}{r} {\scriptstyle (q-m)}\left\{ \right. \\ {\scriptstyle (m)}\left\{ \right. \end{array} \begin{bmatrix} \overbrace{R^{(1)}}^{q-m} \\ R^{(2)} \end{bmatrix} = \begin{bmatrix} I \\ R^{(2)} \end{bmatrix} \tag{3.6}$$

   where the upper matrix of R, denoted as $R^{(1)}$, is an identity matrix $I_{(q-m)\times(q-m)}$.

2. For $k = (q-m), \ldots, (q-1)$,

   (a) Generate convex combinations of all possible pairs of columns in R so as to annihilate the $(k+1)^{th}$ entry of the resulting column. Each combination is formed using a column ii whose $(k+1)^{th}$ entry is positive combined with a column jj whose $(k+1)^{th}$ entry is negative. Following the results from [40] we may perform the operation of bit-wise logical disjunction over the column parts belonging to matrix $R^{(1)}$, while performing the algebraic convex combination over column parts in matrix $R^{(2)}$.

   (b) Eliminate duplicate columns among those generated from $R^{(1)}$ in the previous step.

   (c) Apply the rank test as given in Theorem 2 to each candidate mode, discarding those that fail the test.

   (d) Append matrix R column-wise with the newly computed elementary modes which were accepted by the rank test in the previous step.

   (e) If the $(k+1)^{th}$ reaction is irreversible, discard those old columns whose $(k+1)^{th}$ entry is negative.

   In the next step, the $(k+1)^{th}$ row (the top row of $R^{(2)}$) is moved to become the bottom row of $R^{(1)}$. Following [40], $R^{(1)}$ can be kept only as a bit mask, so the $(k+1)^{th}$ row is converted to a bit mask (a 1 bit stands for a non-zero value).

3. When the computation is complete, matrix $R^{(1)}$ will be of dimension $q \times n_{ems}$, where the n_{ems} is the total number of elementary flux mode columns, while $R^{(2)}$ will be empty. It is then necessary to recalculate the numerical values. This process has linear complexity in the number n_{ems} of elementary modes computed [40].

word (32 or 64 bit). Accordingly, the compressed matrix $R^{(1)}$ as stored in memory has the dimension of $(q/width) \times n_{ems}$, where $width$=32 or 64.

We take advantage of the special row-echelon form of N to obtain a reduced-cost rank test, more properly called a nullity test, given in Theorem 6. Theorem 6 gives a nullity test over a smaller submatrix than previously used and thus reduces the cost of its computation. This reduced nullity test decreases the size of both dimensions of the submatrix by the same value, equal to the number of non-zero entries in the sub-vector $x_{q-m+1,\ldots,k}$.

3.2.1 Complexity of the Nullspace Algorithm

The problem of enumerating elementary flux modes is analogous to the problem of finding vertices or extreme rays in the convex bounded polyhedron (i.e. polytope) as illustrated earlier in the subsection 2.3.3.1. The said enumeration problem differs in its complexity depending if the considered polytope is degenerate or non-degenerate. A non-degenerate polytope is one where each vertex is an intersection of no more than d hyperplanes, while no hyperplane contains $d + 1$ vertices, where d is the dimensionality of the polytope. In other words, if the point satisfies $d + 1$ half-space inequalities with equality the polytope is degenerate. An algorithm for the enumeration of vertices in a non-degenerate polytope which runs in the polynomial total time was earlier proposed, and is known as *local reverse search* [131]. On the other side, for the enumeration of degenerate polytope, the *double description method* was found to perform best in the event where a large number of intermediate results is generated [76]. The complexity of the enumeration of vertices in the degenerate case still remains an open problem [81].

It may be useful to look at the different problems surrounding the discovery of the vertices of polytope such as (1) finding a polytope vertex (2) counting of vertices (3) enumeration of vertices.

Finding if a given set of variables constitutes a valid polytope vertex can be done in time polynomial in the size of the given network using a simple rank test. Similarly, a vertex can be found in polynomial time of the network size where it is not beforehand specified which reactions have to be in the mode. Let `T` be a set of variables which should enter the support set of the vertex. Following results were demonstrated in [132, 82].

Theorem 2 ([132])**.** *For a given polytope and specified variable set* `T`*, enumeration of all vertices having all variables from* `T` *in their support cannot be done in polynomial total time unless P=NP.*

The decision problem is solvable in polynomial time for $\|\texttt{T}\|$, but is NP-complete for $\|\texttt{T}\| > 1$ as illustrated in the following theorem.

Theorem 3 ([132, 82])**.** *For a given polytope and specified variables i and j, to decide if there is a vertex having i and j in its support set is NP-complete.*

Likewise, deciding if there is an elementary mode with at most k reactions having non-zero flux is NP-complete as well.

Theorem 4 ([132]). *For a given polytope and specified integer k, to decide if there is a vertex with at most k variables in its support set is NP-complete.*

Counting of the vertices of a bounded polytope was shown to be #P-complete [133] which was reduced from the problem of counting perfect matchings in a bipartite graph. A #P-complete problem belongs to the complexity class of counting problems which are associated with a decision problem in NP. As such, an #P-complete problem is as hard as the associated decision problem in NP.

In the case of enumeration problems where a large number of intermediate results not entering the output is generated, as is the case in double description method, it is common to express algorithmic complexity as a function of the size of input and output. This is known as the *polynomial total time* complexity. Another related problem of the enumeration of vertices in an unbounded polytope is NP-hard.

Theorem 5 ([81]). *Enumeration of vertices in a general polyhedron is NP-hard.*

Open problem: Is the enumeration of vertices in the bounded polyhedron solvable in polynomial total time?

Degeneracy of the metabolic network solution space: Metabolic network stoichiometry matrix has solutions which lie in the degenerate polytope and thus enumeration of elementary flux modes does not have a known polynomial time algorithm. If K is the right nullspace basis of the stoichiometry matrix N i.e. $N \cdot K$=0 and without loss of generality let all reactions be irreversible. Thus, an admissible metabolic flux vector $\mathbf{x} \geq 0$ can be written as $x = \lambda \cdot K$ for some coefficient vector λ as the right nullspace K is the polytope generating matrix. Thus, the elements of the vector x with zero value correspond to the polytope defining inequalities which are satisfied with equality. As evident, extreme rays of the metabolic network polytope have more zero elements than is the dimensionality of the space implying the degeneracy of the enumeration problem. It remains unclear how may one compare the "levels of degeneracy" between the two metabolic network models.

3.2.2 Serial Nullspace Algorithm Pseudocode

The serial Nullspace Algorithm given in Algorithm 2 takes as input the compressed stoichiometry matrix in the reduced row echelon form (Theorem 1), initial nullspace, and the information on reaction reversibility/irreversibility. The algorithm is executed in m iterations, each of them corresponding to one of the m remaining reactions.

Algorithm 2 is comprised of the generation of the candidate columns (Algorithm 3), sorting (Algorithm 17) and removal of the duplicate candidate columns, numerical rank

Algorithm 2 $[R] =$ `NullspaceAlg`(N, R)

```
Input:
 1: reduced stoichiometry matrix - $N_{m\times q}$
 2: initial nullspace matrix - $R_{q\times(q-m)} = \begin{bmatrix} R^{(\mathsf{bit})}_{(q-m)\times(q-m)} \\ R^{(\mathsf{real})}_{m\times(q-m)} \end{bmatrix} = \begin{bmatrix} I_{(q-m)\times(q-m)} \\ R^{(\mathsf{real})}_{m\times(q-m)} \end{bmatrix}$
Output:
 3: bit-valued matrix of elementary modes -$R^{((\mathsf{bit}))}_{q\times n_{ems}}$
 4: for $k = q-m+1$ to $q$ do
 5:
 6:     ▷ find pairs of columns which when combined form candidate columns. Algorithm 3
 7:         combinations ⇐ GenerateEFMCands($R$)
 8:
 9:     ▷ remove duplicate columns by means of sorting. Algorithm 17
10:         combinations ⇐ RadixSort($R^{(\mathsf{bit})}$, combinations)
11:         combinations ⇐ RemoveDuplicates($R^{(\mathsf{bit})}$, combinations)
12:
13:     ▷ accept those candidate columns which satisfy Theorem 6. Algorithm 18
14:         combinations ⇐ RankTests($N$, $R$, combinations)
15:
16:     ▷ expand R matrix, i.e. its $R^{(\mathsf{bit})}$ and $R^{(\mathsf{real})}$ submatrices. Algorithm 19
17:         $R$ ⇐ ExpandEFM($R$, combinations)
18: end for
```

Algorithm 3 $[combinations] =$ `GenerateEFMCands`(R)

```
Input:
 1: current nullspace matrix - $R_{q\times n_{ems}} = \begin{bmatrix} R^{(\mathsf{bit})} \\ R^{(\mathsf{real})} \end{bmatrix}$
Output:
 2: pairs of indices of columns forming candidates - combinations
 3: $irrev^{+} \Leftarrow \{i : R^{(\mathsf{real})}_{1,i} > 0 \;\&\; (\exists j : j^{th}$ reaction is irreversible, $R^{(\mathsf{bit})}_{j,i} \neq 0)\}$
 4: $irrev^{-} \Leftarrow \{i : R^{(\mathsf{real})}_{1,i} < 0 \;\&\; (\exists j : j^{th}$ reaction is irreversible, $R^{(\mathsf{bit})}_{j,i} \neq 0)\}$
 5: $rev \Leftarrow \{i : R^{(\mathsf{real})}_{1,i} \neq 0 \;\&\; (\forall j : j^{th}$ reaction is reversible or $R^{(\mathsf{bit})}_{j,i} = 0)\}$
 6: ▷ combine columns that can annihilate the element in the current row
 7:     $S \Leftarrow \{(ii,jj) : (ii,jj) \in (irrev^{+} \times irrev^{-}) \cup ((irrev^{+} \cup irrev^{-} \cup rev) \times rev)\}$
 8: for each $(ii,jj) \in S$ do
 9:     form candidate column from the pair of columns indexed by (ii,jj)
10:     if candidate satisfies Theorem 4, add (ii,jj) to combinations
11: end for
```

testing (Algorithm 18) and update of the current nullspace matrix R ($R^{(\mathsf{bit})}$ and $R^{(\mathsf{real})}$) (Algorithm 19). In an effort to eliminate the duplicate bit-valued candidate columns we first sort them according to their binary values and then use one scan to eliminate the duplicates. This operation requires an efficient sorting method to reduce the cost of removing duplicate columns. Candidate columns are sorted using a variation of radixsort algorithm [134] in order to attain linear complexity. We give the outline of the radixsort over an array of bit-valued columns in Algorithm 17.

The idea in Algorithm 17 is to sort bit-columns by first cutting all columns horizontally into chunks of width equal to 2^d (where $d = 3, 4, 5, \ldots,$) and in $q/2^d$ iterations sort the columns using the idea from the radix-sort. In every iteration, columns would be sorted according to the value in the respective chunk. Complexity of this operation is $O(\frac{q}{2^d} \cdot n_{ems})$, where n_{ems} is the number of candidate columns at the given iteration. With the proper selection for width d, we may assume that the constant factor before n_{ems} is small enough to assume linear complexity. In Algorithms 18 and 19 we also give the pseudocode of the subroutines for rank tests and expansion of nullspace matrix in every iteration.

3.3 Reduced algebraic rank test

Theorem 6. *Let the Nullspace Algorithm be in its k^{th} iteration of execution as k ranges over $q - m + 1, \ldots, q$. Let $\overline{Z}_{1,\ldots,q-m}$ be the set of indices corresponding to non-zero entries in $x_{1,\ldots,q-m}$, and let $Z_{q-m+1,\ldots,k}$ be the set of indices corresponding to zero entries in $x_{q-m+1,\ldots,k}$. A vector $\mathbf{x}$ is an elementary flux mode with respect to reactions $1, \ldots, k$ corresponding to the first k columns of matrix N iff*

$$\mathsf{nullity}(N_{Z_{q-m+1,\ldots,k},\overline{Z}_{1,\ldots,q-m}}) = 1. \tag{3.7}$$

Proof. The reduced rank test is derived from the reduced row echelon form of the compressed stoichiometry matrix as is obtained in Theorem 1, which has the form $\tilde{N}^T = \begin{pmatrix} N_1 & I \end{pmatrix}$. We assume without loss of generality that $N = \tilde{N}^T$ is $m \times q$ (by removing redundant rows in advance if necessary) matrix. At the stage k of the Nullspace Algorithm, matrix N can be further decomposed to:

$$N = \begin{pmatrix} N_1 & I \end{pmatrix} = \begin{array}{c} \\ (k-(q-m))\{ \\ (q-k)\{ \end{array} \begin{pmatrix} \overbrace{P}^{(q-m)} & \overbrace{I_{\mathrm{k-(q-m)}}}^{k-(q-m)} & 0 \\ Q & 0 & I_{\mathrm{q-k}} \end{pmatrix}. \tag{3.8}$$

As stated in Theorem 2 we must select all the columns of the stoichiometric matrix whose indices correspond to nonzero elements among $x_1, \ldots, x_k$ at stage k and the first $k-(q-m)$ rows. According to Theorem 2 we would have that:

$$\mathsf{rank}\begin{pmatrix} P_{*,\overline{\mathsf{Z}}_{1\ldots q-m}} & I_{\mathrm{k-(q-m)},\overline{\mathsf{Z}}_{q-m+1\ldots k}} \end{pmatrix} = \left|\overline{\mathsf{Z}}_{1\ldots q-m}\right| + \left|\overline{\mathsf{Z}}_{q-m+1\ldots k}\right| - 1 \tag{3.9}$$

To compute the rank of the submatrix obtained in this way we have:

$$\begin{aligned} \mathsf{rank}\begin{pmatrix} P_{*,\overline{\mathsf{Z}}_{1\ldots q-m}} & I_{\mathrm{k},\overline{\mathsf{Z}}_{q-m+1\ldots k}} \end{pmatrix} &= \mathsf{rank}(P_{\mathsf{Z}_{q-m+1\ldots k},\overline{\mathsf{Z}}_{1\ldots q-m}}) + \mathsf{rank}(I_{\mathrm{k},\overline{\mathsf{Z}}_{q-m+1\ldots k}}) \\ &= \mathsf{rank}(P_{\mathsf{Z}_{q-m+1\ldots k},\overline{\mathsf{Z}}_{1\ldots q-m}}) + \left|\overline{\mathsf{Z}}_{q-m+1\ldots k}\right| \end{aligned} \tag{3.10}$$

and from (3.9) and (3.10) we have that

$$\mathsf{rank}(P_{\mathsf{Z}_{q-m+1\ldots k},\bar{\mathsf{Z}}_{1\ldots q-m}}) = \left|\bar{\mathsf{Z}}_{1\ldots q-m}\right| - 1 \tag{3.11}$$

or expressed in terms of nullity of the matrix

$$\mathsf{nullity}(P_{\mathsf{Z}_{q-m+1\ldots k},\bar{\mathsf{Z}}_{1\ldots q-m}}) = 1. \tag{3.12}$$

□

3.4 Enumeration of Minimal Generating Set

3.4.1 Introduction

The solutions of the stoichiometry equation $\mathbf{S}\cdot\mathbf{x} = 0$, which also satisfies the non-negativity constraints for the flux of its irreversible reactions, describe all possible metabolic states in which the metabolic network may be found. Geometrically this solution space corresponds to the polyhedral cone [135, 80], and it may be fully generated by means of its extreme rays [76]. Extreme rays are conically independent set of vectors and in the convex analysis are also known as a *minimal generating set.* In the stoichiometry network analysis, alongside with the concept of the minimal generating set, stand the extreme pathways and elementary flux modes. It is important to say that the *minimal generating set*, *extreme pathways* and *elementary flux modes* are computed using the standard Double Description Method for the enumeration of extreme rays (i) when no reversible reactions are split, (ii) only internal reversible reactions are split, and (iii) all of the reversible reactions are split, into two irreversible components, respectively [89, 80, 85, 136, 130]. Unlike the minimal generating set, in the original reaction space the extreme pathways and elementary flux modes are not necessarily conically independent which depends on the existence and number of the reversible reactions.

In the absence of reversible reactions, the *minimal generating set*, *extreme pathways* and *elementary flux modes* coincide, are uniquely defined, and correspond to the *extreme rays* of the polyhedral cone. Regarding the directionality of the metabolic pathways which the metabolic network accepts we distinguish two cases.

In the first case, if the metabolic network admits only irreversible pathways (i.e., every pathway contains at least one irreversible reaction), then the minimal generating set is unique, and the corresponding polyhedral cone is said to be *pointed.* On the other side, in the second case, if the metabolic network admits reversible pathways, the minimal generating set is no longer unique and the polyhedral cone is *not pointed.*

Regardless if the cone is pointed or not, the set of the elementary flux modes (or extreme pathways) is a superset of any minimal generating set, and some of the elementary flux modes (or extreme pathways) may lie in the interior of the cone. In addition, putatively exponential hardness and high computational cost of the algorithm used to compute elementary flux modes [132, 82] is another reason to shift the attention from extreme pathways and elementary flux modes to the minimal generating set.

Answering many questions requires the use of extreme pathways [27, 87, 101, 86, 88] and elementary flux modes [98, 1, 112, 4, 137, 138], however there are several applications one can answer with minimal generating sets. This situation especially arises in the case of genome-scale metabolic networks where the computation of elementary modes is prohibitively expensive [139]. Some simple structural properties may be observed, such as whether any reversible reaction appears only in one direction or only in irreversible pathways, or whether some reaction appears in no pathway at all. Flux coupling analysis, a procedure of determining dependencies between network reactions, can be accomplished using the minimal generating set vectors [140]. Control-effective analysis of individual reactions in the network was initially proposed on the basis of computed elementary flux modes [98]. However, minimal generating sets can be used to obtain an analogous control-effective metric, used in the regulatory network analysis and reaction importance assessment [139]. Minimal metabolic behaviors are exposed by the minimal generating set [141], but a simple method to compute it is still needed. In large genome-scale networks, where computation of entire minimal generating sets may be impractical, efforts have been made to compute the K-shortest minimal generating vectors [142] (i.e., pathways involving as few reactions as possible). This was accomplished by means of solving several linear optimization problems and using existing methods for the computation of K-shortest elementary flux modes [143].

It was earlier established that in the metabolic network which corresponds to the pointed cone the minimal generating set is computed using the Nullspace Algorithm after processing all the row constraints corresponding to the irreversible reactions [79]. However, in the case of metabolic network which admits reversible pathways, a said procedure may not yield the correct minimal generating set.

The problem of computing the minimal generating set for the metabolic network which admits reversible pathways is the topic of this section. An earlier analysis of the metabolic networks with reversible pathways by means of two subnetworks, one with no reversible pathways and one with all reversible pathways, can be found in [141]. The computation of the unique minimal generating set for a pointed cone can be accomplished using existing algorithms [118, 41, 144] or using the general paradigm in [130]. But this is considerably more difficult when the cone is not pointed (i.e., there are reversible pathways). This situation can be recognized by computing the rank of the submatrix of **S** consisting of the reversible reactions [130].

The major contribution of this section is to provide a simple procedure to compute the minimal generating set for a stoichiometric network which has reversible pathways. The method is based on combining two existing algorithms: a method to compute the minimal generating set for a pointed cone, and a method to compute a nullspace of a matrix based on the Reduced Row Echelon Form, a classical method in linear algebra. All this is carried out without the necessity to compute all the elementary flux modes for any network. This section is organized as follows. Subsection 3.4.2 gives a theoretical treatment of the representation of reversible and irreversible pathways and the decomposition of the original metabolic network into two subnetworks. Subsection 3.4.3 outlines the algorithm for the computation of the minimal generating set using two subnetworks. Subsection 3.4.4 uses a simple example to illustrate the method and show how the method exposes some of the structure of the network.

3.4.2 Theory

Let $\mathbf{S} = (\mathbf{A}, \mathbf{B}, \mathbf{C})$ be an $m \times n$ stoichiometry matrix with the n columns (reactions) ordered so that $\mathbf{A}$ consists of the irreversible reactions (of which there are n_i) and $\mathbf{B}, \mathbf{C}$ consists of the reversible reactions (of which there are n_r). We assume the reversible reactions $(\mathbf{B}, \mathbf{C})$ form a matrix of rank k_r and that $\mathbf{B}$ consists of k_r columns which are independent, while $\mathbf{C}$ consists of $n_r - k_r$ columns. This implies that all the columns of $\mathbf{C}$ can be written as linear combinations of the columns $\mathbf{B}$: $\mathbf{C} = \mathbf{B}R$ for some $k_r \times (n_r - k_r)$ coefficient matrix R. We remark that the columns of $\mathbf{B}$ can be found by a variety of methods such as the Reduced Row Echelon Form (RREF) [145] where they appear as the "pivot" columns, while the columns $\mathbf{C}$ appear as the "non-pivot" columns. Hence we will refer to $\mathbf{B}$ as the "pivot" columns. The standard RREF algorithm scans the matrix $\mathbf{S}$ left-to-right extracting independent columns $\mathbf{B}$, hence the choice of pivot columns varies depending on the order of columns (reactions) in the original $\mathbf{S}$, but once the latter is fixed, the former is also.

The matrix $(\mathbf{B}, \mathbf{C})$ has a nullspace of dimension $n_r - k_r$, and a suitable basis for this space is $N_R = \begin{pmatrix} -R \\ I \end{pmatrix}$. Any vector in this nullspace is a valid path for the subnetwork $(\mathbf{B}, \mathbf{C})$ and is a reversible path. By prepending zeros, we obtain

$$\widehat{N}_R = \begin{pmatrix} 0 \\ \\ N_R \\ \\ \end{pmatrix} = \begin{pmatrix} 0 \\ -R \\ I \end{pmatrix},$$

which we will show is a minimal basis for the set of all reversible paths in the original network.

A column vector $\mathbf{x}$ is a valid path of the network represented by stoichiometry matrix $\mathbf{S}$ if and only if $\mathbf{Sx} = \mathbf{0}$ and the entries of $\mathbf{x}$ corresponding to irreversible reactions are non-negative. If we split $\mathbf{x} = (\mathbf{x}_\mathrm{a}; \mathbf{x}_\mathrm{b}; \mathbf{x}_\mathrm{c})$ to conform with $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ (where ";" denote vertical concatenation *à la* Matlab), then $\mathbf{x}$ is a valid path if and only if $\mathbf{A}\mathbf{x}_\mathrm{a} + \mathbf{B}\mathbf{x}_\mathrm{b} + \mathbf{C}\mathbf{x}_\mathrm{c} = 0$ and $\mathbf{x}_\mathrm{a} \geq \mathbf{0}$ (elementwise).

We have the following Lemmas:

Lemma 1. Any reversible pathway $\mathbf{x}$ for the stoichiometry matrix $\mathbf{S} = (\mathbf{A}, \mathbf{B}, \mathbf{C})$ split as above can be written in terms of the minimal generating set for the reversible subnetwork $(\mathbf{B}, \mathbf{C})$, as follows:

$$\mathbf{x} \equiv \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{b} \\ \mathbf{x}_\mathrm{c} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{x}_\mathrm{b} \\ \mathbf{x}_\mathrm{c} \end{pmatrix} = \widehat{N}_R \boldsymbol{\alpha} \equiv \begin{pmatrix} \mathbf{0} \\ -R \\ I \end{pmatrix} \boldsymbol{\alpha}$$

for some coefficient vector $\boldsymbol{\alpha}$. ∎

Since there are no sign constraints in the subnetwork represented by $(\mathbf{B}, \mathbf{C})$, the basis N_R is the minimal generating set for all possible reversible paths. In fact any basis for the nullspace of $(\mathbf{B}, \mathbf{C})$ would be a minimal generating set, but we choose this specific one because each column in this basis has a minimal set of non-zeros, i.e., each is also an elementary flux mode. In this sense, we call this a “minimal basis” or “minimal generating set.” There is still freedom to choose any set of k_r independent columns of $(\mathbf{B}, \mathbf{C})$ (reversible reactions) to act as the basis $\mathbf{B}$.

Lemma 2. Any pathway $\mathbf{x}$ for the stoichiometry matrix $\mathbf{S} = (\mathbf{A}, \mathbf{B}, \mathbf{C})$ split as above can be written as the sum of a path involving just reactions indexing columns in $\mathbf{A}, \mathbf{B}$ and a reversible path involving just reactions indexing columns in $\mathbf{B}, \mathbf{C}$:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{b} \\ \mathbf{x}_\mathrm{c} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \tilde{\mathbf{x}_\mathrm{b}} \\ \mathbf{0} \end{pmatrix} + \widehat{N}_R \mathbf{x}_\mathrm{c}$$

where $\tilde{\mathbf{x}_\mathrm{b}} = \mathbf{x}_\mathrm{b} + R\mathbf{x}_\mathrm{c}$. ∎

Proof.

$$\mathbf{x} \equiv \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{b} \\ \mathbf{x}_\mathrm{c} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_\mathrm{a} & + & \mathbf{0} \\ \mathbf{x}_\mathrm{b} + R\mathbf{x}_\mathrm{c} & - & R\mathbf{x}_\mathrm{c} \\ \mathbf{0} & + & \mathbf{x}_\mathrm{c} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \tilde{\mathbf{x}_\mathrm{b}} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} 0 \\ -R \\ I \end{pmatrix} \mathbf{x}_\mathrm{c}.$$

As a valid path, $\mathbf{x}$ lies in the nullspace of $\mathbf{S}$, and so does the term $\widehat{N}_R \mathbf{x}_\mathrm{c}$, hence the remaining term $\begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \tilde{\mathbf{x}_\mathrm{b}} \\ \mathbf{0} \end{pmatrix}$ must also lie in the nullspace of $\mathbf{S}$ and hence is a valid path. Only the component $\mathbf{x}_\mathrm{a}$ is subject to sign constraints. □

Theorem 5. *Any pathway* $\mathbf{x}$ *for the stoichiometry matrix* $\mathbf{S} = (\mathbf{A}, \mathbf{B}, \mathbf{C})$ *split as above can be written as the sum of a linear combination of paths in the minimal generating set* $\widehat{M_I}$ *for the “pointed-cone” subnetwork represented by* $(\mathbf{A}, \mathbf{B})$ *together with a linear combination of the minimal generating set* $\widehat{N}_R$ *for the network of reversible reactions represented by* $(\mathbf{B}, \mathbf{C})$.

Proof. Let M_I be a matrix whose columns form the minimal generating set for the subnetwork $(\mathbf{A}, \mathbf{B})$. This network has no reversible pathways because the columns corresponding to the reversible reactions are linearly independent, i.e., the space of valid paths for this network is a pointed cone [80].

By Lemma 2, any path $\mathbf{x}$ through the entire network can be written as

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \tilde{\mathbf{x}_\mathrm{b}} \\ 0 \end{pmatrix} + \widehat{N}_R \mathbf{x}_\mathrm{c}.$$

Now $\begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \tilde{\mathbf{x}_\mathrm{b}} \end{pmatrix}$ is a path through the subnetwork $(\mathbf{A}, \mathbf{B})$ and hence can be written in terms of the minimal generating set for $(\mathbf{A}, \mathbf{B})$ as $M_I \boldsymbol{\beta}$ for some coefficient vector $\boldsymbol{\beta}$. Hence we have

$$\mathbf{x} = \widehat{M_I} \boldsymbol{\beta} + \widehat{N}_R \alpha, \quad \text{with} \quad \widehat{M_I} = \begin{pmatrix} M_I \\ 0 \end{pmatrix},$$

where we have extended M_I with a block of zeros so that $(\mathbf{A}, \mathbf{B}) \cdot M_I = (\mathbf{A}, \mathbf{B}, \mathbf{C}) \cdot \widehat{M_I}$. □

Lemma 3. The generating set of the stoichiometry matrix $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ formed as a union of the minimal generating sets of matrices $(\mathbf{A}, \mathbf{B})$ and $(\mathbf{B}, \mathbf{C})$ is minimal. ▮

Proof. Assume that the minimal generating set is given by the union $\widehat{M_I} \cup \widehat{N}_R$, and that the given pathway $\mathbf{x} = (\mathbf{x}_\mathrm{a}; \mathbf{x}_\mathrm{b}; \mathbf{x}_\mathrm{c})$ can be written as $\mathbf{x} = \widehat{M_I} \boldsymbol{\beta} + \widehat{N}_R \alpha$. By Lemma 2, given pathway can be written as $\mathbf{x} = \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \tilde{\mathbf{x}_\mathrm{b}} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} 0 \\ -R \\ I \end{pmatrix} \mathbf{x}_\mathrm{c}$, which implies that the coefficients of the vectors in $\widehat{N}_R$ in the decomposition are equal to the elements in $\mathbf{x}_\mathrm{c}$, hence $\alpha = \mathbf{x}_\mathrm{c}$. Since $\widehat{N}_R$ consists of the identity matrix in its lower part, vector $\mathbf{x} - \widehat{N}_R \mathbf{x}_\mathrm{c}$ lies in the pointed polyhedral cone corresponding to the matrix $(\mathbf{A}, \mathbf{B})$ and which has a unique minimal generating set $\widehat{M_I}$. Hence, the subset of minimal generating sets which is used in the decomposition in $\mathbf{x}$ is unique, and therefore minimal. □

3.4.3 Discussion

Theorem 5 implies that we can compute a minimal generating set for a stoichiometry matrix $\mathbf{S}$ by the following procedure.

1. Collect all columns corresponding to irreversible reactions of $\mathbf{S}$ into matrix $\mathbf{A}$.

2. Use the Reduced Row Echelon Form (RREF) (or similar method) on the matrix of all reversible reactions to extract the matrices $\mathbf{B}$ and $\mathbf{C}$, where $\mathbf{B}$ has full column rank and $\mathbf{C}$ can be written as $\mathbf{B}R$ for some R, so that $(\mathbf{B}, \mathbf{C})$ consists of the columns corresponding to all the reversible reactions.

3. Compute a basis N_R as the right nullspace of matrix $(\mathbf{B}, \mathbf{C})$.
4. Compute the minimal generating set M_I for the subnetwork represented by $(\mathbf{A}, \mathbf{B})$ (a pointed cone).
5. Extending M_I with a block of zeros to obtain $\widehat{M_I}$, we obtain a minimal generating set for all valid paths of $\mathbf{S}$, namely the columns of the combined matrix $(\widehat{M_I}, \widehat{N}_R)$.

This procedure allows one to compute the minimal generating set for an arbitrary network by computing the minimal generating sets of two subnetworks, one of which is a pointed cone and the other one having reversible pathways is a non-pointed cone. Since it is well known that the minimal generating set of a network is almost always an order of magnitude smaller than the set of elementary flux modes [141], this procedure allows one to compute the minimal generating set for an arbitrary network at much less cost compared to an algorithm based on a full set of elementary flux modes [136].

We remark that, by construction, the minimal generating set $\widehat{M_I}$, of the subnetwork consisting only of irreversible pathways, is essentially unique once the basis $\mathbf{B}$ for the space of reversible reactions is chosen. They vary only in the combinations of reversible reactions. The patterns of irreversible reactions in the pathways of $\widehat{M_I}$ correspond to the minimal metabolic behaviors of [141].

However, there is quite a large freedom of choice for the minimal generating set $\widehat{N}_R$ corresponding to the subnetwork with reversible pathways. Any basis for the nullspace of $(\mathbf{B}, \mathbf{C})$ will do. By using the nullspace derived from the RREF, we can ensure that each reversible pathway in $\widehat{N}_R$ is minimal (i.e., has a minimal set of non-zero entries).

3.4.4 Example

3.4.4.1 Toy metabolic network

We illustrate the method with a small example derived from [141]. We have made reactions $R6, R7$ reversible in order to illustrate some structure exposed by this method.

Stoichiometry Matrix **S**:

$$\mathbf{S} = \begin{array}{cccccccccccccc}
 & R1\text{r} & R2 & R3\text{r} & R4\text{r} & R5\text{r} & R6\text{r} & R7\text{r} & R8 & R9\text{r} & R10\text{r} & R11\text{r} & R12\text{r} & \\
\hline
 & 1 & -1 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & m1 \\
 & \_ & 1 & -1 & \_ & -1 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & m2 \\
 & \_ & 1 & \_ & -1 & \_ & -1 & \_ & \_ & \_ & \_ & \_ & \_ & m3 \\
 & \_ & \_ & \_ & \_ & 1 & \_ & \_ & 1 & -1 & \_ & -1 & \_ & m4 \\
 & \_ & \_ & \_ & \_ & \_ & 1 & -1 & -1 & \_ & \_ & \_ & \_ & m5 \\
 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & 1 & -1 & m6 \\
 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & 1 & -1 & \_ & \_ & m7 \\
\hline
 & \mathbf{B} & \mathbf{A} & \mathbf{B} & \mathbf{B} & \mathbf{B} & \mathbf{B} & \mathbf{C} & \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{B} & \mathbf{C} & \text{partition}
\end{array}$$

Here we show the partition into which each reaction ends up after extracting the reversible part and applying the RREF. The method consists of three steps as follows.

1. Get minimal generating set for reversible pathways subnetwork.

- Extract all columns from **S** corresponding to reversible reactions into matrix $(\mathbf{B}, \mathbf{C})$.
- Compute basis of right Nullspace of $(\mathbf{B}, \mathbf{C})$ using the Reduced Row Echelon Form (RREF). The rank of the matrix $(\mathbf{B}, \mathbf{C})$ is 7, so there will be 7 pivot columns. The dimension of the nullspace is 3, so the minimal generating set will have 3 entries.
- The pivot reactions correspond to what RREF identified as the pivot columns. The pivot columns form a basis for the entire column space of the reversible columns of **S**.

The columns have been ordered so that the pivot reversible reactions are: $\{R1r$, $R3r$, $R4r$, $R5r$, $R6r$, $R9r$, $R11r\}$. The resulting reversible minimal generating set has 3 entries labeled R in formula (3.13) below. We remark that this minimal generating set is not unique, as discussed further below. In the following paragraphs, we show how the choice of pivot reversible reactions is not completely arbitrary, but depends on the particular partitioning among all the reversible reactions.
Notice in the R columns of (3.13) how $R1r$ is isolated, and $R4r$, $R6r$, $R7r$ form combination disjoint from the remaining reactions. Specifically, the first R column shows that the sum of $R6r$, $R7r$, minus $R4r$ is zero. No R column involves $R1r$. Hence $R1r$ cannot be written as a linear combination of any other reaction, and any one of $R4r$, $R6r$, $R7r$ can be written as a linear combination of the other two, but none of these four reactions can be written as combinations of the remaining reversible reactions. This also shows that $R1r$ is not involved in any reversible pathway.

Regarding the irreversible reactions, there is one irreversible pathway containing just $R2$ and one containing just $R8$, hence those two correspond to the minimal metabolic behaviors in the sense of [141] [1] . Since $R1r$ only appears in the positive direction in a pathway also with $R2$, one might consider $R1$ to be irreversible with respect to this network and the two minimal metabolic behaviors to be $\{R1, R2\}$ and $\{R8\}$.

