UNIVERSITY OF CALIFORNIA, SAN DIEGO

Monte Carlo Simulation in Systems Biology

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Bioinformatics and Systems Biology

by

Jan Schellenberger

Committee in charge:

Professor Bernhard Ø. Palsson, Chair
Professor Vineet Bafna, Co-Chair
Professor Thomas Bewley
Professor Alexander Hoffmann
Professor Andrew McCulloch

2010

The dissertation of Jan Schellenberger is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California, San Diego

2010

DEDICATION

To mom and dad

EPIGRAPH

All models are wrong but some models are useful.

—George E. P. Box

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The last six years have been filled with ups and downs, mostly ups. I have grown both as a scientist and as a person. I would like to acknowledge the people who have made the last six years possible or at least more memorable.

My tenure with the Systems Biology Research Group began as an undergraduate. The person most responsible was Nathan Price who was my TA for BE122A (now BE123) and subsequently let me work for him. The initial experiences in the lab helped get me interested in grad school as well as teach me the ropes of the publication writing process. I owe a lot of gratitude to Nathan as my mentor.

Dr. Palsson was kind enough to take me on as a graduate student and mentored me for (almost) 6 years. He has an unbelievable ability to see the big picture which I have tried to learn from. While not physically in lab as much as other PI's, his presence is always felt. Even when on travels, he is just a quick email away from the sought after answer.

I have worked with many other SBRG members. They have been invaluable resources for solving difficult problems: Vasiliy Portnoy, Christian Barrett, Roger Chang, Harish Nagarajan, Josh Lehrman, Monica Mo, Tom Conrad, Nathan Lewis, Kenyon Applebee, Pep Charusanti, Daniel Hyduke, Jeff Orth, Ines Thiele, Ronan Fleming, Markus Herrgard, Karsten Zengler, BK Cho, Scott Becker, Yeungseob Park, Tim Allen, Jason Papin, Natalie Hurlen, Thuy Vo, Iman Famili. I will miss going to Hare Krishna lunches with you.

The support staff has been of great help in the lab. Lab managers Kathy and Eric took care of the expenses and made it possible for me to travel. Marc Abrams helped out with manuscript revisions.

I would also like to specifically acknowledge Neema Jamshidi. Although we don't have any joint publications, Neema has been a mentor and guide to matters both professional and personal. Having been at UCSD longer than even Dr. Palsson, Neema knows the ins and outs of UCSD, the department and San Diego better than anyone else. Neema - I owe you some Costco Animal Cookies.

I have had the privelege of taking on students of my own. To Ellen, Wing,

Jun, Sunthosh, Sorena, and Joseph - thank you for the opportunity to teach (and learn from!) a group of such talented individuals.

The other members of my committee (Dr. Bafna, Dr. Bewley, Dr. McCulloch and Dr. Hoffmann) were also very helpful in guiding me. Dr. Bewley in particular proved a valuable resource with all things related to numerical issues.

Research does not come cheap and I would like to acknowledge the groups and organizations that have funded me. My first two years were funded by the Bioinformatics program of UCSD and the NIH Ruth L. Kirschstein National Research Service Award GM00806-06 Bioinformatics training grant. Subsequently I was funded by two R01 NIH grants - The *E. coli* computational and Extreme Pathways/Sampling/Kinetics. My work in Japan was funded by the NSF EAPSI program and the Japanese counterpart at the JSPS.

My summer in Japan was a unique experience. I managed to see a different culture and work in a completely different environment. Baek-Seok Lee and Martin Robert at the Institute for Advanced Biosciences took care of me and arranged everything from housing to translating menu items. I am very grateful for their work.

During my graduate studies I have met many fascinating people who have made my life more interesting. Thank you to: the cooking club and it's participants for some good food and good times; our impromptu BMS chamber music group (Julie, Margaret, Navneeta, Diane and Emily) for good music; the rest of the Bioinformatics crew with whom I hung out and played games with; the former nerdhouse inhabitants with whom I'm still in touch; my roommates for putting up with my messes; the Torrey Pines Sailing Club crew for keeping the boats well maintained; Chris Cassidy of the MAE design studio and Dr. Delson for giving me the opportunity to work on the COSMOS program (and also use the lasercam).

Thank you to my parents for always supporting me and helping me along the way. You have always shown me what is right and wrong.

The text of Chapter One, with minor modification, is a reprint of the material as it appears in J. Schellenberger and B.Ø. Palsson. The use of randomized

sampling for analysis of metabolic networks, JBC Minireview. 284(9): 5457-61 (2009). I was the primary author of this publication and the co-author participated and supervised the research, which forms the basis for this chapter.

Chapter Two is unpublished work.

The text of Chapter Three, is work to be submitted to the Biotechnology Journal with authors J. Schellenberger, W. Choi, S. Madireddi, V. Portnoy, D. Scott, J. Reed, A. Ostermann, B.Ø. Palsson. I am the primary author on this study, and did most of the research.

The text of Chapter Four, in its entirety, is a reprint of material as it was submitted in J. Schellenberger, B.Ø. Palsson, Elimination of Thermodynamically Infeasible Loops in Steady State Metabolic Models. Biophysical J (2010 Under Review). I was the primary author of the text.

The text of Chapter Five, in its entirety, is a reprint of material as it appears in J. Schellenberger, Park, J. O., Conrad, T. C., and Palsson, B. Ø., BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions, BMC Bioinformatics, 11:213, (2010). I was the lead developer, and primary author of this manuscript.

The text of Chapter Six, is work to be submitted to Nature Protocols with authors J. Schellenberger, R. Que, Fleming, R., Thiele, I., Orth, J., Feist, A., Zielinski, D., Bordbar, A., Rahmanian, S., Kang, J. Hyduke, D., and B.Ø. Palsson. I was the primary researcher on this project and oversaw the final document for publication.

VITA AND PUBLICATIONS

2004	B.S., Bioengineering: Biotechnology, University of California, San Diego
2004	B.A., Mathematics-Computer Science, University of California, San Diego
2010	Ph.D., Bioinformatics and Systems Biology, University of California, San Diego

PUBLICATIONS

Price, N.D., **Schellenberger, J.**, Palsson, B.Ø., Uniform Sampling of Steady State Flux Spaces: Means to Design Experiments and to Interpret Enzymopathies, Biophysical Journal, 87:2172-2186 (2004)

Schellenberger, J., Palsson, B.Ø., The use of randomized sampling for analysis of metabolic networks, JBC Minireview. 284(9): 5457-61 (2009)

Feist, A.M., Zielinski, D.C., Orth, J.D., **Schellenberger, J.**, Herrgard, M.J., Palsson, B.Ø., Model-driven evaluation of the production potential for growth-coupled products of Escherichia coli, Metabolic Engineering, 12:3 173-186 (2010)

Schellenberger, J., Park, J. O., Conrad, T. C., and Palsson, B. Ø., BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions, BMC Bioinformatics, 11:213, (2010)

Thiele, I., Fleming, R.M.T., Bordbar, A., **Schellenberger, J.**, and Palsson, B.Ø., Functional Characterization of Alternate Optimal Solutions of Escherichia coli's Transcriptional and Translational Machinery, Biophys J 98(10):2072-2081 (2010)

Lewis, N., Schramm, G., **Schellenberger, J.**, Andersen, M., Cheng, J, Patel, N., Yee, A., Lewis, R., Eils, R., Knig, R., Palsson, B.Ø. "Formulating multi-cellular models of metabolism in tissues: application to energy metabolism in the human brain", (Nature Biotech under review) (2010)

Bordbar, A., Jamshidi, N., Lewis, N., **Schellenberger, J.**, Palsson, B.. "Characterizing the metabolic host-pathogen interactions of human alveolar macrophages and Mycobacterium tuberculosis during infection using the genome-scale reconstruction iAB-AMØ-1410-Mt-661, (Mol Sys Bio, under review) (2010)

Schellenberger, J., Palsson, B. Ø., Elimination of Thermodynamically Infeasible Loops in Steady State Metabolic Models, Biophysical J. Under Review (2010)

Schellenberger, J., Choi, W., Madireddi, S., Portnoy, V., Scott, D., Reed, J., Ostermann, A., Palsson, B. Ø., Optimizing the design of Carbon 13 labeling experiments for metabolic elucidation, In preparation (2010)

Schellenberger, J., Que, R., Fleming, R., Thiele, I., Orth, J., Feist, A., Zielinski, D., Bordbar, A., Rahmanian, S., Kang, J., Hyduke, D., Palsson, B.Ø., The COBRA Toolbox version 2.0, In preparation (2010)

ABSTRACT OF THE DISSERTATION

Monte Carlo Simulation in Systems Biology

by

Jan Schellenberger
Doctor of Philosophy in Bioinformatics and Systems Biology
University of California San Diego, 2010
Professor Bernhard Ø. Palsson, Chair
Professor Vineet Bafna, Co-chair

Constraint Based Reconstruction and Analysis (COBRA) is a framework within the field of Systems Biology which aims to understand cellular metabolism through the analysis of large scale metabolic models. These models are based on meticulously curated reconstructions of all chemical reactions in an organism. Instead of attempting to predict the exact state of the biological system, COBRA describes the physiological constraints that the system must satisfy and studies the range of solutions satisfying these constraints.

Monte Carlo Sampling is one of the COBRA methods used to study how biological properties are distributed over the entire solution space. A set of randomly distributed solutions is generated and serves as a proxy for the entire space. Various aspects of Monte Carlo Sampling in Systems Biology are illustrated: 1) Monte Carlo Sampling has been used historically (Chapter 1), 2) A faster and more efficient procedure for generating Monte Carlo Samples is developed (Chapter 2); 3) Carbon 13 tracing experiments are an important tool for measuring reaction rates through a network. Monte Carlo Sampling was used to optimize the choice of label and explain and measure the dimensionality of output data (Chapter 3); 4) It is possible to incorporate the thermodynamic "loop-law" into many COBRA methods including sampling (Chapter 4).

Additionally two software projects are presented which assist in analyzing

COBRA models: 1) the BiGG knowledgebase of reconstructions (Chapter 5) and 2) the COBRA Matlab toolbox v. 2.0 (Chapter 6). These two projects make COBRA methods available to the scientific community.

Chapter 1

The history of Monte Carlo Sampling in Systems Biology

1.1 Abstract

Genome-scale metabolic network reconstructions in microorganisms have been formulated and studied for about 8 years. The constraint-based approach has shown great promise in analyzing the systemic properties of these network reconstructions. Notably, constraint-based models have been used successfully to predict the phenotypic effects of knockouts and for metabolic engineering. The inherent uncertainty in both parameters and variables of large scale models is significant and is well suited to study by Monte Carlo sampling of the solutions space. These techniques have been applied extensively to the reaction rate (flux) space of networks with more recent work focusing on dynamic/kinetic properties. Monte Carlo sampling as an analysis tool has many advantages including: the ability to work with missing data, the ability to apply post processing techniques, ability to quantify uncertainty and optimize experiments to reduce uncertainty. We present an overview of this emerging area of research in systems biology.

Table 1.1: Different variables and constraints used in metabolic analysis
For metabolic analysis, the three major variables are fluxes, concentrations and kinetic parameters. Each of these may be constrained by physiological constraints and experimental measurements. Most research has focused on the flux rates although recently there has been an interest in the kinetic/thermodynamic aspect as well.

Variables	**Constraints**	**Experimental Measurements**
Fluxes	mass balance v_{max}	fluxomic isotope labeling secretion rate profiling
concentration	charge balance osmotic balance volume constraints thermodynamics	metabolomics
kinetic parameters	known dynamic behavior thermodynamics regulatory rules	in vitro assays expression profiling proteomics

1.2 Introduction

The advent of whole genome sequencing has led to the curation of many genome-scale metabolic reconstructions [1, 2]. These reconstructions are mathematically structured knowledge bases containing Biochemical, Genetic and Genomic (BiGG) information about a metabolic network. Whereas the content of these network reconstructions can be fairly complete, the functional (i.e. physiological) states that these networks can achieve are more difficult to determine and such determination is an active area of research. The analysis of whole cell metabolism comes with unknown quantities. Even a description of a steady state condition may require knowledge of a large number of metabolite concentrations and reaction rates (fluxes). Dynamical analysis adds additional complexity.

Although significant effort has gone towards measuring each of these quantities through various high-throughput omics methods, obtaining all the needed numerical values is a significant challenge that may not be met in full for a long time. This incompleteness of data has created the need for analysis methods that

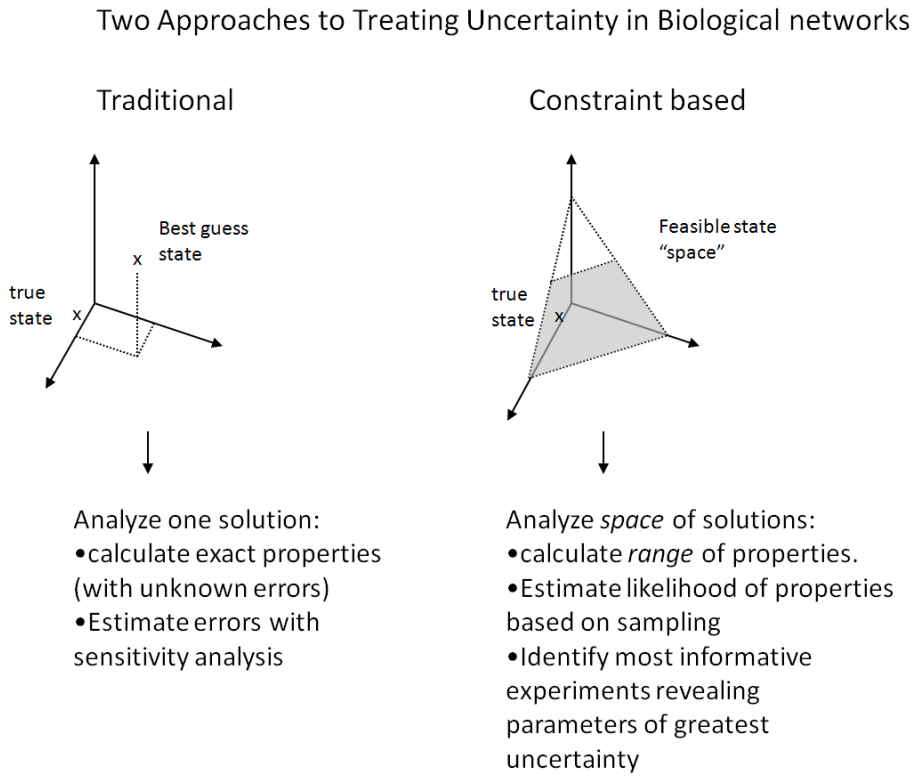


Figure 1.1: Traditional versus Constraint based methods

Traditional analysis methods focus on one solution, which approximates the true biological state as closely as possible. The error in this solution is often unknown although techniques such as sensitivity analysis can give an idea as to its magnitude. In contrast, constraint based analysis does not aim to predict the true biological state but attempts to describe the space within which the solution must lie. The properties of this space as a whole must contain the properties of the true solution.

are able to give meaningful results with only partial measurements. One approach has been successfully used is so called Constraint Based Reconstruction and Analysis (COBRA). This approach emphasizes describing the constraints that a system must satisfy rather than computing an explicit solution. Some of the variables and constraints that have been applied in the past are listed in Table 1.1. This approach leads to the definition of a "solution space," that contains the set of feasible solutions that satisfy all imposed constraints. Figure 1.1 shows a schematic of this approach and how it compares to the traditional simulation approach. If the system equations are set up correctly, the "true state" of the network will lie within the imposed constraints and may then be further analyzed.

Many approaches exist for studying this solution space [3, 4]. One of the more recent approaches is randomized sampling of candidate solutions. In order to study the space of solutions, a random set of points is chosen from it to act as a surrogate for the entire space. Many of the properties that can be calculated for one candidate solution can then be calculated for point throughout the entire space and the properties of this set of solutions can be evaluated in a statistical fashion. This procedure gives information about the how limiting the imposed constraints are, and the results can be used to design further experiments to shrink the size of the solution space.

The workflow associated with the constraint-based paradigm is outlined in Figure 1.2. The scope of the biological system is defined as variables (fluxes, concentrations, pressure, etc.) and parameters (kinetic constants, thermodynamic values, etc.). Experimental measurements, biophysical constraints and other known constraints are imposed on the system yielding a solutions space. It can then be probed with a variety of methods including optimization and Monte Carlo Sampling. The results from such studies are used to design further experiments to further constrain the system.

The ultimate goal of large scale network analysis is to provide a framework for understanding whole cell metabolism. This includes interpreting data from various high throughput omics experiments, creating predictive models of how the cell works, and ultimately being able to manipulate cells for medical or industrial

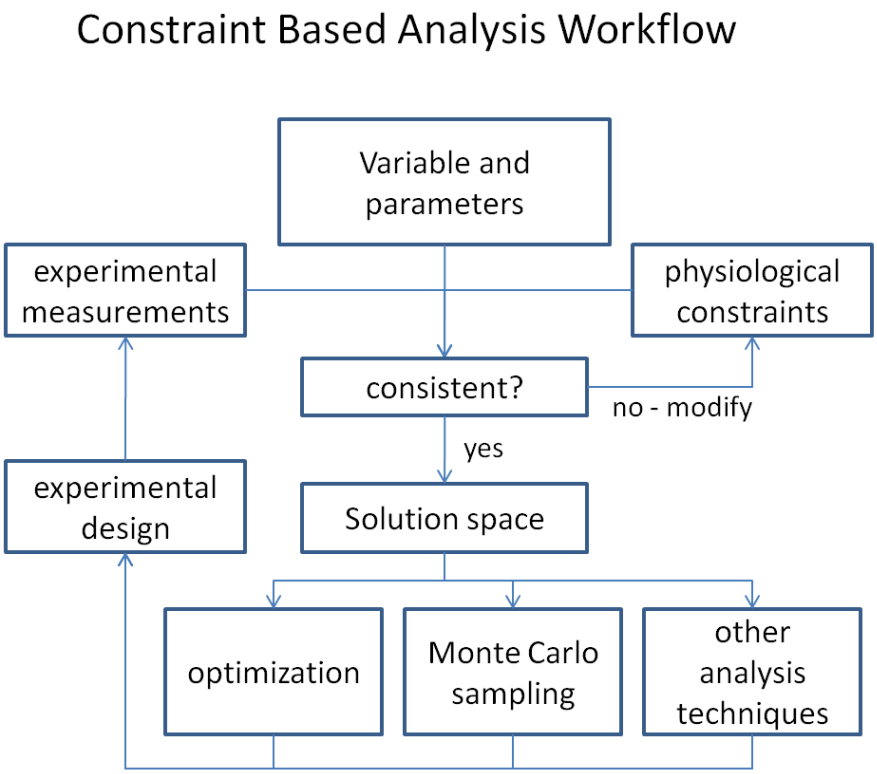


Figure 1.2: Constraint based analysis work flow

The scope of the model is determined by variables and parameters. These are constrained with 1) experimental data and 2) physiological constraints. The solution space is the intersection of all constraints. If this is empty, then the constraints and experimental data are inconsistent and must be modified. If however they are consistent, the solution space may be probed by a variety of methods. The solution space may be shrunk with further experimental data with the current solution space aiding with the experimental design.

purposes.