The R columns of (3.13) is a generalized incidence matrix where two reactions are connected by an edge if they appear in a common path. This shows the set of reversible reactions can be partitioned into three disjoint connected subgraphs: $\{R1r\}$, $\{R4r$, $R6r$, $R7r\}$, $\{R3r$, $R5r$, $R9r$, $R10r$, $R11r$, $R12r\}$, with no connections between the subgraphs. Any pointed cone subset must have a certain number of members from each connected subgraph, where the number of members is equal to the rank of the corresponding columns of the stoichiometry matrix.

So any pointed cone subset in this example must include (in addition to all the irreversible reactions) $R1r$, and any 2 out of $\{R4r$, $R6r$, $R7r\}$, plus some 4 out of $\{R3r$, $R5r$, $R9r$, $R10r$, $R11r$, $R12r\}$ but not any combination. Varying the choice of reversible reactions not only yields a different pointed cone, but also a different minimal generating set for the reversible subnetwork. However, once the minimal generating set for the reversible part is chosen, the rest is all uniquely determined.

2. Get minimal generating set for irreversible pathways subnetwork.

- Combine irreversible reactions with the pivot reversible reactions to form a subnetwork with no reversible pathway (called a Pointed Cone Subset). In this example there are 2 irreversible reactions and 7 pivot reactions to form a subnetwork of 9 reactions.
- Compute minimal generating set for resulting reduced stoichiometry matrix $(\mathbf{A}, \mathbf{B})$ (using usual Nullspace algorithm).

The pointed cone subset has 9 reactions: $\{R2$, $R8\}$, $\{R1r\}$, $\{R4r$, $R6r\}$, $\{R3r$, $R5r$, $R9r$, $R11r\}$ (includes all irreversible reactions plus "pivot" reversible reactions, grouped by partitioning of the reactions induced from above.)

Applying the Nullspace algorithm yields 2 minimal generating vectors labeled I in formula (3.13).

3. Combine the above two minimal generating sets.

All 5 paths in the two computed minimal generating set for this example are combined to represent the minimal generating set of the original metabolic network, a shown in (3.13).

[1] Recall this network has been modified from that of [141] for purposes of illustration

$$\begin{array}{rr|rrrl}
I & I & R & R & R & \\
\hline
\_ & 1 & \_ & \_ & \_ & R1r \\
\_ & 1 & \_ & \_ & \_ & R2 \\
1 & 1 & \_ & -1 & -1 & R3r \\
-1 & 1 & -1 & \_ & \_ & R4r \\
-1 & \_ & \_ & 1 & 1 & R5r \\
1 & \_ & 1 & \_ & \_ & R6r \\
\_ & \_ & 1 & \_ & \_ & R7r \\
1 & \_ & \_ & \_ & \_ & R8 \\
\_ & \_ & \_ & 1 & \_ & R9r \\
\_ & \_ & \_ & 1 & \_ & R10r \\
\_ & \_ & \_ & \_ & 1 & R11r \\
\_ & \_ & \_ & \_ & 1 & R12r
\end{array} \tag{3.13}$$

The label I denotes an irreversible minimal generating vector derived from the pointed cone subnetwork $(\mathbf{A}, \mathbf{B})$, and R denotes a reversible minimal generating vector derived from the reversible subnetwork $(\mathbf{B}, \mathbf{C})$.

This system has 21 EMs consisting of the 5 members of the minimal generating set (3.13) plus the following 16 EMs:

$$\begin{array}{rrrrrrrrrrrrrrrrl}
R & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \\
\hline
\_ & \_ & 1 & 1 & 1 & 1 & \_ & \_ & 1 & 1 & 1 & 1 & \_ & \_ & 1 & 1 & R1r \\
\_ & \_ & 1 & 1 & 1 & 1 & \_ & \_ & 1 & 1 & 1 & 1 & \_ & \_ & 1 & 1 & R2 \\
\_ & 1 & 2 & 1 & \_ & 1 & \_ & \_ & \_ & \_ & \_ & 1 & \_ & \_ & \_ & \_ & R3r \\
\_ & \_ & \_ & \_ & 1 & \_ & -1 & \_ & \_ & \_ & 1 & \_ & -1 & \_ & \_ & \_ & R4r \\
\_ & -1 & -1 & \_ & 1 & \_ & \_ & \_ & 1 & 1 & 1 & \_ & \_ & \_ & 1 & 1 & R5r \\
\_ & \_ & 1 & 1 & \_ & 1 & 1 & \_ & 1 & 1 & \_ & 1 & 1 & \_ & 1 & 1 & R6r \\
\_ & -1 & \_ & 1 & \_ & \_ & \_ & -1 & 1 & \_ & \_ & \_ & \_ & -1 & 1 & \_ & R7r \\
\_ & 1 & 1 & \_ & \_ & 1 & 1 & 1 & \_ & 1 & \_ & 1 & 1 & 1 & \_ & 1 & R8 \\
-1 & \_ & \_ & \_ & 1 & 1 & 1 & 1 & 1 & 2 & \_ & \_ & \_ & \_ & \_ & \_ & R9r \\
-1 & \_ & \_ & \_ & 1 & 1 & 1 & 1 & 1 & 2 & \_ & \_ & \_ & \_ & \_ & \_ & R10r \\
1 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & 1 & 1 & 1 & 1 & 1 & 2 & R11r \\
1 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & 1 & 1 & 1 & 1 & 1 & 2 & R12r
\end{array} \tag{3.14}$$

The third column of (3.14) is an elementary mode for the pointed cone subset $(\mathbf{A}, \mathbf{B})$.

To see that (3.13) is indeed the minimal generating set, we express all the EMs in (3.14) as linear combinations of the paths in (3.13):

$$\begin{array}{cccccccccccccccc l}
\_ & 1 & 1 & \_ & \_ & 1 & 1 & 1 & \_ & 1 & \_ & 1 & 1 & 1 & \_ & 1 & (\mathbf{A},\mathbf{B}) \\
\_ & \_ & 1 & 1 & 1 & 1 & \_ & \_ & 1 & 1 & 1 & 1 & \_ & \_ & 1 & 1 & (\mathbf{A},\mathbf{B}) \\
\_ & -1 & \_ & 1 & \_ & \_ & \_ & -1 & 1 & \_ & \_ & \_ & \_ & -1 & 1 & \_ & (\mathbf{B},\mathbf{C}) \\
-1 & \_ & \_ & \_ & 1 & 1 & 1 & 1 & 1 & 2 & \_ & \_ & \_ & \_ & \_ & \_ & (\mathbf{B},\mathbf{C}) \\
1 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & 1 & 1 & 1 & 1 & 1 & 2 & (\mathbf{B},\mathbf{C})
\end{array}$$

3.4.4.2 Red Blood Cell metabolism

We can apply the algorithm to find minimal generating set for the concrete example of the red blood cell metabolic network [86]. This is also just a simple illustration of a simple biochemical question that can be answered with the minimal generating set: namely that certain reversible reactions operate only in one direction. This network has 32 internal reactions (17 reversible) and 19 external reactions (16 reversible), giving a total of 51 reactions (33 reversible). The network has 6,180 elementary flux modes, but the algorithm of this section shows only 18 of these modes form the minimal generating set (with 1 reversible). In this example **A** consists of the 18 columns corresponding to the 18 irreversible reactions, **B** consists of 32 columns, and **C** has one column. The one reversible pathway is found to consist of reactions `LD`, `TRA3r`, `TRA4r`, `TRA10r`, `TRA11r`, `TRA14r`. A simple examination of the minimal generating set suffices to show that the following reversible reactions appear only in the positive direction: `ALD`, `TPI`, `GAPDH`, `PGM`, `EN`, `LD`, `PGL`, `PRM`, `PNPase`, `TRA4r`, `TRA13r`, `TRA16r`, `TRA17r`, and the following only in the negative direction: `TRA7r`, `TRA12r`, `TRA14r`.

Supplementary MATLAB script which facilitates the computation of minimal generating set using the software for the computation of elementary flux modes is given at `http://elmocomp.sourceforge.net/mingen.zip`. Software for the computation of elementary flux modes has a command line option to perform the processing of constraints only corresponding to irreversible reactions.

3.5 Conclusion

Continuing on the previous development and results in the pathway-based analysis of the metabolic networks this chapter aims to point out the major bottlenecks of the Nullspace Algorithm and possibilities for its improvement. First, the serial Nullspace Algorithm is laid out and decomposed into routines so that it may later be reused in the parallelization. Second, a reduced algebraic rank test is proposed for the case when Nullspace Algorithm is run on a metabolic network without decomposing reversible reactions into irreversible components. Third, an alternative and simple procedure is described which demonstrates how to compute the minimal generating set in a metabolic network which admits reversible metabolic pathways. The approach is based on splitting a metabolic network into two subnetworks, one entirely reversible, and the other without reversible pathways.

Chapter 4

Combinatorial Parallelization of the Nullspace Algorithm

4.1 Introduction

For the metabolic networks which after compression have the number of both metabolites and reactions on the order of $10^2 - 10^3$, the existing software is unable to complete the computation of the elementary flux modes. Thus, in this chapter we resort to the idea of parallelizing the Nullspace Algorithm. Section 4.2 describes the *Combinatorial Parallel Nullspace Algorithm*, named due to the idea of parallelizing the portion of the algorithm which generates the candidate elementary flux modes and which exhibits combinatorial explosion in the amount of candidate modes which are probed for elementarity. Later, section 4.3 shows the results of running the implementation of the Combinatorial Nullspace Algorithm on metabolic networks of different sizes and discusses the existing problems and limitations.

4.2 Parallel Nullspace Algorithm

We assume that the algorithm is designed for a parallel environment of P compute-nodes, where each compute-node refers to an abstract processor, has its own memory and executes an instance of the parallel program in the message-passing distributed memory communication environment. The compute-nodes exchange messages over an unspecified network architecture. This parallel environment corresponds to a distributed-memory system, though our proposed algorithm may be easily expanded into hybrid parallel implementation with the shared-memory paradigm.

Algorithm 4 illustrates the Combinatorial Parallel Nullspace Algorithm named so as the major idea is to parallelize the generation of the candidate elementary modes (Algorithm

3). The portion of the algorithm which introduces the communication among the compute-nodes is given in line 10. We parallelize the tasks of generating candidate columns as in Algorithm 5 and by proper load balancing attain that each participating compute-node generates approximately the same number of candidate columns.

Algorithm 4 $[R] = \texttt{CombParallelNullspaceAlg}(N, R)$

```
Input:
1: reduced stoichiometry matrix - $N_{m\times q}$
2: initial nullspace matrix - $R_{q\times(q-m)} = \begin{bmatrix} R^{(\mathsf{bit})}_{(q-m)\times(q-m)} \\ R^{(\mathsf{real})}_{m\times(q-m)} \end{bmatrix} = \begin{bmatrix} I_{(q-m)\times(q-m)} \\ R^{(\mathsf{real})}_{m\times(q-m)} \end{bmatrix}$
Output:
3: bit-valued matrix of elementary modes - $R_{q\times n_{ems}}$
4: for $k = q - m + 1$ to $q$ do
5:     $combinations \Leftarrow \texttt{ParallelGenerateEFMCands}(R)$
6:     $combinations \Leftarrow \texttt{RadixSortColumnModes}(R^{(\mathsf{bit})}, combinations, width)$
7:     $combinations \Leftarrow \texttt{UniqueColumnModes}(R^{(\mathsf{bit})}, combinations)$
8:     $combinations \Leftarrow \texttt{RankTests}(N, R^{(\mathsf{bit})}, combinations)$
9:     ▷ communicate columns and merge
10:    $combinations \Leftarrow \texttt{Communicate\&Merge}(R^{(\mathsf{bit})}, combinations)$
11:    $R \Leftarrow \texttt{ExpandEfm}(R, combinations)$
12: end for
```

Load balancing is needed to assure that there is no serious time discrepancy among the compute-nodes when they perform the sorting and the evaluation of numerical rank tests. Each compute-node generates its share of candidate elementary mode columns, and filters those which are valid elementary modes at the given iteration according to the same criteria as in serial Nullspace Algorithm. However, different compute-nodes may generate identical candidate elementary mode columns, and compute-nodes will have to communicate to remove these duplicated bit-columns. The result of communication among compute-nodes is the complete set of elementary modes after processing the k^{th} reaction. In a carefully designed communication pattern, compute-nodes would exchange their generated elementary modes, and each compute-node would merge the arrays of bit-columns obtained from other compute-nodes with its local set of elementary mode columns. The disadvantage of this approach, which we have also implemented, is in the `ALL-TO-ALL` merge and communication pattern. The cost of communication among compute-nodes is negligible compared to the total cost of the merge and elimination of duplicated elementary modes which is performed on the compute-nodes locally. In subsection 4.2.2 we analyze the complexity and give an improved communication algorithm for exchange of candidate elementary modes among compute-nodes and efficient merge. Subroutines for sorting, elimination of duplicated candidates, and rank tests remain unmodified from the serial Nullspace Algorithm.

Algorithm 5 $[combinations] = \texttt{ParallelGenerateEFMCands}(R)$

```
Input:
 1: $current\ nullspace\ matrix$ - $R_{q\times n_{ems}} = \begin{bmatrix} R^{(\mathsf{bit})} \\ R^{(\mathsf{real})} \end{bmatrix}$
Output:
 2: $pairs\ of\ indices\ of\ columns\ forming\ candidates$ - $combinations$
 3: $irrev^{+} \Leftarrow \{i : R^{(\mathsf{real})}_{1,i} > 0\ \&\ (\exists j : j^{th}$ reaction is irreversible, $R^{(\mathsf{bit})}_{j,i} \neq 0)\}$
 4: $irrev^{-} \Leftarrow \{i : R^{(\mathsf{real})}_{1,i} < 0\ \&\ (\exists j : j^{th}$ reaction is irreversible, $R^{(\mathsf{bit})}_{j,i} \neq 0)\}$
 5: $rev \Leftarrow \{i : R^{(\mathsf{real})}_{1,i} \neq 0\ \&\ (\forall j : j^{th}$ reaction is reversible or $R^{(\mathsf{bit})}_{j,i} = 0)\}$
 6: $irrev\_p^{-} \Leftarrow \{i : i \in irrev^{-}\ \&\ i = proc\_id(\mathbf{mod} P)\}$
 7: $rev\_p \Leftarrow \{i : i \in rev\ \&\ i = proc\_id(\mathbf{mod} P)\}$
 8: $S \Leftarrow \{(ii,jj) : (ii,jj) \in (irrev^{+} \times irrev\_p^{-}) \cup ((irrev^{+} \cup irrev\_p^{-} \cup rev\_p) \times rev)\}$
 9: for each $(ii,jj) \in S$ do
10:     form candidate column from the pair of columns indexed by $(ii,jj)$
11:     if candidate satisfies Theorem 4 add to combinations
12: end for
```

4.2.1 Load Balancing

As shown in lines 6-7 of Algorithm 5 we partition the arrays of indices of columns of matrix `irrev`$^{-}$ and `rev` among the compute-nodes evenly. However, since the $R^{(\mathsf{bit})}$ bit-matrix remains in sorted order at the beginning of each iteration, the generated candidate elementary mode bit-columns at every compute-node may have non-uniform overall density of non-zero entries. This imbalance would occur if we assigned to each compute-node the contiguous range of indices from arrays `irrev`$^{-}$ and `rev`. If this was the case, compute-nodes would generate the set of candidate columns of non-uniform "sparsity" and thus produce an unequal number of candidate columns which satisfy Theorem 4. This would result in the poor load balancing in the sections of the algorithm corresponding to the "sort & removal of duplicated columns" and "rank tests of candidate columns". As a solution to this problem, we assigned to each compute-node the set of indices from both `irrev`$^{-}$ and `rev` which have values congruent to the compute-node identifier modulo total number of compute-nodes P, as illustrated in Algorithm 5.

The comparison between "sequential" and "interleaved" generation of candidate columns is given in Table 4.1. The imbalance rate in the two sections of algorithm across P compute-nodes is used as a measure, as given in equation (4.1).

$$ImbalanceRate(task) = \frac{\max_{1\leq i\leq P} T^{(i)}_{task}}{\min_{1\leq j\leq P} T^{(i)}_{task}} \tag{4.1}$$

where $task$ corresponds to the "sort & removal of duplicated columns" or "rank tests of candidate columns", while $T^{(i)}_{task}$ is the time i^{th} compute-node spent performing the $task$.

Table 4.1: *Imbalance rate of interleaved and sequential generation of candidates.*

		number of compute-nodes				
		2	4	8	16	32
sequential	sort & removal of duplicated columns	1.91	2.55	4.75	6.49	14.57
	rank tests of candidate columns	2.04	2.97	5.35	10.13	34.34
interleaved	sort & removal of duplicated columns	1.00	1.03	1.03	1.08	1.10
	rank tests of candidate columns	1.02	1.02	1.04	1.07	1.12

4.2.2 Computational Complexity Analysis

In order to estimate the complexity of the Combinatorial parallel Nullspace Algorithm, we have to include the computational complexity term corresponding to the communication among compute-nodes. We try to attain the load balanced situation where every compute-node approximately generates the same number of elementary flux modes as described in subsection 4.2.1. Initially, we implemented the `ALL-TO-ALL` broadcast communication pattern in the environment of P compute-nodes. The network parameters given are the latency t_s and the per-word transfer time t_w [146]. The per-word transfer time is inversely proportional to the available bandwidth between the compute-nodes. Every compute-node generates candidate elementary modes, validates that they represent admissible elementary modes by means of the numerical rank test, and communicates them to the other compute-nodes to eliminate the duplicate columns and merge. The elementary mode columns sent between compute-nodes are in sorted order, and only a proper merge subroutine is needed to eliminate duplicates. In the `ALL-TO-ALL` communication, every compute-node broadcasts its local set of elementary mode columns it generated to all other compute-nodes, and each compute-node does the same task of merging the received sorted columns and eliminates the duplicates from it. Note that the elementary mode columns are communicated as pairs of indices of current nullspace matrix and not as full bit-columns, for the reason of more compactness. At the end of this communication, every compute-node will have the same result, i.e. the complete nullspace matrix of the elementary flux modes at the end of current iteration of the Nullspace Algorithm. For network architectures of ring, 2D-mesh, and hypercube the cost of `ALL-TO-ALL` communication, if we assume that each compute-node has to send the message of approximately the same size M, can be estimated [146]. In the case when very large messages are sent over the network, what is the case in our algorithm, the cost may be approximated as

$$T_{comm}^{all-to-all}(M,P) = \mathcal{O}(t_w M(P-1)), \qquad (4.2)$$

where M is the message length measured in units of pairs of indices being sent over the network, and P is the number of participating compute-nodes. This approximation remains the same, irrespective of the network architecture [146].

In order to sort the received messages, each compute-node has to merge $P-1$ received messages. In each merge, duplicates are being eliminated. Let t_c be the per unit of operation cost in the merge procedure. The computational cost of a single merge of two sorted arrays of length len_1 and len_2, is equal to $t_c(len_1 + len_2)$. We can only give an upper bound on the complexity of this merge task at a single compute-node, as follows:

$$T_{merge}^{all-to-all}(M,P) = t_c 2M + t_c 3M +, \ldots, + t_c(P-1)M = t_c((P-1)P/2-1)M = \mathcal{O}(P^2 M). \tag{4.3}$$

We notice in the case of good load balancing, the product PM remains the same for the given k^{th} iteration as the number of compute-nodes P grows. Accordingly, we note that while the cost of communication will remain the same, the cost of merging the received messages will grow with P. Therefore, this would require the re-design of the communication and merge pattern.

We may reduce the cost of merging the received messages with an alternative communication and merge pattern which corresponds to the hypercube communication. It may be illustrated with a `MERGE-TREE` communication and merge, as a complete binary tree of height $\log P$ and P leaf nodes, where P corresponds to the total number of compute-nodes. The complete binary tree nodes at each level of the tree correspond to those compute-nodes which are being used in the current iteration. We may equally use the term hypercube or tree since the tree may be embedded in a $\log P$-dimensional hypercube almost symmetrically [146]. For convenience we will refer to the `MERGE-TREE` communication and merge in the rest of this section. In the first phase, there will be $\log P$ iterations of unidirectional point-to-point communication among pairs of compute-nodes on the same level of the tree. At the k^{th} iteration ($k \in \{1, \ldots, \log P\}$), each compute-node i such that $i = 0\ (\mathbf{mod}\ 2^k)$ will receive the message from compute-node $j = i+2^{k-1}$. Approximately, the size of the message sent will be of length $2^{k-1}M$. The cost of each iteration has an upper bound equal to the value of merging two messages of length $2^{k-1}M$, i.e. $t_c 2^k M$. At the end of the first phase, the resulting nullspace matrix will be contained in compute-node 0. We assume that the number of compute-nodes P is a power of two, in order to maintain a complete binary tree. Accordingly, we assume that due to proper load balancing, prior to communication each compute-node has precomputed approximately M elementary mode columns and needs to distribute them to other compute-nodes for merge and elimination of duplicates. The cost of this merge operation may be expressed as:

$$\begin{aligned} T_{merge}^{merge-tree}(M,P) &= t_c(2M) + t_c(2^2M) +, \ldots, + t_c(2^k M) \\ &= t_c(2(2^k-1)M) = t_c(2(P-1)M) = \mathcal{O}(PM) \end{aligned} \tag{4.4}$$

Hence, when compared to the result in equation (4.3), the cost of merging given in equation (4.4) is reduced by the factor of P. Since the product PM is constant as P scales, the cost of merge will remain constant as well for the particular iteration of the algorithm.

Algorithm 6 $[combinations] = \texttt{Communicate\&Merge}(R, combinations)$

```
Input:
 1: current nullspace matrix - $R_{q\times n_{ems}} = \begin{bmatrix} R^{(\mathsf{bit})} \\ R^{(\mathsf{real})} \end{bmatrix}$
 2: local set of pairs of column indices which generate new candidates - combinations
Output:
 3: merged set of column-generating pairs of indices - combinations
 4: $proc\_id \Leftarrow$ identifier of the local compute-node
 5: for $i = 1$ to $\log P$ do
 6:     if $proc\_id = 0 \pmod{2^i}$ then
 7:         receive columns from compute-node $proc\_id + 2^{i-1}$
 8:         merge the local set of columns with the received columns
 9:     else
10:         send columns to compute-node $proc\_id - 2^{i-1}$
11:     end if
12: end for
13: if $proc\_id = 0$ then
14:     broadcast the columns to all other compute-nodes
15: end if
```

Apart from estimating the cost of merge, we estimate the cost of `MERGE-TREE` communication across the network. In every k^{th} iteration the cost of exchanging a message of size $2^{k-1}M$ between two compute-nodes is equal to $t_s + t_w 2^{k-1}M$ [146]. The cost of `ONE-TO-ALL` broadcast from the compute-node 0 after all data is merged is equal to $(t_s + t_w PM)\log P$, and thus the total communication cost may be estimated as:

$$\begin{aligned} T_{comm}^{merge-tree}(M,P) &= \left(\sum_{k=1}^{\log P} t_s + t_w 2^{k-1}M\right) + (t_s + t_w PM\log P) \\ &= t_s\log P + t_w M(2^{\log P} - 1) + t_s + t_w MP\log P \\ &= t_s(\log P + 1) + t_w(M(2^{\log P} - 1) + MP\log P) \\ &= t_s(\log P + 1) + t_w(M(P-1) + MP\log P) \\ &= t_w(MP\log P) + \mathcal{O}(t_w MP) \end{aligned} \tag{4.5}$$

The last approximation follows from the assumption that start up time is much smaller than the per-word transfer time [146]. Accordingly, we conclude that the communication cost will grow with a factor of $\log P$, unlike in (4.2) where it remains unchanged.

However, the cost estimates just given are upper bounds. The final set of merged columns which are broadcasted from compute-node 0 may be significantly smaller, because a large share of duplicated elementary mode columns are eliminated before the broadcast. In the experimental results on the computing platforms which were used to test the software, the communication time was negligible compared to the total time required to merge and eliminate duplicates at each compute-node, as will be shown later.

4.3 Results and Discussion

We present the computational times obtained with both the serial and Combinatorial Parallel Nullspace Algorithm. We plot the runtime the metabolic networks of the central metabolism of *Escherichia coli* and *S. cerevisiae* using METATOOL v5.1 [90, 118], our serial implementation ElMo-Comp and EFMTools [41]. During the run of the parallel implementation we time its execution and observe the scalability as the number of compute-nodes grows. For both serial and parallel implementation we use the Template Numerical Toolkit [147] from the National Institute of Technology and the C++ library of linear algebra functions adapted from the Java Matrix Library [148] developed by Mathworks and NIST.

We time the results of our parallel program on the "Calhoun" computing platform of the Minnesota Supercomputing Institute. In the following discussion, we use the terms compute node, processor and core, to describe the hardware.

"Calhoun" has 512 Intel Xeon 5355 (Clovertown) class multi-chip modules (MCMs). Each MCM is composed of two dies. These dies are two separate pieces of silicon connected to each other and arranged on a single module. Each die has two processor cores that share a 4 MB L2 cache. Each MCM communicates with the main memory in the system via a 1,333 MHz front-side bus (FSB). "Calhoun" is configured to have 256 compute nodes, 2 interactive nodes, 5 server nodes, total of 2048 cores, 4TB total main memory. Each node within the system has two quad-core 2.66 GHz Intel Xeon (Clovertown) - class processors and 16GB memory running at 1,333 MHz. All of the systems within Calhoun are interconnected with a 20-gigabit non-blocking InfiniBand fabric used for interprocess communication (IPC). The InfiniBand fabric is a high-bandwidth, low-latency network, the intent of which is to accommodate high-speed communication for large MPI jobs. The nodes are also interconnected with two 1-gigabit ethernet networks for administration and file access, respectively. The parallel program was compiled with Intel C++ compiler and OpenMPI on "Calhoun" platform.

4.3.1 Serial program

The serial program was run and timed on Intel Pentium D CPU 3GHz, dual-core, with 2GB main memory. Algorithm uses metabolic network of the central metabolism of *E. coli* as input, from which five test cases are derived differing by substrate metabolites (e.g. glycerol, glucose, xylose, etc) and reversibility of certain reactions, as shown in Table 4.2. Table 4.2 illustrates the results of running METATOOL, our implementation ElMo-Comp, and EFMTools. As pointed out earlier [40] and in subsection 2.3.1, the compression of the stoichiometric matrix is important in reducing the computational cost. EFMTools and our ElMo-Comp perform the identical iterative compression procedure of the given metabolic network, while METATOOL does not. A major bottleneck of the serial program is in the generation of candidate elementary modes described in Algorithms 3 and 5, followed to smaller extent by the evaluation of the algebraic rank tests on submatrices corresponding to the candidate elementary modes.

Table 4.2: *Result of running serial Nullspace Algorithm on the central metabolic network of E. coli*

network		substrate	Time (sec)			
original[1]	compressed		METATOOL 5.1	ElmoComp	EFMTools	#EFM
$41 \times 61(19)$	$26 \times 40(12)$	Gluc	16	2.65	4.89	38,002
$49 \times 64(19)$	$26 \times 41(12)$	Gluc, Xyl	73	11.64	14.36	92,594
$50 \times 66(19)$	$27 \times 43(13)$	Gluc, Glyc, Xyl	195	39.51	49.04	188,729
$50 \times 66(28)$	$29 \times 45(19)$	Gluc, Glyc, Xyl	NC[2]	1372.77	929.94	1,224,785

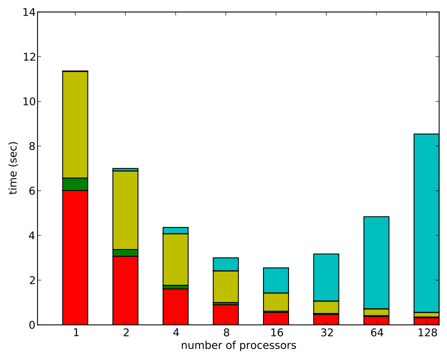
[1] dimensions of stoichiometry matrix; number of reversible reactions given in parentheses

[2] NC (computation did not complete)

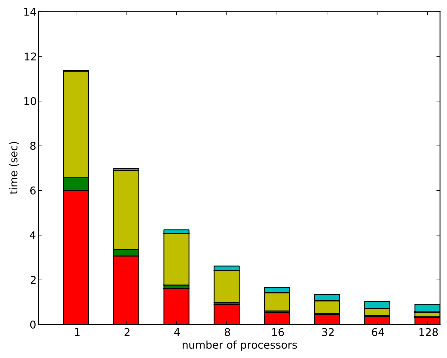
4.3.2 Parallel program

To demonstrate the improved runtime of the Combinatorial Parallel Nullspace Algorithm (Alg. 4) when using `MERGE-TREE` communication pattern over the `ALL-TO-ALL` we use the metabolic network of *E. coli* on glucose substrate which produces 38,001 elementary modes. With the `ALL-TO-ALL` communication and merge implemented there was an increase in the cost of merge proportional to the increase of the number of participating processors P, while in the case of `MERGE-TREE` pattern the cost becomes negligble when run on the 61-reaction metabolic network of *E. coli* using glucose substrate (Figure 4.1). This is consistent with our theoretical prediction that the `MERGE-TREE` communication and merge pattern reduces the overhead.

Further, the Combinatorial Parallel Nullspace Algorithm is run on three metabolic networks for *E. coli* from Table 4.2 having 92,594 , 188,729 and 1,224,785 elementary modes and on one 80-reaction metabolic network of the central metabolism of *S. cerevisiae.*



(a) E. coli 41×61 (19 rev.)



(b) E. coli 41×61 (19 rev.)

Figure 4.1: *Combinatorial Parallel Nullspace Algorithm run for (a)* `ALL-TO-ALL` *and (b)* `MERGE-TREE` *communication and merge pattern as computed using "Calhoun" parallel platform.*

Table 4.3: *Results for Combinatorial Parallel Nullspace Algorithm on Intel Xeon (Clovertown) machine for E. coli metabolic networks.*

		Time (*sec*)								#EFM
		1p	2p	4p	8p	16p	32p	64p	128p	
original	gen. cand.	13.45	7.12	3.73	1.92	1.28	1.21	1.17	0.81	92594
$49 \times 64(19)$[1]	sorting . . .	0.84	0.42	0.33	0.10	0.05	0.05	0.02	0.01	
compressed	rank tests	2.55	1.98	1.35	0.89	0.55	0.39	0.30	0.10	
$26 \times 41(12)$[2]	comm	0.00	0.01	0.01	0.01	0.01	0.02	0.04	0.08	
	merge	0.00	0.02	0.05	0.05	0.05	0.06	0.06	0.07	
	total	17.10	9.72	5.65	3.09	2.16	2.01	1.92	1.39	
original	gen. cand..	46.99	23.86	11.95	6.39	3.73	2.37	1.32	0.73	188729
$50 \times 66(19)$	sorting . . .	2.94	1.48	0.82	0.55	0.27	0.11	0.06	0.03	
compressed	rank tests	8.15	6.27	4.27	2.74	1.54	0.90	0.69	0.47	
$27 \times 43(13)$	comm	0.00	0.01	0.02	0.05	0.05	0.06	0.06	0.08	
	merge	0.00	0.05	0.08	0.09	0.10	0.11	0.11	0.12	
	total	58.90	32.35	17.71	10.31	6.57	3.91	2.31	1.63	
original	gen. cand	2189.32	1077.90	538.30	268.93	135.53	67.48	37.35	21.25	1224785
$50 \times 66(28)$	sorting . . .	84.58	26.60	14.07	10.84	5.55	1.99	1.35	1.32	
compressed	rank tests	91.60	70.40	48.58	30.57	17.89	10.06	5.27	2.85	
$29 \times 45(19)$	comm	0.	0.06	0.14	0.3	0.27	0.28	0.31	0.4	
	merge	0.	0.80	1.26	1.42	1.47	1.56	1.67	1.79	
	total	2381.49	1185.06	609.42	318.80	166.29	86.30	50.97	36.16	

[1] dimensions of stoichiometry matrix of the metabolic network; number of reversible reactions given in brackets

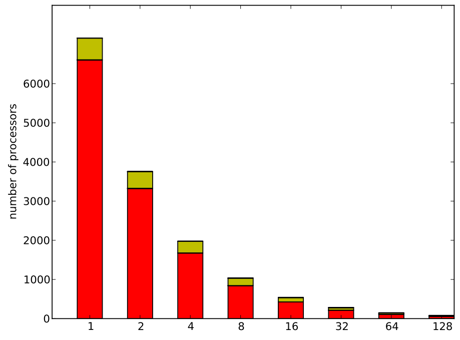
[2] dimensions of stoichiometry matrix of the reduced metabolic network

Table 4.4: *Combinatorial Parallel Nullspace Algorithm on Intel Xeon (Clovertown) machine for S. cerevisiae metabolic network.*

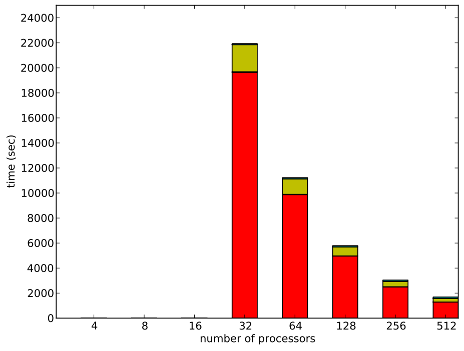
		Time (*sec*)				#EFM
		8p	16p	32p	64p	
original	gen. cand.	32,925.95	15,748.15	8,507.20	5,443.79	13,322,495
$62 \times 80(31)$	sorting	281.13	249.14	85.90	30.68	
compressed	rank tests	1,026.18	640.23	368.33	208.06	
$38 \times 58(20)$	comm	2.25	1.70	1.50	1.15	
	merge	32.94	47.30	37.83	23.54	
	total	34,339.19	16,762.73	9,055.62	5,752.75	
	relative CPU power ratio[1]	1.0	0.976	1.055	1.340	

[1] relative CPU power ratio = $\frac{total\ time \times number\ of\ processors}{total\ time\ on\ 8p \times 8}$

The timing results obtained using "Calhoun" computing platform are given in the tables 4.3 and 4.4. Both tables contain the results for the more efficient `MERGE-TREE` communication and merge pattern. Within the tables, the metabolic networks are annotated with the size of their original and compressed stoichiometry network accompanied with the number of reversible reactions (in the parentheses), since the core Nullspace Algorithm accepts the compressed stoichiometric network as input. Figure 4.2(a) gives the diagram for the parallel program over the *E. coli* 50×66 (28) network, while the figure 4.2(b) gives the similar diagram for the computation from table 4.4 corresponding to 80-reaction *S. cerevisiae* metabolic network.



(a) E. coli 50×66 (28 rev.)



(b) S. cerevisiae 62×80 (31 rev.)

Figure 4.2: *Combinatorial Parallel Nullspace Algorithm using* `MERGE-TREE` *communication pattern on E. coli and S. cerevisiae metabolic networks. Computation was run on "Calhoun" parallel platform.*

Timing results in tables 4.3 for *E. coli* 50×66 (28 rev.) and in table 4.4 for *S. cerevisiae* illustrate that the generation of candidate modes scales well with the increasing number of used processors, while that may not be the case for evaluation of the algebraic rank test. While we used interleaved generation of candidate modes using positive and negative columns, it seems that there may still be space for improvement. Apparently, some of the compute nodes end up having greater fraction of modes with many non-zeros than other nodes, which implies that the rank test is evaluated on larger submatrices.

4.4 Conclusion

Combinatorial Parallel Nullspace Algorithm illustrates a distributed-memory parallelization of the computation of elementary modes performed across the task of generating candidate elementary flux modes. An interleaving strategy is used to assign positive and negative columns in the nullspace matrix and try to attain good load balancing across other portions of the Nullspace Algorithm which follow the generation of the candidate modes. Major disadvantage of the design is the need to keep the entire nullspace matrix during the run of the algorithm at each processor's memory. Hence, it is needed to partition the nullspace matrix across all the processors in the environment and maintain efficient load balancing. One possible solution to this challenge is illustrated in Chapter 6 using Global Arrays library.

Chapter 5

Divide-and-conquer approach in elementary mode computation

5.1 Introduction

An early idea in studying metabolic networks, and biological networks in general, was to perform partitioning of the given network. Splitting network into subsystems which are loosely connected may facilitate the analysis of the functional properties. The idea and efforts to split the stoichiometry metabolic network to facilitate the elementary flux mode computation did not give any concrete procedure or insight. First, if the elementary modes are computed for every partition of the given metabolic network, it is not clear how to glue the modes back into the original network due to the requirement for the quasi-steady state property imposed on internal metabolites. Second, metabolic networks exhibit the scale-free property and few metabolites in the network constitute hubs, hence it may not be possible to split the network into several dense and loosely connected components. Despite that, an alternative reverse partitioning approach was earlier proposed and is a theme of this chapter. If used properly it may attain reduction in the size of utilized memory and computation time. This chapter illustrates the work earlier published in [149] with a correction of the number of modes reported in the original publication for the largest of the metabolic networks which were used.

5.2 Divide-and-conquer

In [150], the authors proposed a parallelization based on the divide-and-conquer approach to compute the complete set of the elementary flux modes. The complete set of the elementary flux modes is partitioned across the selected subset of q_{sub} reactions into $2^{q_{sub}}$ disjoint EFM subsets where the zero/nonzero flux pattern of the elementary flux modes in the i^{th} subset corresponds to the binary representation of the number i, for $i \in 0, \ldots, 2^{q_{sub}} - 1$. The reactions for partitioning elementary flux modes have to be selected after the compression step when the size of the original network is reduced to a smaller size, in order to avoid having the partitioning subset consisting of correlated or null reactions.

This divide-and-conquer approach illustrated in this chapter is based on the following theorem [150, 130].

Theorem 7. *If the Nullspace Algorithm is stopped at its $(q - q')^{th}$ iteration, then the set of elementary flux modes with all the last q' reactions having non-zero flux values coincides with the set of columns in the current nullspace matrix having non-zero flux values in the last q' elements.*

Proof. Assume the Nullspace Algorithm is at the $(q-q')^{th}$ iteration of its run, i.e. there are q' rows left to be processed. If the algorithm continues its run for the remaining q' iterations, every new candidate column will be generated as a linear combination of previous columns and will contain at least one zero element for reactions indexed by $q - q' + 1$ to q. This is due to the fact that the linear combination is performed as to annihilate the element of the column corresponding to the current row (Algorithm 3). Hence, the nullspace matrix at the end of iteration contains all elementary modes having non-zero flux in the last q' reactions. As a last step, one needs to reject those columns in the mentioned current nullspace matrix which do not satisfy the irreversibility constraints. □

Theorem 7 is used to incorporate the divide-and-conquer idea with the combinatorial parallel Nullspace Algorithm (Algorithm 4), as described in the following section.

5.3 Combined Parallel Nullspace Algorithm

In Algorithm 7, we propose the incorporation of the divide-and-conquer approach in the combinatorial parallel Nullspace Algorithm (Algorithm 4). Initially, the reaction subset size q_{sub} is selected and the elementary modes are computed for each of the $2^{q_{sub}}$ subsets. In lines 9 and 10 indices of reactions which should have non-zero and zero flux values are extracted according to the binary representation of the current iteration value k. For the i-th subset, a reduced stoichiometry matrix is formed by removing the columns corresponding to the indices *zfRows* of reactions with zero flux values. This results in the reduced stoichiometry matrix N_i (line 12), and new nullspace matrix R_i is recomputed (line 13). The rows of the nullspace matrix R_i are reordered so that the reactions corresponding to the indices in *nzfRows* are put at the bottom of the matrix (line 15). We then run the combinatorial parallel Nullspace Algorithm (line 18) described in Algorithm 4 on the pair (N_i, R_i). Rows corresponding to those reactions which should have zero reaction flux values are appended back to the matrix EFM_i (lines 21-25) and the iteration is complete.

Algorithm 7 $[R] = \texttt{CombParallelNullspAlg}(N, R, q_{sub})$

```
Input:
 1: reduced stoichiometry matrix ($N_{m\times q}$); initial nullspace of the form $R_{q\times(q-m)} = \begin{bmatrix} R^{(1)} \\ R^{(2)} \end{bmatrix}$
Output:
 2: matrix of elementary modes $EFM_{q\times n_{ems}}$
 3: ▷ Reduce initial metabolic network $(N, R)$ to equivalent smaller network.
 4:   $(N_{red}, R_{red}) \Leftarrow (N, R)$; $EFM = [\,]$;
 5: for $k = 0$ to $2^{q_{sub}} - 1$ do
 6:     ▷ dec2binvec - get binary representation of number as a vector
 7:     ▷ nzfRows - indices of last $q_{sub}$ rows which must have non-zero flux
 8:     ▷ zfRows - indices of last $q_{sub}$ rows which must have zero flux
 9:       $nzfRows = q - q_{sub} + \texttt{find}(\texttt{dec2binvec}(k))$
10:       $zfRows = q - q_{sub} + \texttt{find}(\neg\texttt{dec2binvec}(k))$
11:     ▷ from last $q_{sub}$ rows of nullspace matrix remove rows corresponding to zero reaction flux for the $k^{th}$
    subproblem
12:       $N_i = N_{red}; N_i(:, zfRows) = [\,]$;
13:       $R_i = \texttt{null}(N_i)$
14:     ▷ reorder rows in $R_i$ so that rows corresponding to nzfRows are at the bottom
15:       $R_i = \texttt{ReorderRows}(R_i); N_i = \texttt{ReorderColumns}(N_i)$;
16:       $lastRowIter := q - q_{sub}$
17:     ▷ Run parallel algorithm on the pair $(N_i, R_i)$ until reaching iteration corresponding to $lastRowIter^{th}$
    row
18:       $EFM_i = \texttt{ParallelNullspAlg}(N_i, R_i, N_{nodes}, lastRowIter)$
19:     ▷ keep only those columns in $EFM_i$ which have non-zero values in the last $length(nzfRows)$ rows
20:       $selectedRowInd = q - q_{sub} + (1 : \texttt{length}(nzfRows))$
21:       $EFM_i = EFM_i(:, \texttt{all}(EFM_i(selectedRowInd, :)))$
22:     ▷ add zero-rows to the $EFM_i$ in order to obtain the wanted subset of EFMs
23:       $numEms_i = \texttt{size}(EFM_i, 2)$
24:       $EFM_i(nzfRows, :) = EFM_i(selectedRowInd, :)$
25:       $EFM_i(zfRows, :) = \texttt{zeros}(length(zfRows), numEms)$
26:       $EFM = [EFM\ EFM_i]$
27: end for
28: return $EFM$
```

5.3.1 Example

We now illustrate the divide-and-conquer idea on our sample metabolic network in Figure 5.1 to illustrate our Combined Parallel Nullspace Algorithm. This 'toy' network has five internal metabolites (A,B,C,D,P) and nine reactions ($r_1, r_2, r_3, r_4, r_5, r_{6r}, r_7, r_{8r}, r_9$). All but two reactions are thermodynamically irreversible. Reversible reactions are denoted with a trailing 'r'.

The dotted line in Figure 5.1 marks the boundary between the interior and exterior of the given structure, which may be an entire cell or an internal compartment (organelle).

To represent the metabolic network in an analytical way we use the stoichiometry matrix illustrated in equation (5.1).

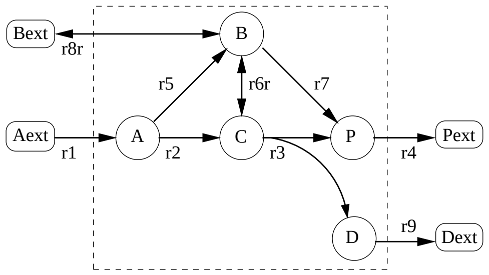


Figure 5.1: *Simple metabolic network [4]*

$$
N = \begin{array}{c} \\ A \\ B \\ C \\ D \\ P \end{array}
\begin{array}{c} \begin{array}{ccccccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_{6r} & r_7 & r_{8r} & r_9 \end{array} \\
\begin{pmatrix} 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 2 & 0 & 0 \end{pmatrix} \end{array}
\tag{5.1}
$$

$$
N_{red} = \begin{array}{c} \\ A \\ B \\ C \\ P \end{array}
\begin{array}{c} \begin{array}{cccccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_{6r} & r_7 & r_{8r} \end{array} \\
\begin{pmatrix} 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 \\ 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 2 & 0 \end{pmatrix} \end{array} .
\tag{5.2}
$$

The Nullspace Algorithm begins by computing a basis for the nullspace of N_{red} (the "nullspace matrix"). This is usually accomplished by reducing N_{red} to row echelon form and permuting the columns so that the stoichiometric matrix takes on the form $(-R^{(2)}, I)$. The resulting nullspace matrix then has the form

$$
R_{redperm} = \begin{pmatrix} R^{(1)} \\ R^{(2)} \end{pmatrix} = \begin{pmatrix} I \\ R^{(2)} \end{pmatrix} =
\begin{array}{c} r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_1 \\ r_3 \\ r_{6r} \\ r_{8r} \end{array}
\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -2 \\ -1 & 1 & 0 & -2 \\ 1 & -1 & 1 & 1 \end{pmatrix}
\tag{5.3}
$$

with the rows corresponding to the identity matrix being pushed to the top. The remaining rows are ordered by the increasing number of non-zero elements in the row [41, 40, 76], a heuristic proven to often improve the efficiency of Nullspace Algorithm. We also reorder the columns of N_{red} to to match the row order of (5.3), obtaining

$$N_{redperm} = \begin{array}{c} \\ A \\ B \\ C \\ P \end{array} \begin{array}{c} \begin{array}{cccccccc} r_2 & r_4 & r_5 & r_7 & r_1 & r_3 & r_{6r} & r_{8r} \end{array} \\ \begin{pmatrix} -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & -1 \\ 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 2 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}. \tag{5.4}$$

If a simple serial or combinatorial parallel Nullspace Algorithm is applied, it will produce the elementary flux mode matrix as given in equation (5.5).

$$EFM = \begin{array}{c} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_{6r} \\ r_7 \\ r_{8r} \\ r_9 \end{array} \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 1 & 2 & 1 & 2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{5.5}$$

Once the initial nullspace matrix is found for the stoichiometry matrix, we consider the four subproblems across the two reactions r_{6r} and r_{8r}.

- *Reactions r_{6r}, r_{8r} should both have zero flux values.* Columns corresponding to the reactions r_{6r}, r_{8r} are removed from the matrix $N_{redperm}$ in equation (5.4) to obtain the matrix given in equation (5.6).

$$N_{r00} = \begin{array}{c} \\ A \\ B \\ C \\ P \end{array} \begin{array}{c} \begin{array}{cccccc} r_2 & r_4 & r_5 & r_7 & r_1 & r_3 \end{array} \\ \begin{pmatrix} -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 2 & 0 & 1 \end{pmatrix} \end{array}. \tag{5.6}$$

The nullpace matrix corresponding to the matrix N_{r00} is given as R_{r00}.

$$R_{r00} = \begin{array}{c} r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_1 \\ r_3 \end{array} \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ -1 & 1 \\ -1 & 1 \\ 1 & 1 \\ 2 & 0 \end{pmatrix} \times \frac{1}{2}. \tag{5.7}$$

When the Nullspace Algorithm is run on the pair (N_{r00}, R_{r00}) two elementary flux modes are obtained:

$$EFM_{r00} = \begin{array}{c} r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_1 \\ r_3 \end{array} \begin{pmatrix} 0 & 1 \\ 2 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}. \tag{5.8}$$

- *Reaction r_{6r} should have zero and reaction r_{8r} should have non-zero flux values.* This requires the removal of the column corresponding to the reaction r_{6r} in the matrix $N_{redperm}$ to obtain matrix N_{r01}

$$N_{r01} = \begin{array}{c|ccccccc} & r_2 & r_4 & r_5 & r_7 & r_1 & r_3 & r_{8r} \\ \hline A & -1 & 0 & -1 & 0 & 1 & 0 & 0 \\ B & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ C & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ P & 0 & -1 & 0 & 2 & 0 & 1 & 0 \end{array}. \tag{5.9}$$

The nullpace matrix corresponding to the matrix N_{r01} is given as R_{r01}.

$$R_{r01} = \begin{array}{c} r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_1 \\ r_3 \\ r_{8r} \end{array} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \\ -1 & 1 & 0 \\ 2 & 0 & 2 \\ 2 & 0 & 0 \\ 1 & -1 & 2 \end{pmatrix} \times \frac{1}{2}. \tag{5.10}$$

Running the Nullspace Algorithm until before the row corresponding to the reaction r_{8r} on a pair (N_{r01}, K_{r01}), and extracting those columns having non-zero flux values for the reaction r_{8r} we obtain two elementary flux modes:

$$EFM_{r01} = \begin{array}{c} r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_1 \\ r_3 \\ r_{8r} \end{array} \begin{pmatrix} 0 & 0 \\ 2 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ -1 & 1 \end{pmatrix}. \tag{5.11}$$

- *Reaction r_{6r} should have non-zero and reaction r_{8r} should have zero flux values.* This requires the removal of the column corresponding to the reaction r_{8r} in the matrix $N_{redperm}$ to obtain matrix N_{r10}.

$$N_{r10} = \begin{array}{c} \\ A \\ B \\ C \\ P \end{array} \begin{array}{c} \begin{array}{ccccccc} r_2 & r_4 & r_5 & r_7 & r_1 & r_3 & r_{6r} \end{array} \\ \begin{pmatrix} -1 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & 2 & 0 & 1 & 0 \end{pmatrix} \end{array}. \tag{5.12}$$

The nullspace matrix corresponding to the matrix N_{r10} is given as R_{r10}.

$$R_{r10} = \begin{array}{c} r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_1 \\ r_3 \\ r_{6r} \end{array} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & -1 \\ 1 & 0 & 1 \\ 2 & -1 & 2 \\ 1 & -1 & 2 \end{pmatrix}. \tag{5.13}$$

Similarly, using the Nullspace Algorithm and running it until before the reaction corresponding to the row r_{6r}, two elementary flux modes are obtained:

$$EFM_{r10} = \begin{array}{c} r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_1 \\ r_3 \\ r_{6r} \end{array} \begin{pmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ -1 & 1 \end{pmatrix}. \tag{5.14}$$

- *reactions r_{6r} and r_{8r} have both non-zero fluxes.*

$$N_{r11} = \begin{array}{c} \\ A \\ B \\ C \\ P \end{array} \begin{array}{c} \begin{array}{cccccccc} r_2 & r_4 & r_5 & r_7 & r_1 & r_3 & r_{6r} & r_{8r} \end{array} \\ \begin{pmatrix} -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & -1 \\ 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 2 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}. \tag{5.15}$$

$$R_{r11} = \begin{array}{c} r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_1 \\ r_3 \\ r_{6r} \\ r_{8r} \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -2 \\ -1 & 1 & 0 & -2 \\ 1 & -1 & 1 & 1 \end{pmatrix}. \tag{5.16}$$

In this last case, the Nullspace Algorithm is run on a pair of matrices (N_{r11}, R_{r11}) until before the row corresponding to the reaction r_{6r}, and the elementary flux modes having non-zero values for both reactions r_{6r} and r_{8r} are extracted. This yields two elementary flux modes:

$$EFM_{r11} = \begin{array}{c} r_2 \\ r_4 \\ r_5 \\ r_7 \\ r_1 \\ r_3 \\ r_{6r} \\ r_{8r} \end{array} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{5.17}$$

We see that the union of elementary flux modes obtained for all four cases coincides with the elementary flux modes given in equation (5.5). In our combined parallel Nullspace Algorithm, each of the four subtasks would be run independently using the Combinatorial Parallel Nullspace Algorithm given earlier in Algorithm 4. ∎

5.4 Results

We used the “Calhoun” and “Itasca” parallel platforms of the Minnesota Supercomputing Institute to test our Combined Parallel Nullspace Algorithm.

“Calhoun” is an SGI Altix XE 1300 Linux cluster. The cluster consists of 256 compute nodes, each containing two quad-core 2.66 GHz Intel Xeon “Clovertown”-class processors sharing 16 GB of main memory. In total, “Calhoun” consists of 2048 compute cores and 4 TB of main memory. Compute node is consists of two multi-chip modules (MCMs) each composed of two dies. These dies are two separate pieces of silicon connected to each other and arranged on a single module. Each die has two processor cores that share a 4 MB

L2 cache. Each MCM communicates with the main memory in the system via a 1,333 MHz front-side bus (FSB). All of the systems within "Calhoun" are interconnected with a 20-gigabit non-blocking InfiniBand fabric used for interprocess communication (IPC). The InfiniBand fabric is a high-bandwidth, low-latency network, the intent of which is to accommodate high-speed communication for large MPI jobs. The nodes are also interconnected with two 1-gigabit ethernet networks for administration and file access, respectively.

"Itasca" platform is an HP Linux Cluster which also hosts an extension in the form of 51 Sandy Bridge blades partitioned into on three groups of 35, 8 and 8 nodes distinguished by the amount of memory available to the single node within each of the groups which ammounts to 64GB, 128GB and 256GB. We used 4 nodes from the second group of 8 nodes where each node consists of 2 eight-core Sandy bridge E5-2670 processor chips (2.6 GHz). This ammounts to the total of 64 cores and 256GB of available memory across 4 nodes.

Algorithm 7 are implemented in software which is distributed under the GNU General Public License (GPL). Source code and documentation are freely available at the web site: `http://elmocomp.sourceforge.net/`.

Table 5.1: *Parallel computation of EFMs on 78-reaction S. cerevisiae metabolic network using Algorithm 4 on Intel Xeon machine*

# nodes	1	2	1	1	4	8	16
# cores per node	1	1	4	8	4	4	4
total # cores	1	2	4	8	16	32	64
memory per core	12gb	12gb	3gb	1.5gb	3gb	3gb	3gb
gen. cand (sec)	2744.76	1383.93	688.60	349.05	179.04	95.44	46.83
rank test (sec)	112.88	77.42	52.80	33.98	20.38	12.21	8.01
communicate (sec)	0	0.06	0.09	0.18	0.17	0.19	0.18
merge (sec)	0	0.68	1.01	1.40	1.45	1.62	1.74
total time (sec)	2894.40	1490.85	761.29	404.33	208.98	115.46	61.87
Total # candidate modes: 159,599,700,951				Total # EFM: 1,515,314			

We used two metabolic networks of *S. cerevisiae* of dimension 62×78 (35×55) and 63×83 (40×61), respectively, where values in parentheses correspond to the size of the reduced metabolic network after elimination of redundant constraints. The list of reactions in the 78- and 83- reaction networks is given in section A.3 of Appendix A. The 83-reaction network accounts for growth on more than one substrate and supports aerobic growth. In the network descriptions reversible reactions are denoted with suffix "r" and external metabolites with suffix "ext". Elementary flux modes were computed for the 78-reaction network using the Combinatorial Parallel Nullspace Algorithm (Algorithm 4) and the Combined parallel Nullspace Algorithm (Algorithm 7) on Intel Xeon (Clovertown)

Table 5.2: *Parallel computation of EFMs on 78-reaction metabolic network of S. cerevisiae using Algorithm 7 on Intel Xeon (Clovertown) machine with partitioning across reactions* $\{R89_r,\ R74_r\}$ *using 16 processors*

subset	~~$R89_r$~~ ~~$R74_r$~~	~~$R89_r$~~ $R74_r$	$R89_r$ ~~$R74_r$~~	$R89_rR74_r$
# EFM	274,919	599,344	207,533	433,518
gen. cand (sec)	17.50	57.36	17.29	24.61
rank test (sec)	2.96	7.18	2.34	3.78
comm (sec)	0.05	0.10	0.05	0.10
merge (sec)	0.16	0.44	0.11	0.36
total time (sec)	21.97	67.77	20.79	31.07
Cumulative total time: 141.6 secs			Total # EFM: 1,515,314	
Total # candidate modes: 81,714,944,316				

machine. The results of the computation using Algorithm 4 are given in table 5.1, while the results of using Algorithm 7 are given in table 5.2. In table 5.2 the row *subset* identifies the subproblem in the divide-and-conquer partitioning, and the zero-flux pattern in the two reactions $R89_r$ and $R74_r$ used (~~R~~ and R denote that the reaction R has zero and non-zero flux value in the given EFM subproblem, respectively). To compute the elementary flux modes for the subproblems in the combined parallel Nullspace Algorithm we used 16 cores across 4 compute nodes and compared that results with the column in Table 5.1 corresponding to 16 cores. The divide-and-conquer splitting in the Algorithm 7 decreased the number of intermediate candidate modes from 159,599,700,951 to 81,714,944,316, what resulted in the effective reduction of the computation time from 208.98 seconds ((Table 5.1)) to 141.6 seconds.

The set of EFMs for the 83-reaction network was computed using the combined parallel Nullspace Algorithm (Algorithm 7) on the 64 cores of the Sandy Bridge cluster earlier described (Table 5.3). We made an estimate that using five reactions to partition the tasks might be sufficient to compute all the modes using given hardware. Fortunately, none of the 32 tasks in that case required further splitting and the computation finished with a total of 68,868,602 elementary modes and maximum requirement for physical memory per compute node equal to 3,708 Mbyte, as shown in Table 5.3. It was established that 12 out of 32 partitions have no elementary modes, though the cost of learning this was paid at high price as is shown by the very large number of generated candidate modes in those partitions. As given in Theorem 3 to decide if there is an elementary mode having non-zero (positive) flux in the k specified reactions is NP-complete decision problem and thus it is not possible to tell in the 12 mentioned cases if the network has no admissible modes before running the

Table 5.3: *Parallel computation of EFMs on 83-reaction metabolic network of S. cerevisiae using Algorithm 7 on Sandy Bridge 64-core cluster with partitioning across reactions* $\{R102r-,\ R57,\ R65,\ R19_r,\ R68\}$

reaction partition	# candidate modes	# EFM	time (sec)
~~$R102_r$–~~, ~~$R57$~~, ~~$R65$~~, ~~$R19_r$~~, ~~$R68$~~	45,095,637,841	1,769,187	107.503
~~$R102_r$–~~, ~~$R57$~~, ~~$R65$~~, ~~$R19_r$~~, $R68$	2,779,842,279,941	2,829,302	1258.898
~~$R102_r$–~~, ~~$R57$~~, ~~$R65$~~, $R19_r$, ~~$R68$~~	35,250,457,892	685,498	102.000
~~$R102_r$–~~, ~~$R57$~~, ~~$R65$~~, $R19_r$, $R68$	354,576,653,413	215,636	298.717
~~$R102_r$–~~, ~~$R57$~~, $R65$, ~~$R19_r$~~, ~~$R68$~~	2,779,842,279,941	907,690	1255.549
~~$R102_r$–~~, ~~$R57$~~, $R65$, ~~$R19_r$~~, $R68$	407,722,024,826	0	359.379
~~$R102_r$–~~, ~~$R57$~~, $R65$, $R19_r$, ~~$R68$~~	354,577,418,542	733,874	353.848
~~$R102_r$–~~, ~~$R57$~~, $R65$, $R19_r$, $R68$	353,223,655,207	0	357.311
~~$R102_r$–~~, $R57$, ~~$R65$~~, ~~$R19_r$~~, ~~$R68$~~	167,684,683,474	1,640,260	213.804
~~$R102_r$–~~, $R57$, ~~$R65$~~, ~~$R19_r$~~, $R68$	5,679,741,563,998	9,488	3561.345
~~$R102_r$–~~, R57, ~~$R65$~~, $R19_r$, ~~$R68$~~	153,276,454,788	0	248.387
~~$R102_r$–~~, R57, ~~$R65$~~, $R19_r$, $R68$	4,647,652,548,048	0	2996.302
~~$R102_r$–~~, $R57$, $R65$, ~~$R19_r$~~, ~~$R68$~~	5,679,741,563,998	14,832,898	3550.382
~~$R102_r$–~~, $R57$, $R65$, ~~$R19_r$~~, $R68$	5,645,989,479,622	0	3042.295
~~$R102_r$–~~, $R57$, $R65$, $R19_r$, ~~$R68$~~	4,647,652,548,048	2,196,698	3002.308
~~$R102_r$–~~, $R57$, $R65$, $R19_r$, $R68$	4,647,649,237,656	0	2823.045
$R102_r$–, ~~$R57$~~, ~~$R65$~~, ~~$R19_r$~~, ~~$R68$~~	171,942,706,750	2,909,673	212.032
$R102_r$–, ~~$R57$~~, ~~$R65$~~, ~~$R19_r$~~, $R68$	1,575,477,661,513	4,602,462	842.725
$R102_r$–, ~~$R57$~~, ~~$R65$~~, $R19_r$, ~~$R68$~~	137,727,557,470	1,185,538	259.513
$R102_r$–, ~~$R57$~~, ~~$R65$~~, $R19_r$, $R68$	1,340,527,709,376	368,820	770.213
$R102_r$–, ~~$R57$~~, $R65$, ~~$R19_r$~~, ~~$R68$~~	1,575,477,661,513	1,498,804	834.200
$R102_r$–, ~~$R57$~~, $R65$, ~~$R19_r$~~, $R68$	1,561,424,250,368	0	714.985
$R102_r$–, ~~$R57$~~, $R65$, $R19_r$, ~~$R68$~~	1,340,527,709,376	1,265,878	774.598
$R102_r$–, ~~$R57$~~, $R65$, $R19_r$, $R68$	1,360,997,901,290	0	709.914
$R102_r$–, $R57$, ~~$R65$~~, ~~$R19_r$~~, ~~$R68$~~	648,315,168,596	2,667,756	473.371
$R102_r$–, $R57$, ~~$R65$~~, ~~$R19_r$~~, $R68$	21,488,036,243,452	17,414	9411.812
$R102_r$–, $R57$, ~~$R65$~~, $R19_r$, ~~$R68$~~	938,248,016,253	0	619.416
$R102_r$–, $R57$, ~~$R65$~~, $R19_r$, $R68$	17,949,749,059,604	0	8174.077
$R102_r$–, $R57$, $R65$, ~~$R19_r$~~, ~~$R68$~~	21,488,036,243,452	24,753,116	9421.823
$R102_r$–, $R57$, $R65$, ~~$R19_r$~~, $R68$	21,851,155,160,634	0	8625.066
$R102_r$–, $R57$, $R65$, $R19_r$, ~~$R68$~~	17,949,749,059,604	3,778,610	8165.555
$R102_r$–, $R57$, $R65$, $R19_r$, $R68$	17,997,956,795,218	0	7357.128
Total:	167,754,867,391,704	68,868,602	22h:28m:18s

Nullspace Algorithm.

5.4.1 Time scalability

Computation time is proportional to the number of generated intermediate elementary modes. Divide-and-conquer approach usually leads to the decrease of the cumulative number of intermediate modes compared to the unsplit problem, and the execution times are proportional to these numbers of modes. It is yet unclear how to select the subset of reactions in divide-and-conquer that may maximally decrease the number of intermediate candidate elementary flux modes.

5.4.2 Memory scalability

The combinatorial parallel Nullspace Algorithm has the disadvantage that it requires the storage of the current nullspace matrix in the local memory across all compute nodes at each step. Hence, until that bottleneck is removed, the combinatorial parallel Nullspace Algorithm may only be used for problems where the current nullspace matrix may fit in the local memory of the compute node. The divide-and-conquer feature of the combined parallel Nullspace algorithm "fits" the larger problem to the available architecture, where combinatorial parallel algorithm only could not be applied. However, cumulative memory requirements for all subproblems compared to the original problem remain the same.

5.5 Discussion

The divide-and-conquer approach requires the selection of the subset of reactions which are used in the partitioning of the computation of modes. If the size of this reaction subset is large, the number of partitions to explore may be impractically high. The estimation of the minimal number of reactions required to enter the partitioning subset is still performed through manual experimentation. An automated method to select the subset and estimate the approximate number of elementary modes across all possible tasks for a given reaction partitioning subset would be lead to a more automated Combined Parallel Nullspace Algorithm. In order to give an intuition about the computational complexity of this problem it was earlier in subsection 3.2.1 cited that to enumerate all the elementary modes having non-zero flux for a specific reaction is NP-hard [132, 82] and to decide if there is an elementary mode with the non-zero flux for the specified subset of reactions is NP-complete.

5.6 Conclusion

This chapter gives an improved parallelization of the Nullspace Algorithm compared to the Combinatorial Parallel Nullspace Algorithm (Algorithm 4) proposed in Chapter 4, in terms of reducing memory requirements per run of task. An earlier established property of the Nullspace Algorithm which allows the computation of modes having non-zero flux for specific subset of reactions was used. The divide-and-conquer idea [150] was incorporated within the Combinatorial Parallel Nullspace Algorithm to yield the Combined Parallel Nullspace Algorithm. The combination of two algorithmic ideas may reduce the computation time by lowering the total number of intermediate candidate elementary modes and fit the larger problems to the available parallel architecture where previously the Combinatorial Parallel Nullspace Algorithm failed. The efficiency of applying the divide-and-conquer approach depends on the proper selection of the reactions subset used to partition the space of elementary flux modes into disjoint subsets computed independently. Using this approach, it was possible to complete the computation of the nearly 70 million elementary flux modes for the 83-reaction *S. cerevisiae* metabolic network which has three distinct substrate reactions. As will be shown in the following chapter, there is more space for improvement within the parallel algorithm proposed here. First, the current nullspace matrix should not be fully stored across all the compute nodes as is currently done in the Combinatorial Parallel Nullspace Algorithm. The replicated data structures should be partitioned in an efficient way across all the processors' memories instead. Second, it would be important to propose a strategy for the selection of the reaction subset used in the divide-and-conquer part of the algorithm so that it generates as few intermediate elementary mode candidates as possible. Finally, rather than partitioning the elementary flux modes across the subset of selected reactions as proposed in this chapter, one may decide to partition the metabolic network graph into smaller units and compute the elementary modes for each of them.

Chapter 6

Parallelization of Nullspace Algorithm using Global Arrays

6.1 Introduction

As outlined earlier, major problem in the proposed parallel Nullspace Algorithms for the distributed memory system is the need to keep the copy of entire nullspace matrix on each compute node. Partitioned Global Addressing Space libraries and languages such as UPC, Co-Array Fortran or Global Arrays[151, 152, 153] may hold promise to attain the partitioning of the data structure across all the compute-nodes and allow the user the shared-memory view of the data. Due to its easy integration and use, Global Arrays library is used to re-implement the Combinatorial Parallel Nullspace Algorithm outlined in Chapter 4, allowing the global view of the distributed memory, which largely facilitates the development of the software with improved memory scalability. Rationale behind this, is that in many cases serial algorithms have a more straightforward and easier parallel design and implementation in a shared memory than in distributed memory systems. Using only MPI library without relying on the Global Arrays, one may have to organize the communication of the data into numerous MPI collective operations in order to attain uniform partitioning of the data and accomplish good time scalability. These reasons, as well as some other features of the Global Arrays library mentioned in continuation of this chapter, allowed the design of an improved parallel implementation of the algorithm for the computation of the elementary flux modes.

This chapter proposes an improved parallel Nullspace Algorithm used to compute elementary flux modes using Global Array library and try to attain good memory and time scalability. Section 6.2 outlines the proposal for the parallel Nullspace Algorithm utilizing Global Arrays. Section 6.3 shows the results obtained for the case of *Sacharomyces cerevisiae* metabolic networks, while section 6.4 lists advantages and disadvantages of the approach outlining possible future work.

6.2 Methods

6.2.1 Global Arrays Library

The Global Arrays library provides a shared-memory view of the data on a distributed-memory computing platform. It was designed to complement the message-passing model and allows the combination of the shared-memory and distributed-memory programming styles. The Global Arrays library allows the creation of data in the form of a global array data structure whose portions are distributed across processors, and every processor is aware of its "remote" and "local" parts. Elements are accessed by means of indexing only, where each processor can access any part of the distributed data, without explicitly requesting the cooperation with other processes. This one-sided asynchronous communication is facilitated using the Aggregate Remote Memory Copy Interface (ARMCI) library [154]. Hence, Global Arrays makes a distributed memory system into a virtual, shared-memory NUMA system.

The Global Arrays library provides operations for initialization and termination of the environment, creation and destruction of arrays, one-sided communication operations, interprocess synchronization, collective array operations, utility operations, mirrored arrays, processor groups, sparse data operations and interfaces to third party software packages. Global Arrays is designed to support use of C and Fortran-77, as well as bindings for C++ and Python.

6.2.2 Parallel Nullspace Algorithm using Global Arrays

The Nullspace Algorithm may be redesigned to utilize Global Arrays library and represent as global arrays those data structures which were replicated across all processors in the Combinatorial Parallel Nullspace Algorithm (Algorithm 8). To facilitate understanding of the pseudocode outlined in this chapter, for data structure X we use the superscript notation $X^{(\mathsf{GA})}$ to denote that it is a global array distributed across all the processors, and $X^{(\mathsf{LC})}$ stands for "local copy" available: data which either resides locally or has been fetched from other processors.