1.2.1 Steady state flux balance analysis

A metabolic network can be concisely described in matrix format using the stoichiometric matrix, S (sometimes called N). Every row in this matrix represents one metabolite and each column represents a reactions. A non-zero entry in $s_{i,j}$ indicates participation of a metabolite i in a reaction j. The rates of every reaction (flux) can be written as a vector v forming the fundamental mass balance equations:

$$dx/dt = S \cdot v(x) \tag{1.1}$$

Integrating this equation over time yields the time course of concentrations $x(t)$. In general, the rates of reaction, v are a function of the concentrations, x, as well as enzyme kinetic constants and other parameters. Since it is hard to measure all the kinetic constants needed to simulate dynamic responses, the steady states of the system are often studied. Steady state implies $dx/dt = 0$ and therefore $S \cdot v = 0$. A different way of stating this is that the (right) null space of S contains the steady state flux distributions. This solutions space is finite in size given enzyme capacity constraints ($v_{min} < v < v_{max}$) has it has been studied extensively and its properties can be studied with a variety of approaches [5]. Recently it has become realized that randomized sampling of solutions in the solutions space is an effective way to characterize its contents.

1.3 Example uses of sampling in small spaces

A number of questions about solution spaces can be addressed using randomized sampling. Three notable examples of studies of small solutions spaces are now described.

1. Designing experiments: The existence of a solution space means that many conceivable flux distributions satisfy the steady state condition. Even with all uptake and secretion rates (inputs and outputs) measured, the internal

flux rates are usually not uniquely specified. If there are two parallel pathways for the same process it will be impossible to distinguish which is being used. This non-uniqueness of flux states was realized early [6], and thus additional measurements are needed to further eliminate candidate solutions. Sampling of the candidate solutions can be used to find the most informative measurements to make [7, 8] . Many possible experiments were simulated and Monte Carlo sampling was used to simulate random experimental noise and propagate it through the network. The ratio of measurement noise to computed noise was statistically quantified thus rejecting experiments with poor design in favor of more informative ones.

2. Determining the shape of solution spaces: Smaller networks (<40 reactions) have flux solution spaces which can be studied directly using techniques from convex analysis. Wiback et. al. defined the flux space of the core Red Blood Cell model and computed its volume using a mathematical techniques called vertex enumeration [9]. A Monte Carlo elimination sampling was used to describe the shape of the space by plotting the distribution of points as a function of flux through each reaction. The shape flux space contains important information about the likelihood of finding the true solution at any particular flux value. A narrow space indicates low likelihood whereas a wide space indicates greater likelihood. An updated method for computing the volume of the space can be found in [10].

3. Consequences of genetic variation: A refined sampling approach was developed by Price et. al [11] which scaled to slightly larger networks. Enzymopathies of the Red Blood Cell were studied by decreasing the V_{max} of the reaction catalyzed by enzymes with known biochemical deficiencies and clinically observable altered phenotypes. It was shown that the enzymopathies that decreased the volume of the flux space most significantly were more likely to have a clinical effect in vivo. This paper first considered looking at the correlations between points as another way of describing the shape of the space and noted that these values tend to shift while simulating enzy-

mopathies.

1.4 Determining global network properties

With the availability of genome-scale network reconstructions there has been significant interested in characterizing them in an unbiased fashion. Two unbiased approaches have emerged [5]. One approach has been the development of network based pathways as convex basis vectors; such as Extreme Pathways [12] and Elementary modes [13]. Monte Carlo Sampling provides an alternate unbiased way. Both have significant numerical challenges at the genome-scale, with the determination of convex pathways being potentially impractical [14], whereas randomized sampling can be achieved at this scale.

1. Sampling methods. With the advent of Monte Carlo methods for the study of the flux space of much larger networks, the dominant algorithm of choice has been The Markov Chain Monte Carlo (MCMC), also known as hit and run sampler. Unlike the elimination algorithm usable for smaller networks, the MCMC algorithm produces a valid solution point at every iteration. An initial valid point is moved repeatedly inside the space according to probabilistic rules. The trail of valid generated points becomes the sample. One key disadvantage of this algorithm is that there is no guarantee that these points cover the entire space in a finite time. This behavior is known as slow mixing. One improvement to the MCMC algorithm is artificial centering (ACHR) [15]. Essentially all publications sampling of the flux space have used this algorithm.

2. Elucidation of a high-flux-backbone: Barabasi and colleagues published two papers that used Monte Carlo Sampling of the flux space to look at global network properties. [16] demonstrates that E. coli flux space is dominated by a few high capacity reactions (the high flux backbone) that is robust between different simulated media conditions. [17] compared the metabolic reconstructions of *H. pylori*, *E. coli* and *S. cerevisiae* under randomly gener-

ated media conditions and used Flux Balance Analysis to compute the flux shifts between them. An activity core was defined based on reactions that are always required for growth. This activity core varies in size between the three reconstructions and is a reflection of the redundancy and robustness built into these systems.

3. Definition of modules: modules can be determined based on correlations between reactions. When two reactions are correlated either perfectly ($r^2 = 1$) or nearly perfectly ($r^2 \sim 1$), the flux through them must be linearly related at steady state. An example of this is two reactions in a linear pathway. This concept was reviewed by Papin et al. [18] as one technique of network classification and applied to the *M. tuberculosis* network [19]. While correlated sets based on extreme pathways and uniform random samples are distinct, a comparison showed that the global network properties to be conserved [20].

1.5 Applications of sampling to study disease states

Whereas global network properties are of academic interest, sampling has also been used to study clinical issues.

1. Human Mitochondria: The human cardiac mitochondria network [21] was analyzed using the sampling approach to characterize network capabilities under different conditions, including various diets and simulating diabetic conditions. A particularly striking result was the observation that the pyruvate dehydrogenase (PDH) flux in diabetic patients is constrained to be lower than non-diabetic patients due to mass conservation constraints alone. This result was surprising, because, although it had been known that PDH has a decreased flux in diabetics, it was believed to be a consequence of unknown regulatory mechanisms. The sampling approach thus demonstrated how bottom-up reconstructions can describe real biochemical and physio-

logical conditions and provide mechanistic insights into the cause and effect relationships.

2. Co-sets and their applications: Another result of this study came from the use of comprehensively sampled points in the flux space to calculate statistically perfectly correlated sets of reactions, termed co-sets [18]. A non-zero flux in one member of a co-set implies a non-zero flux in all other members of the set and vice-versa. Hence co-sets define reactions that are used together as a result of mass-conservation determined by network topology. These reaction co-sets can be used to simplify the network and to enable analysis of disease states and alternative drug targets in terms of causal pathways rather than individual reactions. They can also be used to correlate the causality of SNPs that appear in the enzymes participating in a co-set [22].

3. Human Neural Reconstruction: Occhipinti et al. [23] studied a 5 compartment model of human neuron/astrocyte metabolism with Monte Carlo Sampling and Statistical inference. The method was based on a previous paper [24] and assumes a prior distribution on each flux in the network. Experimental measurements are treated as observations which are used to update the distributions of the fluxes. In this way, it is possible to test specific experimental hypotheses *in silico*. The study asked which metabolic pathways are more active in brain metabolism at steady states with different neural activities and was able to provide quantitative answers.

1.6 Extensions to analysis of dynamic states

Moving beyond a steady state framework to dynamic analysis presents a new set of challenges. Kinetic parameters needed for dynamic analysis are numerous and often unknown. Often *in vitro* techniques fail to give numerical values for kinetic parameters that are consistent with those *in vivo* [25]. Obtaining such a consistent set is expected to require whole systems approaches.

There is a strong need for developing methods that allow for global kinetic

analysis since many of the important systemic properties lie in the dynamic domain. Coupled with the recent development of metabolomic measurements we may be able bridge the gap between steady state and fully dynamic descriptions. Because of the inherent uncertainty in measurements, Monte Carlo Sampling can used to further this aim:

1. K-cone analysis: Famili et al. proposed defining a space of feasibility within which kinetic parameters must be found [26]. This was termed the k-cone. It is analogous to the flux space in that every point represents a feasible set of kinetic parameters. This construction assumes that the concentrations of metabolites are known and that the system is at steady state. Famili was able to show that; 1) Feasible kinetic parameters exist; 2) It is possible to find the set of kinetic parameters that most closely match a measured set (by optimization); and 3) by repeating this procedure under different conditions, the set of kinetic parameters can be narrowed down.

2. Structural Kinetic Modeling: Steuer et al. proposed a different way of looking at whole system kinetics [27, 28, 29]. Structural kinetic modeling makes no assumptions about kinetic parameters and instead focuses on the feedback parameters in a network. Equation 1.1 is linearized about a hypothesized steady state value xss with a steady state flux of v_{ss}. The linearization results in a Jacobian matrix, J, which contains dynamical properties of the system. For example, the eigenvalues of J indicate whether the system is stable or unstable near the steady state. Many of the feedback parameters to build J are unknown. To study their effects Monte Carlo sampling was used and statistical properties were determined (Grimbs, Selbig et al. 2007. A ranking of a feedback sites stability was obtained.

3. Bayesian statistic approaches: Liebermeister and Klipp used Bayesian statistics to describe the distribution of kinetic parameters [30]. Starting with a stoichiometric model, various experimental data types (metabolic concentrations, thermodynamic, kinetic) are integrated in a statistical fashion to produce a set of kinetic models. As a test example, a small threonine metabolism

model was used to simulate noisy experimental measurements. The Bayesian approach was able to narrow the possible range of kinetic parameters from a very large range initially to a much narrower feasible range.

1.7 Challenges for the future

The challenges in this area are several. The main categories are as follows:

1. Computational: Sampling a solution space is difficult as the number of variables increases. This is known as the curse of dimensionality. Various tricks have been employed to get around this limitation (for example ACHR) and this is an area of active research. The flux space and K-cone formalism have the advantage that the commonly used constraints form a convex space, which has some "nice" properties. Writing a sampling procedure capable of sampling a generic set of constraints may be quite difficult. Instead a careful tradeoff between feasibility of sampling and expressiveness of the formalism and constraints is required. The Cobra toolbox [31] contains tools for Monte Carlo Sampling as well as other COBRA techniques.

2. Extend formalism to other parameters: As shown, Monte Carlo sampling has been used to sample the flux space as well as the kinetic space. However, it is conceivable to extend the formalism to many other areas. Recently there has been attempts to describe regulatory networks in a stoichiometric fashion [32]. The formalism uses an R (for regulatory) matrix which is analogous to the S matrix. An R matrix has been constructed, based on the model used in Covert, et al. [33], and interrogated using uniform random sampling. The simulation results exhibited strong concordance for two environments for which microarray experiments have been conducted. In addition, an analysis of the row and column spaces demonstrated that these spaces describe all possible gene expression states for *E. coli* in a given environment [34].

3. Thermodynamic constraints: Kümmel et al. used thermodynamic considerations to describe a set of constraints on the concentrations of metabo-

lites [35]. With the increasing availability of metabolomics data as well as computational estimates of thermodynamic properties, constraints of this type are sure to become more prevalent. Table 1.1 illustrates the variables, parameters and constraints that have been defined for metabolic networks. In all cases, it is not possible to obtain all measurements for a full description with current high-throughput techniques. There are however constraints between variables which suggest that not every variable must be explicitly measured. For some of these applications, Monte Carlo sampling has been applied (Fluxes, and Kinetics). For others this has not been done yet.

One simplification to full thermodynamic considerations is the inclusion of the so-called "loop-law". This law places additional constraints on internal metabolic fluxes to avoid circular flux distributions which are thermodynamically infeasible. Price et al. studied this feasible space by sampling the space of a *H. pylori* reconstruction and eliminating points which violate the loop law [5]. While this method was shown not to scale to larger networks it has the advantage of not requiring knowledge of thermodynamic constants.

1.8 Conclusions

Monte Carlo sampling is emerging as an approach to deal with the analysis of genome-scale metabolic networks. Ideally we would like to know the true internal state of a cell at all times. As this is goal currently intractable, much effort has been focused on giving an accurate description of the solution space within which the true state must lie. Unbiased analysis of this space through randomized has yielded many novel results and provides a framework by which further experiments can be designed.

The text of Chapter One, with some modification, is a reprint of the material as it appears in J. Schellenberger and B.Ø. Palsson. The use of randomized

sampling for analysis of metabolic networks, JBC Minireview. 284(9): 5457-61 (2009). I was the primary author of this publication and the co-author participated and supervised the research, which forms the basis for this chapter.

Chapter 2

Properties Of The GP-ACHR Sampler

2.1 Abstract

Monte Carlo Sampling of convex spaces has been used in metabolic systems biology to explore the shape and dimensions of the space of possible flux distributions. In this chapter we consider two algorithms that have been used for sampling and propose a simple extension to one which provides a significant improvement in computational time and result accuracy.

2.2 Introduction

The theory of Monte Carlo Sampling is well established and is not the focus of this thesis. It is, however, important to point out some of the recent work on the ACHR Sampler as it is part of the COBRA Toolbox (Chapter 6). While conceivably many solution spaces may be defined in systems biology, early work has focused on the flux (v) space which describes the set of possible flux distributions. The space is defined by the bounds:

$$S \cdot v = 0$$

$$lb \leq v \leq ub$$

One important property of this space is that it is convex. The definition of convexity is that for any two points, x and y in a space, the line segment between them ($c \cdot x + (1-c) \cdot y$ for $c \in \{0, 1\}$) is also in the space. This property is directly used in the algorithm for ACHR sampling.

2.2.1 Elimination Sampling

As mentioned in Chapter 1, the first attempts at uniform sampling were based on elimination sampling. The idea is quite intuitive and basically states that in order to sample a complicated space, sample a larger space containing your target space and remove points not in your target space. For example, in order to sample a disk, it is possible to sample a circumscribing square and discard ('eliminate') the points not in the disk. In this case, the so called 'hit fraction' is $\pi/4$. This is an upper bound, though. Choosing any other enclosing rectangle will result in a lower hit fraction and more points are required in order to achieve a fixed number of final points. An illustrated example is shown in Figure 2.1. In practice, elimination sampling works very well for low dimensional spaces however it tends to fail as the dimension increases. This is known as the 'curse of dimensionality' and the reason can be illustrated by the example of the disk in a square. Extending this example to n dimensions, the volume of an n-dimensional unit sphere is $\frac{\pi^(n/2)}{\Gamma(n/2+1)}$[1]. The volume of an enclosing cube is 2^n. The hit fraction is the ratio of these two which for even n looks like:

$$\text{hit fraction} = \left(\frac{\pi}{4}\right)^{(n/2)} \frac{1}{(n/2)!}$$

This ratio goes to 0 rapidly and for $n = 30$ it is already $2 \cdot 10^{-14}$. In order to generate one randomly sampled point from a 30-sphere by elimination sampling requires on the order of 10^{14} samples to be drawn from an enclosing hyper-cube. Figure 2.2 shows this trend. The sphere case is a worst-case analysis if the enclosing shape is a hyper-rectangle and in practice it is easy to sample high dimensional spheres by other means. In Systems Biology is is possible sample higher dimensional spaces through elimination sampling by using parallelpipeds as the enclosing space.

[1] $\Gamma(x)$ is the Gamma function. It behaves similarly to factorial.