Global arrays are created for bit-valued and real-valued nullspace matrices ($R^{(\mathsf{GA,bit})}$ and $R^{(\mathsf{GA,real})}$), arrays of indices of positive, negative and zero columns ($pInd^{(\mathsf{GA})}$, $nInd^{(\mathsf{GA})}$, $zInd^{(\mathsf{GA})}$) in the current nullspace matrix, and arrays of pairs of indices of columns which generate admissible candidate modes ($cPInd^{(\mathsf{GA})}$, $cNInd^{(\mathsf{GA})}$). Algorithm 9 gives a high level description of the parallelization using global arrays. Given the initial nullspace matrix, we create global arrays $R^{(\mathsf{GA,bit})}_{(q-m)\times(q-m)}$ and $R^{(\mathsf{GA,real})}_{m\times(q-m)}$ which represent bit-valued (upper) and real-valued (lower) matrices. In every iteration step, the top row of the

Algorithm 8 $[EFM]$=`CombParallelNullspaceAlg`(N, R)

```
Input:
 1: reduced stoichiometry matrix - $N_{m\times q}$
 2: initial nullspace matrix - $R_{q\times(q-m)} = \begin{bmatrix} R^{(\mathsf{bit})}_{(q-m)\times(q-m)} \\ R^{(\mathsf{real})}_{(m)\times(q-m)} \end{bmatrix}$
Output:
 3: matrix of elementary flux modes $EFM_{q\times nEms}$
 4: $nEms = q - m$
 5: for $currReac = q - m + 1$ to $q$ do
 6:     ▷ current nullspace matrix is $R^{(currReac)}_{q\times(nEms)} = \begin{bmatrix} R^{(\mathsf{bit})}_{(currReac-1)\times nEms} \\ R^{(\mathsf{real})}_{(q-currReac+1)\times nEms} \end{bmatrix}$
 7:     $[pInd, nInd]$ = ScanRealValuedEfm$(currReac, R^{(\mathsf{real})})$
 8:     $[cPInd, cNInd]$ = ParallelGenerateEFMCands$(currReac, pInd, nInd, R^{(\mathsf{bit})})$
 9:     $[cPInd, cNInd]$ = Sort&RemoveDuplicates$(currReac, cPInd, cNInd, R^{(\mathsf{bit})})$
10:     $[cPInd, cNInd]$ = RankTests$(N, cPInd, cNInd, R^{(\mathsf{bit})})$
11:     $[cPInd, cNInd]$ = Communicate&Merge$(cPInd, cNInd, R^{(\mathsf{bit})})$
12:     $[R^{(\mathsf{bit})}, R^{(\mathsf{real})}]$ = ExpandEfm$(cPInd, cNInd, R^{(\mathsf{bit})}, R^{(\mathsf{real})})$
13:     $nEms$ = size$(R^{(\mathsf{real})}, 2)$
14: end for
15: return $R^{(\mathsf{bit},q)}$
```

Algorithm 9 $[EFM]$=`NullspaceAlgParallelGA`(N,R)

```
Input:
 1: reduced stoichiometry matrix $(N_{m\times q})$;
 2: initial right nullspace matrix  $R_{q\times(q-m)} = \begin{bmatrix} R^{(\mathsf{bit})}_{(q-m)\times(q-m)} \\ R^{(\mathsf{real})}_{(m)\times(q-m)} \end{bmatrix}$
Output:
 3: matrix of bit-valued elementary flux modes $EFM_{q\times nEms}$
 4: CREATE$(R^{(\mathsf{GA,bit})}_{(q-m)\times(q-m)})$; initialize to $R^{(\mathsf{bit})}_{(q-m)\times(q-m)}$
 5: CREATE$(R^{(\mathsf{GA,real})}_{m\times(q-m)})$; initialize to $R^{(\mathsf{real})}_{m\times(q-m)}$
 6: $nEms = q - m$
 7: for $currReac = q - m + 1$ to $q$ do
 8:
 9:     ▷ for convenience we will omit the subscript from $R^{(\mathsf{GA,bit})}_{(currReac-1)\times nEms}$ and
        $R^{(\mathsf{GA,real})}_{(q-currReac+1)\times nEms}$
10:     $[pInd^{(\mathsf{GA})}, nInd^{(\mathsf{GA})}, zInd^{(\mathsf{GA})}]$ = ScanRealValuedEfmGA$(currReac, R^{(\mathsf{GA,real})})$
11:     $[cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}]$ = GenerateCandsGA$(currReac, pInd^{(\mathsf{GA})}, nInd^{(\mathsf{GA})}, R^{(\mathsf{GA,bit})})$
12:     $[cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}]$ = LocalPruneCandModesGA$(cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}, R^{(\mathsf{GA,bit})})$
13:     $[cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}]$ = GlobalPruneCandModesGA$(currReac, cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})},$
            $R^{(\mathsf{GA,bit})})$
14:     $[R^{(\mathsf{GA,bit})}, R^{(\mathsf{GA,real})}]$ = ExpandEfmGA$(R^{(\mathsf{GA,bit})}, R^{(\mathsf{GA,real})}, pInd^{(\mathsf{GA})}, nInd^{(\mathsf{GA})}, zInd^{(\mathsf{GA})},$
            $currReac, cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})})$
15: end for
16: return $R^{(\mathsf{GA,bit})}$
```

real-valued part of the nullspace matrix is scanned for positive, negative and zero elements (`ScanRealValueEfmGA` in Alg. 20) to determine indices of positive, negative and zero nullspace matrix columns, respectively. Furthermore, the global arrays of indices of positive and negative columns are used in every processor to fetch the respective columns and generate candidate columns (`GenerateCandsGA` in Alg. 21). The columns are, as earlier shown, pruned for the maximal allowed number of non-zero elements. Procedure `GenerateCandsGA` returns local arrays of indices of positive and negative columns in $cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}$ which can then be locally pruned by eliminating those pairs of indices which generate duplicate candidate columns and running the algebraic rank test (`LocalPruneCandModesGA` in Alg. 22). The resulting arrays of pairs of indices is then pruned globally to eliminate those pairs of indices which generate duplicate column modes (`GlobalPruneCandModesGA` in Alg. 23). The global removal of duplicates is comprised of first running parallel radix-sort (`RadixSortEFMCandsGA` in Alg. 24) on the bit-columns generated by the arrays of pairs of indices. Parallel radix-sort differs from the serial radix-sort in the portion of the algorithm which uses counting sort. Counting sort is parallelized using prefix scan to determine new positions of the local elements in the global array. For an appropriate bucket size $width$=8 or 16, which is observed at every iteration, counts are stored in an array of length 2^{width}. Sorting bit columns across $width$-bit portions is performed in $\frac{numBits}{width}$ steps for each $numBits$-bit word in a column, where $numBits$=32 or 64. The sorted array of pairs of indices is then scanned to retain only the pairs of indices generating unique bit-column modes ($UniqueEFMCandsGA$ in Alg. 25). Once this is complete, the two global arrays $R^{(\mathsf{GA,bit})}$ and $R^{(\mathsf{GA,real})}$ are scanned to eliminate negative columns, i.e. those indexed by elements in $nInd^{(\mathsf{GA})}$, in order to enforce the irreversibility constraint. Finally, the globally pruned and unique column modes are appended to the two global arrays $R^{(\mathsf{GA,bit})}$ and $R^{(\mathsf{GA,real})}$ (`ExpandEfmGA`). Due to limited space we omit the details for `ExpandEfmGA`, since it takes only a small fraction of the total execution time.

The Algorithms sketched here use the collective operations in the Global Array library, denoted here as `CREATE/DESTROY`, `PUT/GET` and `SCATTER/GATHER`.

6.3 Results

Implementation was tested on HP Linux cluster with 1,091 HP ProLiant BL280c G6 blade servers, each with two-socket, quad-core 2.8 GHz Intel Xeon X5560 “Nehalem EP” processors sharing 24 GB of system memory, with a 40-gigabit QDR InfiniBand (IB) interconnect. The Global Array library and elementary mode software were compiled using Intel C++ compiler 12.1.2 (compatible with gcc 4.3.0) and Platform MPI 8.0 for Linux. In addition, the Global Array library was compiled to support communication over the InfiniBand network. Algorithms were implemented using C++ and Standard Template Library.

The HP Linux Cluster utilized was configured so that all processor cores per compute node would have to be used during the run, hence the smallest usable processor count was 8. We decided to use 2 GB of memory per each of the 8 available processor cores. If for the

Table 6.1: *Results of running Algorithm 9 on 78-, 80-, and 83- reactions metabolic networks of S. cerevisiae*

		Time (*sec*)					#EM	# cand. EM
		8p	16p	32p	64p	128p		
original	gen. cands.	156.277	79.067	43.514	23.98	19.293	1,515,315	116,864,715,755
62 × 78(31)	local prune .	25.193	17.051	11.224	8.381	6.991		
compressed	global prune	8.538	7.147	5.963	6.726	8.534		
35 × 54(17)	append	1.056	0.755	0.519	0.400	0.385		
	misc	0.008	0.019	0.444	0.455	0.326		
	total	191.095	104.051	61.270	39.945	35.531		
		64p	128p	256p	512p	1024p		
original	gen. cands.	1348.135	740.007	455.919	499.337	1061.334	13,322,464	8,032,689,186,974
62 × 80(31)	local prune .	1316.965	979.306	819.404	742.346	679.683		
compressed	global prune	85.266	125.281	133.199	136.688	180.026		
38 × 57(19)	append	3.292	2.570	2.505	2.663	3.522		
	misc	0.491	0.513	0.836	1.873	2.374		
	total	2754.173	1847.689	1411.869	1382.910	1926.941		
		512p	1024p	2048p				
original	gen. cands.	3177.564	2908.849	4547.499			68,868,602	66,144,590,373,585
63 × 83(34)	local prune .	2852.662	2266.930	1997.479				
compressed	global prune	72.552	80.616	114.432				
40 × 61(23)	append	3.801	5.668	8.199				
	misc	0.579	1.743	1.575				
	total	6107.177	5263.820	6669.199				

Table 6.2: *Results of running Algorithm 8 on 78-, 80- reactions metabolic networks of S. cerevisiae*

		Time (*sec*)					#EM	# cand. EM
		8p	16p	32p	64p	128p		
original	gen. cands.	239.519	119.939	60.098	30.959	15.277	1,515,315	116,864,715,755
62 × 78(31)	local prune .	20.068	10.958	6.990	5.035	4.017		
compressed	global prune	14.711	15.725	13.344	14.202	14.392		
35 × 54(17)	append	0.356	0.368	0.353	0.370	0.352		
	misc	0.076	3.874	2.100	1.134	0.699		
	total	274.858	150.995	83.022	51.836	34.871		
		64p	128p	256p	512p	1024p		
original	gen. cands.	2204.642	1104.309	552.119	279.801	141.765	13,322,464	8,032,689,186,974
62 × 80(31)	local prune .	761.254	588.728	484.476	443.023	423.616		
compressed	global prune	66.803	72.888	77.400	81.676	88.132		
38 × 57(19)	append	4.053	4.095	4.512	4.818	5.071		
	misc	164.615	83.045	43.155	26.470	11.605		
	total	3201.386	1853.075	1161.666	835.791	670.190		

Table 6.3: *Usage data of the Global Arrays collective operations in the run of Algorithm 9 on the 83-reaction metabolic network of S. cerevisiae*

call	mean value per processor	512p	1024p	2048p
`GET`	# of calls	7,509.07	14,600.10	28,900.05
	# of processors per call	2.26	2.13	2.21
	bytes remotely sent (MB)	142.489	131.597	126.015
`GATHER`	# of calls	7,749.42	14,829.79	29,110.74
	# of processors per call	5.30	4.30	3.41
	bytes remotely sent (MB)	636.339	439.076	340.712
`PUT`	# of calls	237.48	206.75	178.95
	# of processors per call	1.83	1.88	1.94
	bytes remotely sent (MB)	10.717	5.347	2.674
`SCATTER`	# of calls	254.12	227.06	197.03
	# of processors per call	63.05	73.11	77.51
	bytes remotely sent (MB)	16.292	8.141	4.066

specified number of processors and the metabolic network model, the memory requirements per processor exceeded available memory or if it took excessively long to complete the computation of the elementary flux modes, we decided to adopt a larger minimal processor

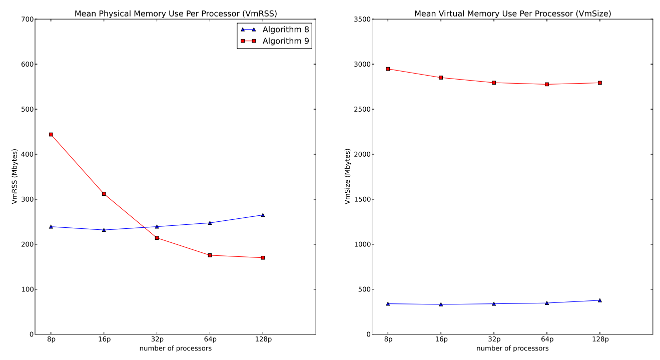


Figure 6.1: *Physical (*`VmRSS`*) and virtual memory (*`VmSize`*) mean usage per processor in the run of Algorithm 8 and 9 on the 78-reaction S. cerevisiae metabolic network*

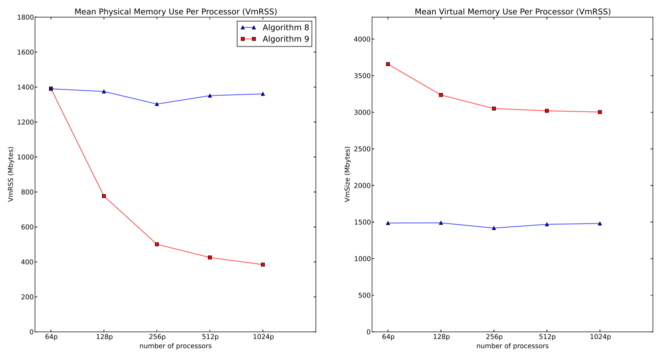


Figure 6.2: *Physical (*`VmRSS`*) and virtual memory (*`VmSize`*) mean usage per processor in the run of Algorithms 8 and 9 on the 80-reaction S. cerevisiae metabolic network*

count for the run of Algorithms 2 and 3 (80- and 83- reaction networks).

The elementary flux modes were computed for the metabolic networks of the central metabolism of *S. cerevisiae* supporting anaerobic (62 metabolites/78 reactions) and aerobic (62 metabolites/80 reactions and 63 metabolites/83 reactions) growth [144, 149], and also given in Table A.3 of Appendix A. In this chapter, we refer to these three networks by their number of reactions as 78-, 80- and 83- reaction networks. The run of Algorithm 9 on the three networks resulted in the computation of 1,515,315, 13,322,464 and 68,868,602 elementary modes, respectively. Results of this computation using up to 2048 processors are given in Table 6.1. In the first column of the table we report both the dimensions of the original and the reduced stoichiometry matrix, since the Nullspace Algorithm is run over the reduced network. To illustrate the complexity of the problem, in the last column we cite the number of intermediate candidates generated during computation before the pruning steps. In the rank tests (procedure `LocalPruneCandModesGA`), the default LU matrix decomposition was used. However, for the 80- and 83- reaction networks, LU gave occasional erroneous rank values, hence we resorted to the more numerically robust SVD (Singular Value Decomposition) at a slight extra expense.

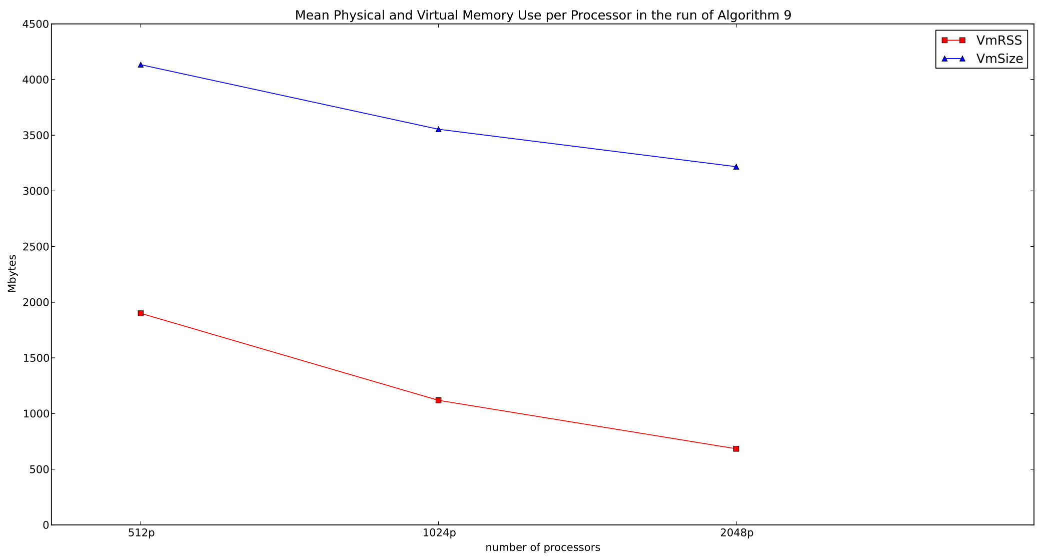


Figure 6.3: *Physical (`VmRSS`) and virtual memory (`VmSize`) mean usage per processor in the run of Algorithm 9 on the 83-reaction S. cerevisiae metabolic network*

For comparison, Table 6.2 gives timing results for the execution of the Combinatorial Parallel Nullspace Algorithm (Algorithm 8) on 78- and 80- reaction networks. Algorithm 9 consistently performs better in the case of the 78-reaction networks as the processor count goes from 8 to 128. However, situation is different in the case of the 80- reaction network where Algorithm 9 exhibits deteriorating time scalability compared to the Algorithm 8 as it transitions from 256 to 512 processors. As is going to be shown in the sequel, this was the price paid to attain an improved memory scalability achieved by using Global Arrays in Algorithm 9.

By the design of the Nullspace Algorithm, we expect that with the increase of used processors and well attained load balancing by means of the interleaved generation of candidates, the generation of candidate modes and the local pruning steps will scale down in time and memory usage. The remaining parts of the algorithm (e.g. global pruning and the expansion of the current nullspace matrix) may see an insignificant increase in the computation time. We do observe this trend in the 78- and 80-reaction networks, except in the cases when moving from 64 to 128 processors and 128 to 256 processors, respectively. These expectations are less evident in the case of the 83-reaction network, as in going from 1024 to 2048 processors the time for the generation of candidate modes almost doubled, which probably occurs due to the increased communication volume and cost. To explain this behavior a more thorough profiling and monitoring of the communication and memory usage will need to be done. Usage data of the collective operations `GET/GATHER` and `PUT/SCATTER` in the Global Array library was collected using the `printStats()` function and presented in the Table 6.3. For each of the four used collective operations the table reports the mean value per processor of the (1) number of calls performed (2) number of processors participating in the operation (3) amount of data sent/received remotely. Prior to analyzing the degrading scalability in the run on the 83-reaction network and during the transition from 1024 to 2048 processors, it is important to note that the scalability of the local pruning part of the algorithm depends on the successful scalability of the generation of candidate modes portion. Accordingly, it is important to analyze separately the load balancing and scalability of the `GenerateCandsGA` routine.

While the number of `GET` and `GATHER` calls scales up with the number of processors, the total amount of data sent per processor using `GET` and `GATHER` does not scale down with the same factor. On the other side, while the total amount of data sent using `PUT` and `SCATTER` scales down with the increase of the number of processors, the number of calls of respective operations does not scale up with the same factor. We also noticed that the share of data used in all collective operations that is sent locally is just a very small fraction of the total data. The deteriorating scalability of the routine `GenerateCandsGA` shown for the 83-reaction network is likely caused by the performance of the `GET` and `GATHER` functions and poorly exploited locality.

To track the memory usage, we read the values of the size of the physical and virtual memory (`VmRSS` and `VmSize` fields from the `/proc/pid/status` files of each of the running processes, respectively) and plot the mean value. The two values are sampled at the end of the last iteration of the Nullspace Algorithm. Figures 6.1 and 6.2 show the memory usage data for the case of the 78- and 80- reaction yeast networks as sampled during the runs of Algorithms 8 and 9. Similarly, Figure 6.3 shows the `VmRSS` and `VmSize` values for the case of the run of Algorithm 9 on the 83-reaction yeast network, as we were unable to fit this metabolic network model as input to the Algorithm 8. As profiled, the Global Arrays implementation attains significantly better physical memory scalability, unlike the MPI-only based implementation in Algorithm 8 where the scalability is negligible. The reported virtual memory size is significantly larger in the case of the Global Arrays implementation, but it should not be of concern since that corresponds to the heap and stack memory reserved by the Memory Allocator of the Global Arrays library during the initialization of the environment.

Finally, the computation of the elementary modes in the 83-reaction network using the Global Arrays supported parallel Nullspace Algorithm is an improvement as we were previously unable to solely use the Combinatorial Parallel Nullspace Algorithm (Algorithm 8) for its computation. Instead, the Combinatorial Parallel Nullspace Algorithm had to be combined with the divide-and-conquer approach [149] to reduce the memory footprint and complete the computation as shown in chapter 5. Table 6.4 compares the run of Global

Arrays supporte dand Combined parallel Nullspace Algorithm on 64 and 128 cores using Sandy Bridge extension of the "Itasca" parallel platform described in section 5.4.

Table 6.4: *Computation of EFMs on 83-reaction S. cerevisiae metabolic network using Algorithms 9 and 7 Sandy Bridge extension*

algorithm	divide-and-conquer	GA-based
# cores	64	128
max memory per core used (GB)	3.708	7.505
max memory per core available (GB)	16	8
# cand modes	167,754,867,391,704	66,144,590,373,585
cumul. time	22 h 28 min 18 sec	3 h 21 min 47 sec
Total # EFM: 68,868,602		

6.4 Conclusion and Future Work

This chapter details the proposed and implemented parallel Nullspace Algorithm for the computation of elementary flux modes using the Global Arrays library. To the extent possible, it merges the design concepts from the earlier MPI-only based implementation, the Combinatorial Parallel Nullspace Algorithm [144], where interleaved generation of candidates yielded good time scalability and load balancing, with the capability and convenience of the Global Arrays to uniformly partition the data across all processors. This effort was countered by the need to communicate the large amount of data between processors which in methods such as `GET` and `GATHER` did not scale as expected. It was possible to compute the elementary flux modes on a large yeast metabolic network that was too large to fit in the previous MPI-only implementation. This computation was performed using 512 processors and yielded close to 70 million elementary flux modes.

Further efforts will be to explore third party profiling tools which may aid in the reorganization of the communication between the processors, as well as to better exploit the locality and processor adjacency, and to reduce the amount of data sent using collective operations of the Global Arrays library. Adding to the difficulty in profiling this Global Arrays based implementation was the existence of the intermediate ARMCI communication layer. There is also a possibility and plan to add the feature of out-of-core computation.

Chapter 7

Rational strain design using elementary modes

7.1 Introduction

This chapter builds up on the pathway-based approach used to enumerate multiple reaction knockout subsets as illustrated in subsection 2.3.3. In section 7.2 we propose the algorithms for *direct* and *indirect* enumeration of the efficient reaction knockout subsets using elementary flux modes. Section 7.3 outlines the results obtained with both direct and indirect algorithms on the metabolic networks of *E. coli* and *S. cerevisiae*. Finally, section 7.4 compares the algorithms proposed in this chapter to some of the earlier proposed pathway-based and optimization-based methods, particularly emphasizing the design and performance when compared to the exhaustive algorithm used to compute constrained minimal cut sets [117].

7.2 Methods

The algorithms proposed in this chapter directly or indirectly utilize elementary flux modes to enumerate multiple reaction knockout subsets. Their use is tied to the expense of computing all the elementary flux modes and as such are intended for those users who can afford such cost and aim to fully characterize the space of possible metabolic states. The concepts from Minimal Metabolic Functionality and CASOP frameworks are subsumed and quantitatively represented in the proposed algorithms. The Minimal Metabolic Functionality procedure, which was heretofore only manually used, is presented as one variation of the new algorithms. Prior to proposing our algorithms we outline the metabolic design criteria in subsection 7.2.1. Enumeration of multiple reaction knockout subsets allows the user to select the solutions which may fit best to the experimental circumstances. The direct enumeration algorithm is detailed in the subsection 7.2.2, while the indirect enumeration algorithm is given in the subsection 7.2.3. Algorithms are based on known graph theory algorithms of exponential complexity, but incorporate heuristics which aim to reduce the search space of possible knockout subset solutions by properly tuning algorithmic parameters.

Throughout the chapter, the *yield* of an elementary flux mode with respect to the target chemical or biomass reaction is defined as the ratio of the fluxes of either target chemical or biomass reaction versus the substrate reaction.

7.2.1 Metabolic design criteria

We summarize the two major design criteria which are later incorporated into the algorithms.

Network flexibility refers to the number of alternate pathways available for the conversion of substrate into target chemical. The Minimal Metabolic Functionality ([1]) concept aims for a low network flexibility, with as few preserved high-yielding modes as possible. In contrast, CASOP framework ([114]) strives to attain high network flexibility with high target chemical yield.

Growth-coupled production refers to the high target chemical yield which is coupled to the high biomass yield in the elementary mode or overall flux distribution. Optimization-based methods such as OptKnock require that in all conditions the high target chemical yield is coupled to the high biomass reaction yield. However, the design of Minimal Metabolic Functionality requests the existence of both growth-coupled and growth-uncoupled elementary flux modes, with growth-uncoupled modes capable of producing the target chemical at the maximum possible yield even in the absence of substrates needed for growth (ammonia in our example). Algorithms outlined in sequel can incorporate both growth-coupled and growth-uncoupled design features.

7.2.2 Direct enumeration of reaction knockout subsets using elementary flux modes

We begin with some basic definitions.

Definition 8. An elementary flux mode is called efficient if the yield of the target chemical reactions (e.g. ethanol) is not smaller than some specified minimum yield value, with respect to the given substrate (e.g. glucose). Efficient elementary flux modes may be growth-coupled or growth-uncoupled, where biomass yield is above or below some specified minimum yield value. □

Following the definition, we denote all the efficient elementary flux modes as $\mathtt{EFM}^{(\mathtt{eff})} = \mathtt{EFM}^{(\mathtt{eff})}_{\mathtt{growth}} \cup \mathtt{EFM}^{(\mathtt{eff})}_{\mathtt{nogrowth}}$.

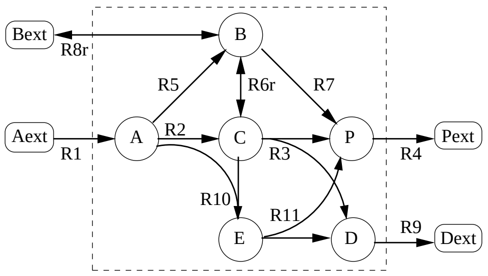


Figure 7.1: *Simple metabolic network*

Definition 9. Elementary flux mode is called <u>inefficient</u> if its target chemical reaction yield is below the specified minimum yield value. ($\texttt{EFM}^{(\texttt{ineff})} = \texttt{EFM}^{(\texttt{all})} \setminus \texttt{EFM}^{(\texttt{eff})}$)). □

Example 1. The matrix of elementary modes (7.2) corresponds to the metabolic network (7.1) with 6 metabolites and 11 reactions depicted in Figure 7.1. Let reaction R_1 correspond to the substrate reaction, R_4 to the target chemical reaction and R_9 to biomass. The maximum possible yields for both target chemical and biomass reactions is 2.0. Let the efficient modes be defined as having the target chemical yield of at least 50% of the maximum possible, while efficient modes are growth coupled if the biomass yield is at least 50% of the maximum possible. Modes M_1 and M_2 are not considered due to their zero substrate flux. Accordingly, columns M_7, M_8, M_9, M_{10} and M_{11} are the growth-coupled efficient modes, columns M_4 and M_5 are growth-uncoupled efficient modes and columns M_3 and M_6 are the inefficient modes. ■

$$N = \begin{array}{c|ccccccccccc} & R_1 & R_2 & R_3 & R_4 & R_5 & R_{6r} & R_7 & R_{8r} & R_9 & R_{10} & R_{11} \\ A & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ B & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & 0 & 0 \\ C & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ D & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ P & 0 & 0 & 1 & -1 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\ E & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 \end{array} \tag{7.1}$$

$$EFM = \begin{array}{c|ccccccccccc} & M_1 & M_2 & M_3 & M_4 & M_5 & M_6 & M_7 & M_8 & M_9 & M_{10} & M_{11} \\ R_1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ R_2 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1/2 & 0 & 0 \\ R_3 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ R_4 & 1 & 2 & 0 & 2 & 2 & 0 & 1 & 1 & 1 & 1 & 2 \\ R_5 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1/2 & 0 \\ R_6 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 1 & 0 & 1/2 & 1 \\ R_7 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ R_8 & -1 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ R_9 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 \\ R_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 1 \\ R_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \end{array} \tag{7.2}$$

Given a set of elementary flux modes, one may use a greedy algorithm to select reactions for knockout one at a time, adding it to a partial list of reactions previously selected. To attain low network flexibility, at each step a reaction r is added to a knockout subset based on maximizing the number of collapsed inefficient elementary modes. Specifically, reaction r is selected using the following score:

$$score(r; KO) = (1 - f_D) \cdot I(\mathtt{EFM}^{(\mathtt{eff})}_{\mathtt{growth}} \neq \emptyset) \cdot I(\mathtt{EFM}^{(\mathtt{eff})}_{\mathtt{nogrowth}} \neq \emptyset), \tag{7.3}$$

where $f_D = \frac{N^{(\mathtt{eff})}_{\mathtt{growth},KO\cup\{r\}} + N^{(\mathtt{ineff})}_{KO\cup\{r\}}}{N^{(\mathtt{eff})}_{\mathtt{growth},KO} + N^{(\mathtt{ineff})}_{KO}}$ is the fraction of efficient growth-coupled and inefficient modes which remain after the addition of the candidate reaction r to the current reaction knockout subset KO. $N^{(\mathtt{eff})}_{\mathtt{growth},KO}$ and $N^{(\mathtt{ineff})}_{KO}$ are equal to the number of efficient growth-coupled and inefficient modes that remain after the deletion of reactions from subset KO. Similarly, $N^{(\mathtt{eff})}_{\mathtt{growth},KO\cup\{r\}}$ and $N^{(\mathtt{ineff})}_{KO\cup\{r\}}$ are equal to the number of efficient growth-coupled and inefficient modes that remain after the deletion of reactions from subset $KO \cup \{r\}$. The notation $I(\cdot)$ denotes an indicator function, equal to one if the underlying condition is true, otherwise equal to zero. Here they are used to enforce the condition that at least one efficient growth-coupled and efficient growth-uncoupled mode are preserved.

At the opposite extreme, to attain high network flexibility it may be decided to maximize the number of efficient growth-coupled modes which are preserved, while collapsing as many inefficient modes as possible. This can be modeled by the following score:

$$score(r; KO) = f_{ED} \cdot (1 - f_{ID}) \cdot I(\mathtt{EFM}^{(\mathtt{eff})}_{\mathtt{nogrowth}} \neq \emptyset), \tag{7.4}$$

where $f_{ED} = \frac{N^{(\mathtt{eff})}_{\mathtt{growth},KO\cup\{r\}}}{N^{(\mathtt{eff})}_{\mathtt{growth},KO}}$ and $f_{ID} = \frac{N^{(\mathtt{ineff})}_{KO\cup\{r\}}}{N^{(\mathtt{ineff})}_{KO}}$ are the fraction of efficient growth-coupled and inefficient modes which remain after the addition of the candidate reaction r to the reaction knockout subset KO. $N^{(\mathtt{eff})}_{\mathtt{growth},KO}$ and $N^{(\mathtt{eff})}_{\mathtt{growth},KO\cup\{r\}}$ are equal to the number of efficient growth-coupled modes which remain after the deletion of reactions from subsets KO and $KO \cup \{r\}$, respectively. Similarly, $N^{(\mathtt{ineff})}_{KO}$ and $N^{(\mathtt{ineff})}_{KO\cup\{r\}}$ denote the number of the inefficient modes which remain after the deletion of reactions in the reaction knockout subsets KO and $KO \cup \{r\}$, respectively.

The greedy algorithm for direct enumeration of knockouts is implemented in Algorithm 10. Enumeration is carried out with a breadth-first branch-and-bound algorithm, implemented in a non-recursive manner. We construct a partial search tree with each node representing a partial knockout subset (a set of reactions). The root node corresponds to the empty set. Every other node has a non-empty knockout subset consisting of one reaction added to the knockout subset of its parent node. At each stage in the algorithm, the first unexpanded node is fetched. If this node's corresponding knockout subset `u` collapses all efficient modes, this knockout set is discarded. If all inefficient modes are collapsed, this knockout set `u` is added to `T`, the collection of successful knockout subsets. Otherwise if the size of the knockout set does not exceed the user set upper limit `max_ko_length`, each possible reaction r is added to `u` – each possible $\texttt{u} \cup \{r\}$ forms a child node which is put on the queue Q for later expansion. Instead of considering all possible new reactions `r`, we score each combination $\texttt{u} \cup \{r\}$ using (7.3) or (7.4) and examine only the top `max_reac_cands` candidates. In this way we prune the search space to a manageable size.

In order to further control the search space, the branching factor `max_reac_cands` is adjusted in Algorithm 10 during the search: for knockout subsets of length m, the branching factor is limited to $\texttt{max\_reac\_cands}/2^{\texttt{m}}$. In addition, the set of reactions eligible to extend a knockout subset may be limited to the set `cand_reacs`. For example, one may decide that exchange reactions are ineligible and omit them from `cand_reacs`.

Algorithm 10 [T] = direct_enum_ko($\texttt{EFM}^{(\texttt{all})}$, cand_reacs, ...)

```
Input:
 1: $\texttt{EFM}^{(\texttt{all})}$ - set of all elementary flux modes
 2: cand_reacs - reactions available for knockout
 3: max_ko_length - maximal length of efficient reaction knockout subset
 4: max_reac_cands - maximal number of candidate reactions to consider for expansion at single enumeration step
Output:
 5: T - efficient reaction knockout subsets
 6: $(\texttt{EFM}^{(\texttt{eff})}_{\texttt{growth}}, \texttt{EFM}^{(\texttt{eff})}_{\texttt{nogrowth}}, \texttt{EFM}^{(\texttt{ineff})}) \leftarrow \texttt{EFM}^{(\texttt{all})}$
 7: T ← ∅
 8: ▷ Q contains reaction subsets which are not efficient
 9: create a queue Q
10: enqueue {∅} onto Q
11: while Q is not empty:
12:     dequeue an item from Q into u
13:     if u collapses all modes in $\texttt{EFM}^{(\texttt{ineff})}$:
14:         add u to T
15:         continue
16:     if u has max_ko_length elements:
17:         continue
18:     ▷ for reactions in cand_reacs \ u compute score using equations (7.3) or (7.4)
19:     ▷ and rank reactions in decreasing order of the score value
20:     F ← return top $\max(1, \frac{\texttt{max\_reac\_cands}}{2^{length(\texttt{u})}})$ ranked
21:           candidates among reactions cand_reacs \ u
22:           (omitting those with score 0)
23:     for each candidate reaction f in F :
24:         enqueue u ∪ {f} onto Q
```

7.2.3 Indirect enumeration of reaction knockouts from elementary modes

An alternative search strategy is to first search for feasible sets of elementary modes, where a set of modes is "feasible" if there exists a reaction knockout subset which collapses all inefficient elementary modes leaving those in the set intact. Formally, we have:

Definition 10. A subset of efficient elementary flux modes is called <u>feasible</u> if there is at least one subset of reactions which when deleted from the metabolic network will collapse all the inefficient elementary flux modes leaving the modes in the efficient subset intact. □

Definition 11. A feasible subset of efficient elementary flux modes is called <u>maximal</u> if it cannot be expanded by adding another efficient mode and remain feasible at the same time. □

In the sequel we use the acronym MFES to denote a Maximal Feasible efficient Elementary flux mode Subset.

It is possible to identify (Algorithm 12) feasible sets of elementary modes (MFES's) directly without computing minimal knockout sets. This may be useful in case it is desired to preserve certain side-effects not included in the stoichiometric model. It also indicates alternatives available to achieve a given yield using a knockout strategy. Once such feasible sets have been identified, one may use Algorithm 13, a slight variation of Algorithm 10, differing only in the contents of `cand_reacs` and the score used to select the next candidate reaction. The score is computed as given in (7.5), given that $f_{ID} = \frac{N^{(\mathtt{ineff})}_{KO\cup\{r\}}}{N^{(\mathtt{ineff})}_{KO}}$ is a fraction of inefficient modes which remain after expanding the current knockout subset KO with the candidate reaction `r`, where `r` $\in$ `cand_reacs`. It is combined with the requirement that the currently considered MFES should be intact.

$$score(r; KO) = 1 - f_{ID} \tag{7.5}$$

The combined procedure is represented by Algorithm 11. As different MFESs are processed, hashing is used to eliminate duplicate knockout subsets.

Example 1. (continued) We will illustrate the concept of feasibility and maximality of the elementary flux mode subset. As earlier mentioned, efficient modes are columns M_4, M_5, M_7, M_8, M_9, M_{10}, M_{11}, while inefficient modes are M_3, M_6. Let the set of reactions that can be used for knockout be `cand_reacs` $= \{R_2, R_3, R_5, R_{6r}, R_7, R_{8r}, R_{10}, R_{11}\}$. Any single efficient elementary flux mode by itself constitutes a feasible efficient elementary mode subset. The subset of efficient modes comprised of columns M_4, M_5, M_7, M_8, M_9, M_{10} is also feasible, as there is a reaction knockout subset $\{R_{8r}\}$ which can collapse both inefficient modes M_3, M_6, and leave the efficient subset intact. In addition, this subset is also maximal, as the addition of the remaining efficient mode M_{11} to the subset would make it infeasible. The feasibility and maximality may also be defined only on the efficient growth-coupled modes, as is done in the algorithms. ■

7.2.4 Complexity analysis

Algorithms 10, 12 and 13 in worst case have an exponential complexity. The complexity of the algorithms is controlled by the number of candidate reactions or candidate elementary flux modes considered for addition at every step, as well as by the length of enumerated

Algorithm 11 `[T]= indirect_enum_ko(`$\texttt{EFM}^{(\texttt{all})}$`)`

```
Input:
 1: $EFM^{(all)}$ - elementary flux modes
Output:
 2: T - efficient reaction knockout subsets
 3: ($EFM^{(eff)}_{growth}$,$EFM^{(eff)}_{nogrowth}$,$EFM^{(ineff)}$) ← $EFM^{(all)}$
 4: S ← ∅; T ← ∅;
 5: for every mode e ∈ $EFM^{(eff)}_{growth}$:
 6:         S ← S ∪ enum_mfes(e,$EFM^{(eff)}_{growth}$,$EFM^{(ineff)}$)
 7: for every MFES s ∈ S
 8:         T ← T ∪ indirect_enum_ko_from_mfes(s,$EFM^{(all)}$)
```

Algorithm 12 `[S]= enum_mfes(`$e^{(eff)}_{growth}$,$EFM^{(eff)}_{growth}$,$EFM^{(ineff)}$`, ...)`

```
Input:
 1: $e^{(eff)}_{growth}$ - initial efficient growth-coupled elementary mode
 2: $EFM^{(eff)}_{growth}$ - set of all efficient growth-coupled elementary modes
 3: $EFM^{(ineff)}$ - set of inefficient elementary modes
 4: max_mfes_length - maximal length of MFES
 5: max_efm_cands - maximal number of candidate modes to consider for expansion
Output:
 6: S - maximal efficient elementary modes subsets
 7: S ← ∅
 8: ▷ Q contains only subsets of modes that are feasible; $e^{(eff)}_{growth}$ is always FES;
 9: create a queue Q
10: enqueue $e^{(eff)}_{growth}$ onto Q
11: while Q is not empty:
12:     dequeue an item from Q into v
13:     if v has max_mfes_length elements:
14:         add v to S
15:         continue
16:     index_last ← index of last element in v,($EFM^{(eff)}_{growth}$(index_last) = v(end))
17:     ▷ rank modes in $EFM^{(eff)}_{growth}$(index_last + 1 : end) by # of reactions remaining for knockout when each mode
    is added to the subset v
18:     G ← find no more than max(1,$\frac{max\_efm\_cands}{2^{length(v)}}$) top ranked candidate modes in
    $EFM^{(eff)}_{growth}$(index_last + 1 : end)
19:     ▷ if there are no modes in $EFM^{(eff)}_{growth}$(index_last + 1 : end) that may be appended to v, then v is MFES
20:     if G is empty:
21:         add v to S
22:     for each candidate mode g in G:
23:         enqueue v ∪ {g } onto Q
```

Algorithm 13 `[T] = indirect_enum_ko_from_mfes(mfes,`$EFM^{(all)}$`)`

```
Input:
 1: mfes - maximal feasible efficient elementary flux mode subset
 2: $EFM^{(all)}$ - set of all elementary flux modes
Output:
 3: T - efficient reaction knockout subsets
 4: T ← ∅
 5: cand_reacs ← reactions not used in mfes and available for knockout
 6: [T] = direct_enum_ko($EFM^{(all)}$,cand_reacs)
```

subsets. In *indirect* enumeration of the reaction knockout subsets, the number of reaction candidates that is considered is reduced for each MFES as depicted in Algorithm 13.

Plots in figure 7.2 illustrate the remaining elementary modes and their position with respect to the biomass and target chemical yield. The remaining modes seem to be lying on

Table 7.1: *Elementary modes and yield values in 68-reaction E. coli central metabolism network using different substrates ([1])*

substrate	xylose or arabinose	manose	galactose	glucose
no. of modes	15,185	27,033	34,016	38,001
anaerobic growth				
no. of modes ($\mathtt{EFM}^{(\mathtt{all})}$)	1,004	2,841	1,620	5,010
maximum ethanol yield	1.667	2.00	2.00	2.00
maximum biomass yield	0.0257	0.0513	0.0342	0.0513
size of $\mathtt{EFM}^{(\mathtt{eff})}_{\mathtt{growth}}$ [1]	66	124	208	176
size of $\mathtt{EFM}^{(\mathtt{eff})}_{\mathtt{nogrowth}}$ [2]	12	22	12	28

[1]efficient growth-coupled modes defined as having biomass and ethanol yield at least 50% and 60% of the maximum, respectively.

[2]efficient growth-uncoupled modes defined as having ethanol yield at 100% of the maximum.

the curve which corresponds to the Pareto surface of the two-objective optimization task. This aligns with the observations discussed in [155] that the likeliest state of the metabolic network is at some point along the Pareto surface.

7.3 Results

We run the direct and indirect enumeration algorithms on the models of the central metabolism of *E. coli* and *S. cerevisiae*. Elementary flux modes and their yield values are computed using the software ElMo-Comp (http://elmocomp.sourceforge.net), while they may be easily loaded and explored from MATLAB or other scripting language. Algorithms for reaction knockout subset enumeration are implemented in C++ and are available as ElMo-Knock program at `http://elmocomp.sourceforge.net/elmoknock.php`. Program was compiled using GNU C++ compiler tool on a Linux machine with Intel Core i7-2600 processor running at 3.40 GHz, with 8 MB cache memory and 16 GB main memory. The resulting list of reaction knockout subsets for both *E. coli* and *S. cerevisiae* is available at `http://elmocomp.sourceforge.net/elmo_knock/ecoli_and_yeast_results.zip`.

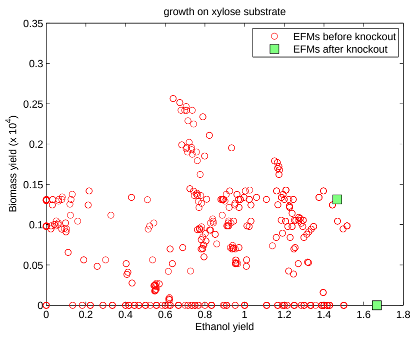
7.3.1 Results on *Escherichia coli*

The model of the central metabolism network of *E. coli* ([1]) which we used has 68 reactions and uses pentose (xylose or arabinose) and hexose (glucose, manose, galactose) substrates. In this chapter we observe the cases when only a single substrate is consumed at a time – the networks are identical except for the substrate uptake reactions. In all cases of the substrate uptake, it is assumed that the growth conditions are anaerobic, hence eliminating from consideration elementary modes with non-zero flux in one of the reactions OPM1 and OPM2.

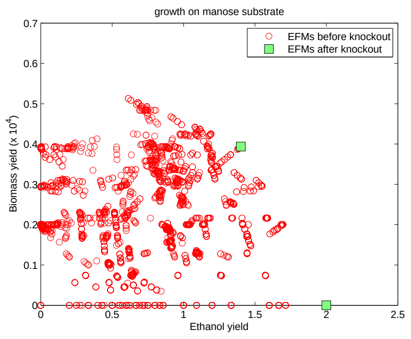
Determination of the threshold values to define efficient and inefficient modes requires prior exploration of yield values across all elementary flux modes. In order to replicate the design earlier attained in Minimal Metabolic Functionality ([1, 112]), the efficient growth-coupled modes are defined as having at least 50% of the maximal possible biomass yield and 60% of the maximum possible ethanol yield. Similarly, efficient growth-uncoupled modes are defined as having maximum possible ethanol yield, regardless of the biomass yield.

Table 7.2 shows the results obtained running three algorithmic variations using four different carbohydrate substrates in each case. Algorithms are run without constraining the knockout subset length, and the number of enumerated reaction knockout subsets of length 5 to 13 is given in respective table columns. Following the execution of Algorithm 10 (score equations (7.3) and (7.4) for low and high network flexibility) and Algorithm 11, the number of shared reaction knockout subsets of the same length between Algorithm 10 (low flexibility) and Algorithm 10 (high flexibility), as well as between Algorithm 10 (high flexibility) and Algorithm 11 are given. The pairwise numbers of shared knockout subsets demonstrate the discriminatory power of each individual algorithmic variation. In the case of indirect enumeration, time to enumerate MFES (t_{MFES}) is orders of magnitude smaller than the time required to enumerate efficient reaction knockout subsets (t_{KO}). The branching factors in enumeration of efficient reaction knockout subsets in Algorithms 10 and 11 are 32 and 16, respectively, while the branching factor for the enumeration of MFES in Algorithm 11 is 32.

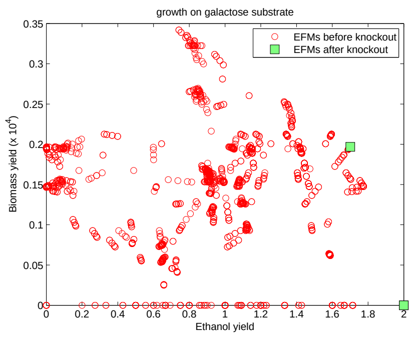
Table 7.3 shows 30 reaction knockout subsets found using Algorithm 10 for the four substrates of xylose, manose, galactose and glucose. In this specific case, any of the 30 enumerated knockout subsets collapses the metabolic network into 12 elementary flux modes, which do not necessarily coincide. We found one of the 30 solutions (PPP1, ANA2, FEM3, FEM7, TRA5, OPM4r) to almost coincide with the solution (PPP1, ANA2, FEM2, FEM3, FEM7, TCA10, OPM4r) given in ([1]), differing only in using TCA10 in place of the succinate transport reaction TRA5, and in the addition of FEM2. An analysis of the stoichiometric model shows FEM2 is redundant in this particular knockout set. Biochemical considerations not represented in the stoichiometric model allow one to eliminate the production of succinate using TCA10 instead [1]. Using the stoichiometric model, elimination of succinate production requires additional knockouts, such as TCA5 and TCA9r. Indeed, when Algorithm 10 is applied with the exchange reactions marked ineligible for knockout selection, one of the resulting knockouts is (PPP1, TCA5, TCA10, ANA2, FEM3, FEM7, TCA9r, OPM4r). In other words, all the enumerated subsets contain at least one reaction of the anapleurotic pathway (e.g. ANA2, GLB1, GLB2) and one from the tricarboxylic acid



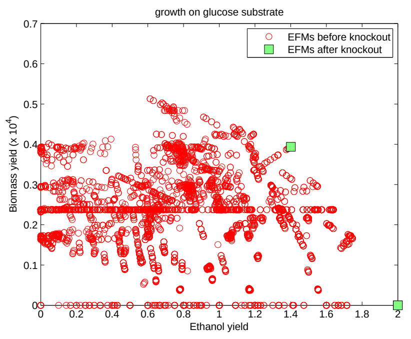
(a)



(b)



(c)



(d)

Figure 7.2: *Plot of all vs. remaining modes after applying the knockout of reactions $PPP1$, $ANA2$, $FEM3$, $FEM7$, $TRA5$, $OPM4r$ on the E. coli network grown on xylose (7.2(a)), manose (7.2(b)), galactose (7.2(c)) and glucose (7.2(d)) substrate, respectively.*

cycle.

7.3.2 Results on *Saccharomyces cerevisiae*

The metabolic network of anaerobic growth of *S. cerevisiae* on glucose was used to run the direct and indirect enumeration algorithms (Table 7.4). This network has 62 metabolites and 78 reactions (Appendix A, Table A.3) with 1,515,315 elementary modes ([149]). Initially, 5 modes with zero glucose uptake flux are excluded from consideration. After exploration of biomass and ethanol yields across all elementary flux modes, efficient growth-coupled elementary modes are defined as having the biomass and ethanol yield of at least 64% of maximum possible value each, while efficient growth-uncoupled modes are defined as having 100% of the maximum possible ethanol yield. This results in 15,088 efficient growth-coupled and 6 efficient growth-uncoupled elementary modes.

The results of running Algorithms 10 (low and high network flexibility) and 11 are given in Table 7.5. The number of reaction knockout subsets of length 3 to 11 are given in table columns, accompanied by the number of common subsets between subsets obtained running Algorithms 10 (low and high network flexibility), as well as Algorithms 10 (high network flexibility) and 11. In all runs shown in the table, both internal and exchange reactions were allowed to be used. All the enumerated reaction subsets include at least one exchange reaction. The branching factor for the enumeration of reaction knockout subsets in both variations of Algorithms 10 is 32, while the branching factor for the enumeration of MFES and reaction knockout subsets in Algorithm 11 is 2 each. Results of executing algorithms for the case when exchange reactions are marked ineligible for use in a knockout are available online in the supplementary archive file. We remark that while the length of the shortest subset which use exchange reactions is equal to 3, the length of the shortest subset when exchange reactions cannot be used is equal to 8.

A brief interpretation of the few shortest knockout subsets obtained using three algorithms can be done. Algorithm 10 with score function for low network flexibility finds one knockout subsets with four reactions (secretion of acetate ($R63$), lactate($R64$) and succinate ($R67$), conversion of acetyl-CoA into ethanol and CoA ($R32r$)). For the case of high network flexibility, the Algorithm 10 finds two subsets with four reactions ((i) secretion of acetate ($R63$) and lactate($R64$), glycerol-3-phosphate dehydrogenase ($R13r$), malate dehydrogenase ($R29r$) and (ii) secretion of acetate ($R63$), glycerol-3-phosphate dehydrogenase ($R13r$), malate dehydrogenase ($R29r$), lactate dehydrogenase ($R30r$)).

When we ran the Algorithm 11 we obtained one knockout subset with three reactions (secretion of acetate ($R63$), lactate ($R64$) and succinate ($R67$)), and three knockout subsets with four reactions ((i) secretion of acetate ($R63$), lactate($R64$) and succinate ($R67$), α-ketoglutarate dehydrogenase complex in mitochondria ($R24r$), (ii) secretion of acetate ($R63$), lactate($R64$) and succinate ($R67$), Glycerol-3-phosphate dehydrogenase ($R13r$), (iii) secretion of acetate ($R63$), lactate($R64$), isocitrate lyase ($R46$), succinate/malate antiport to mitochondria ($R89r$)).

Despite higher cost, Algorithm 11 found one solution of shorter length in the case of the three-reaction solution, and in the case of subset with four reactions, the result in both biomass and ethanol yield is at least as high as the ones produced using Algorithm 10. We believe that it motivates one to further improve the Algorithm 11 and the way feasible elementary flux mode subsets are formed, as it may allow better control to the user over the final metabolic network design.

7.4 Discussion

7.4.1 Parameterization of algorithms

Proposed enumeration algorithms can constrain the size of the search space in several ways such as by limiting the reactions that can be used in the knockout (`cand_reacs`), initial reaction knockouts prior to running the algorithms, maximal length of efficient reaction knockout subset (`max_ko_length`), as well as limiting the size of the feasible elementary flux mode subsets (`max_mfes_length`) in Algorithm 11. Similarly, both Algorithms 10 and 12, can control the size of search space by varying branching factors `max_reac_cands` and `max_efm_cands`.

7.4.2 Comparison to related pathways-based methods

Algorithms for direct and indirect enumeration are compared to related methods which were proposed earlier such as (a) Minimal Metabolic Functionality, (b) CASOP and (c) constrained Minimal Cut Sets.

7.4.2.1 Minimal Metabolic Functionality

The metabolic strain design embodied in the concept of Minimal Metabolic Functionality and previously manually used is contained and formalized in Algorithm 10 and score equation (7.3). The score equation (7.3) is proposed according to the original description of the Minimal Metabolic Functionality. The stopping condition for the algorithm, previously unspecified, is defined as the one of having at least one efficient growth-coupled and one efficient growth-uncoupled mode remain each.

7.4.2.2 CASOP (Computational Approach for Strain Optimization aiming at high Productivity)

A feature of the specific productivity described within the CASOP framework ([114]) is implemented using the score equation (7.4) in Algorithm 10.

7.4.2.3 Constrained Minimal Cut Sets

The enumeration of the cMCS (constrained Minimal Cut Sets) ([117]) is illustrated here using Algorithm 14 as proposed in the original work. This algorithm was run along with algorithms 10 and 11 for comparison and results are shown in tables 7.2 and 7.5.

Algorithm 14 is parameterized with the number of efficient growth-coupled and -uncoupled modes which should remain after knockout, specified by $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{growth}}$ and $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{nogrowth}}$. It performs an exhaustive enumeration of the reaction knockout subsets, which for a large number of elementary flux modes may be a limitation, as inefficient elementary modes are sequentially processed in the outer loop. The complexity of the algorithm grows significantly if the strict equality constraint on the parameters $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{growth}}$ and $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{nogrowth}}$ is replaced with the inequality constraint which is demonstrated in the table 7.2. In contrast, algorithms 10 and 11 proposed in this chapter are non-exhaustive and use a quantitative score to select the next candidate reaction for knockout, and at a lesser cost stand as an alternative for enumeration of a fraction of the entire solution set while still allowing the user to specify the metabolic network criteria in the mutant cell.

Algorithm 14 `[M]= enumerate_cMCS(`$\mathtt{EFM}^{(\mathtt{all})}, \mathtt{n}^{(\mathtt{eff})}_{\mathtt{growth}}, \mathtt{n}^{(\mathtt{eff})}_{\mathtt{nogrowth}}$`)`
`(based on [117])`

Input:
```
1: $EFM^{(all)}$ - elementary flux modes
2: $n^{(eff)}_{growth}$ - minimal number of growth-coupled efficient modes to retain
3: $n^{(eff)}_{nogrowth}$ - minimal number of growth-uncoupled efficient modes to retain
```
Output:
```
4: M - collection of cMCS

5: for every mode e in $EFM^{(ineff)}$
6:   for every mcs in M
7:     if mcs does not collapse e
8:       remove mcs from $T$
9:       for every reaction r in mode e
10:         if (mcs $\cup$ {r} knockout leaves $n^{(eff)}_{growth}$ and $n^{(eff)}_{nogrowth}$
11:             modes in $EFM^{(eff)}_{growth}$ and $EFM^{(eff)}_{nogrowth}$)
12:               add mcs $\cup$ {r} to M
13:         end
14:       end
15:     end
16:   end
17:   remove non-minimal cMCS from M
18: end
```

In addition, it was recently demonstrated ([95]) that the set of constrained minimal cut sets for the specified target reaction subset corresponds to the elementary modes of the dual network obtained from the original metabolic network under specified constraints. While that approach gives an alternate view of the problem, it does not eliminate the computational cost of computing elementary flux modes for the network which may in some cases even be larger than the original network.

7.4.3 Comparison to optimization-based methods

Results of running enumeration algorithms proposed in this chapter are compared to the results obtained using optimization-based methods, such as OptKnock or RobustKnock. Both OptKnock and RobustKnock aim to attain a metabolic network design where maximal biomass yield will be coupled to a high target chemical yield. RobustKnock is an improvement over OptKnock in the way that it guarantees to raise the low bound on the target chemical yield, avoiding low-yield competing pathways. The optimization-based methods assume that genetic modification of the metabolic network will be followed by the adaptive evolution ([59, 156, 157, 158, 159]), hence the cell will function across the elementary modes with maximal possible growth. This implies that the elementary flux modes with lower-than-maximal biomass yield do not need to be collapsed with the reaction knockout, as they will not be active. Reactions should be chosen for knockout only to collapse the modes with high biomass but low target chemical yield.

In order to enable the pathway-based algorithms to mimic this behavior a few changes are required. Elementary flux modes are split into (1) efficient modes (2) inefficient modes and (3) ignored modes. Two values are given to specify the minimal acceptable yield for (a) biomass reaction (usually 5% of the maximal possible value) and (b) target chemical reaction (usually above 50% of the maximal possible value). *Efficient modes* are modes with biomass and target chemical reaction flux values above the specified low bound value. *Inefficient modes* are elementary flux modes with biomass value above the specified low bound, and the target chemical flux value below the specified low bound. Finally, *ignored modes* are those elementary flux modes with the biomass flux value below the specified input low bound value. The goal is to find reaction knockout subsets which will collapse all the inefficient modes, leaving some of the efficient modes intact, preferably those with as high target chemical flux value as possible.

We will illustrate this using the *core model of the Escherichia coli* [2, 3] which has 72 metabolites and 95 reactions and is comprised of glycolysis, tricarboxylic acid cycle, pentose phosphate pathway, glyoxylate shunt, gluconeogenesis, nitrogen metabolism, electron transport chain. In our study, we consider the growth on glucose media and optimize the production of the succinate and ethanol. The network itself has 100,274 elementary flux modes. OptKnock and RobustKnock [62] were used as implemented in MATLAB running on the top of TOMLAB CPLEX optimizer (trial version) ([160]) to solve the underlying mixed-integer linear programming problem.

We ran OptKnock, RobustKnock and Algorithm 10 allowing not more than six reaction knockouts. The results of this run are given in Table 7.6. To rank the quality of the knockout subsets, one may consider the criteria of maximal biomass value, and the minimal and maximal guaranteed flux value of the chemical exchange reaction. We compared elementary mode based algorithm to the RobustKnock, since OptKnock cannot guarantee the lower maximal chemical production flux. In the case of the succinate production, elementary mode based algorithm attains equal or higher maximal chemical flux when compared to RobustKnock, with high enough minimal guaranteed chemical flux. On the other side, in the case of ethanol production, the elementary mode based method attains somewhat lower, but comparable maximal chemical yield when compared to RobustKnock, which seems to happen at the expense of higher biomass flux.

7.5 Conclusion

We propose and implement the algorithms for the enumeration of reaction knockout subsets in metabolic networks for efficient cellular strain design. The algorithms use the elementary flux modes and perform a non-recursive constrained breadth-first search to enumerate multiple reaction knockout subsets. The concept is demonstrated on two distinct networks of the central metabolism of *Escherichia coli* and *Sacharomyces cerevisiae* for the growth-coupled production of ethanol, and compared to previous known pathways-based and optimization-based methods. In particular, the proposed algorithms stand as alternatives to the exhaustive enumeration performed in the previously proposed algorithm for the enumeration of constrained minimal cut sets. There may be several potential improvements to the algorithms described in this chapter. First, we may explore different forms of score functions and heuristics used to rank knockout candidate reactions for inclusion in the knockout subset during the course of algorithm execution. Second, beside considering reaction deletion one may also consider up- or down- regulating reaction expression, though that approach requires experimental results and knowledge of the reference wild-type metabolic network flux distribution. Third, gene regulatory rules may be incorporated into the implementation as one possible way of constraining the solution space.

Table 7.2: *Results of applying Algorithm 10 for the case of low and high network flexibility (score criteria in equations (7.3) and (7.4)), Algorithm 11 and Algorithm 14 on 68-reaction E. coli metabolic network ([1]) using xylose (or arabinose), galactose, manose and glucose substrates. The knockout subset of 30 reactions of length 6 that is marked in a box turned out to be common to all four substrate cases.*

	algorithm	# MFES	# eff. reaction KO subsets	# of eff. reaction knockout subsets by length							time (sec)	
				5	6	7	8	9	10	11-13	t_{MFES}	t_{KO}
xylose or arabines	Alg. 10 low flexibility	N\A	2,822[1]	20	296	797	689	470	266	284	N\A	1.4
	Alg. 10 high flexibility	N\A	2,598[1]	16	129	483	1113	389	60	400	N\A	1.54
	Alg. 11	112[2]	1,165[2]	10	123	383	351	190	95	13	0.02	4.28
	Alg. 14	N\A	528[3]	0	20	144	244	120	0	0	N\A	5.41
	Alg. 14	N\A	1568[4]	20	176	624	604	144	0	0	N\A	99.72
	shared (Alg. 10 low flexibility, Alg. 10 high flexibility)			16	91	96	45	8	9	164		
	shared (Alg. 10 high flexibility, Alg. 11)			8	66	156	161	75	20	2		
manose	Alg. 10 low flexibility	N\A	2,531	36	300	1076	747	294	14	64	N\A	2.6
	Alg. 10 high flexibility	N\A	3,249	10	272	1052	887	259	89	635	N\A	13.55
	Alg. 11	357	2,917	23	297	854	911	486	319	27	0.024	12.3
	Alg. 14	N\A	888[3]	0	4	84	400	400	0	0	N\A	88.10
	Alg. 14	N\A	2200[4]	46	322	612	780	440	0	0	N\A	647.28
	shared (Alg. 10 low flexibility, Alg. 10 high flexibility)			0	46	233	73	13	2	30		
	shared (Alg. 10 high flexibility, Alg. 11)			5	96	244	128	30	7	0		
galactose	Alg. 10 low flexibility	N\A	3,010	36	278	608	607	913	382	186	N\A	1.97
	Alg. 10 high flexibility	N\A	1,984	20	82	224	617	492	80	433	N\A	2.56
	Alg. 11	1,444	2,708	18	225	726	904	553	226	56	0.062	13.78
	Alg. 14	N\A	1084[3]	0	4	104	496	480	0	0	N\A	28.89
	Alg. 14	N\A	2432[4]	36	268	512	1040	576	0	0	N\A	447.04
	shared (Alg. 10 low flexibility, Alg. 10 high flexibility)			20	24	24	10	15	0	66		
	shared (Alg. 10 high flexibility, Alg. 11)			10	29	98	148	66	0	58		
glucose	Alg. 10 low flexibility	N\A	2,281	0	$\boxed{30}$	431	808	633	207	172	N\A	5.46
	Alg. 10 high flexibility	N\A	2,712	0	21	227	537	691	392	795	N\A	7.17
	Alg. 11	359	2,907	0	44	429	776	698	527	433	0.005	33.43
	Alg. 14	N\A	1992[3]	0	60	428	784	720	0	0	N\A	266.54
	Alg. 14	N\A	2722[4]	0	106	676	1180	760	0	0	N\A	2082.75
	shared (Alg. 10 low flexibility, Alg. 10 high flexibility)			0	0	18	2	0	0	34		
	shared (Alg. 10 high flexibility, Alg. 11)			0	12	67	59	6	0	0		

[1] *branching factor* (`max_reac_cands`) value for enumeration of reaction knockout subsets is 32

[2] *branching factor* (`max_efm_cands`) value is 32 for maximal feasible elementary mode subset enumeration and 16 for efficient reaction knockout subset enumeration

[3] algorithm is run fixing the parameters $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{growth}}$ =8 and $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{nogrowth}}$ =4

[4] algorithm is run for $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{growth}}$ >0 and $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{nogrowth}}$ >0

Table 7.3: *List of enumerated reaction knockout subsets as minimal length subsets found across all four used substrates (xylose or arabinose, manose, galactose and glucose) for the efficient ethanol production in 68-reaction E. coli network ([1]). The 30 subsets are extracted from the results presented in Table 7.2 in the case when Algorithm 10 (low flexibility) was applied. Lower table illustrates minimal required biomass and ethanol yields as required for the existence of growth-coupled and growth-uncoupled modes, number of modes which remain after applying one of 30 knockouts, and minimum biomass and ethanol yields in the residual network.*

REACTION KNOCKOUT SUBSETS	
PPP1 TCA5 GLB1 FEM3 TRA2 TCA9r	PPP2 TCA5 GLB1 FEM3 TRA2 TCA9r
PPP1 TCA5 GLB2 FEM3 TRA2 TCA9r	PPP2 TCA5 GLB2 FEM3 TRA2 TCA9r
PPP1 TCA7 ANA2 FEM3 TRA2 TRA5	PPP2 TCA7 ANA2 FEM3 TRA2 TRA5
PPP1 TCA7 FEM3 TRA2 TRA5 TCA9r	PPP2 TCA7 FEM3 TRA2 TRA5 TCA9r
PPP1 GLB1 FEM3 TRA2 TCA6r TCA9r	PPP2 GLB1 FEM3 TRA2 TCA6r TCA9r
PPP1 GLB2 FEM3 TRA2 TCA6r TCA9r	PPP2 GLB2 FEM3 TRA2 TCA6r TCA9r
PPP1 ANA2 FEM3 FEM7 TRA5 OPM4r	PPP2 ANA2 FEM3 FEM7 TRA5 OPM4r
PPP1 ANA2 FEM3 FEM8 TRA5 OPM4r	PPP2 ANA2 FEM3 FEM8 TRA5 OPM4r
PPP1 ANA2 FEM3 TRA2 TRA5 TCA8r	PPP2 ANA2 FEM3 TRA2 TRA5 TCA8r
PPP1 ANA2 FEM3 TRA2 TRA5 TCA9r	PPP2 ANA2 FEM3 TRA2 TRA5 TCA9r
PPP1 ANA2 FEM3 TRA2 TRA5 OPM4r	PPP2 ANA2 FEM3 TRA2 TRA5 OPM4r
PPP1 FEM3 FEM7 TRA5 TCA9r OPM4r	PPP2 FEM3 FEM7 TRA5 TCA9r OPM4r
PPP1 FEM3 FEM8 TRA5 TCA9r OPM4r	PPP2 FEM3 FEM8 TRA5 TCA9r OPM4r
PPP1 FEM3 TRA2 TRA5 TCA8r TCA9r	PPP2 FEM3 TRA2 TRA5 TCA8r TCA9r
PPP1 FEM3 TRA2 TRA5 TCA9r OPM4r	PPP2 FEM3 TRA2 TRA5 TCA9r OPM4r

		Xyl (or Ara)	Man	Gal	Gluc
growth-coupled modes	min biomass	50%			
	min ethanol	60%			
	# of modes satisfying constraints	66	124	208	176
	# of modes remaining after knockout	4	4	4	4
	min biomass after knockout	51%	77%	58%	77%
	min ethanol after knockout	88%	70%	85%	70%
growth-uncoupled modes	min biomass	0%			
	min ethanol	100%			
	# of modes satisfying constraints	12	22	12	28
	# of modes remaining after knockout	8	8	8	8
	min biomass after knockout	0%	0%	0%	0%
	min ethanol after knockout	100%	100%	100%	100%

Table 7.4: *Elementary modes and yield values in S. cerevisiae central metabolism network on glucose substrate*

total # of modes ($\texttt{EFM}^{(\texttt{all})}$)	1,515,315
maximum ethanol yield	2.00
maximum biomass yield	0.0355
size of $\texttt{EFM}^{(\texttt{eff})}_{\texttt{growth}}$ [1]	15,088
size of $\texttt{EFM}^{(\texttt{eff})}_{\texttt{nogrowth}}$ [2]	6

[1]efficient growth-coupled modes defined as having biomass and ethanol yield at least 64% of the maximum each.

[2]efficient growth-uncoupled modes defined as having ethanol yield at least 100% of the maximum.

Table 7.5: *Results of applying Algorithm 10 for the case of low and high network flexibility (score criteria in equations (7.3) and (7.4)), Algorithm 11 and Algorithm 14 on S. cerevisiae metabolic network using glucose substrate*

algorithm	# MFES	# eff. KO	# of eff. reaction knockout subsets by length							time (sec)	
		subsets	3	4	5	6	7	8	$\geq$9	t_{MFES}	t_{KO}
Alg. 10 LF	N\A	6,320[3]	0	1	36	342	1507	2317	1239	N\A	32.26
Alg. 10 HF	N\A	3,032[3]	0	2	74	745	1444	345	284	N\A	249.23
Alg. 11	15,088[4]	2,954[4]	1	3	20	90	115	89	76	89.95	2380.64
Alg. 14	N\A	8[5]	2	6	0	0	0	0	0	0	685.56
shared (Alg. 10 LF, Alg. 10 HF)			0	0	0	0	0	0	0		
shared(Alg. 10 LF, Alg. 11)			0	0	0	2	6	10	18		

[1] *LF* - low network flexibility

[2] *HF* - high network flexibility

[3] *branching factor* (`max_reac_cands`) value for enumeration of reaction knockout subsets is 32

[4] *branching factor* (`max_efm_cands`) value for Alg. 11 is 2 for maximal feasible elementary mode subset enumeration and 2 for efficient reaction knockout subset enumeration

[5] algorithm is run for $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{growth}}$ >0 and $\mathtt{n}^{(\mathtt{eff})}_{\mathtt{nogrowth}}$ >0 and maximum knockout length 4

Table 7.6: *Knockout strategies predicted by Algorithm 10, OptKnock and RobustKnock on 95-reaction core E. coli metabolic network model ([2, 3]) for the production of succinate and ethanol. Glucose uptake flux is fixed to be 10.*

Chem.	KO size	Method	Knockout Subset	Max. Bio	Min. Chem	Max. Chem.
Succ	-	Wild-Type	-	0.9166	0	0
	2	Alg. 1	CO2t, PGI	0.2766	9.449	9.449
		OptKnock	CO2t, PGI	0.2766	9.449	9.449
		RobustKnock	CO2t, PGI	0.2766	9.449	9.449
	3	Alg. 1	CO2t, EX_ac, PGI	0.2528	9.497	11.169
		OptKnock	CO2t, EX_ac, PGI	0.2528	9.497	11.169
		RobustKnock	CO2t, PFL, PGI	0.2195	10.622	10.622
	4	Alg. 1	EX_pyr, FORti, CO2t, PGI	0.1252	12.193	12.193
		OptKnock	CO2t, EX_ac, PGI	0.2528	9.497	11.169
		RobustKnock	ACK, CO2t, PFL, PGI	0.2195	10.622	10.622
	5	Alg. 1	EX_lac_D, FORti, THD2, ACALD, EX_o2	0.1095	10.156	12.778
		OptKnock	CO2t, PFL, PGI, THD2	0.2195	10.622	10.622
		RobustKnock	CO2t, EX_ac, PGI	0.2528	9.497	11.169
	6	Alg. 1	EX_lac_D, GLUSy, PFL, THD2, ACALD, EX_o2	0.1095	10.156	12.778
		OptKnock	CO2t, EX_ac, PGI	0.2528	9.497	11.169
		RobustKnock	AKGt2, CO2t, GND, ME2, PFL, THD2	0.2304	12.549	12.549
Etoh	-	Wild-Type	-	0.9166	0	0
	2	Alg. 1	EX_h2o, PGI	0.3264	11.703	11.761
		OptKnock	EX_for, EX_o2	0.2709	14.8621	14.8621
		RobustKnock	FORti, O2t	0.2709	14.862	14.862
	3	Alg. 1	CYTBD, ATPS4, MDH	0.2867	14.149	15.327
		OptKnock	ACK, O2t, PYK	0.2072	16.071	16.851
		RobustKnock	ACK, CYTBD, PYK	0.2072	16.071	16.851
	4	Alg. 1	CYTBD, ATPS4, G6PDH2, MDH	0.2598	16.052	16.052
		OptKnock	ACK, ATPS4, PYK, SUCOAS	0.1620	0	17.538
		RobustKnock	FRD7, O2t, PTA, PYK	0.2072	16.851	16.851
	5	Alg. 1	AKGDH, GND, ATPS4r, EX_o2, MDH	0.2598	16.052	16.052
		OptKnock	ATPS4, EX_ac, GLUDy, PYK, SUCOAS	0.1494	0	17.729
		RobustKnock	ACK, FRD7, GLUDy, O2t, PYK	0.1933	17.062	17.062
	6	Alg. 1	AKGDH, ATPM, GND, ATPS4, EX_o2, MDH	0.2598	16.052	16.052
		OptKnock	AKGDH, ATPS4, GLUDy, PFL, PTA, PYK	0.1494	0	17.729
		RobustKnock	CYTBD, FUM, G6PDH2, ME2, PTA, PYK	0.1852	17.185	17.185

Chapter 8

Use of optimization methods in cellular strain design and elementary mode analysis

8.1 Introduction

As illustrated in previous chapters, the computation of all the elementary flux modes still presents an unsurmountable challenge when metabolic networks comprising of several hundred reactions are used. In the event when the cost of computing dozens of millions of elementary flux modes can be afforded, it remains still unclear how to distinguish modes and their relative contributions to the functioning of the cell and overall reaction of the external metabolites. This chapter proposes an idea of using optimization methods to reduce the original network to a smaller one, which would be used to either compute elementary flux modes or use some of the enumeration algorithms presented in chapter 7. The rationale behind the proposed optimization framework is to infer the likely inactive reactions the residual efficient metabolic subnetwork. In the subsequent sections several approaches will be illustrated which are based on solving a regularized non-linear convex optimization problem. Beforehand, it would be useful to put a reminder of the earlier work surrounding the possible evolutionary goals of a biological cell and existing hypotheses.