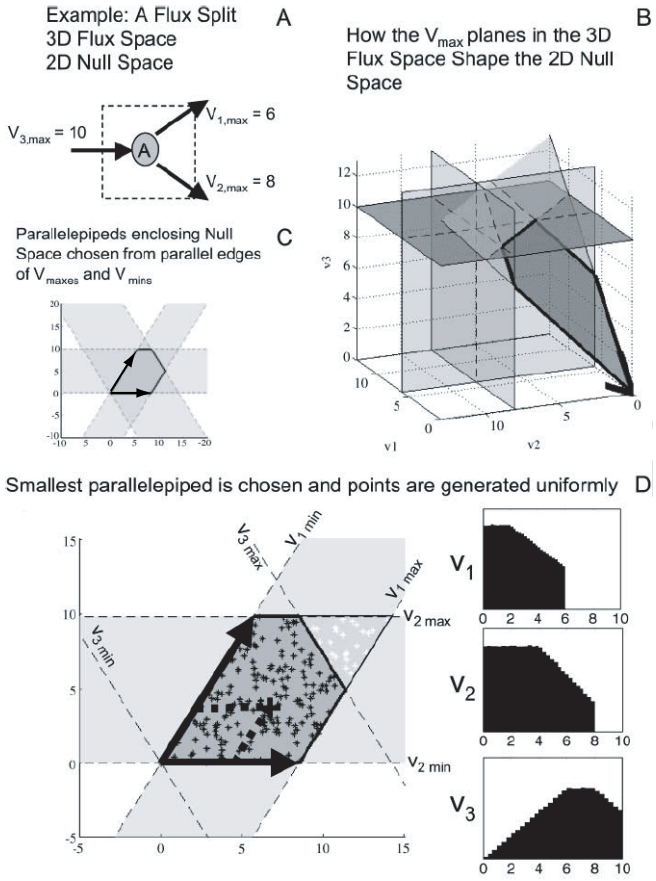


Figure 2.1: Elimination Sampling

A simple network is used to illustrate the method of elimination sampling. (A) a simple network consisting of just one metabolite and three reactions. Plotting the resulting steady state solutions in 3-D yields the bounds shown in (B). The steady state portion of this space is shown in (C) and is a pentagon bounded on all sides by v_{min} and v_{max}. (D) shows one of the possible parallelpipeds enclosing the space. In this case $v_{max,3}$ is temporarily ignored and v_3 becomes a dependent variable on v_1 and v_2. Points can now be randomly sampled by choosing v_1 and v_2 independently. If the resulting v_3 is greater than $v_{max,3}$ then this point must be eliminated. (D) also shows the resulting histograms for all three reactions. Figure previously published in [4]

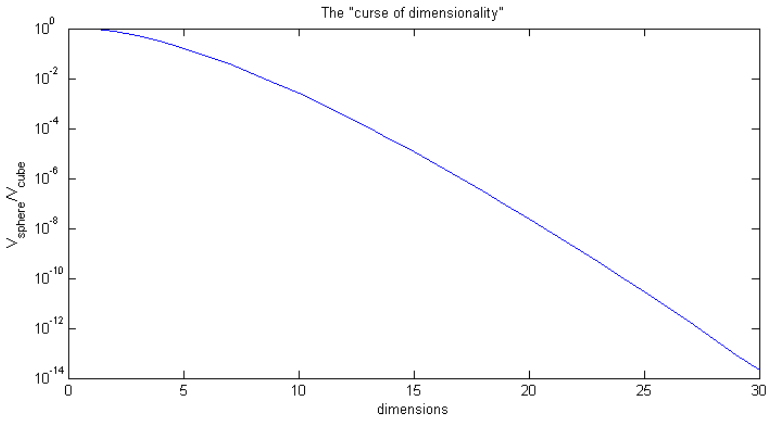


Figure 2.2: The Curse of Dimensionality

Plot shows the ratio of volumes of an n-dimensional sphere to an n-dimensional hypercube of side length 2. The exact formula of the sphere is $\frac{\pi^(n/2)}{\Gamma(n/2+1)}$. As a function of n, the decay of this ratio is sub-exponential.

Parallelpipeds are the high dimensional extension of parallelograms and they match the shape of the solution quite well such that the space is a parallelpiped with corners missing. Nonetheless, it was shown that the hit fraction gets very small for medium size networks and elimination sampling is not feasible.

2.2.2 Hit and Run Sampling

The alternative method to elimination sampling is hit-and-run sampling. This method eliminates the problem of low hit fractions by only generating points inside the space. The rough outline of the procedure is as follows:

1. Choose any point, x_0 inside your space[2].

2. At each iteration, choose a random direction, c.

3. Compute the limits of a line through x_0 along direction c. In other words, compute the limits of α (α_{min}, α_{max}) so that the point $x_{k+1} = x_k + \alpha c$ is still inside the space.

[2]In the case of the flux space, this can be done by linear programming with a random objective

4. Randomly choose an α between α_{min} and α_{max} and choose $x_{k+1} = x_k + \alpha c$.

This procedure generates a series of points $x_0, x_1, ..., x_k$ which form a uniform random sample of the space as $k \longrightarrow \infty$.

While Hit and Run (HR) sampling does not have the low hit fraction problem, it suffers a problem known as 'poor mixing'. This is because each point is dependent on the previous point. The number of steps required to achieve 'mixing', where point x_k and point $x_k + n$ are independent can be huge. To illustrate this problem, consider the two dimensional space defined by $|x_1| \leq 1$ and $|x_2| < a$ for a parameter $a << 1$. Any point in this space will be very near the boundary $x_2 = \pm a$ and a line in a random direction will hit these boundaries in a distance that is on the order of a. Then the series of sample points in this space can be thought of as a random walk along one dimension, with step size on the order of a. After n steps, a point will be expected to move a distance that is on the order of $a\sqrt{n}$. So if $a \sim 10^{-3}$ then it would take on the order of 10^6 steps for a point to move from one part of the space to another.

Generalizing a bit from this two dimensional case, Hit and Run sampling shows poor mixing behavior if the sampled space is scaled poorly - i.e. it is much 'longer' in one dimension than in another. However this is precisely the property seen in biological spaces where dimensions are fluxes. It is know that fluxes follow a power law distribution [16] with low-flux reactions having rates several orders of magnitude lower than high-flux reactions.

Fortunately, an enhancement to the Hit and Run Sampler was developed to help with this problem [36]. The main idea is to choose a direction, c, not uniformly randomly but in such a way as to maximize the step size and mix more rapidly. The method suggested by Kaufmann et. al uses the history of points $x_0...x_k$ to as a basis for choosing the direction. First the midpoint is computed $\bar{x} = \sum_{i=1}^{k} x_i$ and then c is chosen $c = x_i - \bar{x}$ for a random i. In other words, the direction is chosen as the difference between a random previously visited point and the center point. Why does this work? It assumes that the previously visited points already have the shape of the space and therefore a random point will tend to be farther away from the center precisely in the dimensions where the space is longest.

A slight modification is necessary in the start procedure as it is impossible to choose a direction until a sizeable set (at least the size of the dimension of the sampled space) has already been drawn. The solution is to generate some (minimum = dimension of space) so called 'warmup' points by other methods and then employ the ACHR sampling technique subsequently. Generally, the warmup points are generated the same way x_0 is generated - by Linear Programming with a random objective (see section 2.3.4).

2.3 Topics of Interest

There are several things that must be considered when running the ACHR sampler. Here we introduce the generalized and parallel ACHR (GP-ACHR) sampler.

2.3.1 Generalizing the ACHR Sampler

The initial ACHR Sampler as used for metabolic Systems Biology could only sample spaces running through the origin. The space description was:

$$S \cdot v = 0$$

$$v_{min} < v < v_{max}$$

In general though, we may be interested in sampling 1) an affine space (one which does not go through the origin) and 2) a space bounded by linear inequalities ($A \cdot v < b$). Together these two conditions can be written as the general:

$$A \cdot x \leq = \geq b$$

$$v_{min} < v < v_{max}$$

where $\leq = \geq$ is a vector of $\leq$, $=$ or $\geq$.

Sampling an affine space

As will be shown later, it is beneficial to have all the equality constraints on the space be equal to 0. This indicates that the space is a subspace of R^n going through the origin and it has some properties which make sampling easier. A space not containing the origin may be sampled by applying a linear transformation of variables. Consider the part of the constraints containing equalities: $A_{eq} \cdot x = b$. Find any solution x_0 to this system such that $A_{eq} \cdot x_0 = b$. Now transform variables to y such that $y = x - x_0$ or $x = y + x_0$.

We now have that $A_{eq} \cdot (y + x_0) = b$.

$$A_{eq} \cdot y + A_{eq} \cdot x_0 = b$$

$$A_{eq} \cdot y + b = b$$

$$A_{eq} \cdot y = 0$$

We can therefore sample this homogeneous space (in y) and transform to x later. From now on we can assume that this trick was performed and that without loss of generality, $A \cdot x \leq = \geq 0$.

Sampling with Inequality Constraints

Adding inequality constraints is actually not a great challenge either. Without loss of generality it can be assumed that all inequalities are of the form $A \leq 0$ as all constraints of the form $A \geq 0$ can be reversed by multiplying both sides by -1.

The ACHR algorithm must compute α_{min} and α_{max} at each step. For a direction c and a point x_0, this requires solving the equations:

$$\text{lb} \leq x_0 + \alpha \cdot c \leq \text{ub}$$

This vector inequality is easily solved one component at a time.

The equality constraint $A \cdot x = 0$ is automatically satisfied because x_0 satisfies it as well as the direction c.

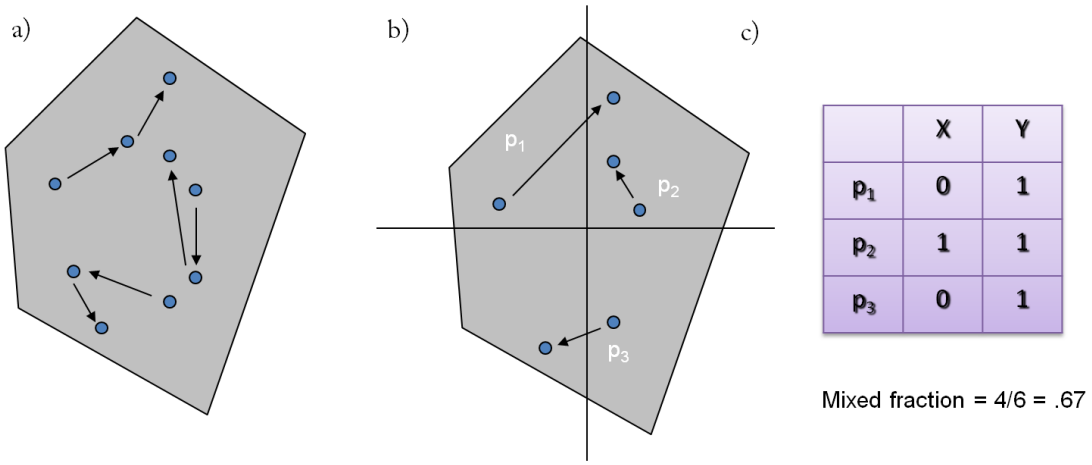


Figure 2.3: Parallel Sampler and the Mixed Fraction
Several points are moved throughout the space of interest in parallel (a). The mixed fraction is computed as follows: Axes are drawn along all principle directions (b) and a count is tabulated of which points cross which boundaries (c). A '0' indicates that a point crossed a certain axis and a '1' indicates the point is still on the same side. The mean of all these entries is the mixed fraction. It is 1 initially ($x_0 = x_f$) and tends towards .5 as the perfect mixing is achieved.

In order to add inequality constraints, an additional constraint is placed on α.

$$x_0 + \alpha \cdot c \leq b$$

This too can be solved one component at a time.

$$\alpha \begin{matrix} \leq (b - x_0)/c_i \text{ for } c_i > 0 \\ \geq (b - x_0)/c_i \text{ for } c_i < 0 \end{matrix}$$

The minimum feasible range of α is computed based on these constraints.

2.3.2 Parallelization

The 'P' in 'GP' sampler stands for parallelization. The basic premise is instead of moving one point throughout the space, move a large number of points

in parallel. Figure 2.3 illustrates this idea. There are several reasons why parallelization is an improvement:

1. Speed - When points move in parallel it is possible to move them in parallel computationally. Most desktop computers and workstations are now multicore capable and our tests have shown that the running in parallel results in a speedup of $\sim 70\%$ what would be expected with perfect scaling (results not shown).

2. Constant Memory - When performing conventional ACHR, the number of points is always growing. At some point it exhausts the allocated memory and then has to be written to a disk. If the ACHR algorithm has not been run long enough for sufficient mixing then performing additional mixing will necessarily generate more points. Very often the application requires a fixed number of points that is known in advance. The parallel sampler can keep the points constant while performing additional mixing. As long as the number of points can be kept in main memory, no file accesses are required.

3. Independence of points - The ACHR sampler generates a sequence of points which form a uniform random sample of the space. However the points themselves are not independent from one another. This may not be an issue in all cases but often having identical independently distrubted (iid) points is desirable. The parallel sampler inherently solves this problem because each point starts independently.

4. Termination Condition - Measuring when sufficient mixing has occured is difficult. With the new parallel sampler I propose a parameter called the 'mixed fraction' (detailed below) which measures the degree of mixing. While it is not a sufficient condition to guarantee mixing, it is necessary and in practice seems to work quite well.

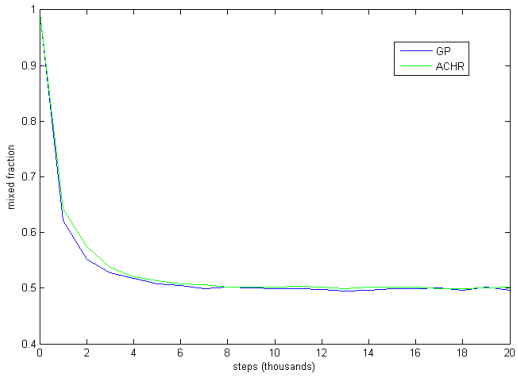


Figure 2.4: ACHR vs GP-Sampler

The hit fraction is plotted as a function of number of steps. The GP-Sampler is indicated by the blue line and the ACHR sampler the green line. After 20,000 steps, both samplers have reached a mixed fraction of .5. The trajectory of the GP sampler is faster however.

The Mixed Fraction Parameter

The mixed fraction (mf) measures the fraction of points that have crossed the median in any direction. Figure 2.3c illustrates how this is done. The sampled space is partitioned by a linear boundary. This boundary is chosen such that half of the points are on each side. After a certain number of steps, the partition is re-created. For each point it is tabulated whether it is on the same or opposite side of the partition. If the points have been perfectly mixed and there is no dependence between the initial position and final position, then the chance of crossing the partition is exactly 50%. Therefore taking an average of all points would result in a mixed fraction of .5 when mixing is achieved. Initially, before points have moved at all, the mixed fraction is exactly 1 and the mixed fraction would be expected to decrease exponentially towards .5.

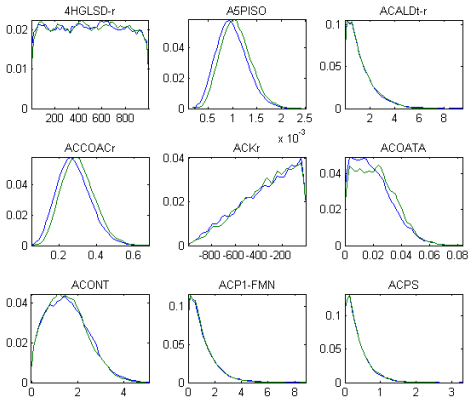


Figure 2.5: ACHR vs GP-Sampler distributions
The ACHR sampler and GP-ACHR sampler are compared for the reduced *H. pylori* iIT341 model. Both samplers started with the same warmup points. In both cases, after 20,000 points, the distributions look identical

Comparing Parallel ACHR and Parallel ACHR

To compare the ACHR versus GP-Sampler, a network was sampled using both methods. The network used was the iIT341 model of *H. pylori*[37]. In both cases the model was first reduced by eliminating extraneous reactions that carry 0 flux (reduceModel.m in the COBRA Toolbox). To create a fair comparison, both the ACHR and GP-Sampler were run for an identical number of steps. 5000 points were generated with 20000 steps between points (ACHR) or 20000 steps total (parallel ACHR).

Results are shown in Figure 2.4. After 20,000 steps, both the GP-Sampler and the ACHR sampler had a mixed fraction of .5. The GP sampler however reaches this value faster than the ACHR sampler. For larger models, the effect is more noticable (results not shown). Figure 2.5 shows that both samplers produce identical distributions (positive control).

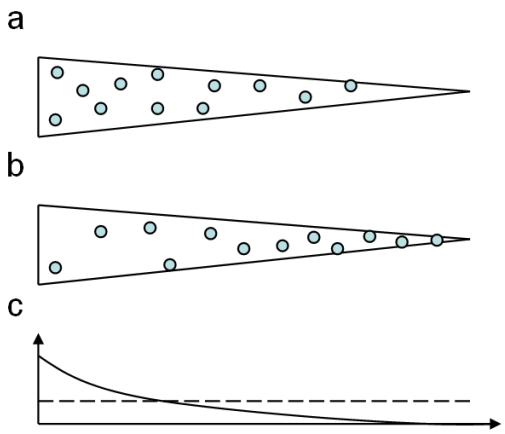


Figure 2.6: Biased Sampling

Sampling a skewed polytope with unbiased (a) and biased (b) sampling. Sampling in a uniform fashion produces a marginal distribution with infrequent sampling of the tail region to the right (solid line c).In this case the sampling bias makes the marginal distribution uniform over the feasible flux range (dotted line in c) even though the polytope is not sampled uniformly.

2.3.3 Biased Sampling

The current 'hit and run' sampling algorithm [36] is designed to uniformly sample a convex space. Applied to the steady state flux space, *in silico* uniform sampling assumes that an *in vivo* flux has equal probability of residing in all regions of the space. However sometimes it maybe desirable to sample a space in a way that is biased. Figure 2.6 shows an example where the marginal distribution over one particular reaction is kept uniform. An unbiased sampling tends to move all points to just one region of the solution space.

While the general solution to sampling with arbitrary marginal distributions is quite difficult, there is a rather simple trick with the GP-ACHR sampler that allows specifying one arbitrary marginal distribution. First, points are chosen from the marginal distribution of the reaction of choice. The warmup points must then be chosen to already match this distribution. With the conventional linear programming approach this is not difficult. Finally, the only modification required to the GP-ACHR algorithm is that the direction c that each point travels in is modified such that $c_i = 0$ where i is the direction of the imposed marginal distribution. This small modification ensures that once the marginal distribution is set in the warmup points, it never changes. Note that this trick is not possible with the conventional ACHR algorithm as fixing a direction of motion would not allow the points to cover the entire space. Also note that this trick cannot be applied when more than one marginal distribution is imposed as the the joint distributions must be fixed during the warmup point generation but are not known.