8.2 Cellular evolutionary objectives

Several possible notions of cellular objective among wild-type and mutant cells exist. It was earlier established and experimentally matched in several cases that the cell of a microorganism such as *E. coli* aims to improve its fitness by maximizing the cellular growth i.e. flux of the biomass reaction. This lead to the case of the secretion of possible cellular by-products usually being uncoupled from growth and at very low levels. Genetic modifications in the cell were used to enforce the growth-coupled production of the desired chemical by-product assuring that flux in it never goes below specified level.

Contrary to this goal, other authors have established that the maximal growth objective is valid only in the event of limited nutrient supply, while in the contrary case the cell aims to function on the Pareto surface of optimal ATP and biomass production [155], as well as 1-norm of the metabolic flux vector. In [107, 161], using elementary flux modes and the assumption that the cell will function at the state of maximal entropy production, authors demonstrated both experimentally and theoretically that the cell will function with the weight of elementary modes proportional to the logarithm of the elementary mode reaction entropy.

8.3 Enumerating efficient elementary modes using L1-regularized quadratic programming

An end user may be interested in being able to compute as many efficient elementary flux modes as possible, where efficiency of a mode is defined in terms of its high target chemical reaction yield. Rather than running the Nullspace Algorithm on the original network, if it were possible to infer the subnetwork which contains as many efficient modes as possible it might significantly lower the computational cost.

Lets assume that the metabolic network is given with its stoichiometry matrix $N_{m\times q}$ and that the consumption of substrate reaction v_{subst} is fixed to unit value. Using flux balance analysis, let the flux vector $v^{(wt)}$ correspond the the wild-type network with maximum possible biomass for fixed substrate uptake. Further, let the goal be that as many efficient target chemical producing modes should be computed which we request by imposing an inequality constraint $v_{chem} \geq \alpha \cdot v_{chem}^{(wt,max)}$ for some specified threshold parameter α. From this point, the greatest challenge is to propose an objective function which mimics the desired cellular behavior as closely as possible.

Assume that an objective is proposed incorporating the terms which model the (1) "kinetic energy" of the metabolic network and (2) number of active reactions. Given the flux vector v, the objective function may be represented as $\sum_{i=1}^{q} v_i^2 + \lambda\|v\|_1$, i.e. in a L1-regularized quadratic programming form. With the properly chosen value for λ this function aims to minimize the spread of the flux across the network while at the same time minimizing the number of active reactions. In addition, we may want to account for the wild-type flux distribution vector $v^{(wt)}$ which models the cell with maximum biomass production. The quadratic term in the objective function $\sum_{i=1}^{q} v_i^2$ may be substituted for $\sum_{i=1}^{q}(v_i - v_i^{(wt)})^2$ to model the earlier evolutionary assumption adopted in the MOMA framework (subsection 2.3.2.2). Main assumption in this approach is that the solution of the proposed L1-regularized quadratic program should be indicative of the possible role of reactions according to their flux values and zero/non-zero pattern.

The proposed non-linear program is illustrated in equation (8.1) as:

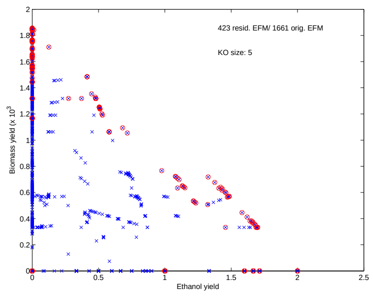
$$\begin{aligned} \min_{v} \quad & \sum_{i=1}^{q} (v_i - w_i)^2 + \lambda \|v\|_1 \\ \text{subject to} \quad & N \cdot v = 0, \\ & v_{chem} \geq \alpha \cdot v_{chem}^{(wt,max)} \\ & v_{chem} \geq \beta \cdot v_{bio}^{(wt)} \\ & v_{subst} = 1 \end{aligned} \tag{8.1}$$

Further, a procedure to enumerate as many efficient modes as possible is illustrated in the following algorithmic steps:

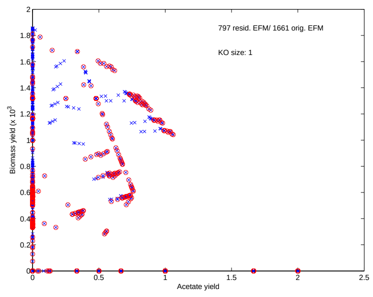
1. For the given metabolic network, stoichiometric matrix and the minimal flux value of the target chemical solve the non-linear problem in (8.1)

2. From the original network remove those reactions which have zero or near-zero flux in the solution flux vector obtained in the previous step resulting in the residual network N_R.

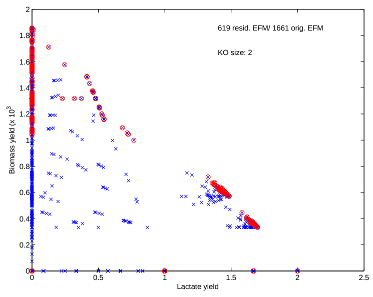
3. Compute elementary modes EFM_{N_R} for the residual network N_R. Extract desired efficient modes for further exploration.



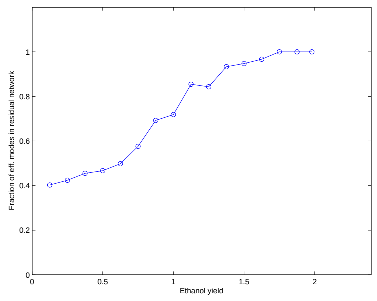
(a) Ethanol



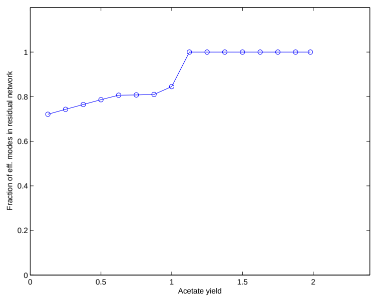
(b) Acetate



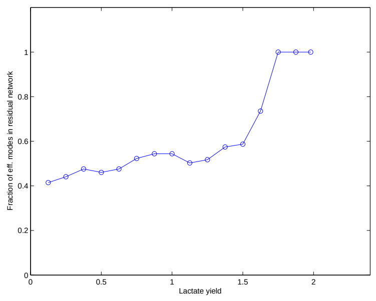
(c) Lactate



(d) Ethanol



(e) Acetate

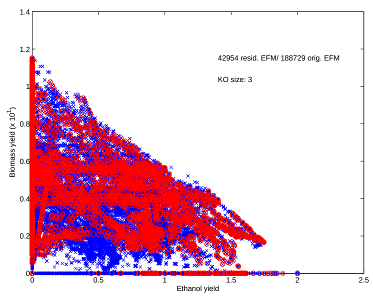


(f) Lactate

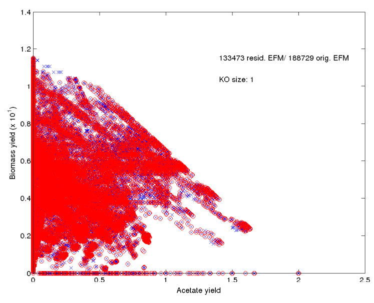
Figure 8.1: *Computed elementary modes in residual networks for growth-coupled production of ethanol (8.1(a),8.1(d)), acetate (8.1(b),8.1(e)) and lactate (8.1(c),8.1(f)) in a 44-reaction S. cerevisiae metabolic network.*

8.3.1 Results

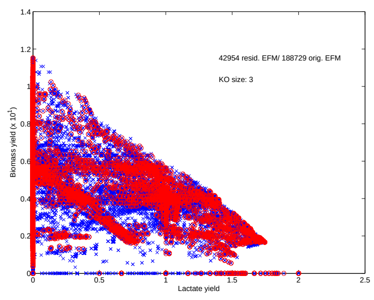
Using the 44-reaction metabolic network of the recombinant *S. cerevisiae* earlier used in [162] and given in Appendix A, Table A.4, the procedure outlined in the previous subsection is executed for different target chemicals such as ethanol, acetate and lactate.



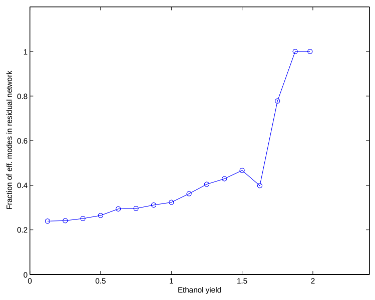
(a) Ethanol



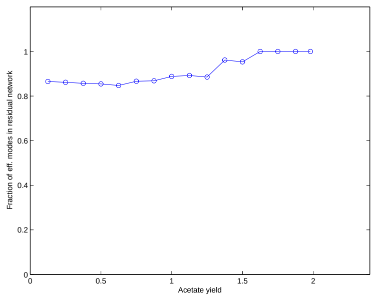
(b) Acetate



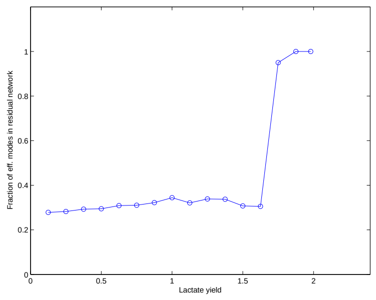
(c) Lactate



(d) Ethanol



(e) Acetate



(f) Lactate

Figure 8.2: *Computed elementary modes in residual networks for growth-coupled production of ethanol (8.2(a),8.2(d)), acetate (8.2(b),8.2(e)) and lactate (8.2(c),8.2(f)) in a 66-reaction E. coli metabolic network.*

The results of the run are shown in Figure 8.1. The union red dots correspond to the elementary flux modes in the residual network, while the union of blue and red dots comprises all of the elementary flux modes of the original metabolic netwokr prior to knocking out the reactions which were inferred as targets for deletion using the framework described in (8.1).

Similarly, Figure 8.2 shows the result of the run of the same framework on the 66-reaction E. coli network which uses three different substrates of glucose, xylose and glycerol (Appendix A, Table A.1) and supports the production of ethanol, acetate or lactate.

8.4 Combined optimization-based and EFM-based enumeration of multiple knockout subsets

The L1-regularized quadratic program formulation may be used to infer the few most likely candidate reactions for knockout and continue on to execution of one of the algorithms for the enumeration of multiple reaction knockout subsets using algorithms proposed in Chapter 7. This idea may be illustrated in the following steps:

Algorithm 15 Combined optimization- and EFM-based finding of reaction knockout subset

```
Input:
 1: stoichiometry matrix - $N_{m\times q}$
 2: substrates, biomass and target chemical reactions - $subst$, $bio$ and $chem$
Output:
 3: reaction knockout set - $knockoutSet$
 4: maximal size of knockout set - $maxSizeKO$
 5: $availReacs \Leftarrow$ list of reactions available for knockout
 6: Let $v^{(wt)}$ be the flux vector in the wild-type cell in the conditions of maximal biomass
    production.
 7: Initialize $knockoutSet \leftarrow \emptyset$
 8: while length($knockoutSet$) $< sizeKO^{(opt)}$ do
 9:     Let objFunc($KO$) for some knockout set $KO$ be a minimal value of objective func-
        tion optimized in:
```

$$\begin{aligned} \min_{v} \quad & \sum_{i=1}^{q} v_i^2 + \lambda \|v\|_1 \\ \text{subject to} \quad & N \cdot v = 0, \\ & v_{chem} \geq \alpha \cdot v_{chem}^{max} \\ & v_{chem} \geq \beta \cdot v_{bio}^{(wt,max)} \\ & v_{subst} = 1 \\ & v_j = 0 \; for \; j \in KO \end{aligned} \tag{8.2}$$

```
10:     Find reaction $r \in availReacs$ such that objFunc($knockoutSet$)-
        objFunc($knockoutSet \cup r$) is maximal.
11:     Expand the current knockout subset as: $knockoutSet \leftarrow knockoutSet \cup \{r\}$ and
        $availReacs \leftarrow availReacs \setminus \{r\}$.
12: end while
13: From original network remove reactions in $knockoutSet$ and compute all elementary
    flux modes.
14: Run one of the earlier proposed algorithms in Chapter 7 for enumeration of multiple
    reaction knockout subsets of length not larger than $sizeKO^{(efm)} = maxSizeKO -$
    $sizeKO^{(opt)}$.
15: Append $knockoutSet$ to each subset obtained in the previous step and return all of
    them as result.
```

8.5 Finding of optimal knockout subset using non-linear optimization

Rather than resorting to computing any of the elementary flux modes, one may decide to solely rely on using a non-linear objective function and solve it iteratively to enumerate an optimal reaction knockout subset whose collapse will assure optimal growth-coupled target chemical production. As illustrated in subsections 2.3.2.2, 2.3.2.8 and 2.3.2.14, non-linear objectives were used in MOMA [56] and BiMOMA [74] (e.g. MOMAKnock in [75]) in the form of a quadratic program, and in OptGene's biomass-product coupled yield. The novelty here is the inclusion of the L1 regularization term which accounts for the minimization of the kind proposed in [155].

Algorithm 16 Iterative optimization-based finding of reaction knockout subset

```
Input:
 1: stoichiometry matrix - $N_{m \times q}$
 2: substrates, biomass and target chemical reactions - subst, bio and chem
Output:
 3: reaction knockout set - knockoutSet
 4: maximal size of knockout set - maxSizeKO
 5: availReacs ⇐ list of reactions available for knockout
 6: Let $v^{(wt)}$ be the flux vector in the wild-type cell in the conditions of maximal biomass
    production.
 7: Initialize knockoutSet ← ∅
 8: while length(knockoutSet) < maxSizeKO do
 9:     Let objFunc(KO) for some knockout set KO be defined as in Algorithm 15, line 9
10:     Find reaction $r \in$ availReacs such that objFunc(knockoutSet)-
    objFunc(knockoutSet ∪ r) is maximal.
11:     Expand the current knockout subset as: knockoutSet ← knockoutSet ∪ {r} and
    availReacs ← availReacs{r}.
12: end while
13: return knockoutSet
```

8.6 Conclusion

There is still insufficient intuition surrounding the problem of using the non-linear optimization methods to model the cellular phenotype as proposed in this chapter. Nevertheless, this chapter may serve as a base for future research work and exploration. One may need to account for the more comprehensive domain knowledge in biophysics, biochemistry and computational learning.

Chapter 9

Concluding Remarks

9.1 Summary of contributions

The preceding chapters of this thesis treat the problem of computational analysis of the steady state cellular metabolic networks from the aspect of metabolic pathways. Elementary flux modes, the metabolic pathways which also meet the requirement of the enzymatic minimality, are used as major constituents for the study of the cellar phenotype. The software implementations of the algorithms presented in the thesis and the accompanying data sets are available at `http://elmocomp.sourceforge.net`. Major contributions of this thesis are listed as:

Theoretical development and improvement of the Nullspace Algorithm

Chapter 3 brings several theoretical treatments and considerations of the Nullspace Algorithm used in the computation of elementary flux modes. First, the Nullspace Algorithm is decomposed into subroutines and analyzed for its later use in parallel computing of both elementary flux modes and the minimal generating set. Second, the reduced algebraic rank test is proposed which brings cost reductions in the Nullspace Algrotihms when run on the metabolic networks without previous splitting of reversible reactions. The said rank test is run against the submatrix smaller than the one previously used for every candidate elementary flux mode during the course of executing Nullspace Algorithm. Third, a simple method for the computation of the minimal generating set in the metabolic networks which admit the reversible pathway is given. The minimal generating set is a minimal subset of elementary flux modes which describes the polyhedral cone corresponding to the solution space of the stoichiometric problem.

Parallelization of the Nullspace Algorithm

Combinatorial Parallel Nullspace Algorithm is the first parallelization of the Nullspace

Algorithm targeted for the distributed-memory system, as illustrated in capter 4. It uses the MPI library of communication routines to perform task partitioning, while still exhibits insufficient memory scalability due to replicating major data structure across all compute nodes. To assure load balancing, an interleaved parallel generation of candidate elementary flux modes is performed across the so-called positive and negative candidate columns.

As a second step in the effort to efficient parallel run of the Nullspace Algorithm and improvement of the memory scalability the divide-and-conquer idea was incorporated. chapter 5 incorporates the Combinatorial Nullspace Algorithm with the divide-and-conquer approach across a selected subset of partitioning reactions. This has a potential to reduce the cumulative memory footprint and computation time, and therefore fit larger metabolic networks into the Nullspace Algorithm. However, as major challenge remains how to optimally choose the set of partitioning reactions in order to assure that the amount of memory used and execution time are minimal.

In order to improve the memory scalability within the Combinatorial Parallel Nullspace Algorith, chapter 6 describes the use of the Global Arrays library of partitioned global address paradigm which provides the shared-memory view of the distributed-memory data and significantly reduces the time required for the development and implementation of the parallel algorithms. Using the Global Arrays the data structures which were previously replicated across all compute nodes were now instead partitioned to attain improved memory scalability. The Global Arrays library runs on the top of the MPI library and uses its routines for one-sided asynchronous communication underneath.

Design of algorithms for efficient cellular design using elementary flux modes

In chapter 7, elementary flux modes are used in the enumeration of multiple reaction knockout subsets which can collapse the cellular metabolic network to the subnetwork which attains desired metabolic goals of the elevated production of the target chemical coupled with the cellular biomass growth, assuring either low or high network flexibility. Proposed algorithms use the quantitative score which models the cellular design goals and perform the branch-and-bound search. The proposed algorithms are compared to the existing methods, particularly to the variation of the Berge's algorithm used for the enumeration of constrained minimal cut sets.

Analysis of metabolic networks using non-linear optimization methods

Finally, chapter 8 proposes a collection of algorithms based on L1-regularized non-linear optimization (e.g. quadratic programming) which can be used to reduce the original metabolic network to the efficent subnetwork. Such reduced subnetwork can then be used to (1) compute efficient elementary flux modes and (2) infer remaining reaction knockout targets with or without computing elementary flux modes. In the first case, the optimization framework is used to infer which reactions are less likely to be active in the efficient elementary flux modes, where efficiency is defined by high target chemical reaction yield. This allows the computation of the large fraction of the elementary flux modes which may later be used in the cellular design. Second case may allow the run of some of the mixed-inteera linear programming frameworks described in chapter 2 or the algorithms for enumeration of multiple reaction knockout subsets given in chapter 7. Major goal of this chapter was to model the cellular phenotype using a non-linear objective function with existing linear constraints accounting for known cellular constraints and goals.

9.2 Future research directions

Considering the previous body of research and undertaken work, future actions may consist of several directions. Some of the major ideas may be outlined as:

Improvement of the Global Arrays based Nullspace algorithm

While the Global Arrays parallelization of the Nullspace algorithm allowed the computation of the network with nearly 70 million elementary modes with demonstrated improved performance, it still suffers from deteriorating scalability. This will require the design of strategy for more comprehensive profiling, especially at points where collective communication calls are performed. Another point of improvement will be to add the feature for out-of-core computation by means of being able to efficiently store and retrieve the data on the disk. Finally, we experienced imprecision issues during the evaluation of the matrix rank on larger metabolic networks where SVD decomposition reported accurate value, while the LU decomposition failed. While SVD may be more accurate, its run exhibits higher computational cost and one may look into a strateg for combined use of LU and SVD matrix decomposition routines.

Divide-and-conquer strategies in elementary mode computation

Continuing on the implementation of the Combined Parallel Nullspace Algorithm, one may look at deciding how to optimally select a subset of reactions which will assure good time and memory load balancing. It is not clear yet how this may be attained, but one approach may be to try to estimate the number of modes in which a given reaction may be active.

Characterization of elementary flux modes

Elementary flux modes are still largely indistinguishable and their categorization and characterization into groups by relative importance and contribution to the phenotype may be an interesting area for further research. This may help to better understand which elementary modes should be retained in the final cellular design and which should be collapsed.

Modeling of the cellular behavior using optimization methods

Understanding the cellular behavior across different growth and environment conditions may lead to proposing a more comprehensive cellular objective function which would model the phenotype more accurately. The methods proposed in the thesis and which use non-linear optimization require considerable improvement and experimentation.

Application of algorithm for enumeration of vertices in degenerate polytope

Algorithm for the enumeration of vertices in the degenerate polytope found its application in bioinformatics. One may still continue looking for the application of this algorithm in other problem domains such as computer networks, social network analysis, etc.

References

[1] C.T. Trinh, P. Unrean, and F. Srienc. A minimal Escherichia coli cell for most efficient ethanol production from hexoses and pentoses. *Applied and Environmental Microbiology*, 74(12):3634–3643, 2008.

[2] B.Ø. Palsson. *Systems Biology : Properties of Reconstructed Networks.* Cambridge University Press, 2006.

[3] J. Schellenberger, R. Que, R.M.T. Fleming, I. Thiele, J.D. Orth, A.M. Feist, D.C. Zielinski, A. Bordbar, N.E. Lewis, S. Rahmanian, J. Kang, D.R. Hyduke, and B.O. Palsson. Quantitative prediction of cellular metabolism with constraint-based models: the cobra toolbox v2.0. *Nature Protocols*, 6(12901307), 2011.

[4] C.T. Trinh, A. Wlaschin, and F. Srienc. Elementary mode analysis: a useful metabolic pathway analysis tool for characterizing cellular metabolism. *Appl. Microbiol. Biotechnol*, 22(5):813–826, 2009.

[5] E. Klipp, W. Liebermeister, C. Wierling, A. Kowald, H. Lehrach, and R. Herwig. *Systems Biology.* John Wiley and Sons, first edition, 2009.

[6] L. Barabasi and Z.N. Oltvai. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[7] M. Buchanan, G. Caldarelli, P. De Los Rios, F. Rao, and M. Vendruscolo. *Networks in Cell Biology.* Cambridge University Press, first edition, 2010.

[8] D.R. Hyduke and B.Ø. Palsson. Towards genome-scale signalling-network reconstructions. *Nature Reviews Genetics*, 11:297–307, 2010.

[9] M. Papini, M. Salazar, and J. Nielsen. Systems biology of industrial microorganisms. *Journal of Biomedicine and Biotechnology*, 2010, 2010. Article ID 518743.

[10] C. Dellomonaco, F. Fava, and R. Gonzalez. The path to next generation biofuels: successes and challenges in the era of synthetic biology. *Microbial Cell Factories*, 9(3), 2010.

[11] T.M. Mata, A.A. Martins, and N.S. Caetano. Microalgae for biodiesel production and other applications: A review. *Renewable and Sustainable Energy Reviews*, 14:217–232, 2010.

[12] C.H. Schilling, S. Schuster, B.Ø. Palsson, and R. Heinrich. Metabolic pathway analysis: Basic concepts and scientific applications in the post-genomic era. *Biotechnol. Prog.*, 15:296–303, 1999.

[13] V. Lacroix, L. Cottret, P. Thebault, and M. Sagot. An introduction to metabolic network analysis and their structural analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(4):594–617, 2008.

[14] L.F. de Figueiredo, A. Podhorski, A. Rubio, C. Kaleta, J.E. Beasley, S. Schuster, and F.J. Planes. Computing the shortest elementary flux modes in genome-scale metabolic networks. *Bioinformatics*, 25(23):3158–3165, 2009.

[15] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28:27–30, 2012.

[16] M. Kanehisa, S. Goto, Y. Sato, M. Furumichi, and M. Tanabe. KEGG for integration and interpretation of large-scale molecular datasets. *Nucleic Acids Research*, 40:D109–D114, 2012.

[17] I.M. Keseler et al. EcoCyc: a comprehensive database of Escherichia coli biology. *Nucleic Acids Research*, 39:D583–590, 2011.

[18] R. Caspi, H. Foerster, C. A. Fulcher, R. Hopkinson, H. Ingraham, P. Kaipa, M. Krummenacker, S. Paley, J. Pick, S. Y. Rhee, C. Tissier, P. Zhang, and P. D. Karp. MetaCyc: a multiorganism database of metabolic pathways and enzymes. *Nucleic Acids Research*, 34:511–516, 2006.

[19] R. Caspi et al. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Research*, 36:623–631, 2008.

[20] M.L. Green D. Kaiser M. Krummenacker P. Romero, J. Wagg and P.D. Karp. Computational prediction of human metabolic pathways from the complete human genome. *Genome Biology*, 6(1), 2004.

[21] M. Covert, C.H. Schilling, I. Famili, J.S. Edwards, I.I. Goryanin, E. Selkov, and B.Ø. Palsson. Metabolic modeling of microbial strains in sillico. *Trends in Biochemical Sciences*, 26(3):179–186, 2001.

[22] J. Schellenberger, J.O. Park, T.M. Conrad, and B.Ø. Palsson. BiGG: a biochemical genetic and genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinformatics*, 11(213), 2010.

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

[24] A.M. Feist, M.J. Herrgård, I. Thiele, J.L. Reed, and B.Ø. Palsson. Reconstruction of biochemical networks in microorganisms. *Nature Reviews Microbiology*, 7:129–143, 2009.

[25] J. Förster, I. Famili, P. Fu, and B Palsson. Genome-scale reconstruction of the Saccharomyces cerevisiae metabolic network. *Genome Research*, 13:644–653, 2003.

[26] N.C. Duarte, M.J. Herrgård, and B.Ø. Palsson. Reconstruction and validation of Saccharomyces cerevisiae iND750, a fully compartmentalized genomescale metabolic model. *Genome Research*, 14(7):1298–309, 2004.

[27] C. H. Schilling and B.Ø. Palsson. Assessment of the Metabolic Capabilities of Haemophilus Influenzae Rd through a Genome-scale Pathway Analysis. *Journal of Theoretical Biology*, 203(3):249–283, 2000.

[28] C.H. Schilling, M.W. Covert, I. Famili, G.M. Church, J.S. Edwards, and B.Ø. Palsson. Genome-scale metabolic model of helicobacter pylori 26695. *Journal of Bacteriology*, 184(16):4582–4593, 2002.

[29] I. Thiele, T.D. Vo, N.D. Price, and B.Ø. Palsson. Expanded metabolic reconstruction of helicobacter pylori (iit341 gsm/gpr): an in silico genome-scale characterization of single- and double-deletion mutants. *Journal of Bacteriology*, 187(16):5818–5830, 2005.

[30] P. Suthers, M. Dasika, V. Kumar, G. Denisov, J. Glass, and C. Maranas. A genome-scale metabolic reconstruction of Mycoplasma genitalium iPS189. *PLoS Computational Biology*, 5(2), 2009.

[31] S.A. Becker and B.Ø. Palsson. Genome-scale reconstruction of the metabolic network in staphylococcus aureus n315: an initial draft to the two-dimensional annotation. *BMC Microbiology*, 5, 2005.

[32] N. C. Duarte, S. A. Becker, N. Jamshidi, I. Thiele, M. L. Mo, T. D. Vo, R. Srivas, and B. Ø. Palsson. Global reconstruction of the human metabolic network based on genomic and bibliomic data. *Proceedings of the National Academy of Sciences of the USA*, 104(6):1777–1782, 2007.

[33] I. Thiele et al. A community-driven global reconstruction of human metabolism. *Nature Biotechnology*, 31(5):419425, 2013.

[34] B.J. Bornstein, S.M. Keating, A. Jouraku, and M. Hucka. Libsbml: An api library for sbml. *Bioinformatics*, 24(6):880–881, 2008.

[35] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, and the rest of the SBML Forum. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network model. *Bioinformatics*, 19(4):524–531, 2003.

[36] B. Clarke. Stoichiometric network analysis. *Cell Biophysics*, 12:237–53, 1988.

[37] M.W. Covert, I. Famili, and B. Palsson. Identifying constraints that govern cell behavior: A key to converting conceptual to computational models in biology? *Biotechnology and Bioengineering*, 84(7):309–325, 2003.

[38] J.M. Park, T.Y. Kim, and S.Y. Lee. Constraints-based genome-scale metabolic simulation for systems metabolic engineering. *Biotechnology Advances*, 27:979–988, 2009.

[39] D. G. Luenberger. *Linear and nonlinear programming*. Springer, second edition, 2003.

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

[41] M. Terzer and J. Stelling. Large Scale computation of elementary flux modes with bit pattern trees. *Bioinformatics*, 2008.

[42] A.M. Feist and B.Ø. Palsson. The biomass objective function. *Current Opinion in Microbiology*, 13:344–349, 2010.

[43] A. R. Zomorrodi, P. F. Suthers, S. Ranganathan, and C. D. Maranas. Mathematical optimization applications in metabolic networks. *Metabolic Engineering*, 12:672–686, 2012.

[44] A. Varma and B.Ø. Palsson. Metabolic Flux Balancing: Basic Concepts, Scientific and Practical Use. *Nature Biotechnology*, 12:994–998, 1994.

[45] C.H. Schilling, J. S. Edwards, and B.Ø. Palsson. Towards metabolic phenomics: Analysis of genomic data using flux balances. *Biotechnol. Prog.*, 15(3):288–295, 1999.

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

[47] J.M. Lee, E.P. Gianchandani, and J.A. Papin. Flux balance analysis in the era of metabolomics. *Briefings in Bioinformatics*, 7(2):140–150, 2006.

[48] K. Raman and N. Chandra. Flux balance analysis of biological systems: applications and challenges. *Briefings in Bioinformatics*, 10(4):435–449, 2009.

[49] J.D. Orth, I. Thiele, and B.Ø. Palsson. What is flux balance analysis? *Nature Biotechnology*, 28(3):245248, 2010.

[50] M.W. Covert, C.H. Schilling, and B.Ø. Palsson. Regulation of gene expression in flux balance models of metabolism. *J. theor. Biol.*, 213:73–88, 2001.

[51] M. Covert and B. Palsson. Transcriptional regulation in constraints-based metabolic models of escherichia coli. *The Journal of Biological Chemistry*, 27:28058–28064, 2002.

[52] M. Covert and B. Palsson. Constraints-based models: Regulation of gene expression reduces the steady-state solution space. *The Journal of Biological Chemistry*, (221):309–325, 2003.

[53] T. Shlomi, Y. Eisenberg, R. Sharan, and E. Ruppin. A genome-scale computational study of the interplay between transcriptional regulation and metabolism. *Molecular Systems Biology*, 3(101), 2007.

[54] S. Chandrasekarana and N. D. Price. Probabilistic integrative modeling of genome-scale metabolic and regulatory networks in Escherichia coli and Mycobacterium tuberculosis. *Proceedings of the National Academy of Sciences of the USA*, 107(41):17845–17850, 2010.

[55] T. Benyamini, O. Folger, E. Ruppin, and T. Shlomi. Flux balance analysis accounting for metabolite dilution. *Genome Biology*, 11(4), 2010.

[56] D. Segrè, D. Vitkup, and G.M. Church*. Analysis of optimality in natural and perturbed metabolic networks. *Proceedings of the National Academy of Sciences USA*, 99(29):15112–15117, 2002.

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

[58] R. Mahadevan and C.H. Schilling. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metabolic Engineering*, 5(4):264–276, 2003.

[59] R.U. Ibarra, J.S. Edwards, and B.O. Palsson. Escherichia coli k-12 undergoes adaptive evolution to achieve in silico predicted optimal growth. *Nature*, 420:186–189, 2002.

[60] A.P. Burgard, P. Pharkya, and C.D. Maranas. OptKnock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and Bioengineering*, 84(6):647–657, 2003.

[61] P. Pharkya, A.P. Burgard, and C.D. Maranas. Exploring the overproduction of amino acids using the bilevel optimization framework OptKnock. *Biotechnology and Bioengineering*, 84(7):887–99, 2003.

[62] N. Tepper and T. Shlomi. Predicting metabolic engineering knockout strategies for chemical production: accounting for competing pathways. *Bioinformatics*, 26(4):536–543, 2009.

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

[64] P. Pharkya and C.D. Maranas. An optimization framework for identifying reaction activation/inhibition or elimination candidates for overproduction in microbial systems. *Metabolic Engineering*, 6(8):1–13, 2006.

[65] K.R. Patil, I. Rocha, J. Forster, and J. Nielsen. Evolutionary programming as a platform for in silico metabolic engineering. *BMC Bioinformatics*, 6(1):308, 2005.

[66] M.Rocha, P. Maia, R. Mendes, J.P. Pinto, E.C. Ferreira, J. Nielsen, K.R. Patil, and I. Rocha. Natural computation meta-heuristics for the in silico optimization of microbial strains. *BMC Bioinformatics*, 9, 2008.

[67] D.S. Lun, G. Rockwell, N.J. Guido, M. Baym, J.A. Kelner, B. Berger, J.E. Galagan, and G.M. Church. Large-scale identification of genetic design strategies using local search. *Molecular Systems Biology*, 5(296), 2009.

[68] H.S. Choi, S.Y. Lee, T.Y. Kim, and H.M. Woo. In Silico Identification of Gene Amplification Targets for Improvement of Lycopene Production. *Applied and Environmental Microbiology*, 76(10):3097–3105, 2010.

[69] J. Kim and J. Reed. OptORF: Optimal metabolic and regulatory perturbations for metabolic engineering of microbial strains. *BMC Systems Biology*, 4(53):53–71, 2010.