2.3.4 Warmup Point Generation

The ACHR algorithm requires a set of warmup points to initialize the algorithm. The method used to generate these points has a important effect on the mixing time on the algorithm. Ideally, of course, the warmup point generator would generate points which are already uniformly distributed within the desired space. If this were possible, mixing would not be necessary at all. However, having points in different parts of the space is still desirable. Another desirable (in fact

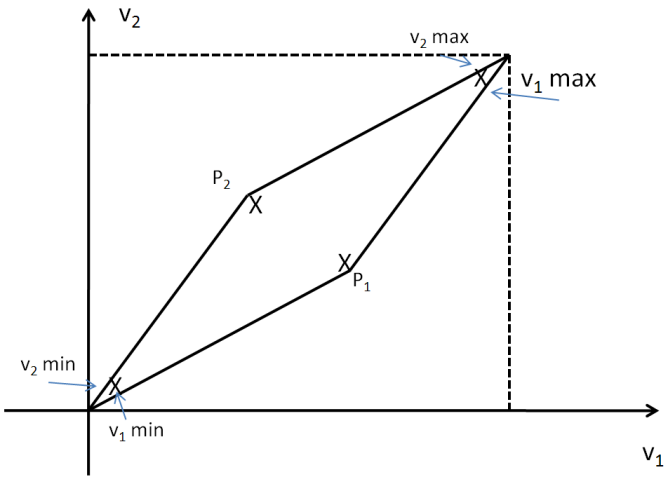


Figure 2.7: Problems with Warmup Point Stategies

Two wramup point strategies are compared. The **orthonormal strategy** minimizes/maximizes each reaction in turn. In this case there are two reactions (dimensions) however the maximum/minimum coincide for both reactions, yielding only two unique points. The **random strategy** maximizes and minimizes in a random direction. Because of the shape of the space, most points will again end up in the upper and lower corners. Only a small fraction of points will end up at P_1 and P_2.

necessary) property is that the warmup points span the entire space. In other words, the entire space is reachable as a linear combination of warmup points.

Two strategies for linear programming have been proposed and used. The first involves maximizing and minimizing each reaction sequentially and the other involves maximizing and minimizing random objective functions. Both of these strategies intuitively work but potentially have significant problems. One conceivable problem is illustrated in Figure 2.7. In this aniosotropic space, maximizing and minimizing in each direction yields points that do not span the entire space. The random direction strategy would also have problems in this case as points P_1 and P_2 are improbable to be obtained by chance, depending on the degree to which the space is anisotropic. This makes both strategies undesirable.

A solution to this problem is to use an adaptive strategy which ensures that the points generated by the warmup sampler do in fact span the entire space. A set of warmup points W can be checked whether it completely spans the solution space by optimizing in all directions perpendicular to the space spanned by W. First, the space of directions in W is computed by moving the space to the origin. $\tilde{W} = W_{2..k} - W_1$ The first point is subtracted from all subsequent points. Now the orthonormal basis to this set is generated:

$$B = null(orth(\tilde{W}^T))$$

where 'null' is a function that returns a basis for the null space. 'Orth' is optional but will improve computation time if $\tilde{W}$ already has many points. Maximizing and minimizing in the directions given by B is guaranteed to find any points not already spanned by W. If the dimensionality of W does not increase with the addition of the new points then W already spans the full solution space.

2.4 Conclusion

The ACHR sampler has been used in various studies to study the properties of the flux space. With the recent development of the GP-ACHR sampler, it is now a very practical algorithm for even large scale networks.

Chapter 3

Carbon 13 Analysis with Monte Carlo Sampling

3.1 Abstract

Background Carbon 13 tracing experiments have been used to indirectly measure rates of reaction in large biological networks. The choice of labels is an important consideration when designing these experiments. We present a novel Monte Carlo algorithm for finding the optimal substrate input label for a variety of experimental objectives. Unlike previous methods, this method does not require knowledge of the flux distribution beforehand.

Results Using a large *E. coli* isotopomer model, it was shown that the choice of optimal label is a function of the experimental objective. There is no universally best label able to answer all experimental objectives. Many commercially available labels were predicted to be outperformed by complex synthetic labeling patterns. Based on Monte Carlo Sampling, the dimensionality of experimental data was found to be considerably less than anticipated thus reducing effectiveness of C13 experiments in general.

Conclusion While Carbon 13 experiments are a useful tool in systems biology, the high redundancy in the measured values reduces the number of degrees of

freedom that can be obtained from each measurement. It is however possible to compute these drawbacks before the experiment is run and predict whether, and to what degree, a reaction can be elucidated.

3.2 Introduction

Metabolic systems biology aims to study metabolism through the use of large scale metabolic models. One framework that is popularly used is COnstraint Based Reconstruction and Analysis (COBRA) [3, 1]. COBRA relies on accurate, manually curated, chemical reconstructions as a basis for models. Many of these reconstructions have been generated [38] and the procedure well established [1, 39]. Biochemical models are generated from reconstructions by the imposition of physicochemical constraints - most commonly: steady state, reaction reversibility, enzyme capacities and reaction bounds based on experimental measurements. These models can then be used for a variety of methods such as computing growth rates [40, 41], predicting the effects of gene knockouts [42, 40, 33], predicting the endpoint of adaptive evolutions [43], and designing strains for industrial production [44, 45]. A review of these methods can be found here [46, 1, 3].

One of the principle interests in metabolic systems biology is the rate of chemical reactions through the metabolic network. Most internal reaction rates cannot be directly measured and COBRA models tend to be under-determined [7] yielding many possible flux states. One of the widely used experimental techniques which can indirectly measuring the flux rates indirectly is isotope labeling [47]. By measuring the enrichment for C13 in key metabolites after growing on a C13 labeled substrate, inferences about the internal flux state can be made.

An overview of the general C13 methods is described in Figure 3.1. An isotopomer model is created which describes the fate of each carbon atom through the stoichiometric network (Figure 3.1a). At steady state, with a known distribution of reaction rates (v) the isotopomer distributions vector (IDV) can be computed given a known carbon input label (Figure 3.1b). The isotopomer distributions are compared to measured mass distributions vectors (MDV) from either Mass Spec-

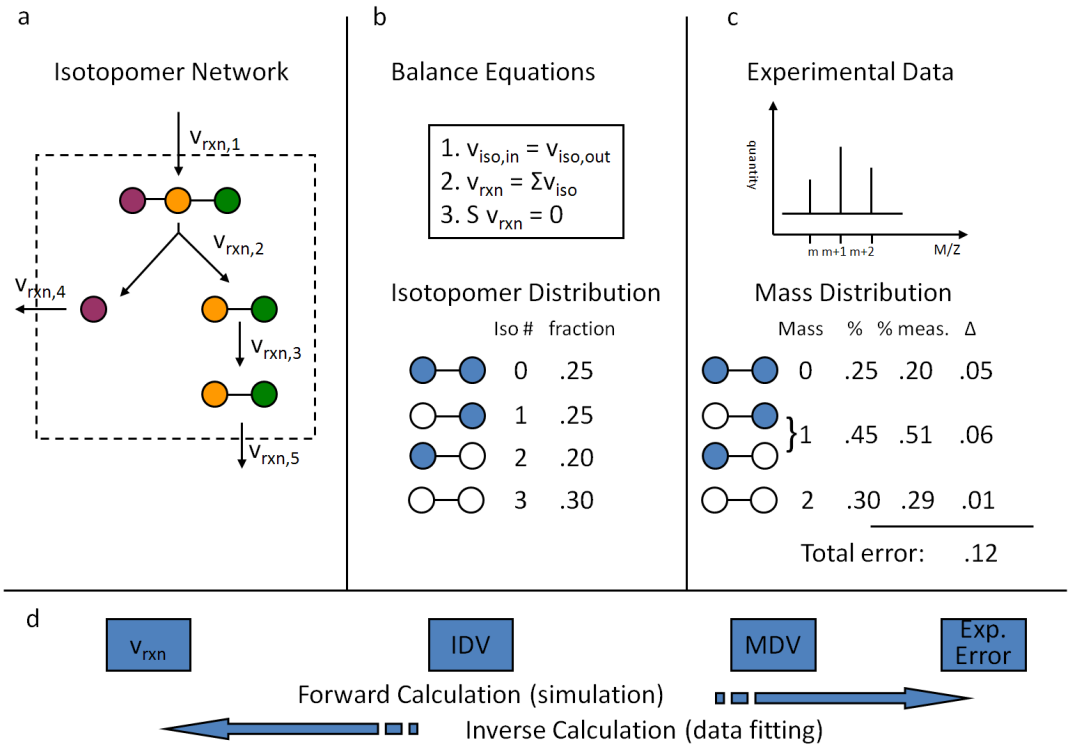


Figure 3.1: Isotopomer Overview

a) definition of the network including carbon fates. b) isotopomer balance equations. Solving these equations yields the Isotopomer Distribution Vector (IDV) c) experimental data are compared to computed Mass Distribution Vectors (MDV) yielding experimental fit. d) illustrates the two types of possible computations. The forward computation uses a flux distribution as input to compute the MDV. The inverse problem attempts to find the flux distribution which minimizes the experimental discrepancy.

trometry experiments or Nuclear Magnetic Resonance (NMR) experiment (Figure 3.1c). An experimental error is computed which summarizes how well a given flux distribution (v) is explained by the C13 experiment. Then through the use of a global optimization problem, the flux distribution v can be computed which minimizes the experimental error. While this forward calculation, ie. simulating C13 enrichment given a flux distribution is deterministic and easy, the inverse problem is of greater interest yet significantly more difficult (Figure 3.1d). This problem has been shown to be under-determined [48] indicating that many possible fluxes (vs) have the same minimum error and explain the data equally well. Additionally, the error minimization is computationally intensive and most methods are not guaranteed to find the global optimum. A review of these methods and challenges can be found in [49, 50, 51].

As these experiments are quite expensive both in terms of time and money, there is the possibility to enhance the experimental success by computationally optimizing experimental parameters before performing the experiment. Of particular note is the choice of substrate label. For a given n-carbon compound there are 2^n possible C13 labels (and possibly mixtures as well) and choosing the best label is known to affect the ability to elucidate reactions. Some work has been done in this area[47, 52] however all these methods have required knowing (or guessing) the flux distribution, v, of the organism. This is a restriction on the utility of such methods. We propose a Monte Carlo sampling based method for choosing the optimal label which does not require knowledge of the flux distribution of the cell.

The intuition for this method comes from realizing that while the inverse, data-fitting, problem is difficult and expensive, the forward computation is comparatively easy and fast. Initially, the flux distribution inside the cell is not known, however it is a bounded set as described by COBRA methods. A general C13 experiment reduces the feasible space in which cell must operate. A good experiment is one which reduces the space in a favorable way. This can be estimated with Monte Carlo Sampling. An experiment is envisioned as determining which of two regions a flux distribution came from (henceforth known as the ‘experi-

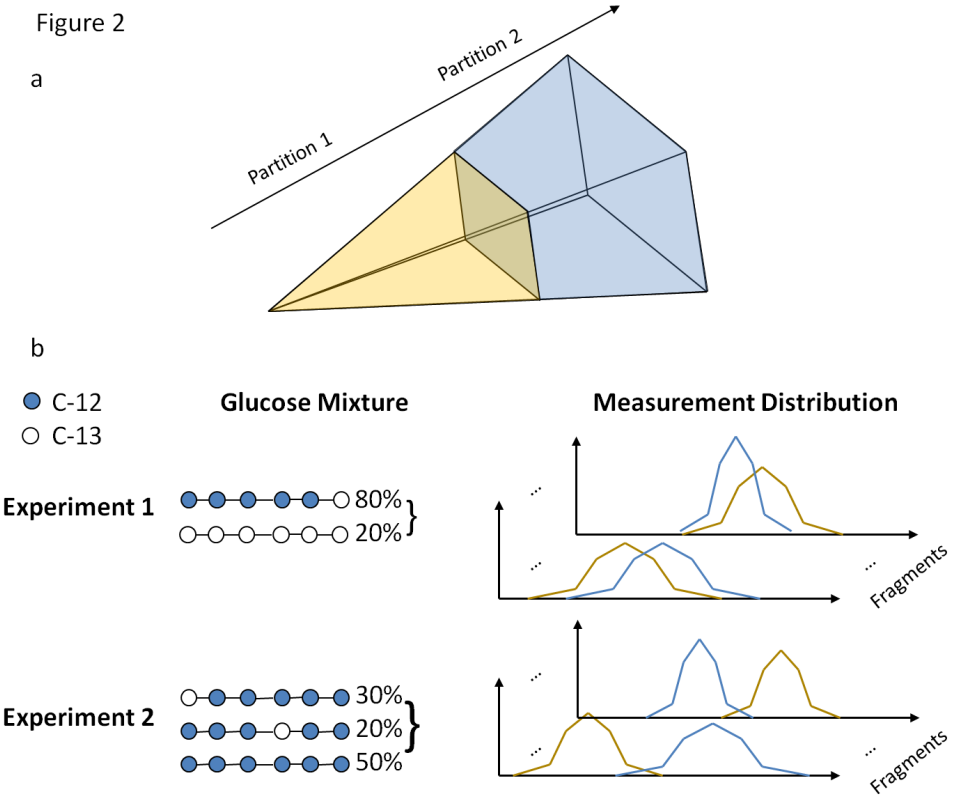


Figure 3.2: Method overview

a) The space of flux distributions is partitioned in two parts corresponding to high flux versus low flux. A uniform random sample is drawn from the space and is also partitioned into partition 1 and partition 2. b) For each point in the space the distribution of experimental measurements is simulated. Hypothetical Experiment 1 and Experiment 2 with different glucose label mixtures produce different measurement distributions. Experiment 2s distributions are more separated, indicating parameters of experiment 2 are more conducive for differentiating between the high and low partition.

mental hypothesis'). A well designed experiment is one where the expected C13 measurements of one subspace differ greatly from the C13 measurements of another. As seen in Figure 3.2, a hypothetical experiment 1 produces measurement distributions which overlap whereas Experiment 2 shows greater separation. If one were interested in differentiating between partition 1 and partition 2, experiment 2 would be much preferable. This method allows for the scoring of any label for any given experimental hypothesis without first knowing the true cellular flux distribution v.

3.3 Materials and Methods

3.3.1 Isotopomer Network Description

The isotopomer network was derived from the iJR904 *E. coli* reconstruction. The content is described in Appendix A. There are a total of 335 irreversible reactions including 278 which track Carbon. All carbon tracking reactions are broken into elementary forward and reverse reactions.

A central metabolic isotopomer model was generated that is equivalent in reaction content to most previously published isotopomer models for *E. coli* [53, 54]. The model includes a total of 85 reactions. These 85 reactions include a biomass production reaction, which drains the precursor metabolites used to make biomass, and 14 system boundary exchange fluxes (for glucose, oxygen, phosphate, NO_2, NO_3, acetate, CO_2, ethanol, formate, fumarate, glycerol, D-lactate, pyruvate, and succinate). The biomass composition is based on one that was reported previously [55, 40] and used in the biosynthetic isotopomer model (see details below), but where the biomass components are replaced by the amount of ATP, NADH, NADPH, and central metabolic precursors needed to synthesize the biomass components (Appendix A). The remaining 70 reactions participate in glycolysis, TCA cycle, pentose phosphate pathway, oxidative phosphorylation, pyruvate metabolism, and anaplerotic metabolism. The central metabolic isotopomer model includes linear mass balance equations for 67 metabolites. Carbon

atoms are tracked through 46 metabolites in the network.

To this central metabolic model were added additional catabolic and anabolic reactions. Changes to the central metabolic reactions include assigning fumarate reductase to reactions which utilize menaquinone and demethylmenaquinone rather than ubiquinone, and using a phosphate transport reaction coupled to proton symport. The biomass reaction also differs since it uses the amino acids, nucleotides, co-factors, and macromolecules rather than their precursor metabolites. In addition, the biosynthetic model balances intracellular protons as well as water molecules similar to iJR904 [40].

Aside from the central metabolic reactions contained in the central isotopomer model, the biosynthetic model also includes a number of other catabolic and anabolic reactions. To build the biosynthetic isotopomer model, the iMC1010 metabolic network [33], derived from iJR904 [40], was evaluated to determine which reactions can sustain non-zero fluxes during growth on glucose, acetate, or lactate when only certain by-products are allowed to be secreted (acetate, formate, D-lactate, pyruvate, succinate, glycerol, CO_2, and ethanol). The blocked reactions, which must have zero net flux at steady state, were subsequently omitted from the biosynthetic isotopomer model along with reactions that were not expected to be used. Analysis of the remaining reactions and metabolites identified groups of reactions that could be merged together in order to reduce the number of variables without affecting model results (see Appendix A). Large sets of biosynthesis reactions that produce phospholipids, nucleotides, co-factors were also combined. Since there are no experimental measurements for these high carbon metabolites, their isotopomers were not specifically accounted for in the model; however by-products that are formed as a result of their production (eg. CO_2, formate, succinate, fumarate, and pyruvate) that can enter back into the metabolic network, were tracked and accounted for.

The resulting biosynthetic isotopomer model includes 189 metabolites (126 of which have tracked carbon atoms), 250 metabolic reactions (63 of which are reversible and involve tracked carbon atoms), and 8,612 isotopomer variables (which is equal to the number of non-linear isotopomer mass balance constraints). The

model also includes a biomass reaction and 19 system boundary exchange reactions. Of the original 932 reactions in the complete metabolic iMC1010 network, nearly a third is represented in the biosynthetic isotopomer model, either individually or as grouped (or combined) reactions. A complete listing of the reactions and metabolites in the biosynthetic network can be found in Appendix A.

The fluxes were calculated with an additional constraint that flux through formyltetrahydrofolate deformylase (which removes the C1 unit from 10-formyltetrahydrofolate) was less than or equal to the measured formate secretion flux. When higher flux through this reaction was allowed the minimum error improved by only 0.3%, but the flux through this reaction was high (around half the glucose uptake rate). As a result, the optimal flux distributions and confidence intervals were calculated with this additional constraint on the formyltetrahydrofolate deformylase flux.

3.3.2 Sampling Of The Network

To compute possible flux distributions, v of the *E. coli* model, The network was sampled using a Markov Chain, Monte Carlo (MCMC) sampler. With traditional MCMC, a point is selected within the space which is then iteratively moved around. At each step, a random direction is chosen and the next point is chosen uniformly along this line. The set of points that this algorithm visits will converge to a uniformly distributed set. Two modifications were made: 1) Artificial Centering [36] Because these biological spaces tend to be elongated in one direction, it is often beneficial to choose directions along the "long" direction rather than uniformly. This can be done by choosing the direction based on previously visited points. At each step, the direction is chosen by drawing a vector from the center of the previous points to one of the previous points chosen at random. 2) In place sampling Instead of moving just one point throughout the space, many points are moved simultaneously. In this way, no "history" is kept, only the updated position of all the points. This method is described in greater detail in Chapter 2.

3.3.3 Computing the Isotopomer Distribution

Each flux distribution and glucose input result in a unique isotopomer distribution. We used the cumomer method [56] as well as the EMU method [57], and implemented it in Matlab. These methods involve solving several linear systems of equations to compute different groups of isotopomers. One small change implemented for numerical reasons is that at every step, a routine is introduced which checks whether all parts of the network are still connected. Disconnected components can occur when fluxes to and from the component are zero. It is then impossible to compute the isotopomer distribution within this subnetwork as many isotopomers will satisfy the balance equations. By removing these components first, the other metabolites can be solved in a numerically stable fashion. A breadth first search (starting with the glucose node) is computed through the network through reactions containing non-zero flux. All remaining metabolites are discarded and are assigned a distribution of 100% unlabeled.

For each flux distribution, the resulting isotopomers for the amino acids were transformed to mass distributions and concatenated into a long vector (length). This way, any experiment was abstracted to a 5000 x (number of fragments) matrix.

3.3.4 Generating and Evaluating C13 Experimental Hypotheses

An experimental hypothesis is defined as a partition of the solution space. While many possible hypotheses could be considered, only two particular kinds were studied. The first case attempts to elucidate whether a reaction has high or low flux. The solution space is partitioned into all points with $v_i >$ threshold versus $v_i <$ threshold. A different hypothesis is generated for each reaction i. The threshold has chosen to be the median of all v_i so that half (2500) points would be in each of the two partitions. The second set of hypotheses tested was for biologically relevant flux ratios. For each point the ratio of two reactions, v_i/v_j, was determined to be above or below some threshold forming the partition.

Scoring Hypotheses

Intuitively, a hypothesis score should be high if the isotopomer points coming from one partition are distinguishable from points in the other partition. While there are several ways of doing this, we chose a familiar one: Z-score The conventional Z-score is used to determine the difference between two normally distributed samples. While the samples of the MDVs are not always normally distributed, the Z-scores is nonetheless appropriate as an approximation:

$$Z_i = \frac{|\bar{x_{hi}} - \bar{x_{lo}}|}{\sqrt{s_{hi}^2 + s_{lo}^2}}$$

The Z score of each fragment is added together to give the Z-score of the experiment.

$$Z = \sum_i Z_i$$

A slight modification is introduced.

$$Z_i = \frac{|\bar{x_{hi}} - \bar{x_{lo}}|}{\sqrt{s_{hi}^2 + s_{lo}^2 + \sigma^2}}$$

Where $\sigma = .014$. σ is on the order of magnitude of the uncertainty in measurements. This slight modification accomplishes two tasks. First if both fragment i is of very low abundance then in practice it cannot be measured accurately and the Z-score will be low. Second, it is conceivable of having a high Z_i by having tiny s^2 values. In practice this cannot happen due to the experimental uncertainty.

3.3.5 Singular Value Decomposition of Samples

Singular Value Decomposition is a data reduction technique which allows the estimation of data dimensionality. A data matrix M is decomposed into $M = U \cdot \Sigma \cdot V^T$ where U and V are orthonormal basis and Σ is a diagonal matrix containing the primary diagonal contains singular values in descending order. A partial reconstruction of M is possible by taking only a subset of the largest singular values. For each condition, the number of singular values greater than some threshold. These thresholds have a direct interpretation as the uncertainty with

which a datapoint can be measured. A threshold cutoff of .01 indicates that the remaining uncertainty of the data falls within .01 or 1% measurement error.

3.3.6 Code and Equipment

The code was written in the MATLAB environment and the COBRA toolbox. Linear Programming was done with the Tomlab/CPLEX package and non linear optimization with the TOMLAB/SNOPT interface. The EMU and cumomer method were written in native Matlab but generated in Perl. Computations were performed on a Dell Studio XPS desktops (2.6 Ghz core i7 with 9-12 GB ram) and a custom Rocks cluster (100 dual Xeon 5500 series nodes).

3.3.7 Sample Preparation and C13 Measurement

Culture labeling

Prior to labeling, single colonies of *E. coli* K12 MG1655 were selected from stock plates and inoculated directly into 250 ml M9 medium in 500 Erlenmeyer flasks aerated by stirring at 1000 rpm. Cells were grown overnight, harvested, washed twice with water and used to inoculate 50 ml flasks containing 25 ml medium with 2 g/L 13C-labeled D-glucose, with initial OD600 0.005-0.01. Glucose was supplied as either 100% 1-13C-labeled, 100% 6-13C-labeled, or a mixture of 20% uniformly (U-13C-) labeled with 80% natural glucose (which is randomly 1% 13C). Cells were grown to mid-log phase, corresponding to OD600 of 0.6 (WT) or 0.25 (mutant). 3 ml (WT) or 10 ml (mutant) of each culture was harvested by centrifugation at 4C. Media were aspirated and analyzed with HPLC to determine the remaining glucose concentration. Cell pellets were placed at -80C prior to further analysis.

Derivatization and GC-MS analysis

Cells were resuspended in 0.1 ml 6 M HCl, transferred to glass vials and protein was digested into amino acids under a nitrogen atmosphere for 18 hr at 105C in an Eldex H/D Work Station. Digested samples were dried to remove residual

HCl, resuspended with 75 l each tetrahydrofuran and N-tert-butyldimethylsilyl-N-methyltrifluoroacetamide (Aldrich), and incubated for 1 hr at 80C to derivatize amino acids. Samples were filtered through 0.2 m PVDF filters, and injected into a Shimadzu QP2010 Plus GC-MS (0.5 l with 1:50 split ratio). GC injection temperature was 250C and the GC oven temperature was initially 130C for 4 min, rising to 230C at 4C/min and to 280C at 20C/min with a final hold at this temperature for 2 min. GC flow rate with helium carrier gas was 50 cm/s. The GC column used was a 15 m x 0.25 mm x 0.25 m SHRXI-5ms (Shimadzu). GC-MS interface temperature was 300C and (electron impact) ion source temperature was 200C, with 70 eV ionization voltage. The mass spectrometer was set to scan m/z range 50-600.

Processing of GC-MS data

Mass data were retrieved from the GC-MS for fragments of 14 derivatized amino acids: cysteine and tryptophan were degraded during amino acid hydrolysis; asparagine and glutamine were converted respectively to aspartate and glutamate; arginine was not stable to the derivatization procedure. For each fragment, these data comprised mass intensities for the base isotopomer (without any heavy isotopes, M+0), and isotopomers with increasing unit mass (up to M+6) relative to M+0. These mass distributions were normalized by dividing by the sum of M+0 to M+6, and corrected for naturally-occurring heavy isotopes of the elements H, N, O, Si, S, and (in moieties from the derivatizing reagent) C, using matrix-based probabilistic methods as described [58, 59] implemented in Microsoft Excel. Data were also corrected for carry-over of unlabeled inoculum [58].

3.3.8 Computing Reaction rates from C13 Data

The inverse problem finding the flux distribution (reaction rates) that best explains a set of C13 data is formulated as a non-linear optimization problem:

$$\min_{v} \mathrm{Error(v)}$$

subject to

$$v_{min} < v < v_{max}$$

$$S \cdot v = 0$$

The function Error(v) is a score of how well a given flux distribution fits the experimental data. It is defined as:

$$Error(v) = \sum_{i \in \text{fragments}} \frac{(\text{fragment}_i(v) - \text{measured}_i)^2}{\sigma^2}$$

where measured_i is the measured fractional enrichment of fragment i, and $\text{fragment}_i(v)$ is the computed fractional enrichment of fragment i as a function of the flux distribution v and σ is the standard deviation of the fragments as calculated from repeat experiments.

A small variation was introduced to reduce the number of variables and remove constraints. Let N be a basis for the null space of S. Then all valid fluxes can be written as: $v = N \cdot \alpha$.

$$\min_{\alpha} \text{Error}(N \cdot \alpha)$$

subject to:

$$v_{min} < N \cdot \alpha < v_{max}$$

This reduced the number of variables from $|v| = 335$ to $|\alpha| = 139$.

Optimization was performed with the Tomlab SNOPT package. This method is an iterative local optimization and is therefore not guaranteed to find the optimal solution. The procedure was therefore run with many randomly generated starting points.

Computing confidence intervals

Confidence intervals for reaction rates were also computed by maximizing and minimizing the value of each reaction in turn subject to a slightly relaxed score.

$$\min_{\alpha} / \max_{\alpha} c_i^T \cdot N \cdot \alpha$$

Table 3.1: Computational Evaluation of Glucoses

Potential glucose labels are evaluated based on three criteria a) Absolute Z-score for various fluxes b) Z-scores normalized with respect to the best glucose and c) the number of singular values of the sample greater than a threshold (.03, .01, .003, .001). Glucose labels are listed on top including exotic labeling patterns (#111000 through #011111) and commercially available labels (C1 through C6). C12 = C1,2 double labeled, CU = uniform labeled and C0 = unlabeled. Reaction and reaction ratio hypotheses are listed on the left. The random hypotheses, as described in the methods, shows the level of noise.

Glucose / reaction	max	'#111000'	'#110010'	'#011000'	'#000011'	'#011111'	'80%C1+20%CU'	'C1'	'80%C1+20%CU'	'C2'	'C3'	'C4'	'C5'	'C6'	'C12'	'#111000'	'#110010'	'#011000'	'#000011'	'#011111'	'80%C1+20%CU'	'C1'	'80%C1+20%CU'	'C2'	'C3'	'C4'	'C5'	'C6'	'C12'
	Z-scores															Relative Z-scores													
'EX_nh4'	24.0	18.3	24.0	16.3	13.1	13.4	11.5	13.4	12.1	9.7	11.0	9.7	22.5	8.9	16.7	0.6	0.9	0.5	0.4	0.4	0.3	0.4	0.4	0.3	0.3	0.3	0.8	0.2	0.6
'EX_for'	27.2	15.1	21.5	21.9	9.4	21.5	9.4	21.5	18.5	10.4	22.9	3.6	4.4	6.0	27.2	0.4	0.7	0.7	0.2	0.7	0.2	0.7	0.6	0.3	0.7	0	0	0.1	0.9
'CS'	54.4	44.0	38.7	54.4	43.9	28.4	15.4	28.4	22.8	39.7	36.9	12.1	20.6	43.1	47.9	0.7	0.7	0.9	0.7	0.5	0.2	0.5	0.4	0.7	0.6	0.2	0.3	0.7	0.8
'EDA'	29.3	28.6	16.7	20.1	23.3	19.6	5.4	19.6	16.1	21.5	12.2	21.4	18.4	11.2	29.3	0.9	0.5	0.6	0.7	0.6	0.1	0.6	0.4	0.6	0.3	0.6	0.5	0.3	0.9
'FTHFD'	30.8	12.2	13.2	21.7	9.3	30.8	5.0	30.8	25.4	16.6	9.9	2.6	8.6	5.5	16.0	0.3	0.3	0.6	0.2	0.9	0.1	0.9	0.7	0.4	0.2	0	0.2	0.1	0.4
'GLYK'	13.0	2.4	9.6	10.0	1.8	10.0	1.0	10.0	8.1	1.4	13.0	0.9	1.2	1.0	11.2	0	0.5	0.5	0	0.5	0	0.5	0.4	0	0.8	0	0	0	0.6
'MALS'	47.7	36.3	21.4	34.5	17.8	36.6	21.6	36.6	32.0	23.6	17.6	8.6	10.3	7.1	47.7	0.7	0.4	0.7	0.3	0.7	0.4	0.7	0.6	0.4	0.3	0.1	0.1	0.1	0.9
'PGI'	68.1	59.8	44.0	52.4	68.1	34.1	11.4	34.1	24.8	45.4	32.7	13.6	31.9	60.5	54.0	0.8	0.6	0.7	1	0.5	0.1	0.5	0.3	0.6	0.4	0.2	0.4	0.8	0.7
'PGK'	61.2	52.8	34.9	41.8	61.2	32.5	8.5	32.5	24.1	39.4	25.4	15.3	30.3	49.3	47.2	0.8	0.5	0.6	0.9	0.5	0.1	0.5	0.3	0.6	0.4	0.2	0.4	0.8	0.7
'POX'	3.4	3.0	2.6	2.8	2.1	2.2	2.5	2.2	1.8	2.2	2.3	1.5	0.8	1.8	3.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'PFL'	22.8	13.3	17.2	21.3	7.2	21.4	7.8	21.4	18.3	9.9	18.5	4.1	3.9	2.6	22.8	0.4	0.6	0.8	0.2	0.8	0.2	0.8	0.7	0.3	0.7	0	0	0	0.9
'PPC'	58.8	41.6	35.1	39.8	22.7	49.3	21.9	49.3	41.9	24.8	30.7	9.5	12.7	8.6	58.8	0.7	0.5	0.6	0.3	0.8	0.3	0.8	0.7	0.4	0.5	0.1	0.2	0.1	0.9
'rFUM'	55.0	40.6	40.6	55.0	33.6	35.0	18.2	35.0	29.4	35.4	41.2	11.2	16.1	32.5	52.2	0.7	0.7	0.9	0.6	0.6	0.3	0.6	0.5	0.6	0.7	0.1	0.2	0.5	0.9
'PFK_FBP'	28.0	28.0	7.6	12.2	9.2	7.3	4.6	7.3	6.0	4.5	10.3	4.7	4.3	2.6	18.5	0.9	0.2	0.3	0.2	0.1	0.1	0.1	0.1	0	0.3	0.1	0	0	0.5
'GAPD_G6PDH2r'	69.1	60.3	43.7	52.1	69.1	34.6	11.0	34.6	25.3	45.7	32.5	13.8	32.6	60.6	54.3	0.8	0.6	0.7	1	0.5	0.1	0.5	0.3	0.6	0.4	0.2	0.4	0.8	0.7
'PYK_PPS'	25.1	19.6	17.8	24.6	17.3	17.1	10.1	17.1	15.3	16.7	16.0	5.7	6.5	17.9	25.1	0.7	0.6	0.9	0.6	0.6	0.3	0.6	0.5	0.5	0.5	0.1	0.1	0.6	0.9
'PPC_PPCK'	11.1	10.8	4.1	7.5	4.1	7.4	5.6	7.4	6.4	6.2	3.4	1.8	2.0	2.2	11.1	0.7	0.1	0.4	0.1	0.4	0.2	0.4	0.3	0.3	0	0	0	0	0.7
'rFUM_ENO'	50.7	37.2	39.0	50.0	28.0	35.6	17.7	35.6	29.8	30.3	39.0	11.5	14.5	26.2	50.7	0.7	0.7	0.9	0.5	0.6	0.3	0.6	0.5	0.5	0.7	0.2	0.2	0.5	0.9
'rACONT'	54.4	44.0	38.7	54.4	43.9	28.4	15.4	28.4	22.8	39.7	36.9	12.1	20.6	43.1	47.9	0.7	0.7	0.9	0.7	0.5	0.2	0.5	0.4	0.7	0.6	0.2	0.3	0.7	0.8
'PDH_rFUM'	20.7	11.5	17.0	20.7	8.1	16.2	5.2	16.2	13.6	9.8	19.7	3.6	3.7	6.7	19.0	0.4	0.7	0.8	0.2	0.6	0.1	0.6	0.5	0.3	0.8	0	0	0.2	0.8
'MALS_rACONT'	44.3	35.6	18.9	28.1	20.5	34.8	19.1	34.8	30.3	19.8	14.2	7.8	11.4	6.6	44.3	0.7	0.4	0.6	0.4	0.7	0.4	0.7	0.6	0.4	0.2	0.1	0.2	0.1	0.9
'GLUDy_GLUSy'	2.1	1.4	1.4	2.1	1.2	1.8	0.6	1.8	1.5	1.6	1.0	1.4	0.9	1.1	1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'LDHD_PDH'	7.9	6.2	7.9	4.8	6.0	5.6	1.5	5.6	4.5	5.3	4.8	2.5	3.7	5.5	7.5	0.4	0.6	0.2	0.4	0.3	0	0.3	0.2	0.3	0.2	0	0.1	0.3	0.5
'PYK_PDH'	20.9	15.0	14.9	20.6	12.0	16.4	7.6	16.4	14.1	11.3	16.3	4.6	5.1	11.1	20.9	0.6	0.6	0.8	0.4	0.6	0.2	0.6	0.5	0.4	0.6	0.1	0.1	0.4	0.8
'FRD2_SUCD1i'	3.8	3.2	2.7	3.4	1.4	3.4	2.0	3.4	3.1	2.2	2.3	0.9	1.5	1.6	3.8	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0.2
'random1'	3.1	3.0	2.6	1.7	2.2	2.5	1.0	2.5	2.0	1.5	1.8	1.4	1.0	1.3	3.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
'random2'	2.5	1.8	2.5	1.6	1.3	2.1	1.3	2.1	2.0	0.9	1.3	1.5	2.1	1.4	2.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Singular Value counts															Relative Singular Value counts													
SVD 0.03	54	48	53	50	54	39	27	39	44	38	26	41	41	39	47	0.9	1	0.9	1	0.7	0.5	0.7	0.8	0.7	0.5	0.8	0.8	0.7	0.9
SVD 0.01	73	64	73	66	71	53	35	53	64	49	33	56	57	53	66	0.9	1	0.9	1	0.7	0.5	0.7	0.9	0.7	0.5	0.8	0.8	0.7	0.9
SVD 0.003	91	84	91	82	90	65	42	65	82	58	42	70	69	68	82	0.9	1	0.9	1	0.7	0.5	0.7	0.9	0.6	0.5	0.8	0.8	0.7	0.9
SVD0 .001	100	95	99	88	100	77	49	77	93	66	50	80	81	77	94	1	1	0.9	1	0.8	0.5	0.8	0.9	0.7	0.5	0.8	0.8	0.8	0.9

subject to:

$$v_{min} \leq N \cdot \alpha \leq v_{max}$$

$$\text{Error}(N \cdot \alpha) \leq Error_{max}$$

Where $c_i = (0, 0, ...0, 1, 0...0)^T$ is a vector of all zeros with a 1 in position i. The $Error_{max}$ depends on the confidence value. Because different data sets provide different levels of consistency, Error_{max} was chosen to be 30 more than the minimum error found.