[70] J.L. Reed, T.D. Vo, C.H. Schilling, and B.Ø. Palsson. An expanded genome-scale model of Escherichia coli K-12 (iJR904 GSM/GPR). *Genome Biology*, 4(9):435–443, 2003.

[71] M.W. Covert, E.M. Knight, J.L. Reed, M.J. Herrgard, and B. Palsson. Integrating high-throughput and computational data elucidates bacterial. *Nature*, (429):92–96, 2004.

[72] S. Ranganathan, P.F. Suthers, and C.D. Maranas. Optforce: An optimization procedure for identifying all genetic manipulations leading to targeted overproductions. *PLoS Comput Biol*, 6(4), 2010.

[73] P. Suthers, A. Zomorrodi, and C. Maranas. Genome-scale gene/reaction essentiality and synthetic lethality analysis. *Molecular Systems Biology*, 5(301), 2009.

[74] J. Kim, J. Reed, and Christos T. Maravelias. Large-Scale Bi-Level Strain Design Approaches and Mixed-Integer Programming Solution Techniques. *PLoS ONE*, 6(9), 2011.

[75] S. Ren, B. Zeng, and X. Qian. Adaptive bi-level programming for optimal gene knockouts for targeted overproduction under phenotypic constraints. *Proceedings of Eleventh Asia Pacific Bioinformatics Conference (APBC 2013): Bioinformatics*, 14(Suppl. 2), 2013.

[76] K. Fukuda and A. Prodon. Double description method revisited. In M. Deza, R. Euler, and I. Manoussakis, editors, *Combinatorics and Computer Science*, pages 91–111. Springer, 1996. Also tech. report, Mathematics, ETH, 1995.

[77] S. Schuster, C. Hilgetag, J.H. Woods, and D.A. Fell. Reaction routes in biochemical reaction systems: Algebraic properties, validated calculation procedure and example from nucleotide metabolism. *Mathematical Biology*, 45(2):153–181, 2002.

[78] C. Wagner. Nullspace approach to determine the elementary modes of chemical reaction systems. *J. Phys. Chem.*, 108(7):2425–2431, 2004.

[79] R. Urbanczik and C. Wagner. An improved algorithm for stoichiometric network analysis: theory and applications. *Bioinformatics*, 21(7):1203–1210, 2005.

[80] C. Wagner and R. Urbanczik. The geometry of the flux cone of a metabolic network. *Biophysics Journal*, 89(6):3837–3845, 2005.

[81] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich. Generating all vertices of a polyhedron is hard. *Discrete Comput Geom*, 39:174–190, 2008.

[82] V. Acuña, A. Marchetti-Spaccamela, M. Sagot, and L. Stougie. A note on the complexity of finding and enumerating elementary modes. *Biosystems*, 99(3):210–214, 2010.

[83] S. Klamt and J. Stelling. Combinatorial complexity of pathway analysis in metabolic networks. *Molecular Biology Reports*, 29(1-2):233–236, 2002.

[84] M. Yeung, Ines Thiele, and B.Ø. Palsson. Estimation of the number of extreme pathways for metabolic networks. *BMC Bioinformatics*, 8(363), 2007.

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

[86] S. Wiback and B.Ø. Palsson. Extreme pathway analysis of human red blood cell metabolism. *Biophysics Journal*, 83(2):808–818, 2002.

[87] J.A. Papin, N.D. Price, J.S. Edwards, and B.Ø. Palsson. The Genome-Scale Metabolic Extreme Pathway Structure in Haemophilus influenzae Shows Significant Network Redundancy. *Journal of Theoretical Biology*, 215:67–82, 2002.

[88] N.D. Price, J.A. Papin, and B.Ø. Palsson. Determination of redundancy and systems properties of the metabolic network of helicobacter pylori using genome-scale extreme pathway analysis. *Genome Research*, 12:760–769, 2002.

[89] S. Schuster and C. Hilgetag. On elementary flux modes in biochemical reaction systems at steady state. *Journal of Biological Systems*, 2(2):165–182, 1994.

[90] T. Pfeiffer, I. Sanchez-Valdenebro, J.C. Nuno, F. Montero, and S. Schuster. META-TOOL: for studying metabolic networks. *Bioinformatics*, 15(3):251–257, 1999.

[91] S. Klamt and J. Stelling. Two approaches for metabolic pathway analysis? *Trends in Biotechnology*, 21(2):64–69, 2003.

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

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

[94] U. Haus, S. Klamt, and T. Stephen. Computing knock-out strategies in metabolic networks. *Journal of Computational Biology*, 15(3):259–268, 2008.

[95] K. Ballerstein, A. von Kamp, S. Klamt, and U. Haus. Minimal cut sets in a metabolic network are elementary modes in a dual network. *Bioinformatics*, 28(3):381–387, 2012.

[96] C. Kaleta, L. F. de Figueiredo, and S. Schuster. Can the whole be less than the sum of its parts? pathway analysis in genome-scale metabolic networks using elementary flux patterns. *Genome Research*, 19:1872–1883, 2009.

[97] J. Cakir, B. Kirdar, Z.I. Onsan, K.O. Ulgen, and J. Nielsen. Effect of carbon source perturbations on transcriptional regulation of metabolic fluxes in saccharomyces cerevisiae. *BMC Systems Biology*, 1(18), 2007.

[98] J. Stelling, S. Klamt, K. Bettenbrock, S. Schuster, and E. D. Gilles. Metabolic network structure determines key aspects of functionality and regulation. *Nature*, 420(6912):190–193, 2002.

[99] Q.Zhao and H.Kurata. Genetic modification of flux for flux prediction of mutants. *Bioinformatics*, 25(13):1702–1708, 2009.

[100] M.A. Orman, F. Berthiaume, I.P. Androulakis, and M.G. Ierapetritou. Pathway analysis of liver metabolism under stressed condition. *Journal of Theoretical Biology*, 272(1):131–140, 2011.

[101] S.J. Wiback, R. Mahadevan, and B.Ø. Palsson. Reconstructing metabolic flux vectors form extreme pathways: defining the alpha-spectrum. *Journal of Theoretical Biology*, 224(3):313–324, 2003.

[102] M.G. Poolman, K.V. Venkatesh, M.K. Pidcock, and D.A. Fell. A method for the determination of flux in elementary modes, and its application to lactobacillus rhamnosus. *Biotechnology and Bioengineering*, 88(5):601–612, 2004.

[103] J.M. Schwartz and M. Kanehisa. A quadratic programming approach for decomposing steady-state metabolic flux distributions onto elementary modes. *Bioinformatics*, 21(2):204–205, 2005.

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

[105] Q.Zhao and H. Kurata. Maximum entropy decomposition of flux distribution at steady state to elementary modes. *Journal of Bioscience and Bioengineering*, 107(1):84–89, 2009.

[106] Q. Zhao and H. Kurata. Use of maximum entropy principle with lagrange multipliers extends the feasibility of elementary mode analysis. *Journal of Bioscience and Bioengineering*, 110(2):254–261, 2010.

[107] A.P. Wlaschin, C.T. Trinh, R. Carlsson, and F. Srienc. The fractional contributions of elementary modes to the metabolism of escherichia coli and their estimation from reaction entropies. *Metabolic Engineering*, 8:338–352, 2006.

[108] H. Kurata, Q. Zhao, R. Okuda, and K. Shimizu. Integration of enzyme activities into metabolic flux distributions by elementary mode analysis. *BMC Systems Biology*, 1(31), 2007.

[109] K. Ip, C. Colijn, and D.S. Lun. Analysis of complex metabolic behavior through pathway decomposition. *BMC Systems Biology*, 5(11), 2011.

[110] J. Behre, T. Wilhelm, A. von Kamp, E. Ruppin, and S. Schuster. Structural robustness of metabolic networks with respect to multiple knockouts. *Journal of Theoretical Biology*, 252(3):433–441, 2008.

[111] C.T. Trinh, R. Carlson, A. Wlaschin, and F. Srienc. Design, construction and performance of the most efficient biomass producing e. coli bacterium. *Metabolic Engineering*, 8(6):628–638, 2006.

[112] C.T. Trinh and F. Srienc. Metabolic engineering of escherichia coli for efficient conversion of glycerol to ethanol. *Applied and Environmental Microbiology*, 75(21):6696–6705, 2009.

[113] G. Melzer, M.E. Esfandabadi, E. Franco-Lara, and C. Wittmann. Flux design: In silico design of cell factories based on correlation of pathway fluxes to desired properties. *BMC Systems Biology*, 3:120–135, 2009.

[114] O. Hädicke and S. Klamt. CASOP: A computational approach for strain optimization aiming at high productivity. *Journal of Biotechnology*, 147(2):88–101, 2010.

[115] K. Bohl, L.F. de Figueiredo, O. Hädicke, S. Klamt, C. Kost, S. Schuster, and C. Kaleta. CASOP GS: Computing intervention strategies targeted at production improvement in genome-scale metabolic networks. *Proceedings of the 25th German Conference on Bioinformatics*, pages 71–80, 2010.

[116] B.A. Boghigian, H. Shi, K. Lee, and B.A. Pfeifer. Utilizing elementary mode analysis, pathway thermodynamics, and a genetic algorithm for metabolic flux determination and optimal metabolic network design. *BMC Systems Biology*, 4, 2010.

[117] O. Hädicke and S. Klamt. Computing complex metabolic intervention strategies using constrained minimal cut sets. *Metabolic Engineering*, 13(2):204–213, 2011.

[118] von Kamp, A., and S. Schuster. Metatool 5.0: fast and flexible elementary modes analysis. *Bioinformatics*, 22(15):1930–1931, 2006.

[119] S. Klamt, J. Saez-Rodriguez, and E. D. Gilles. Structural and functional analysis of cellular networks with CellNetAnalyzer. *BMC Systems Biology*, 1(2), 2007.

[120] S. Hoops, S. Sahle R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. COPASI a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074, 2006.

[121] S. Bell and B.Ø. Palsson. Expa: Program for calculating extreme pathways in biochemical reaction networks. *Bioinformatics*, 21(8):1739–1740, 2005.

[122] K. Yizhak, T. Benyamini, W. Liebermeister, and E. Ruppin. Integrating quantitative proteomics and metabolomics with a genome-scale metabolic network model. *Bioinformatics*, 26(12):255–260, 2010.

[123] T. Shlomi, M.N. Cabili, M.J. Herrgard, B.O. Palsson, and E. Ruppin. Network-based prediction of human tissue-specific metabolism. *Nature Biotechnology*, 26(9), 2008.

[124] H. Zur, E. Ruppin, and T. Shlomi. imat: an integrative metabolic analysis tool. *Bioinformatics*, 26(24):3140–3142, 2010.

[125] M.W. Covert, N. Xiao, T.J. Chen, and J.R. Karr. Integrating metabolic, transcriptional regulatory and signal transduction models in escherichia coli. *Bioinfrmatics*, 24(18):2044–2050, 2008.

[126] J. R. Karr, J. C. Sanghvi, D. N. Macklin, M. V. Gutschow, J. M. Jacobs, B. Bolival Jr., N. Assad-Garcia, J. I. Glass, and M. W. Covert. A Whole-Cell Computational Model Predicts Phenotype from Genotype. *Cell*, 150(2):389–401, 2012.

[127] J. Gunawardena. Sillicon dreams of cells into symbols. *Nature Biotechnology*, 30(9):838–840, 2012.

[128] M. Isalan. A cell in a computer. *Nature*, 488:40–41, 2012.

[129] S. Klamt, J. Stelling, M. Ginkel, and E.D. Gilles. FluxAnalyzer: Exploring structure, pathways, and flux distributions in metabolic networks on interactive flux maps. *Bioinformatics*, 19(2):261–269, 2003.

[130] D. Jevremovic, C.T. Trinh, F. Srienc, and D. Boley. On algebraic properties of extreme pathways in metabolic networks. *Journal of Computational Biology*, 17(2):107–119, 2010.

[131] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Computational Geometry*, 8(1):295–313, 1992.

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

[133] M.E. Dyer. The Complexity of Vertex Enumeration Methods. *Mathematics of Operations Research*, 8(3):381–402, 1983.

[134] T. H. Corment, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, 2nd edition, 2001.

[135] A. Schrijver. *Theory of linear and integer programming.* Wiley, 1988.

[136] F. Llaneras and J. Pico. Which metabolic pathways generate and characterize the flux space? a comparison among elementary modes, extreme pathways and minimal generators. *J of Biomedicine and Biotechnology*, 2010, 2010. Article ID 753904.

[137] S. Pérès, F. Vallée, M. Beurton-Aimar, and J.P. Mazat. ACoM: A classication method for elementary ux modes based on motif nding. *Biosystems*, 103:410–419, 2011.

[138] C.M. Flynn, K.A. Hunt, J.A. Gralnick, and F. Srienc. Construction and elementary mode analysis of a metabolic model for Shewanella oneidensis MR-1. *Biosystems*, 107(2):120–128, 2012.

[139] A. Larhlimi. *New Concepts and Tools in Constraint-based Analysis of Metabolic Networks.* PhD thesis, Freie Universität, Berlin, July 2008.

[140] A. Larhlimi and A. Bockmayr. A new approach to flux coupling analysis of metabolic networks. In *Computational Life Sciences II*, volume 4216 of *Lecture Notes in Computer Science*, pages 205–215. Springer Berlin / Heidelberg, 2006.

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

[142] A. Rezola, L.F. de Figueiredo, M. Brock, J. Pey, A. Podhorski, C. Wittmann, S. Schuster, A. Bockmayr, and F. J. Planes. Exploring metabolic pathways in genome-scale networks via generating flux modes. *Bioinformatics*, 27(4):534–540, 2011.

[143] L. F. de Figueiredo, A. Podhorski, A. Rubio, C. Kaleta, J. Beasley, S. Schuster, and F. J. Planes. Computing the shortest elementary flux modes in genome-scale metabolic networks. *Bioinformatics*, 25(23):3158–3165, 2009.

[144] D. Jevremovic, C. T. Trinh, F. Srienc, C. Sosa, and D. Boley. Parallelization of nullspace algorithm for the computation of elementary flux modes. *Parallel Computing Journal*, 37(6-7):261–278, 2011.

[145] David Lay. *Linear Algebra and Its Applications.* Addison Wesley, 4 edition, 2012.

[146] A. Grama, G. Karypis, V. Kumar, and A. Gupta. *Introduction to Parallel Computing.* Addison-Wesley, 2nd edition, 2003.

[147] Template Numerical Toolkit. `http://math.nist.gov/tnt/`.

[148] Java Matrix Library. `http://math.nist.gov/javanumerics/jama`.

[149] D. Jevremovic, D. Boley, and C. Sosa. Divide-and-conquer approach to the parallel computation of elementary flux modes in metabolic networks. In *IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, pages 502–511, 2011.

[150] S. Klamt, J. Gagneur, and A. von Kamp. Algorithmic approaches for computing elementary modes in large biochemical reaction networks. *Systems Biology, IEE Proceedings*, 152(4):249–255, 2005.

[151] J. Nieplocha, B. Palmer, V. Tipparaju, M. Krishnan, H. Trease, and E. Apra. Advances, Applications and Performance of the Global Arrays Shared Memory Programming Toolkit. *International Journal of High Performance Computing Applications*, 20(2):203–231, 2006.

[152] Global arrays webpage. `http://www.emsl.pnl.gov/docs/global/`. Accessed: 03/04/2012.

[153] M. Krishnan, B. Palmer, A. Vishnu, S. Krishnamoorthy, J. Daily, and D. Chavarria. The Global Arrays User Manual, November 2011.

[154] J. Nieplocha and V. Tipparaju and M. Krishnan and D. Panda. High Performance Remote Memory Access Comunications: The ARMCI Approach. *International Journal of High Performance Computing and Applications*, 20(2), 2006.

[155] R. Schuetz, N. Zamboni, M. Zampieri, M. Heinemann, and U. Sauer. Multidimensional Optimality of Microbial Metabolism. *Science*, 336(6081):601–604, 2012.

[156] S.S. Fong, J.Y. Marciniak, and B.O. Palsson. Description and interpretation of adaptive evolution of escherichia coli k-12 mg1655 by using a genome-scale in silico metabolic model. *Journal of Bacteriology*, 185(21):6400–6408, 2003.

[157] S.S. Fong and B.O. Palsson. Metabolic genedeletion strains of escherichia coli evolve to computationally predicted growth phenotypes. *Nature Genetics*, 36:1056–1058, 2004.

[158] S.S. Fong, A.P. Burgard, C.D. Herring, E.M. Knight, F.R. Blattner, C.D. Maranas, and B.O. Palsson. In silico design and adaptive evolution of escherichia coli for production of lactic acid. *Biotechnology and Bioengineering*, 91(5):643648, 2005.

[159] T.M. Conrad, N.E. Lewis, and B.O. Palsson. Microbial laboratory evolution in the era of genome-scale science. *Molecular Systems Biology*, 7:509, 2011.

[160] K. Holmström, A. O. Göran, and M. M. Edvall. *User's Guide for TOMLAB 7*, May 2010.

[161] F. Srienc and P. Unrean. A statistical thermodynamical interpretation of metabolism. *Entropy*, 12(8):1921–1935, 2010.

[162] R. Carlson, D. Fell, and F. Srienc. Metabolic pathway analysis of a recombinant yeast for rational strain development. *Biotechnology and bioengineering*, 79(2):121–134, 2002.

Appendix A

Metabolic Network Models

Metabolites in the metabolic networks listed here have compartmentalized metabolites. For notation, the metabolite name is followed by a bracketed single letter such as GAL[e], denoting the extracellular galactose. Mitochondrial, extracellular and cytosol compartments are denoted as [m], [e] and [c], respectively. Cytosol metabolites are given without the notation in brackets as the default metabolite location.

A.1 *Escherichia coli*

Table A.1: *Central metabolism of Escherichia coli [1]*

		(*reactions which are present when xylose is used as substrate*)
$XYL1$	:	XYLO + ATP $\Longrightarrow$ XYLU + ADP
$XYL2$	:	XYLU $\Longrightarrow$ X5P
		(*reactions which are present when manose is used as substrate*)
$MAN1$	:	MAN[e] + ATP $\Longrightarrow$ MAN6P + ADP
$MAN2$	:	MAN6P $\Longrightarrow$ F6P
		(*reactions which are present when galactose is used as substrate*)
$GAL1$	:	GAL + ATP $\Longrightarrow$ G6P + ADP
$TRA9$	:	GAL[e] + ATP $\Longrightarrow$ GAL + ADP
		(*reactions which are present when glucose is used as substrate*)
$GG1$	:	GLC[e] + PEP $\Longrightarrow$ G6P + PYR
		(*reactions which are present when glycerol is used as substrate*)
$GLYCER1$	:	GLYCEROL[e] + ATP $\Longrightarrow$ GLY3P + ADP
$GLYCER2$	:	GLY3P + NAD $\Longrightarrow$ DHAP + NADH
		(*reactions which are common for all substrates*)
$GG2_r$	:	G6P $\Longleftrightarrow$ F6P
$GG3$	:	F6P + ATP $\Longrightarrow$ F16BP + ADP

$GG4$	:	F16BP $\Longrightarrow$ F6P
$GG5_r$	:	F16BP $\Longleftrightarrow$ DHAP + GA3P
$GG6_r$	:	GA3P $\Longleftrightarrow$ DHAP
$GG7_r$	:	GA3P + NAD $\Longleftrightarrow$ 3PGP + NADH
$GG8_r$	:	3PGP + ADP $\Longleftrightarrow$ 3PG + ATP
$GG9_r$	:	3PG $\Longleftrightarrow$ 2PG
$GG10_r$	:	2PG $\Longleftrightarrow$ PEP
$GG11$	:	PEP + ADP $\Longrightarrow$ PYR + ATP
$GG12$	:	PYR + ATP $\Longrightarrow$ PEP + AMP
$GG13$	:	PYR + CoASH + NAD $\Longrightarrow$ ACoA + CO2 + NADH
$PPP1$	:	G6P + NADP $\Longrightarrow$ GL6P + NADPH
$PPP2$	:	GL6P $\Longrightarrow$ 6PG
$PPP3$	:	6PG + NADP $\Longrightarrow$ R5P + CO2 + NADPH
$PPP4_r$	:	R5P $\Longleftrightarrow$ X5P
$PPP5_r$	:	R5P $\Longleftrightarrow$ RIBO5P
$PPP6_r$	:	RIBO5P + X5P $\Longleftrightarrow$ S7P + GA3P
$PPP7_r$	:	GA3P + S7P $\Longleftrightarrow$ ERY4P + F6P
$PPP8_r$	:	ERY4P + X5P $\Longleftrightarrow$ GA3P + F6P
$TCA1$	:	OAA + ACoA $\Longrightarrow$ CIT + CoASH
$TCA2_r$	:	CIT $\Longleftrightarrow$ CACO
$TCA3_r$	:	CACO $\Longleftrightarrow$ ICIT
$TCA4$	:	ICIT + NADP $\Longrightarrow$ AKG + CO2 + NADPH
$TCA5$	:	AKG + NAD + CoASH $\Longrightarrow$ NADH + SCoA + CO2
$TCA6_r$	:	SCoA + ADP $\Longleftrightarrow$ SUCC + ATP + CoASH
$TCA7$	:	SUCC + Q $\Longrightarrow$ FUM + QH2
$TCA8_r$	:	FUM $\Longleftrightarrow$ MAL
$TCA9_r$	:	MAL + NAD $\Longleftrightarrow$ OAA + NADH
$TCA10$	:	FUM + QH2 $\Longrightarrow$ SUCC + Q
$GLB1$	:	ICIT $\Longrightarrow$ GLYOXY + SUCC
$GLB2$	:	GLYOXY + ACoA $\Longrightarrow$ MAL + CoASH
$ANA1$	:	PEP + CO2 $\Longrightarrow$ OAA
$ANA2$	:	MAL + NAD $\Longrightarrow$ PYR + CO2 + NADH
$ANA3$	:	OAA + ATP $\Longrightarrow$ PEP + ADP + CO2
$FEM1$	:	PYR + CoASH $\Longrightarrow$ ACoA + FOR
$FEM2$	:	PYR + Q $\Longrightarrow$ ACE + CO2 + QH2
$FEM3$	:	PYR + NADH $\Longrightarrow$ LAC + NAD
$FEM4$	:	FOR $\Longrightarrow$ CO2 + H2[e]

$FEM5$	:	ACoA + NADH $\Longrightarrow$ ACA + NAD + CoASH
$FEM6$	:	ACA + NADH $\Longrightarrow$ ETOH + NAD
$FEM7$	:	ACoA $\Longrightarrow$ ACP + CoASH
$FEM8$	:	ACP + ADP $\Longrightarrow$ ACE + ATP
$FEM9$	:	PYR $\Longrightarrow$ ACA + CO2
$EDP1$	:	6PG $\Longrightarrow$ KDPG
$EDP2$	:	KDPG $\Longrightarrow$ PYR + GA3P
BIO	:	49 G6P + 17 F6P + 860 RIBO5P + 1426 AKG + 2355 OAA + 512 ERY4P + 960 PEP + 3920 PYR + 1642 3PG + 31 GA3P + 1207 ACoA + 40680 ATP + 4079 NAD + 18320 NADPH + 12502 NH3 $\Longrightarrow$ BIOMASS + 1207 CoASH + 40680 ADP + 4079 NADH + 18320 NADP
$OPM1$	:	NADH + 2 ADP + O2[e] $\Longrightarrow$ NAD + 2 ATP
$OPM2$	:	QH2 + ADP + O2[e] $\Longrightarrow$ Q + ATP
$OPM3$	:	ATP $\Longrightarrow$ ADP + ATP_base
$OPM4_r$	:	NADH + Q $\Longleftrightarrow$ NAD + QH2
$TRA1$	:	ETOH $\Longrightarrow$ ETOH[e]
$TRA2$	:	ACE $\Longrightarrow$ ACE[e]
$TRA3$	:	NH3[e] $\Longrightarrow$ NH3
$TRA4$	:	LAC $\Longrightarrow$ LAC[e]
$TRA5$	:	SUCC $\Longrightarrow$ SUCC[e]
$TRA6$	:	FOR $\Longrightarrow$ FOR[e]
$TRA7$	:	CO2 $\Longrightarrow$ CO2[e]
$TRA8$	:	XYLO[e] + ATP $\Longrightarrow$ XYLO + ADP
$FC1_r$	:	NAD + NADPH $\Longleftrightarrow$ NADP + NADH
$FC2$	:	AMP + ATP $\Longrightarrow$ 2 ADP

Table A.2: *Core metabolic network of Escherichia coli [2, 3]*

		(*Glycolysis/Gluconeogenesis*)
ENO	:	2PG $\Longleftrightarrow$ H2O + PEP
FBA	:	FDP $\Longleftrightarrow$ DHAP + G3P
FBP	:	FDP + H2O $\Longrightarrow$ F6P + Pi
$GAPD$	:	G3P + NAD + Pi $\Longleftrightarrow$ 13DPG + H + NADH
PDH	:	COA + NAD + PYR $\Longrightarrow$ ACCOA + CO2 + NADH
PFK	:	ATP + F6P $\Longrightarrow$ ADP + FDP + H
PGI	:	G6P $\Longleftrightarrow$ F6P
PGK	:	3PG + ATP $\Longleftrightarrow$ 13DPG + ADP
PGM	:	2PG $\Longleftrightarrow$ 3PG
PPS	:	ATP + H2O + PYR $\Longrightarrow$ AMP + 2 H + PEP + Pi
PYK	:	ADP + H + PEP $\Longleftrightarrow$ ATP + PYR
TPI	:	DHAP $\Longleftrightarrow$ G3P
$G6PDH2$	:	G6P + NADP $\Longleftrightarrow$ 6PGL + H + NADPH
		(*Pentose Phosphate pathway*)
GND	:	6PGC + NADP $\Longrightarrow$ CO2 + NADPH + RU5P_D
PGL	:	6PGL + H2O $\Longrightarrow$ 6PGC + H
RPE	:	RU5P_D $\Longleftrightarrow$ XU5P_D
RPI	:	R5P $\Longleftrightarrow$ RU5P_D
$TALA$	:	G3P + S7P $\Longleftrightarrow$ E4P + F6P
$TKT1$	:	R5P + XU5P_D $\Longleftrightarrow$ G3P + S7P
$TKT2$	:	E4P + XU5P_D $\Longleftrightarrow$ F6P + G3P
		(*Oxidative Phosphorylation*)
$ADK1$	:	AMP + ATP $\Longleftrightarrow$ 2 ADP
$ATPM$	:	ATP + H2O $\Longrightarrow$ ADP + H + Pi
$ATPS4$	:	ADP + 4 H[e] + Pi $\Longleftrightarrow$ ATP + H2O + 3 H
$CYTBD$	:	4 H + 1 O2 + 2 Q8H2 $\Longrightarrow$ 2 H2O + 4 H[e] + 2 Q8
$NADH16$	:	4 H + NADH + Q8 $\Longrightarrow$ 3 H[e] + NAD + Q8H2
$NADTRHD$	:	NAD + NADPH $\Longrightarrow$ NADH + NADP
$THD2$	:	2 H[e] + NADH + NADP $\Longrightarrow$ 2 H + NAD + NADPH
		(*Pyruvate metabolism*)
ACK	:	AC + ATP $\Longleftrightarrow$ ACTP + ADP
$ACALD$	:	ACALD + COA + NAD $\Longleftrightarrow$ ACCOA + H + NADH
$ALCD2x$	:	ETOH + NAD $\Longleftrightarrow$ ACALD + H + NADH
LDH_D	:	LAC_D + NAD $\Longleftrightarrow$ H + NADH + PYR
PFL	:	COA + PYR $\Longrightarrow$ ACCOA + FOR

PTA	:	ACCOA + Pi $\Longleftrightarrow$ ACTP + COA
		(*Anaplerotic reactions*)
ICL	:	ICIT $\Longrightarrow$ GLX + SUCC
MALS	:	ACCOA + GLX + H2O $\Longrightarrow$ COA + H + MAL_L
*ME*1	:	MAL_L + NAD $\Longrightarrow$ CO2 + NADH + PYR
*ME*2	:	MAL_L + NADP $\Longrightarrow$ CO2 + NADPH + PYR
PPC	:	CO2 + H2O + PEP $\Longrightarrow$ H + OAA + Pi
PPCK	:	ATP + OAA $\Longrightarrow$ ADP + CO2 + PEP
		(*TCA cycle*)
ICDHy	:	ICIT + NADP $\Longleftrightarrow$ AKG + CO2 + NADPH
CS	:	ACCOA + H2O + OAA $\Longrightarrow$ CIT + COA + H
FUM	:	FUM + H2O $\Longrightarrow$ MAL_L
MDH	:	MAL_L + NAD $\Longleftrightarrow$ H + NADH + OAA
SUCDi	:	Q8 + SUCC $\Longrightarrow$ FUM + q8h2
AKGDH	:	AKG + COA + NAD $\Longrightarrow$ CO2 + NADH + SUCCOA
SUCOAS	:	ATP + COA + SUCC $\Longleftrightarrow$ ADP + Pi + SUCCOA
*FRD*7	:	FUM + q8h2 $\Longrightarrow$ q8 + SUCC
ACONTa	:	CIT $\Longleftrightarrow$ ACON_C + H2O
ACONTb	:	ACON_C + H2O $\Longleftrightarrow$ ICIT
		(*Transport reactions*)
GLCpts	:	GLC_D[e] + PEP $\Longrightarrow$ G6P + PYR
*MALt*2_2	:	2 H[e] + MAL_L[e] $\Longrightarrow$ 2 H + MAL_L
*H*2*Ot*	:	H2O[e] $\Longleftrightarrow$ H2O
*GLUt*2	:	GLU_L[e] + H[e] $\Longleftrightarrow$ GLU_L + H
*NH*4*t*	:	NH4[e] $\Longleftrightarrow$ NH4
*O*2*t*	:	O2[e] $\Longleftrightarrow$ O2
*FORt*2	:	FOR[e] + H[e] $\Longrightarrow$ FOR + H
FORti	:	FOR $\Longrightarrow$ FOR[e]
ACALDt	:	ACALD[e] $\Longleftrightarrow$ ACALD
*ACt*2	:	AC[e] + H[e] $\Longleftrightarrow$ AC + H
*AKGt*2	:	AKG[e] + H[e] $\Longleftrightarrow$ AKG + H
*CO*2*t*	:	CO2[e] $\Longleftrightarrow$ CO2
*D_LACt*2	:	H[e] + LAC_D[e] $\Longleftrightarrow$ H + LAC_D
*ETOHt*2	:	ETOH[e] + H[e] $\Longleftrightarrow$ ETOH + H
*FUMt*2_2	:	FUM[e] + 2 H[e] $\Longrightarrow$ FUM + 2 H
*PIt*2	:	H[e] + Pi[e] $\Longleftrightarrow$ H + Pi
*PYRt*2	:	H[e] + PYR[e] $\Longleftrightarrow$ H + PYR

SUCCt2_2	:	2 H[e] + SUCC[e] $\Longrightarrow$ 2 H + SUCC
SUCCt3	:	H[e] + SUCC $\Longrightarrow$ H + SUCC[e]
FRUpts2	:	FRU[e] + PEP $\Longrightarrow$ F6P + PYR
EX_GLC	:	GLC_D[e] $\Longleftrightarrow$ GLC_D[b]
EX_GLN–L	:	GLN_L[e] $\Longleftrightarrow$ GLN_L[b]
EX_GLU–L	:	GLU_L[e] $\Longleftrightarrow$ GLU_L[b]
EX_H	:	H[e] $\Longleftrightarrow$ H2O[b]
EX_H2O	:	H2O[e] $\Longleftrightarrow$ H[b]
EX_LAC–D	:	LAC_D[e] $\Longleftrightarrow$ LAC_D[b]
EX_MAL–L	:	MAL_L[e] $\Longleftrightarrow$ MAL_L[b]
EX_NH4	:	NH4[e] $\Longleftrightarrow$ NH4[b]
EX_O2	:	O2[e] $\Longleftrightarrow$ O2[b]
EX_Pi	:	Pi[e] $\Longleftrightarrow$ Pi[b]
EX_PYR	:	PYR[e] $\Longleftrightarrow$ PYR[b]
EX_SUCC	:	SUCC[e] $\Longleftrightarrow$ SUCC[b]
EX_AC	:	AC_e $\Longleftrightarrow$ AC[b]
EX_ACALD	:	ACALD[e] $\Longleftrightarrow$ ACALD[b]
EX_AKG	:	AKG[e] $\Longleftrightarrow$ AKG[b]
EX_CO2	:	CO2[e] $\Longleftrightarrow$ CO2[b]
EX_ETOH	:	ETOH[e] $\Longleftrightarrow$ ETOH[b]
EX_FOR	:	FOR[e] $\Longleftrightarrow$ FOR[b]
EX_FRU	:	FRU[e] $\Longleftrightarrow$ FRU[b]
EX_FUM	:	FUM[e] $\Longleftrightarrow$ FUM[b]
		(*amino acid biosynthesis*)
GLNS	:	ATP + GLU_L + NH4 $\Longrightarrow$ ADP + GLN_L + H + Pi
GLNabc	:	ATP + GLN_L[e] + H2O $\Longrightarrow$ ADP + GLN_L + H + Pi
GLUDy	:	GLU_L + H2O + NADP $\Longleftrightarrow$ AKG + H + NADPH + NH4
GLUN	:	GLN_L + H2O $\Longrightarrow$ GLU_L + NH4
GLUSy	:	AKG + GLN_L + H + NADPH $\Longrightarrow$ 2 GLU_L + NADP
		(*Cell growth*)
BIO	:	14960 3PG + 37478 ACCOA + 598100 ATP + 3610 E4P + 709 F6P + 1290 G3P + 2050 G6P + 2557 GLN_L + 49414 GLU_L + 598100 H2O + 35470 NAD + 130279 NADPH + 17867 OAA + 5191 PEP + 28328 PYR + 8977 R5P $\Longrightarrow$ 598100 ADP + 41182 AKG + 37478 COA + 598100 H + 35470 NADH + 130279 NADP + 598100 Pi

A.2 *Sacharomyces cerevisiae*

Table A.3 illustrates the central metabolic network where, the 80-reaction network extends the 78-reaction network, while the 83-reaction network extends the 80-reaction network, respectively. Lists of irreversible and reversible reaction are given separately in the anaerobic 78-reaction network, while the extra reactions and modifications are listed at the end of the table.

Table A.3: *Central metabolism of S. cerevisiae [149]*

S. cerevisiae metabolic network I with 62 metabolites and 78 reactions:		
the irreversible reactions.		
$R4$	:	F6P + ATP $\Longrightarrow$ FDP + ADP
$R5$	:	FDP $\Longrightarrow$ F6P
$R9$	:	PYR + ATP $\Longrightarrow$ PEP + ADP
$R10$	:	PEP + ADP $\Longrightarrow$ PYR + ATP
$R12$	:	GL3P + FAD[m] $\Longrightarrow$ DHAP + FADH[m]
$R26$	:	GL3P $\Longrightarrow$ GLY
$R15$	:	G6P + 2 NADP $\Longrightarrow$ 2 NADPH + CO2 + RL5P
$R21$	:	ACCOA + OA $\Longrightarrow$ COA + CIT
$R23$	:	ICIT + NADP $\Longrightarrow$ CO2 + NADPH + AKG
$R24$	:	AKG[m] + NAD[m] + COA[m] $\Longrightarrow$ CO2 + NADH[m] + SUCCOA[m]
$R27$	:	FUM + FADH $\Longrightarrow$ SUCC + FAD
$R33$	:	PYR + COA $\Longrightarrow$ ACCOA + FOR
$R37$	:	PYR + ATP + CO2 $\Longrightarrow$ ADP + OA
$R38$	:	PYR $\Longrightarrow$ ACEADH + CO2
$R40$	:	ACEADH + NADH $\Longrightarrow$ ETOH + NAD
$R41$	:	ACEADH + NADP $\Longrightarrow$ AC + NADPH
$R42$	:	OA + ATP $\Longrightarrow$ PEP + CO2 + ADP
$R43$	:	PEP + CO2 $\Longrightarrow$ OA
$R46$	:	ICIT $\Longrightarrow$ GLX + SUCC
$R47$	:	ACCOA + GLX $\Longrightarrow$ COA + MAL
$R53$	:	ACEADH + NAD $\Longrightarrow$ AC + NADH
$R54$	:	ATP $\Longrightarrow$ ADP
$R58$	:	NADH + NAD[m] $\Longrightarrow$ NAD + NADH[m]
$R59$	:	NH3[e] $\Longrightarrow$ NH3
$R60$	:	GLY $\Longrightarrow$ GLY[e]
$R62$	:	GLC[e] + PEP $\Longrightarrow$ G6P + PYR
$R63$	:	AC $\Longrightarrow$ AC[e]

$R64$: LAC $\Longrightarrow$ LAC[e]

$R65$: FOR $\Longrightarrow$ FOR[e]

$R66$: ETOH $\Longrightarrow$ ETOH[e]

$R67$: SUCC $\Longrightarrow$ SUCC[e]

$R68$: O2[e] $\Longrightarrow$ O2

$R69$: CO2 $\Longrightarrow$ CO2[e]

$R70$: 7437 G6P + 611 G3P + 437 R5P + 130 E4P + 500 PEP + 2060 PYR + 45 ACCOA[m] + 362 ACCOA + 733 AKG + 1232 OA + 1158 NAD + 434 NAD[m] + 6413 NADPH + 1568 NADPH[m] + 40141 ATP + 5587 NH3 $\Longrightarrow$ 1000 BIO + 247 CO2 + 45 COA[m] + 362 COA + 1158 NADH + 434 NADH[m] + 6413 NADP + 1568 NADP[m] + 40141 ADP

$R72$: PYR[m] + COA[m] + NAD[m] $\Longrightarrow$ ACCOA[m] + NADH[m] + CO2

$R73$: OA[m] + ACCOA[m] $\Longrightarrow$ CIT[m] + COA[m]

$R75$: ICIT[m] + NAD[m] $\Longrightarrow$ AKG[m] + NADH[m] + CO2

$R76$: ICIT[m] + NADP[m] $\Longrightarrow$ AKG[m] + NADPH[m] + CO2

$R77$: ICIT + NADP $\Longrightarrow$ AKG + NADPH + CO2

$R82$: MAL[m] + NADP[m] $\Longrightarrow$ PYR[m] + NADPH[m] + CO2

$R85$: ETOH[m] + COA[m] + 2 ATP[m] + 2 NAD[m] $\Longrightarrow$ ACCOA[m] + 2 ADP[m] + 2 NADH[m]

$R86$: ACEADH[m] + NAD[m] $\Longrightarrow$ AC[m] + NADH[m]

$R87$: ACEADH[m] + NADP[m] $\Longrightarrow$ AC[m] + NADPH[m]

$R93$: ADP + ATP[m] $\Longrightarrow$ ADP[m] + ATP

$R98$: FUM[m] + SUCC $\Longrightarrow$ SUCC[m] + FUM

$R100$: SUCC $\Longrightarrow$ SUCC[m]

$R101$: AKG + MAL[m] $\Longrightarrow$ AKG[m] + MAL

S. cerevisiae metabolic network I with 62 metabolites and 78 reactions:

the reversible reactions

$R3_r$	:	$\mathrm{G6P} \Longleftrightarrow \mathrm{F6P}$
$R6_r$	:	$\mathrm{FDP} \Longleftrightarrow \mathrm{G3P} + \mathrm{DHAP}$
$R7_r$	:	$\mathrm{G3P} \Longleftrightarrow \mathrm{DHAP}$
$R8_r$	:	$\mathrm{G3P} + \mathrm{NAD} + \mathrm{ADP} \Longleftrightarrow \mathrm{PEP} + \mathrm{ATP} + \mathrm{NADH}$
$R13_r$	:	$\mathrm{DHAP} + \mathrm{NADH} \Longleftrightarrow \mathrm{GL3P} + \mathrm{NAD}$
$R16_r$	:	$\mathrm{RL5P} \Longleftrightarrow \mathrm{R5P}$
$R17_r$	:	$\mathrm{RL5P} \Longleftrightarrow \mathrm{X5P}$
$R18_r$	:	$\mathrm{R5P} + \mathrm{X5P} \Longleftrightarrow \mathrm{G3P} + \mathrm{S7P}$
$R19_r$	:	$\mathrm{X5P} + \mathrm{E4P} \Longleftrightarrow \mathrm{F6P} + \mathrm{G3P}$
$R20_r$	:	$\mathrm{G3P} + \mathrm{S7P} \Longleftrightarrow \mathrm{E4P} + \mathrm{F6P}$
$R22_r$	:	$\mathrm{CIT} \Longleftrightarrow \mathrm{ICIT}$
$R25_r$	:	$\mathrm{SUCCOA[m]} + \mathrm{ADP[m]} \Longleftrightarrow \mathrm{ATP[m]} + \mathrm{COA[m]} + \mathrm{SUCC[m]}$
$R28_r$	:	$\mathrm{FUM} \Longleftrightarrow \mathrm{MAL}$
$R29_r$	:	$\mathrm{MAL} + \mathrm{NAD} \Longleftrightarrow \mathrm{NADH} + \mathrm{OA}$
$R30_r$	:	$\mathrm{PYR} + \mathrm{NADH} \Longleftrightarrow \mathrm{NAD} + \mathrm{LAC}$
$R32_r$	:	$\mathrm{ACCOA} + 2\ \mathrm{NADH} \Longleftrightarrow \mathrm{ETOH} + 2\ \mathrm{NAD} + \mathrm{COA}$
$R36_r$	:	$\mathrm{ATP} + \mathrm{AC} + \mathrm{COA} \Longleftrightarrow \mathrm{ADP} + \mathrm{ACCOA}$
$R74_r$	:	$\mathrm{CIT[m]} \Longleftrightarrow \mathrm{ICIT[m]}$
$R78_r$	:	$\mathrm{ACEADH[m]} + \mathrm{NADH[m]} \Longleftrightarrow \mathrm{ETOH[m]} + \mathrm{NAD[m]}$
$R79_r$	:	$\mathrm{SUCC[m]} + \mathrm{FAD[m]} \Longleftrightarrow \mathrm{FUM[m]} + \mathrm{FADH[m]}$
$R80_r$	:	$\mathrm{FUM[m]} \Longleftrightarrow \mathrm{MAL[m]}$
$R81_r$	:	$\mathrm{MAL[m]} + \mathrm{NAD[m]} \Longleftrightarrow \mathrm{OA[m]} + \mathrm{NADH[m]}$
$R88_r$	:	$\mathrm{CIT} + \mathrm{MAL[m]} \Longleftrightarrow \mathrm{CIT[m]} + \mathrm{MAL}$
$R89_r$	:	$\mathrm{MAL} + \mathrm{SUCC[m]} \Longleftrightarrow \mathrm{MAL[m]} + \mathrm{SUCC}$
$R90_r$	:	$\mathrm{CIT} + \mathrm{ICIT[m]} \Longleftrightarrow \mathrm{CIT[m]} + \mathrm{ICIT}$
$R92_r$	:	$\mathrm{AC[m]} \Longleftrightarrow \mathrm{AC}$
$R94_r$	:	$\mathrm{PYR} \Longrightarrow \mathrm{PYR[m]}$
$R95_r$	:	$\mathrm{ETOH} \Longrightarrow \mathrm{ETOH[m]}$
$R96_r$	:	$\mathrm{MAL[m]} \Longrightarrow \mathrm{MAL}$
$R97_r$	:	$\mathrm{ACCOA[m]} \Longrightarrow \mathrm{ACCOA}$
$R102r$	:	$\mathrm{OA} \Longleftrightarrow \mathrm{OA[m]}$

S. cerevisiae metabolic network II with 62 metabolites and 80 reactions:

added to network I.

$R56$	:	$24\ \mathrm{ADP} + 20\ \mathrm{NADH[m]} + 10\ \mathrm{O2} \Longrightarrow 24\ \mathrm{ATP} + 20\ \mathrm{NAD[m]}$
$R57$	:	$24\ \mathrm{ADP} + 20\ \mathrm{FADH} + 10\ \mathrm{O2} \Longrightarrow 24\ \mathrm{ATP} + 20\ \mathrm{FAD}$

S. cerevisiae metabolic network III with 63 metabolites and 83 reactions: differences from network I and II.

additional internal metabolite:

GLC

added reactions:

$R1$	:	GLC + ATP $\Longrightarrow$ G6P + ADP
$R14$	:	GLY + ATP $\Longrightarrow$ GL3P + ADP
$R56$	:	24 ADP + 20 NADH[m] + 10 O2 $\Longrightarrow$ 24 ATP + 20 NAD[m]
$R57$	:	24 ADP + 20 FADH + 10 O2 $\Longrightarrow$ 24 ATP + 20 FAD
$R61$	:	GLC[e] $\Longrightarrow$ GLC

reactions made reversible:

$R54_r$	:	ATP $\Longleftrightarrow$ ADP
$R60_r$	:	GLY $\Longleftrightarrow$ GLY[e]
$R63_r$	:	AC $\Longleftrightarrow$ AC[e]

modified reaction:

$R62$	:	GLC + PEP $\Longrightarrow$ G6P + PYR

Table A.4: *Recombinant metabolic network of S. cerevisiae [162]*

$R1$	:	GLU[e] + PEP $\Longrightarrow$ GLU6P + PYR
$R2r$	:	GLU6P $\Longleftrightarrow$ FRU6P
$R3$	:	FRU6P + ATP $\Longrightarrow$ FRUBISP + ADP
$R4$	:	FRUBISP $\Longrightarrow$ FRU6P
$R5r$	:	FRUBISP $\Longleftrightarrow$ DHAP + GA3P
$R6r$	:	GA3P $\Longleftrightarrow$ DHAP
$R7r$	:	GA3P + ADP + NAD $\Longleftrightarrow$ PG + ATP + NADH
$R8r$	:	PG $\Longleftrightarrow$ PEP
$R9$	:	PEP + ADP $\Longrightarrow$ PYR + ATP
$RR9$	:	PYR + 2 ATP $\Longrightarrow$ PEP + 2 ADP
$R10$	:	GLU6P + 2 NAD $\Longrightarrow$ RIBULOSE5P + 2 NADH + CO2
$R11r$	:	RIBULOSE5P $\Longleftrightarrow$ XYL5P
$R12r$	:	RIBULOSE5P $\Longleftrightarrow$ RIBOSE5P
$R13r$	:	RIBOSE5P + XYL5P $\Longleftrightarrow$ SED7P + GA3P
$R14r$	:	GA3P + SED7P $\Longleftrightarrow$ ERYTH4P + FRU6P
$R15r$	:	ERYTH4P + XYL5P $\Longleftrightarrow$ GA3P + FRU6P
$R20$	:	PYR + CoASH $\Longrightarrow$ ACETYLCoA + FORMATE

$R21$	:	PYR + NAD + CoASH $\Longrightarrow$ ACETYLCoA + CO2 + NADH
$R22$	:	OXALO + ACETYLCoA $\Longrightarrow$ CITRATE + CoASH
$R23r$	:	CITRATE $\Longleftrightarrow$ ISOCIT
$R24$	:	ISOCIT + NAD $\Longrightarrow$ AKG + NADH + CO2
$R25$	:	AKG + NAD + CoASH $\Longrightarrow$ NADH + SUCCCoA + CO2
$R26r$	:	SUCCCoA + ADP $\Longleftrightarrow$ SUCC + ATP + CoASH
$R27r$	:	SUCC + FAD $\Longleftrightarrow$ FUMARATE + FADH
$R28r$	:	FUMARATE $\Longleftrightarrow$ MALATE
$R29r$	:	MALATE + NAD $\Longleftrightarrow$ OXALO + NADH
$R40$	:	PEP + CO2 $\Longrightarrow$ OXALO
$R41$	:	MALATE + NAD $\Longrightarrow$ PYR + NADH + CO2
$R42$	:	OXALO + ATP $\Longrightarrow$ PEP + ADP + CO2
$R53r$	:	PYR + NADH $\Longleftrightarrow$ LACTATE + NAD
$R54r$	:	ACETYLCoA + 2 NADH $\Longleftrightarrow$ ETOH + 2 NAD + CoASH
$R55$	:	ACETYLCoA + ADP $\Longrightarrow$ ACETATE + CoASH + ATP
$R70$	:	4 GLU6P + 46 RIBOSE5P + 31 ERYTH4P + 156 PEP + 237 PYR + 72 ACETYLCoA + 86 AKG + 139 OXALO + 2921 ATP + 856 NADH + 731 NH3 $\Longrightarrow$ BIOMASS + 72 CoASH + 2921 ADP + 856 NAD + 35 CO2
$R80$	:	NADH + 2 ADP + OXY[e] $\Longrightarrow$ NAD + 2 ATP
$R81$	:	FADH + ADP + OXY[e] $\Longrightarrow$ FAD + ATP
$R82$	:	ATP $\Longrightarrow$ ADP + ATPmain
$R83$	:	NADH + FAD $\Longrightarrow$ NAD + FADH
$R90$	:	ETOH $\Longrightarrow$ ETOH[e]
$R91$	:	ACETATE $\Longrightarrow$ ACETATE[e]
$R93$	:	NH3[e] $\Longrightarrow$ NH3
$R94$	:	LACTATE $\Longrightarrow$ LACTATE[e]
$R95$	:	SUCC $\Longrightarrow$ SUCC[e]
$R96$	:	FORMATE $\Longrightarrow$ FORMATE[e]
$R97r$	:	CO2 $\Longleftrightarrow$ CO2[e]

Appendix B

Pseudocode

B.1 Serial Nullspace Algorithm

Algorithm 17 $[combinations] = \texttt{RadixSortEFMCands}(R^{(\mathsf{bit})}, combinations, width)$

```
Input:
 1: bit pattern matrix used to generate new candidates - $R^{(\mathsf{bit})}$
 2: pairs of column indices which generate new candidates - combinations
 3: width of the sequence of bits over which elementary radix sort is performed - width
Output:
 4: reordered pairs of indices so that corresponding columns are sorted - combinations
 5: $r \Leftarrow \texttt{size}(R^{(\mathsf{bit})}, 1)$
 6: col_length ⇐ number of machine words in r bits
 7: for $i = 1$ to col_length do
 8:     $factor \Leftarrow \frac{r}{width}$        ▷ number of sequences of length width in current word
 9:     for $j = 1$ to $factor$ do
10:         $counting \Leftarrow \texttt{zeros}(1, 2^{width})$
11:         for $k = 1$ to $\texttt{length}(combinations)$ do
12:             $(ii, jj) \Leftarrow combinations_k$
13:             $aa \Leftarrow R^{(\mathsf{bit})}_{*,ii}$ or $R^{(\mathsf{bit})}_{*,jj}$
14:             $aa \Leftarrow (aa$ shl $j \cdot width)$ and $(2^{width} - 1)$
15:             $counting_{aa} \Leftarrow counting_{aa} + 1$
16:         end for
17:         for $k = 1$ to $2^{width}$ do
18:             $counting_k \Leftarrow counting_k + counting_{k-1}$
19:         end for
20:         for $k = \texttt{length}(combinations)$ downto 1 do
21:             $(ii, jj) \Leftarrow combinations(k)$
22:             $aa \Leftarrow R^{(\mathsf{bit})}_{*,ii}$ or $R^{(\mathsf{bit})}_{*,jj}$
23:             $aa \Leftarrow (aa$ shl $j \cdot width)$ and $2^{width} - 1$
```

Algorithm 17 (continued)

```
24:             $combinations\_sorted[counting_{aa} - 1] \Leftarrow combinations_k$
25:             $counting_{aa} \Leftarrow counting_{aa} - 1$
26:         end for
27:         $combinations \Leftarrow combinations\_sorted$
28:     end for
29: end for
```

Algorithm 18 $[combinations]$ = `RankTests`$(N, R, combinations)$

Input:

```
1: reduced stoichiometry matrix - $N_{m\times q}$
2: initial nullspace matrix - $R_{q\times(q-m)} = \begin{bmatrix} R^{(\mathsf{bit})} \\ R^{(\mathsf{real})} \end{bmatrix}$
3: array of pairs of column-generating indices - $combinations$
```

Output:

```
4: array column-generating pairs valid elementary modes -$combinations$)
5: $k \Leftarrow$ size$(R^{(\mathsf{bit})}, 1)$
6: for each $(ii, jj) \in combinations$ do
7:     $x_{1\times k} \Leftarrow R^{(\mathsf{bit})}_{*,ii}$ or $R^{(\mathsf{bit})}_{*,jj}$
8:     $aa \Leftarrow$ indices of non-zero entries in $x_{1\ldots q-m}$
9:     $bb \Leftarrow$ indices of zero entries in $x_{q-m+1\ldots k}$
10:    $\triangleright$ if Theorem 6 is not satisfied reject candidate
11:    if NULLITY$(N_{bb,aa}) \neq 1$ then
12:        $combinations \Leftarrow combinations \setminus (ii, jj)$
13:    end if
14: end for
```

Algorithm 19 $[R]$ = `ExpandEfm`$(R, combinations)$

Input:

```
1: current nullspace matrix - $R_{q\times(q-m)} = \begin{bmatrix} R^{(\mathsf{bit})} \\ R^{(\mathsf{real})} \end{bmatrix}$
2: array of column-generating pairs of indices - $combinations$
```

Output:

```
3: updated nullspace matrix - $R$
4: $k \Leftarrow$ size$(R^{(\mathsf{bit})}, 1)$
5: $eps \Leftarrow 10^{-10}$
6: for each $(ii, jj) \in combinations$ do
7:     $x_{k\times 1} \Leftarrow R^{(\mathsf{bit})}_{*,ii}$ or $R^{(\mathsf{bit})}_{*,jj}$
8:     $y_{(q-r)\times 1} \Leftarrow$ linear combination of $R^{(\mathsf{real})}_{*,ii}$ and $R^{(\mathsf{real})}_{*,jj}$ so that $y_1 = 0$
```

Algorithm 19 (continued)

```
 9:     $y(\texttt{fabs}(y) < eps) \Leftarrow 0$     ▷ *for simplicity we omit the check if improperly adopted tolerance assigns zero value erroneously*
10:     $y \Leftarrow y/\|y\|_1$
11: **end for**
12: $R^{(\mathsf{bit})} \Leftarrow [R^{(\mathsf{bit})}\ x]$
13: $R^{(\mathsf{real})} \Leftarrow [R^{(\mathsf{real})}\ y]$
14: **if** $k^{th}$ reaction is irreversible **then**
15:     delete from R columns with negative elements in current row
16: **end if**
```

B.2 Global Arrays-based Parallel Nullspace Algorithm

Algorithm 20 $[pInd^{(\mathsf{GA})}, nInd^{(\mathsf{GA})}, zInd^{(\mathsf{GA})}] = \texttt{ScanRealValueEfmGA}(currReac, R^{(\mathsf{GA,real})})$

```
Input:
 1: currently processed reaction-row - $currReac$
 2: global array of real-valued part of current right nullspace matrix - $R^{(\mathsf{GA,real})}_{(q-currReac+1)\times nEms}$
Output:
 3: global arrays of indices of positive, negative and zero columns - $pInd^{(\mathsf{GA})}$, $nInd^{(\mathsf{GA})}$, $zInd^{(\mathsf{GA})}$
 4: at processor $procId$:
 5:     $numLocalCols = \texttt{min}((procId+1)\cdot \frac{nEms}{P}, nEms)\text{-}procId\cdot\frac{nEms}{P}$
 6:     $R^{(\mathsf{LC,real})} = \texttt{GET}(R^{(\mathsf{GA,real})}, procId\cdot\frac{nEms}{P}\ldots procId\cdot\frac{nEms}{P}+numLocalCols)$
 7:     scan first row in $R^{(\mathsf{LC,real})}$ to generate indices $pInd^{(\mathsf{LC})}$, $nInd^{(\mathsf{LC})}$ and $zInd^{(\mathsf{LC})}$
 8:     **for** $k = 1$ to $numLocalCols$
 9:         **if** $R^{(\mathsf{LC,real})}[1,k] > 0$
10:             $pInd^{(\mathsf{LC})} = pInd^{(\mathsf{LC})} \cup \{procId\cdot\frac{nEms}{P}+k\}$
11:         **elsif** $R^{(\mathsf{LC,real})}[1,k] < 0$
12:             $nInd^{(\mathsf{LC})} = nInd^{(\mathsf{LC})} \cup \{procId\cdot\frac{nEms}{P}+k\}$
13:         **else**
14:             $zInd^{(\mathsf{LC})} = zInd^{(\mathsf{LC})} \cup \{procId\cdot\frac{nEms}{P}+k\}$
15:         **end**
16:     **end**
17:     $[prefixPInd]$=MPI_PREFIX_SUM(length($pInd^{(\mathsf{LC})}$))
18:     $[prefixNInd]$=MPI_PREFIX_SUM(length($nInd^{(\mathsf{LC})}$))
19:     $[prefixZInd]$=MPI_PREFIX_SUM(length($zInd^{(\mathsf{LC})}$))
20:     **if** $procId == P-1$
21:         $totalPInd = prefixPInd$, $totalNInd = prefixNInd$, $totalZInd = prefixZInd$
22:     **end**
23:     broadcast values $totalPInd$, $totalNInd$ and $totalZInd$ from processor $P-1$
24:     ▷ create global arrays of positive, negative and zero columns of length $totalPInd$ , $totalNInd$ and $totalZInd$
25:     $pInd^{(\mathsf{GA})}$=CREATE($totalPInd$),$nInd^{(\mathsf{GA})}$=CREATE($totalNInd$),$zInd^{(\mathsf{GA})}$=CREATE($totalZInd$)
26:     PUT($pInd^{(\mathsf{GA})}$, $pInd^{(\mathsf{LC})}$), PUT($nInd^{(\mathsf{GA})}$, $nInd^{(\mathsf{LC})}$), PUT($zInd^{(\mathsf{GA})}$, $zInd^{(\mathsf{LC})}$)
27:
28: **return** $[pInd^{(\mathsf{GA})}, nInd^{(\mathsf{GA})}, zInd^{(\mathsf{GA})}]$
```

Algorithm 21 $[cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}] = \texttt{GenerateCandsGA}(currReac, pInd^{(\mathsf{GA})}, nInd^{(\mathsf{GA})}, R^{(\mathsf{GA,bit})})$

```
Input:
 1: currently processed reaction-row - $currReac$
 2: global arrays of indices of positive and negative columns - $pInd^{(\mathsf{GA})}$, $nInd^{(\mathsf{GA})}$
 3: global array of bit-valued part of current right nullspace matrix - $R^{(\mathsf{GA,bit})}$
Output:
 4: local candidate mode column pairs of indices - $cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}$
 5: at processor $procId$:
 6:     $numPosCols = \texttt{min}((procId + 1) \cdot \frac{\texttt{length}(pInd^{(\mathsf{GA})})}{P}, \texttt{length}(pInd^{(\mathsf{GA})}))\text{-}procId \cdot \frac{\texttt{length}(pInd^{(\mathsf{GA})})}{P}$
 7:     $pIndInd$ = $numPosCols$ indices to elements in $pInd^{(\mathsf{GA})}$
 8:     $pInd^{(\mathsf{LC})} = \texttt{GATHER}(pInd^{(\mathsf{GA})}, pIndInd)$
 9:     $posCols^{(\mathsf{LC,bit})} = \texttt{GATHER}(R^{(\mathsf{GA,bit})}, pInd^{(\mathsf{LC})})$
10:     for $it = 0$ to $P - 1$
11:         $numNegCols = \texttt{min}(it \cdot \frac{\texttt{length}(nInd^{(\mathsf{GA})})}{P}, \texttt{length}(nInd^{(\mathsf{GA})}))\text{-}it \cdot \frac{\texttt{length}(nInd^{(\mathsf{GA})})}{P}$
12:         $nInd^{(\mathsf{LC})} = \texttt{GET}(nInd^{(\mathsf{GA})}, it \cdot \frac{\texttt{length}(nInd^{(\mathsf{GA})})}{P} \dots it \cdot \frac{\texttt{length}(nInd^{(\mathsf{GA})})}{P} + numNegCols)$
13:         $negCols^{(\mathsf{LC,bit})} = \texttt{GATHER}(R^{(\mathsf{GA,bit})}, nInd^{(\mathsf{LC})})$
14:         for $s = 1$ to $\texttt{length}(pInd^{(\mathsf{LC})})$
15:             for $t = 1$ to $\texttt{length}(nInd^{(\mathsf{LC})})$
16:                 $candColumn = posCols^{(\mathsf{LC,bit})}[s]$ bit-OR $negCols^{(\mathsf{LC,bit})}[t]$
17:                 if ($candColumn$ is admissible)
18:                     $cPInd^{(\mathsf{LC})}.add(pInd^{(\mathsf{LC})}[s])$
19:                     $cNInd^{(\mathsf{LC})}.add(nInd^{(\mathsf{LC})}[t])$
20:                 end
21:             end
22:         end
23:     end
24: return $[cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}]$
```

Algorithm 22 $[cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}] = \texttt{LocalPruneCandModesGA}(cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}, R^{(\mathsf{GA,bit})})$

```
Input:
 1: local pairs of indices of accepted column modes - $cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}$
 2: global array of bit-valued part of current right nullspace matrix - $R^{(\mathsf{GA,bit})}$
Output:
 3: array of pairs of indices of accepted unique column modes $cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}$
 4: at node $procId$:
 5:     sort local pairs of indices ($cPInd^{(\mathsf{LC})}$, $cNInd^{(\mathsf{LC})}$) which generate candidate columns using serial radix-sort
 6:     remove pairs of indices in ($cPInd^{(\mathsf{LC})}$, $cNInd^{(\mathsf{LC})}$) which generate duplicate candidate columns in single scan
```

Algorithm 23 $[cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}]$=`GlobalPruneCandModesGA`$(currReac, cPInd^{(\mathsf{LC})}, cNInd^{(\mathsf{LC})}, R^{(\mathsf{GA,bit})})$

```
Input:
 1: currently processed reaction-row - $currReac$
 2: global arrays of indices of positive and negative columns - $cPInd^{(\mathsf{LC})}$, $cNInd^{(\mathsf{LC})}$
 3: global array of bit-valued part of current right nullspace matrix - $R^{(\mathsf{GA,bit})}$
Output:
 4: local candidate mode column pairs of indices - $cPInd^{(\mathsf{GA})}$, $cNInd^{(\mathsf{GA})}$
 5: at processor $procId$:
 6:     $\triangleright$ do prefix scan to find number of all candidate mode pairs of indices
 7:     $[prefixNumPairs]$=MPI_PREFIX_SUM(length($cPInd^{(\mathsf{LC})}$))
 8:     if $procId == P-1$
 9:        $totalNumPairs = prefixNumPairs$
10:     end
11:     broadcast value $totalNumPairs$ from processor $P-1$
12:     $cPInd^{(\mathsf{GA})}$=CREATE($totalNumPairs$), $cNInd^{(\mathsf{GA})}$=CREATE($totalNumPairs$)
13:     PUT($cPInd^{(\mathsf{GA})}$, $cPInd^{(\mathsf{LC})}$), PUT($cNInd^{(\mathsf{GA})}$, $cNInd^{(\mathsf{LC})}$)
14:     [$cPInd^{(\mathsf{GA})}$, $cNInd^{(\mathsf{GA})}$]=RadixSortEFMCandsGA($currReac, cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}, R^{(\mathsf{GA,bit})}$)
15:     [$cPInd^{(\mathsf{GA})}$, $cNInd^{(\mathsf{GA})}$]=UniqueEFMCandsGA($currReac, cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}, R^{(\mathsf{GA,bit})}$)
16: return ($cPInd^{(\mathsf{GA})}$, $cNInd^{(\mathsf{GA})}$)
```

Algorithm 24 $[cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}]$ = `RadixSortEFMCandsGA`$(currReac, cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}, R^{(\mathsf{GA,bit})})$

```
Input:
 1: index of currently processed reaction-row - $currReac$
 2: global arrays of pairs of indices of accepted column modes - $cPInd^{(\mathsf{GA})}$,$cNInd^{(\mathsf{GA})}$
 3: bit-valued part of current nullspace matrix as Global Array - $R^{(\mathsf{GA,bit})}$
Output:
 4: sorted global arrays of pairs of indices of accepted column modes - $cPInd^{(\mathsf{GA})}$,$cNInd^{(\mathsf{GA})}$
 5: at processor $procId$:
 6:        $numLocalCols = \texttt{min}(procId \cdot \frac{\texttt{length}(cPInd^{(\mathsf{GA})})}{P}, \texttt{length}(cPInd^{(\mathsf{GA})}))\text{-}procId \cdot \frac{\texttt{length}(cPInd^{(\mathsf{GA})})}{P}$
 7:        $numBits$ = 32-bit or 64-bit, $width$ = 8 or 16
 8:        for $i = 1$ to $\frac{currReac}{numBits}$
 9:           for $j = 1$ to $\frac{numBits}{width}$
10:              $cPInd^{(\mathsf{LC})}$ = GET($cPInd^{(\mathsf{GA})}$, $procId \cdot \frac{\texttt{length}(cPInd^{(\mathsf{GA})})}{P} \dots procId \cdot \frac{\texttt{length}(cPInd^{(\mathsf{GA})})}{P} + numLocalCols$)
11:              $cNInd^{(\mathsf{LC})}$ = GET($cNInd^{(\mathsf{GA})}$, $procId \cdot \frac{\texttt{length}(cNInd^{(\mathsf{GA})})}{P} \dots procId \cdot \frac{\texttt{length}(cNInd^{(\mathsf{GA})})}{P} + numLocalCols$)
12:              $R^{(\mathsf{bit,LC})}$ = bit-OR columns in $R^{(\mathsf{GA,bit})}$ indexed by $cPInd^{(\mathsf{LC})}$ and $cNInd^{(\mathsf{LC})}$
13:              $counting$=scan bit interval of length $width$ in columns of $R^{(\mathsf{bit,LC})}$ and increment respective elements of $counting$ array
14:              [$prefixCount$]=MPI_PREFIX_SUM($counting$)
15:              if $procId == P-1$
16:                 $totalPrefixCount$=$prefixCount$
17:              end
18:              broadcast array $totalPrefixCount$ from processor $P-1$
19:              shift elements in array $totalPrefixCount[1 \dots K]$ one position to the right
```

Algorithm 24 (continued)

```
20:           $totalPrefixCount[k]$=0
21:             for $k = 2$ to $K$
22:               $totalPrefixCount[k]$=$totalPrefixCount[k-1]$+$totalPrefixCount[k]$
23:             end
24:             ▷ determine new position of local column indices in the global array
25:             for $k = 1$ to $numLocalColumns$
26:               $bitVal$ = $width$-bit sequence at position $j \times width$ in $i^{\mathrm{th}}$ word
27:               $columnIndPosition[k]$=$totalPrefixCount[bitVal]$+$prefixCount[bitVal]$
28:             end
29:             ▷ scatter indices in $cPInd^{(\mathsf{LC})}$ and $cNInd^{(\mathsf{LC})}$ into proper positions in respective global arrays
30:             SCATTER($cPInd^{(\mathsf{GA})}$,$columnIndPosition$,$cPInd^{(\mathsf{LC})}$)
31:             SCATTER($cNInd^{(\mathsf{GA})}$,$columnIndPosition$,$cNInd^{(\mathsf{LC})}$)
32:         end
33:       end
34:
35: return [$cPInd^{(\mathsf{GA})}$, $cNInd^{(\mathsf{GA})}$]
```

Algorithm 25 [$cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}$] = UniqueEFMCandsGA($currReac, cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}, R^{(\mathsf{GA,bit})}$)

```
Input:
 1: index of currently processed reaction-row - $currReac$
 2: global arrays of pairs of indices of accepted column modes - $cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}$
 3: global array of bit-valued part of current nullspace matrix - $R^{(\mathsf{GA,bit})}$
Output:
 4: global arrays of pairs of indices generating unique accepted modes - $cPInd^{(\mathsf{GA})}, cNInd^{(\mathsf{GA})}$
 5: at node $procId$:
 6:     $numLocalCols$ = min($procId \cdot \frac{\mathtt{length}(cPInd^{(\mathsf{GA})})}{P}$, length($cPInd^{(\mathsf{GA})}$))-$procId \cdot \frac{\mathtt{length}(cPInd^{(\mathsf{GA})})}{P}$
 7:     fetch $\frac{\mathtt{length}(cPInd^{(\mathsf{GA})})}{P}$ elements from both index arrays
 8:     $cPInd^{(\mathsf{LC})}$ = GET($cPInd^{(\mathsf{GA})}$, $procId \cdot \frac{\mathtt{length}(cPInd^{(\mathsf{GA})})}{P} \ldots procId \cdot \frac{\mathtt{length}(cPInd^{(\mathsf{GA})})}{P} + numLocalCols$)
 9:     $cNInd^{(\mathsf{LC})}$ = GET($cNInd^{(\mathsf{GA})}$, $procId \cdot \frac{\mathtt{length}(cNInd^{(\mathsf{GA})})}{P} \ldots procId \cdot \frac{\mathtt{length}(cNInd^{(\mathsf{GA})})}{P} + numLocalCols$)
10:     scan the candidate columns which are formed by columns indexed using pairs from ($cPInd^{(\mathsf{LC})}$, $cNInd^{(\mathsf{LC})}$)
11:     ($uniqPosInd^{(\mathsf{LC})}$,$uniqNegInd^{(\mathsf{LC})}$) = unique($cPInd^{(\mathsf{LC})}$,$cNInd^{(\mathsf{LC})}$)
12:     [$uniqCnt$]=MPI_PREFIX_SUM(length($uniqPosInd^{(\mathsf{LC})}$))
13:     broadcast value $uniqCnt$ from processor $P-1$
14:     DESTROY($cPInd^{(\mathsf{GA})}$), DESTROY($cNInd^{(\mathsf{GA})}$)
15:     $cPInd^{(\mathsf{GA})}$= CREATE($uniqCnt$), $cNInd^{(\mathsf{GA})}$=CREATE($uniqCnt$)
16:     PUT($uniqPosInd^{(\mathsf{LC})}$,$cPInd^{(\mathsf{GA})}$),PUT($uniqNegInd^{(\mathsf{LC})}$,$cNInd^{(\mathsf{GA})}$)
17:
18: return [$cPInd^{(\mathsf{GA})}$, $cNInd^{(\mathsf{GA})}$]
```