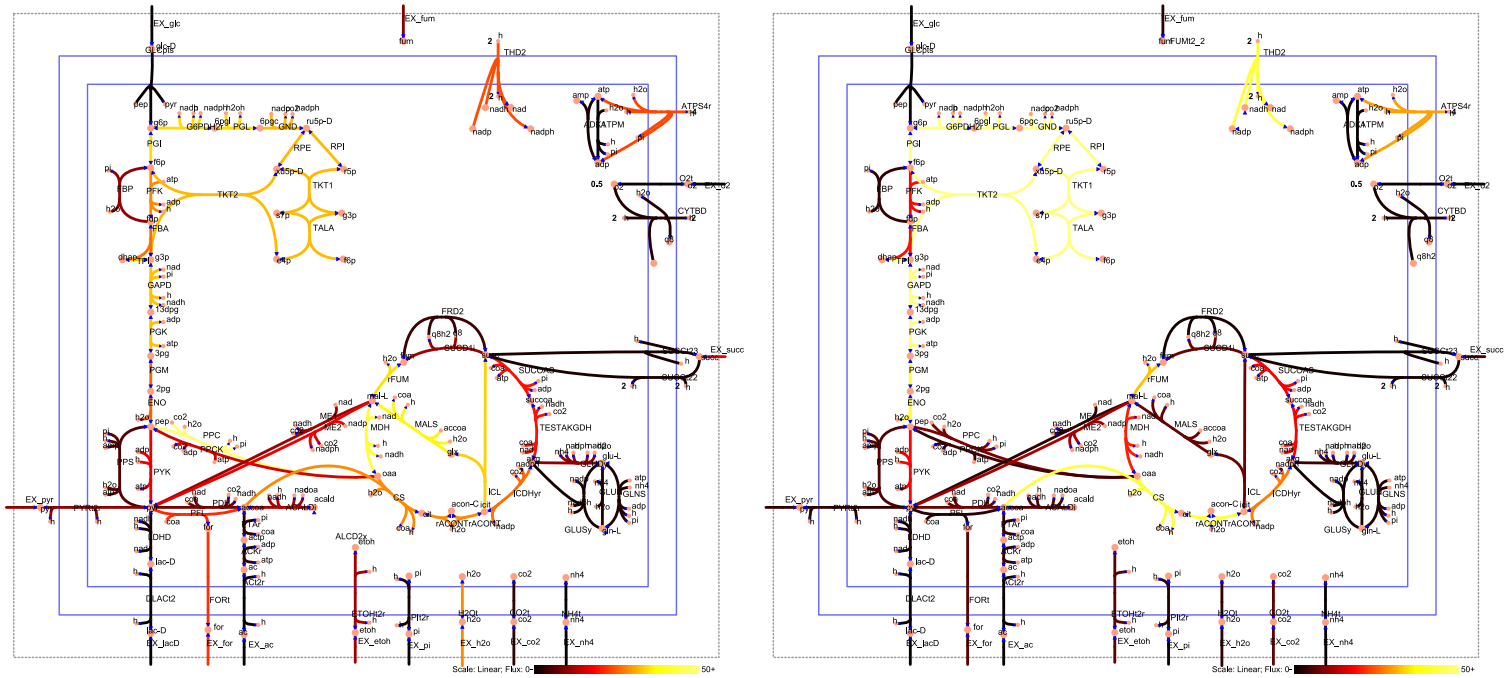
3.4 Results and Discussion

A large scale isotopomer model was constructed of *E. coli* as described in the methods section.

3.4.1 Scoring Experiments

From this model, 5000 candidate flux states were sampled uniformly and experimental hypotheses tested. Z-scores were calculated for the hi-lo hypothesis corresponding to 1) individual reactions 2) reaction ratios and 3) two random hi-lo experiments. Raw Z-scores as well as normalized ones are presented in Table 3.1. The random hypotheses (random 1 and random2) illustrate the level of noise associated with taking 5000 points and are on the order of 3.0. The reaction and reaction ratio scores varied from the level of noise (example C4 + GLYK) to a maximum of 69.1 (#000011 + GAPD/G6PDH2r).

The normalized Z-scores clearly show that there is no universally best label. ie. there is no single label that yields the best results for all experimental objectives. The exchange of formate ('EX_for') could be easiest measured with a 1,2 labeled glucose however this labeling pattern is bested by a Carbon 1 label for the measurement of reaction 'FTHFD'. This non-universality of labels in line with expectations as it has been previously shown that the choice of labels can affect the experimental outcome.



(a) C1

(b) C6

Figure 3.3: Two Maps with Z-scores

Two possible glucose label patterns show different strengths in evaluating different parts of the network. C1 shows illustrates a hypothetical experiment with 100% Carbon 1 labeled glucose. C6 shows the same network evaluated with Carbon 6 labeled glucose.

There are certain reactions which are predicted to be difficult to measure. The Z-scores for reaction 'POX' for example all lie within the level of noise.

For certain reactions, the best experiment that could be performed involves exotic (non-commercially available) labels. One example is the ratio of 'PFK' to 'FBP'. The best label was the 1,2,3 triple label (Z = 28.0) which is significantly higher than the best commercially available label (1,2 double label, Z = 18.5). It may be necessary to synthesize these compounds by other means in order to best measure this reaction flux.

Visualization

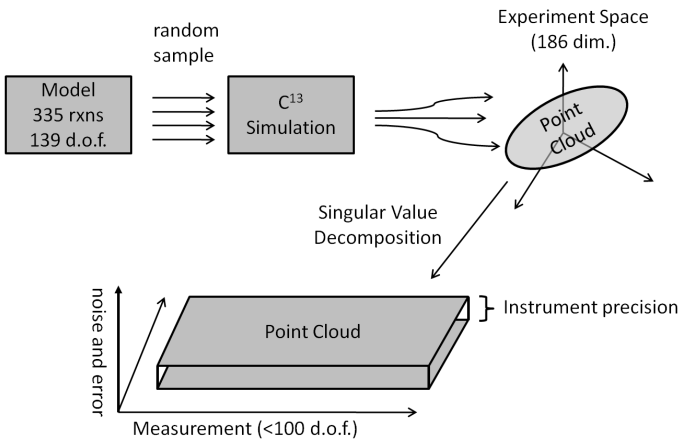
Two sets of Z-scores corresponding to glucose lables 'C1' and 'C6' are also plotted in Figure 3.3. Lighter colors indicate higher Z-scores and ease of measurement. With this overview it is easy to compare two labels and their success at measuring different pathways in the network. In this case, 'C6' scores higher at measuring the Pentose Phosephate Pathway and most of lower glycolysis whereas 'C1' scores much higher at measuring malate synthase ('MALS' in the citric acid cycle).

3.4.2 Dimensionality of Isotopomer Data

To determine the dimensionality of the isotopomer data, Singular Value Decomposition (SVD) was performed on the entire solution space of several glucose labels. The results are summarized in Figure 3.4B. Globally, the choice of glucose labels affects the dimensionality of the resulting isotopomer data set. At the 1% (.01) threshold, the label with the highest dimensionality was the exotic label C110010 with 73. The three experiments performed in this study (C1, C6, CU) had dimensions 53, 53 and 35 respectively. These values are all significantly lower than the best label and in particular the uniform labeled experiment only produces half of the dimensionality as the optimal experiment.

This result is highly important. Whereas 186 dimensions (pieces of information) which are measured is enough to specify a unique flux vector, v (at least

a)



SVD \ GLCS	'#111000'	'#110010'	'#011000'	'#000011'	'#011111'	80%C0 +20%CU'	'C1'	80%C1 +20%CU'	'C2'	'C3'	'C4'	'C5'	'C6'	'C12'
0.03	48	53	50	54	39	27	39	44	38	26	41	41	39	47
0.01	64	73	66	71	53	35	53	64	49	33	56	57	53	66
0.003	84	91	82	90	65	42	65	82	58	42	70	69	68	82
0.001	95	99	88	100	77	49	77	93	66	50	80	81	77	94

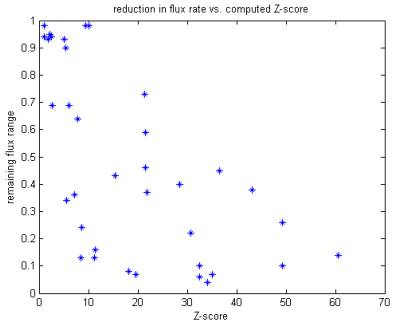
(b)

Figure 3.4: Data Dimensionality with SVD

The linear dimensionality of experimental data space is measured with Singular Value Decomposition. a) The E. coli model has 335 reactions and 139 degrees of freedom. The isotopomer fragments were computed for a random sample of flux distributions and plotted in the 186 dimensional space of simulated measurements. The upper bound on the number of degrees of freedom in this space was determined by Singular Value Decomposition on the samples. The number of singular values was counted until the magnitude of the next singular value fell below the instrument threshold. b) The number of significant singular values is tabulated against the choice of glucose with values ranging from 26 to 100.

RXN	Z-SCORES			FVA				NORM FVA		
	C1	C6	CU	none	C1	C6	CU	C1	C6	CU
'EX_for'	21.5	6.0	9.4	19.5	11.5	13.5	19.2	0.59	0.69	0.98
'CS'	28.4	43.1	15.4	14.5	5.7	5.5	6.3	0.40	0.38	0.43
'EDA'	19.6	11.2	5.4	13.6	1.0	1.8	12.2	0.07	0.13	0.90
'FTHFD'	30.8	5.5	5.0	15.8	3.5	5.4	14.7	0.22	0.34	0.93
'GLYK'	10.0	1.0	1.0	24.7	24.1	23.2	24.1	0.98	0.94	0.98
'MALS'	36.6	7.1	21.6	8.6	3.9	3.1	4.0	0.45	0.36	0.46
'PGI'	34.1	60.5	11.4	100.0	4.2	13.6	15.6	0.04	0.14	0.16
'PGK'	32.5	49.3	8.5	96.8	9.9	9.6	12.6	0.10	0.10	0.13
'POX'	2.2	1.8	2.5	13.0	12.4	12.1	12.2	0.95	0.93	0.94
'PFL'	21.4	2.6	7.8	19.7	14.4	13.6	12.6	0.73	0.69	0.64
'PPC'	49.3	8.6	21.9	27.7	7.1	6.6	10.1	0.26	0.24	0.37
'rFUM'	35.0	32.5	18.2	100.0	6.8	5.9	8.3	0.07	0.06	0.08

(a)



(b)

Figure 3.5: Comparing Predictions and Experimental Flux Ranges
a) For several reactions, the computed Z scores are compared to the resulting measured flux ranges. Z-scores show (color coded) Z-scores for each of the 12 reactions and three glucose labels. FVA indicates the absolute allowable flux ranges for three glucoses ('C1', 'C6', 'CU') as well as the range if no C13 data is imposed ('none'). A normalized version of this table is also presented where all flux ranges are divided by the FVA range thus showing the fraction of flux range remaining. This quantity ranges from 0 (range fully specified) to 1 (no additional information). b) A scatterplot of Z-scores versus the reduction of flux ranges. The correlation is r = -.62.

in theory), the much lower true dimensionality is nowhere near enough. The best setup specifies just over half (73/139 = .52) the degrees of freedom required and the CU label only about 1/4 (.26).

The SVD computational is a linear operation and thus actually *overestimates* the true dimensionality of the data.

3.4.3 Experimental Verification

Wild type *E. coli* was grown under three glucose media conditions: 1) 100% C1 labeled glucose ('C1'). 2) 100% C6 labeled glucose ('C6') 3) 20% Uniform labeled + 80% natural labeled glucose ('CU'). The experimental measurements were corrected for natural isotope labeling and confidence intervals were generated

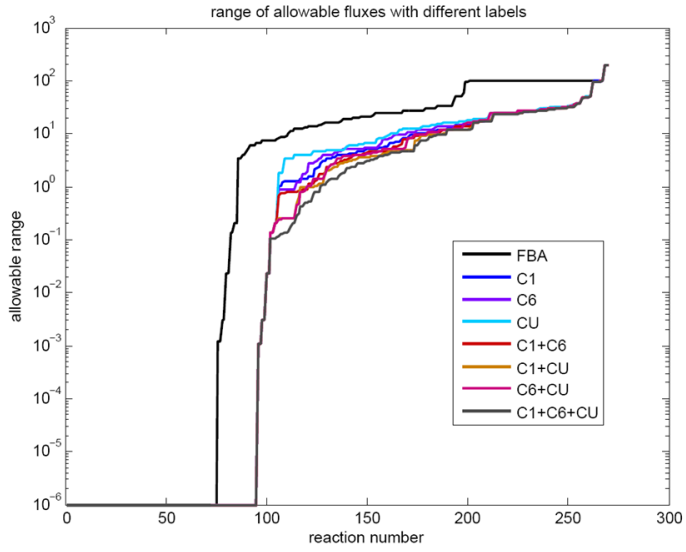


Figure 3.6: Solution space size with different data sets
The range of allowable fluxes ($v_{max}v_{min}$) were computed for each reaction, constrained by none, one, two or three sets of C13 data. Reactions are rank ordered by range. The reaction order is potentially different between conditions.

for each reaction. Figure 3.5 compares Z-scores and computed reaction ranges.

From the color coding of the Z-scores (green = high score, red = low) and the relative reaction ranges (green = low range, red = high range) it is apparent that the Z-scores are an imperfect predictor of flux range. In fact a scatter plot of Z-scores vs. relative flux ranges (Figure 3.5B) shows a correlation of -.62. This is statistically significant and in particular the reactions which were most difficult to elucidate (high relative flux range) all had Z-scores less than 10.

Global Glucose Properties

Reaction confidence ranges were computed for all reactions using all three sets of C13 data and all combinations thereof. The reactions were rank ordered by allowable flux in Figure 3.6. Using different labels provides different levels of reaction confidence. Including no C13 data generates the largest solution ranges (upper black line). Adding C13 data reduces the ranges. With almost no exception,

including one experiment yields larger confidence intervals than any combination of two carbon sources which in turn is a larger range than including all three sets. Of the single experiment curves, the 'CU' curve provides notably worse ranges than the other two experiments which agrees with the earlier result that 'CU' provides less information to the model.

If a reaction confidence of 1 ($mm \cdot gDW^{-1} \cdot h^{-1}$) is desired, then without any C13 data, 85 reactions meet this criterium. Performing the worst C13 experiment ('CU') yields 105 reactions whereas the combination of all three C13 experiments yields 125 reactions. In other words, performing all three experiments will increase the number of elucidated reactions by 40 reactions or about 50%. This is somewhat less than but on the same order of magnitude as the dimensionality of the C13 data.

3.5 Concluding Remarks

We introduce a new framework for dealing with the uncertainty inherent to C13 experiments using Monte Carlo Sampling. This allows us to predict the success of actual wet lab experiments before actually performing them. This framework reveals several key findings:

- The choice of input label is important. Different labels perform better than others. In particular, a 20% mixture of uniform label + 80% natural label ('CU') was shown to elucidate reaction rates worse than either a 'C1' or 'C6' label. This can be established before performing any actual experiment and without having information about the true flux distribution like other methods [47, 52].

- There is no universally best label. The best label depends on the experimental objective. Certain reactions are easier measured with some labels than others and no label is best at elucidating all reactions.

- With our method we were able to predict that certain exotic C13 labels of glucose, in particular C1,2,3 and C1,2,5 triple labels are predicted to perform

superior for elucidating many reactions to commercially available single labels. It may be necessary to synthesize these isotopomer compounds to best measure these key reactions.

- The C13 data dimensionality is less than anticipated. Whereas each C13 experiment can measure 186 pieces of information at a time, there is a high degree of interdependence. We measured the true data dimensionality to be in the 35-50 range for experiments tested and as high as 73 for certain exotic labels. This high data redundancy can partially explain why C13 experiments still leave so many reactions with high uncertainties.

While steady state C13 analysis is clearly useful, it may be less useful at restricting flux than would be expected.

The authors declare no conflicts of interest.

The text of Chapter Three, is work to be submitted to the Biotechnology Journal with authors J. Schellenberger, W. Choi, S. Madireddi, V. Portnoy, D. Scott, J. Reed, A. Ostermann, B.Ø. Palsson. I am the primary author on this study, and did most of the research.

Chapter 4

Elimination of Thermodynamically Infeasible Loops in Steady State Metabolic Models

4.1 Abstract

The Constraint-Based Reconstruction and Analysis (COBRA) framework has proved useful for studying steady state flux solutions in genome-scale metabolic networks. One shortcoming of current COBRA methods is the possible violation of the so called "loop law" in the computed steady state flux solutions. The loop law is analogous to Kirchhoffs second law for electric circuits and it states that there can be no net flux around a closed network cycle in the steady state. While the consequences of loop law have been known for years, it has been computationally difficult to work with and thus the limitations that it imposes have been overlooked. We present a general Mixed Integer Programming (MIP) approach, called loopless COBRA (ll-COBRA), to eliminate all steady state flux solutions not compatible with the loop law. We demonstrate the approach to augment COBRA on two previously published COBRA methods, Flux Variability Analysis and Monte Carlo

sampling of the flux space, and find that in both cases the imposition of the loop law improves published results.

4.2 Introduction

Systems Biology aims to understand the properties of large-scale biochemical networks through the construction of predictive *in silico* models. One of the most commonly used approach is the Constraint-Based Reconstruction and Analysis (COBRA) framework [60, 3, 4]. Genome-scale models for metabolism are built in a bottom-up fashion from various sources of biological knowledge, such as genome annotations, metabolic databases and bibliographic "legacy" data; reviewed in [1, 61, 39]. This quality controlled reconstruction process results in validated mathematical models, capable of making predictions about the rates of reaction inside a cell, leading to a variety of applications [46, 62, 63]. As these models are generally under-determined, steady state flux solutions are calculated by imposing constraints on the system and optimization of an objective function [3, 64, 46, 65]. Popular constraints include the steady state assumption, reaction reversibilities and reaction capacities. A list of methods developed under this framework have been reviewed [3, 1, 61, 66].

A COBRA model consists of at minimum a stoichiometric matrix (S), and reaction lower and upper bounds (lb, ub). The stoichiometric matrix encodes information about reactions (columns) and metabolites (rows) such that entry $S_{i,j}$ is the stoichiometric participation of metabolite i in reaction j. The stoichiometric matrix relates the rates of reaction (v) to the change in concentration of metabolites (x) by the formula:

$$\frac{dx}{dt} = S \cdot v$$

At steady state, concentrations are assumed not changing and this equation reduces to $S \cdot v = 0$. The reaction bounds impose an upper and lower limit on the rates of each reaction. Many reactions are considered irreversible ($v_i > 0$). Others, especially uptake and secretion reactions, are experimentally measured ($v_i = v_{exp}$). If no information is available, arbitrarily large bounds are set (ie

$-10000 < v_i < 10000$). Together, the flux bounds and the steady state equation , define a closed space of possible flux states. Flux Balance analysis (FBA) attempts to fine compute the likely state by optimizing a metabolic objective [67, 68]. Often this is the production of biomass [46], production of ATP [69] or production of biomass per unit input [65].

Classical FBA and many other COBRA methods often ignore the imposition of the so-called 'loop law' [70]. The loop law is analogous to Kirchhoffs second law for electrical circuits and it states that the thermodynamic driving forces around a metabolic loop must add up to zero. As such, there cannot be a net flux around a closed cycle in a network in a steady state. Methods for determining whether a computed flux distribution has a loop have been developed [71] although they are quite cumbersome and not flexible enough to be included in optimization computations [5].

An alternative approach to treating the loop law is to include additional thermodynamic information. Methods that do so rely on the relation $\Delta G_r = \Delta G_0 + RT \ln Q$ where Q is a ratio of metabolic concentrations and ΔG_r is the Gibbs energy of a reaction. ΔG_r directly relates to the sign of the flux through the associated reaction. If $\Delta G_r > 0$ and $v_{net} < 0$ and vice versa. This places additional constraints on the reaction directions as well as concentrations. This approach was used to compute potential regulatory sites [35], determine reaction directionality [72] and compute feasible concentration ranges [73]. A slightly different formalism with decoupled forward and reverse reactions has been used to modify FBA [74, 75]. All these methods require knowing the standard free energy change of reactions (ΔG_r) or the standard energies of formation ΔG_f of all metabolites in the network. These values can be found in databases [76], or estimated computationally with methods such as group contribution theory [77, 78] although accuracy and coverage pose some challenges.

We have developed a simple method to incorporate the loop law constraints into many current COBRA methods. This method requires no additional inputs or data (such as concentrations, ΔG_f, etc.) and turns any Linear, Quadratic or Mixed Integer Problem (LP, QP, MIP) into a modified Mixed Integer Problem.

The solution to this expanded problem solves the initial problem with the added constraint that the fluxes may not contain loops. We demonstrate this technique on three popular COBRA methods: Flux Balance Analysis (FBA), Flux Variability Analysis (FVA) and Monte Carlo Sampling to produce loopless versions of each (ll-FBA, ll-FVA, ll-sampling). Given the extensive use of COBRA methods this simple method to eliminate steady state flux solutions with net fluxes through loops is likely to find widespread use.

4.3 Methods

4.3.1 The 'loopless' condition

First, lets consider the simpler problem of determining if a given flux solution, v, contains a loop. In order for v to satisfy the loop law the reaction energies around any cycle must add to zero. This condition can be written concisely as $v^T \cdot G = 0$ where G is a vector of energies of each reaction. Extreme Pathway (ExPa) [12] and Elementary Mode analysis [79] have shown that the number of loops (type III pathways) grows rapidly and enumerating all loops (v) has not been possible for medium to large scale networks [14]. As it turns out, it is not necessary to enumerate all loops. As shown in Fig. 4.1, all loops lie within the internal network, S_{int}. Any steady state path in S_{int} is a loop and all such paths can be expressed as a linear combination of the null basis of S_{int} [4]. All loops can be expressed in the form $v = N_{int} \cdot \alpha_i$ where $N_{int} = \text{null}(S_{int})$ and α_i are weights. If it can be shown that $N_{int}^T \cdot G = 0$ then it automatically follows that $v^T \cdot G = 0$ for all loops v.

The loopless condition forms a Linear Programming (LP) problem. A vector of continuous variables (Gi) indicates the driving force of each reaction. This quantity can be thought of as being analogous the ΔG_r of each reaction in that $\text{sign}(G) = \text{sign}(\Delta G_r)$ although numerically they may be quite different. A loop is purely defined by the sign (direction) of the flux distribution [71] and therefore any solution to $N_{int} \cdot G_i = 0$ indicates that no loop is present. Verifying a

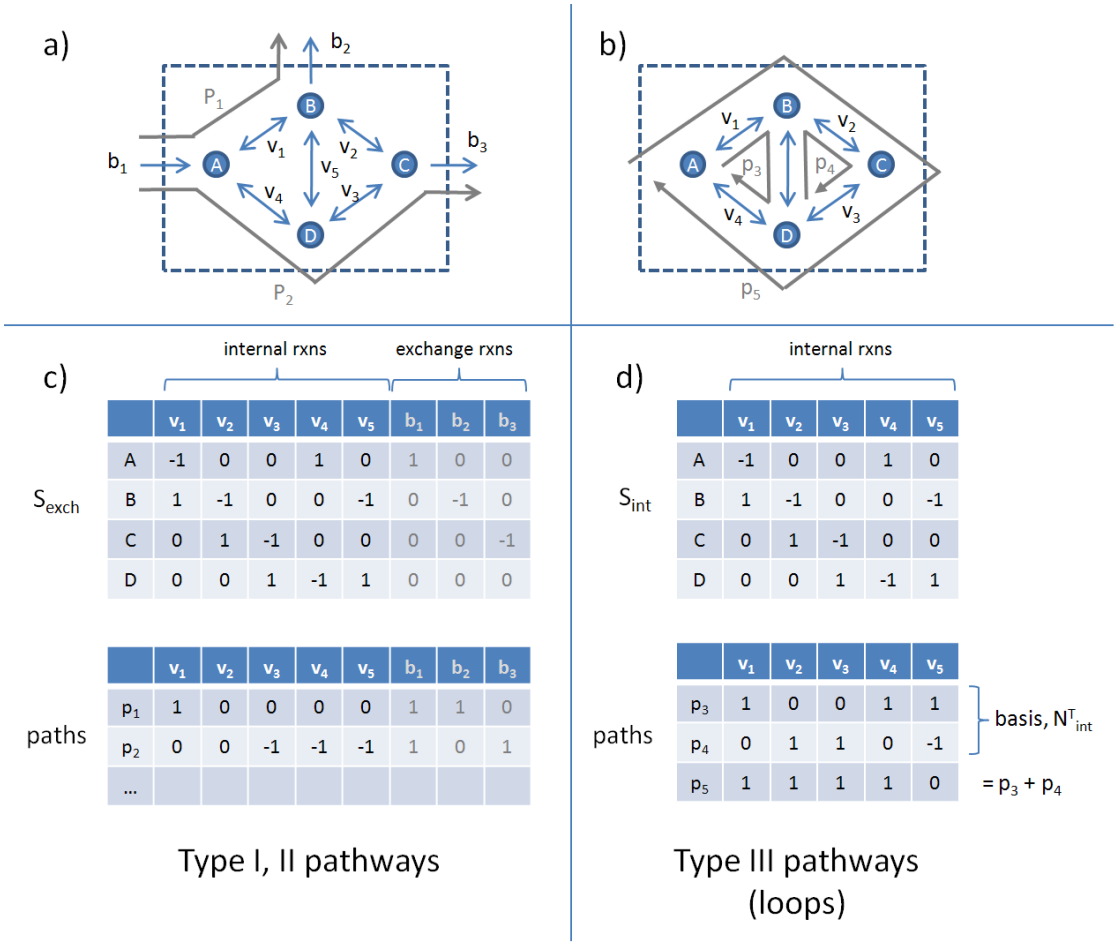


Figure 4.1: Loops in metabolic networks

A small network illustrates pathways with and without loops. A network contains five internal reactions and three exchanges (a,c). The internal part of the network is shown in (b,d). Steady state pathways are superimposed in gray. Extreme Pathway analysis [12] indicates that Type I and II pathways use exchange reactions and a partial subset listed in c). Type III pathways do not contain exchange reactions and form a set of loops. In this example there are 3 type III pathways (d). The first two pathways form a basis of the internal null space (N_{int}) and the third pathway can be written as a linear combination of the first two pathways.

flux distribution therefore involves finding a solution to $N_{int} \cdot G = 0$ with these constraints:

$$G_i < 0 \text{ for all } v_i > 0$$

$$G_i > 0 \text{ for all } v_i < 0$$

$$G_i \in \Re \text{ for all } v_i = 0$$

$$N_{int} \cdot G = 0$$

In practice, it is necessary to restrict G_i to be strictly positive or strictly negative to avoid the degenerate solution $G_i = 0$ for all i. This necessity is another reason why G_i may not be interpreted directly as ΔG_rs. The following correction restricts G_i to [-1000,-1] or [1,1000] and G_i may never be exactly 0.

$$-1000 < G_i < -1 \text{ for all } v_i > 0$$

$$1 < G_i < 1000 \text{ for all } v_i < 0$$

$$G_i \in \Re \text{ for all } v_i = 0$$

$$N_{int} \cdot G = 0$$

If a solution exists, then v contains no loops. If no solution exists, v contains a loop. Note that unlike most LP problems, the objective (max $c^T \cdot \Delta G$) is of no concern; only the feasibility is relevant.

4.3.2 Adding the Loop Law Constraints to COBRA problems: ll-FBA

The linear loop law constraints described above can be added to almost any COBRA LP, MILP, QP or MIQP problem as long as this problem contains a variable, v, for each of the (internal) fluxes in the model. The only necessary addition is the condition that ensure sign(v) = -sign(G). This is achieved by

addition of a binary indicator variable (a_i) for each internal reaction. The full set of constraints can be expressed as follows:

$$a_i = \begin{matrix} 0 \text{ if } v_i < 0 \\ 1 \text{ if } v_i > 0 \end{matrix}$$

$$G_i > 0 \text{ if } a_i < 0$$

$$G_i < 0 \text{ if } a_i > 0$$

$$N_{int} \cdot G = 0$$

This is converted to the following MILP problem:

$$-1000(a_i) + 1(1 - a_i) \leq G_i \leq -1(a_i) + 1000(1 - a_i)$$

$$-1000(a_i) \leq G_i \leq 1000(1 - a_i)$$

$$N_{int} \cdot G = 0$$

$$a_i \in 0, 1$$

$$G_i \in \Re$$

As before, G_i is now allowed to be 0 to avoid degenerate solutions. These constraints may be added to almost any LP COBRA method. For example, the full formulation for loopless Flux Balance Analysis (ll-FBA) is as follows:

$$\max_v c^T \cdot v$$

subject to

$$\sum_k S_{kj} v_k = 0$$

$$lb_j \leq v_j \leq ub_j$$

$$-1000(1 - a_i) \leq v_i \leq 1000 a_i$$

$$-1000(a_i) + 1(1 - a_i) \leq G_i \leq -1(a_i) + 1000(1 - a_i)$$

$$N_{int} \cdot G = 0$$

$$a_i \in \{0, 1\}$$

$$G_i \in \Re$$

$$i \in \text{`internal'}$$

Where S_{kj} is the stoichiometric matrix, j iterates over all reactions, i iterates over internal reactions, lb_j, ub_j are the lower and upper bounds of all reactions and cj are the coefficients of optimization.

4.3.3 Performance enhancements

Several performance considerations can be made in order to speed up this computation:

1. Null basis computation: The algorithm requires a set of basis vectors of the internal part of the stoichiometric matrix, S. $N_{int} = \text{null}(S_{int})$. By default, MATLAB will compute a dense orthonormal basis based on Singular Value Decomposition. It was found to be many fold faster to use a sparse representation of N_{int} based on a LU decomposition. As an added benefit, calculating N_{int} in this fashion can be faster than computing the orthonormal basis.

2. Elimination of unnecessary reactions: If there is no flux through a reaction, it can be eliminated from consideration for the loop law formulation. This elimination saves an indicator variable ai and its corresponding G_i. Reactions can have no flux if lb = ub = 0, which is very easy to check. A more complete check requires performing a Flux Variability Analysis (FVA) computation that is itself expensive but may be worthwhile especially if many loop removing computations will be computed. Depending on the MILP/MIQP solver used, this optimization may not bring much benefit as a pre-processor may catch these conditions automatically.

3. Combining reactions with the same direction: Often in a network reactions are coupled such that their fluxes are either both active or both inactive. This

Table 4.1: Network Description

Five models of increasing size are used throughout this paper. 1) The toy network described in Fig. 4.2. 2) The *Escherichia coli* core model [80] 3) *Helicobacter pylori* iIT341 [37] 4) *Staphylococcus aureus* iSA619 [81] 5) *E. coli* iAF1260 [82] and 6) *Homo sapiens* Recon 1 [83].

Network	**Rxns**	**Mets**	**Genes**
Toy Network	5	3	-
Core *E. coli*	95	72	137
H. pylori iIT341	554	485	339
S. aureus iSB619	743	655	619
E. coli iAF1260	2382	1668	1261
Human Recon 1	3742	2776	1905

happens for example if two reactions are part of the same linear pathway. If two reactions i and j are coupled, then it is known that $a_i = a_j$ and one variable can be used instead. G_i and G_j must remain two separate variables, however. In rare instances, two reactions are coupled inversely where $v_i = -v_j$. In this case it is possible to combine ai and a_j as $a_i = 1 - a_j$. Coupled reactions can be easily computed because their rows in N_{int} will be similar ($N_{int}(i,:) = kN_{int}(j,:)$. $k > 0$ implies positive coupling. $k < 0$ implies negative coupling).

4. Flux Variability Analysis only: It is comparatively inexpensive to compute the classic FVA procedure before going to the loopless method. As a solution pool of loopless distributions is formed, they can be checked for optimality for any of the subsequent computations. If two reactions are in a linear pathway, then a flux solution that optimally uses one reaction will also optimally use the second. This saves time by eliminating expensive MILP computations. For the *E. coli* model, the number of computations is cut by a factor of 2.

4.3.4 Models

All COBRA models used below except the toy network were previously published and the reader should refer to Table 4.1 for the references. The models were exported from the BiGG knowledgebase [38] as SBML files and imported with default parameters in the COBRA toolbox. The *S. aureus* model contains several potential biomass objective functions and all but 'biomass_SA_8a' (the default) were removed.

4.3.5 Flux Variability Analysis: ll-FVA

Loopless Flux Variability Analysis (ll-FVA) was performed by using the ll-FBA method described above and sequentially maximizing/minimizing each reaction in the model. This computation was performed both with and without the loop law constraints. Reactions where ($\max_i \min_i$) differed by more than 10^{-6} between FVA and ll-FVA were counted.

Sampling of the steady state solution space

Monte Carlo Sampling was used to generate a set of uniform flux distributions, possibly containing loops. The method is based on the Artificially Centered Hit and Run (ACHR) algorithm with slight modifications [21]. Initially a set of non-uniform pseudo-random points, called warm-up points, is generated.

In a series of iterations, each point is randomly moved, always remaining within the feasible flux space. This procedure is achieved by 1) choosing a random direction, 2) computing the limits of how far one can travel in that direction in either positive or negative direction and 3) choosing a new point randomly along this line. After many iterations, the set of points is mixed and approach a uniform sample of the solution space.

Linear Programming (LP) was used to generate warm-up points. For each point, the objective coefficients are set to a random vector with values in [-1,1]. This procedure generates a point at random corners in the solution space. The direction of movement is chosen as described in [36]. The center point of all points

is computed and the direction is the difference of a randomly selected point and the center point. This method of choosing the direction has the effect of biasing the directions in the longer directions of the solution space and speeds up the rate of mixing while maintaining sample uniformity.

One drawback of the ACHR sampler is that the termination condition is not clearly defined. Here we introduce the concept of the mixed fraction as a measure of the number of iterations required for proper mixing. A partition is created over the set of points by drawing a line at the median value with half the points on either side of the line. The mixed fraction is a count of how many points cross this line between the beginning of sampling and the end as a fraction of the total number of points. Initially the mixed fraction is 1 as all points are on the same side of the partition. When perfect mixing is achieved, each point has a 50% chance of crossing the partition line so the mixed fraction will be close to 0.5. 2000 points were generated and mixed for 1 hour at which point the mixed fraction was 0.495.

Removing Loops: ll-sampling

Sampling the loopless solution space can be implemented as a post-processing step to this Monte Carlo Sampling. Once a set of flux distributions has been generated, the loops can be removed. One of the several ways to do this is to find the nearest loopless flux (call it w_i) to a given flux distribution (v_i):

$$\min_{w} |w_j - v_j|$$

subject to:

$$\sum_{k} S_{kj} \cdot w_k = 0$$

$$lb_j \leq w \leq ub_j$$

$$-1000(1 - a_i) \leq w_i \leq 1000a_i$$

$$-1000(a_i) + 1(1 - a_i) \leq G_i \leq -1(a_i) + 1000(1 - a_i)$$

$$N_{int}G = 0$$

$$a_i \in 0, 1$$

$$G_i \in R$$

where $|w_i - v_i|$ is the distance to be minimized. If the distance metric used is the Euclidian norm (2-norm), then this becomes an MIQP problem. We minimize $(w_j - v_j)^2$. The Manhattan norm (1-norm) may be implemented as an MILP problem by introducing two helper variables $v^+$and v^- such that $|v-w| = (v^+ + v^-)$.

$$\min_w (v_j^+ + v_j^-)$$

subject to:

$$v_j^+ \geq v_j - w_j$$

$$v_j^- \geq w_j - v_j$$

$$\sum_k S_{kj} v_k = 0$$

$$lb_j \leq v_j \leq ub_j$$

$$-1000(1 - a_i) \leq v_i \leq 1000 a_i$$

$$-1000(a_i) + 1(1 - a_i) \leq G_i \leq -1(a_i) + 1000(1 - a_i)$$

$$N_{int} \cdot G = 0$$

$$a_i \in \{0, 1\}$$

$$G_i \in \Re$$

$$v_j^+, v_j^- > 0$$

4.3.6 Computer Configuration and Availability

Computations were performed in the MATLAB (Mathworks; Natick, MA) version 2009b environment with the COBRA toolbox [31] and SBML toolbox [84]. Linear and Mixed Integer Linear programming was performed by the TOMLAB/-CPLEX (Tomlab Research; Pullman, WA) package version 7.4. All computations were run on a Dell Studio XPS workstation (core i7 920 processor, 12GB ram, Windows 2003 R2, 64 bit). The parallel toolbox was used for both ll-FVA and ll-sampling application to take advantage of multi-core architecture.

The code used for the computations in this publication will be made available as part of the next release of the COBRA toolbox [31]. The optimizeCbModel (FBA) and fluxVariability (FVA) functions now have an optional flag to exclude loops. Monte Carlo Sampling is called through the new nearestLoopLessFlux function which returns the nearest loopless flux using the MILP or MIQP methods. Internally, these three methods call an addLoopLawConstraints function which adds loop law constraints to any COBRA LP, MILP, QP or MIQP problem and turn it into an ll-COBRA problem.

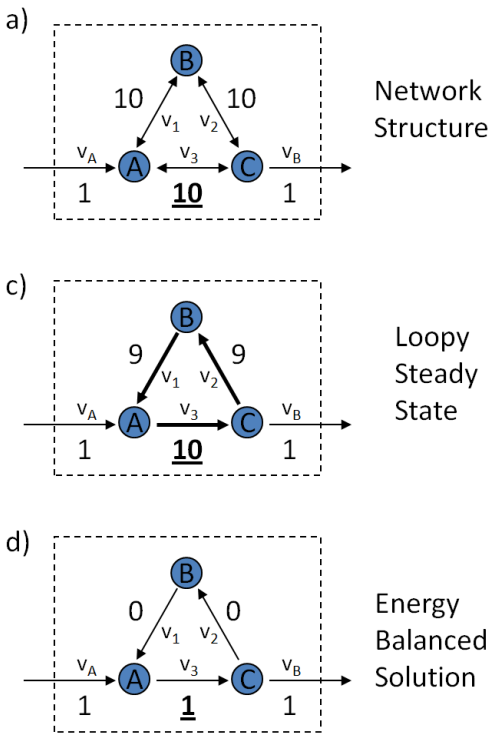
4.4 Results

4.4.1 Flux Balance Analysis with a toy network

A toy network illustrating Flux Balance Analysis is shown in Fig. 4.2, a and b. Its S matrix defines the topology of this network mathematically. Upper and lower bounds are imposed on the reaction rates. In this case the internal reactions (v_1, v_2 and v_3) are reversible and bounded in the range [-10, 10]. Metabolite A and C can be exchanged with the environment at lower rates [0,1]. This setup is common as uptake and secretion rates may be constrained to experimentally measured values whereas internal reaction bounds are often unknown. Maximizing reaction v_3 by conventional FBA yields the flux distribution shown in Fig. 4.3 c. This solution is not unique, however, all the solutions have a flux of 10 through v_3 and at least 9 units through reactions v_1, v_2 and v_3 forming a loop around $A \rightarrow B \rightarrow C \rightarrow A$. This scenario violates the second law of thermodynamics, as there can be no chemical driving force for all three reactions at the same time. The ll-FBA solution that maximizes flux through v_2 is shown in Fig. 4.2 d. In this case only 1 unit of flux through v_3 and v_1 and v_2 remain unutilized.

4.4.2 Flux Variability Analysis

While the previous example is synthetic, imposing the looplaw also has an effect on real biological networks. FVA is a common technique for evaluating the



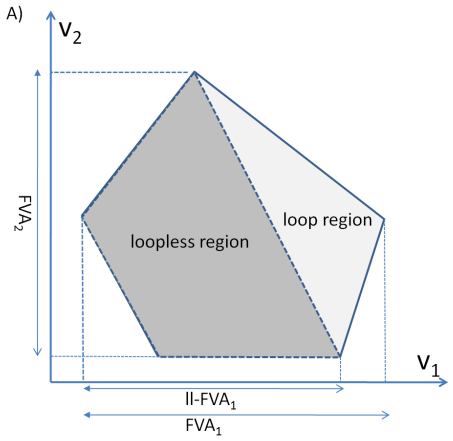
b)

S	v_1	v_2	v_3	v_A	v_B
A	-1	0	-1	1	0
B	1	-1	0	0	0
C	0	1	1	0	-1

	v_1	v_2	v_3	v_A	v_B
lb	-10	-10	-10	0	0
ub	10	10	10	1	1
C	0	0	1	0	0

Figure 4.2: A toy network with loops

A five-reaction toy network is used to illustrate the effects of the loop law. A) The structure of the network along with reaction bounds are represented graphically. Reaction v_1, v_2 and v_3 form a loop. B) The stoichiometric matrix (S), lower bounds (lb) and upper bounds (ub) mathematically describe the network. The objective coefficients (c) indicate which reaction should be maximized, in this case, reaction v_c. C) Classical FBA returns a solution that contains a loop. The objective value is 10. D) Eliminating the loop results in a lower objective of 1 unit of flux through reaction v_c and none flowing around the center loop.



Model	# rxns	FVA time (s)	FVA-loops time (s)	time/ iteration (s)	# diff	Changed Reactions
Toy Network	5	0.81	0.59	0.12	3	v1 v2 v3
Core *E. coli*	95	1.5	5.70	0.06	2	FRD7 SUCDi
H. pylori iIT341	554	11.4	119	0.21	22	4HGLSD ACKr AHSERL2 H2CO3D H2CO3D2 HCO3E HPROa HPROx HSERTA HSK METB1r NAt3_1 PHCD PHCHGS PROt2r PROt4r PTAr SHSL1r SHSL2r SHSL4r THRD_L THRS
S. aureus iSB619	730	19.5	1217	1.67	3	GLUt2 NADTRHD NAt3
E. coli iAF1260	2382	229.6	17657	7.4	69	ABUTt2pp ACACT1r ACCOAL ACKr ACOAD7f ACS ACt2rpp ACt4pp ADK1 ADK3 ADNt2pp ADNt2rpp ALATA_L CA2t3pp CAt6pp CRNDt2rpp CRNt2rpp CRNt8pp CYTDt2pp CYTDt2rpp GLBRAN2 GLCP GLCP2 GLCS1 GLCtex GLCtexi GLDBRAN2 GLGC GLUABUTt7pp GLUt2rpp GLUt4pp GLYCLTt2rpp GLYCLTt4pp HPYRI HPYRRx ICHORS ICHORSi INDOLEt2pp INDOLEt2rpp INSt2pp INSt2rpp KAT1 NAt3pp NDPK1 PPAKr PPCSCT PPKr PPM PROt2rpp PROt4pp PRPPS PTA2 PTAr R15BPK R1PK SERt2rpp SERt4pp SUCOAS THMDt2pp THMDt2rpp THRt2rpp THRt4pp TRSARr URAt2pp URAt2rpp URIt2pp URIt2rpp VALTA VPAMT
Human Recon 1	3742	609.64	99638	26.6	34	10FTHFtm 25HVITD2t 25HVITD2tm 4HGLSDm ACCOAtn ACHtn CHAT CHATn CHOLtn COAtn CYTK1n CYTK2n DCK1n DCK2n GALT H2CO3D2m H2CO3Dm HCO3Em NADHtpu NADtpu NDPK1n NDPK2n NDPK3n NDPK5n NDPK7n NDPK8n NNATr P5CRx PHCDm PHCHGSm PPDOx UMPKn VITD3t VITD3tm

Figure 4.3: Flux Variability Analysis

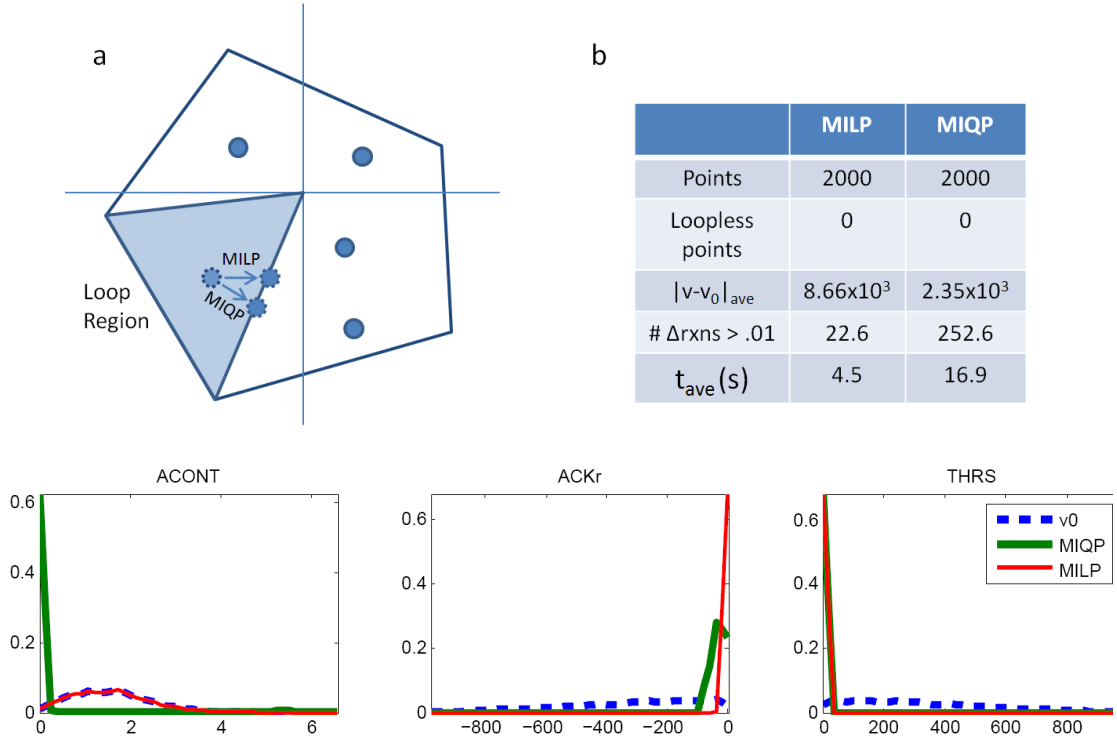
Flux Variability Analysis (FVA) computes the range of each reaction in a network. a) An illustrated example of the looped and loopless region shows that removing loops (ll-FVA) reduces the range of reaction 1 but not reaction 2. b) FVA was performed on various models using both conventional flux balance analysis and ll-FVA. The running times for both methods are listed. The number of discrepancies is listed as well as a list of reactions in which these discrepancies occur. In all cases ll-FVA is as or more constraining than classical FVA.

scope of possible metabolic states a network can achieve (Fig. 4.3 A). Each reaction is in turn minimized and maximized giving the range of use of a given reaction. Several models of increasing size were tested with FVA (Table 4.1). They range from the toy network with only 5 reactions to a whole human network containing 3742 reactions. The results of the six networks are shown in Fig. 4.3 B. In all cases, two or more reactions can have a larger range if the loop law is not imposed. Performance Considerations There is a marked reduction in computational performance when ll-FVA is computed as compared to FVA alone. MILP is known to be NP Complete and the running time is highly problem- and solver- specific. Average running times for the Tomlab/CPLEX solver are shown in Fig. 4.3. The running time of each iteration varied by several orders of magnitude (data not shown) and the average is dominated by a few very slow iterations. For a medium size problem such as the *H. pylori* model, the addition of the loop law constraints increases the computation time by a factor of 12. A different solver (GLPK) was not able to solve this problem after several hours.

4.4.3 Monte Carlo Sampling of networks

Monte Carlo Sampling of the steady state solution space is one of the techniques to study the set of feasible flux distributions a network is capable of supporting. For this study the iIT431 model was chosen. 2000 points were sampled by a modified Artificially Centered Hit and Run (ACHR) method and then post-processed to remove the loops by either the MILP or MIQP method as shown in Fig. 4.4 a and described in the Methods section. All 2000 points contain loop fluxes (Fig. 4.4 b). The MILP and MIQP methods show different properties in removing looping reactions. On average, the MILP method altered 22.6 reactions in the flux distribution, which is very close to the number of reactions with altered FVA range [72]. The MIQP method on average shifted 252.6 reactions by 0.01 units or more.

These samples allow the visualization of the solution space within which the cell operates. By plotting the samples on a histogram, the projection across the solution space can be obtained along any reaction. The projections of 3 reactions



	MILP	MIQP
Points	2000	2000
Loopless points	0	0
$\lvert v\text{-}v_0 \rvert_{ave}$	8.66×10^3	2.35×10^3
# Δrxns > .01	22.6	252.6
$t_{ave}(s)$	4.5	16.9

Figure 4.4: Monte Carlo Sampling

A random Monte Carlo Sample of 2000 points was generated for the *H. pylori* model [37]. a) Points are initially sampled from the complete steady state solution space. Points which are in the infeasible region (grey) are moved to the nearest feasible point using either a Euclidian distance (MIQP) or a Manhattan distance (MILP). b) Differences between the two samples including the average distance moved ($|v - v_0|$), and the average number of reactions moving as the point is corrected (#rxns $>$ .01). c) Histograms showing the distributions across three reactions (ACONT, ACKr, THRS) of initial points (v_0), and MILP/MIQP loopless points.

are shown in Fig. 4.4 c. For many reactions, such as ACONT, the MILP loop removal does not shift points whereas MIQP does. For other reactions, removing loops shifts the bulk of points towards higher (ACKr) or lower (THRS) fluxes.

Performance Considerations

Removing loops in the iIT431 model with 554 reactions requires about 4.5 seconds of post processing per point using the MILP method. The MIQP method requires about 16.9 seconds. It is possible to sample larger networks (iJR904 with 1075 reactions being the largest successfully sampled, results not shown) but there is a limit. The larger *E. coli* model, iAF1260 and the human reconstruction could not be sampled as described in the Methods section due to scaling issues. The MILP/MIQP solver was not able to find the nearest solution in a reasonable time.

4.5 Discussion

We have presented an enhancement to classical FBA, and other COBRA methods, that incorporate the constraints associated with the loop law by disallowing steady state flux solutions that contain closed loops. We apply this method to FVA as well as Monte Carlo Sampling as examples of many methods to which it could be applied. In both cases we are able to show that the traditional method (which allows loops) gives answers that are known to be infeasible and that ll-COBRA provides a better solution.

The COBRA framework and methods such as FBA were in part developed as an alternative to detailed kinetic models. FBA models lack of requirement for a large numbers of parameters allows them to scale to larger size. Given the utility of FBA-models, interest developed in creating hybrid methods that combine the classical FBA approach with some elements of thermodynamics. In fact, there is a spectrum of potential modeling methodologies, shown in Table 4.2, that strike different points in the tradeoff between descriptive power and computational difficulty and number of required parameters. The loop law method proposed in this paper is again a hybrid method of classical FBA and the inclusion of a subset of

Table 4.2: Comparison of Formalisms

Different methodologies for computational biology are compared according to assumptions made, parameters required, computation type and complexity. FBA "classical" Flux Balance Analysis; FBA + Loop Law the method proposed in this manuscript; FBA + thermo methods incorporating ΔG_f information but still assuming steady state [35, 73]; Kinetics Kinetic modeling where reaction rates are functions of concentrations. *Depending on implementation this may be MILP [73] or convex optimization or generalized NLP [74, 35]). ** One either has to assume mass action kinetics or know the mechanism of reaction.

Formalism	FBA	ll-FBA	FBA + thermo	Kinetics
Assumptions and Constraints	Steady State Well Mixed	Steady State Well Mixed Loop Law	Steady State Well Mixed $\Delta G = \Delta G_0 + RTln(Q)$	Well Mixed (mass action)**
Parameters	Enzyme Capacities Uptake Rates	Enzyme Capacities Uptake Rates	Enzyme Capacities Uptake Rates Met. Conc. ΔG_0	Kinetic Parameters Met. Conc. Initial Conditions (mechanism)**
Computation Type	Linear	Mixed Integer	Non Linear*	Non Linear
Computational Difficulty	Easy	Medium	Hard*	Hard

thermodynamic constraints. The benefits over classical FBA are the exclusion of loop containing fluxes. The benefits if the loopless approach is essentially "free," in the sense that it requires no additional parameters over classical FBA.

The loop law is a direct consequence of the second law of thermodynamics. Incorporating thermodynamics into COBRA methods has been an ongoing challenge. Most methods rely on ΔG_f information of metabolites [35, 73]. These quantities are often known in the literature or can be estimated computationally [77, 85]. In theory the ΔG_f along with the metabolic concentrations are all that is needed to compute the reaction directionality of reactions and as such can be exploited. One direct consequence of this formalism is that no flux distributions with loops are allowed however the thermodynamic constraints go further than that as well. The disadvantage of such methods is the requirement for accurate ΔG_f values. When these values are not available, many of the thermodynamic constraints can no longer be evaluated, with the loop law being an exception; thus the loopless method is useful. It does not require any additional input not already provided by the COBRA framework. The only reason to not include the loop law these would be computational difficulty; however as Fig. 4.3 shows, for moderately sized problems, it is quite tractable.

The reconstruction process for creating COBRA models has been described before in great detail [1, 39]. Every reaction must be evaluated by hand and among other things, its directionality determined. One of the important steps towards their construction is ensuring that they produce viable results when optimizing for physiological functions such as ATP production or biomass production. What can often occur is an under constrained model will predict unbounded internal reaction rates due to some loop. This unrealistic prediction is problematic and very often the remedy involves changing the directionality of a reaction to eliminate the loop. While this solves the problem of unbounded solutions, it may inadvertently introduce an artificial constraint on the system. All reactions are inherently reversible and marking them as irreversible may restrict the accuracy of the model under some extreme conditions. Several studies [86, 87] indicate that in the iJR904 model of *E. coli* most reactions are estimated to be near equilibrium. One reason

that loopless FVA affects so few reactions (Fig. 4.3) is precisely because the model was constructed to avoid loops. Using loopless FBA instead of classical FBA may in the future produce better models in the future because many of these artificial directionalities are simply not needed and eliminating loops when computing is more accurate than eliminating them from the network description.

This study only covers two commonly used COBRA methods and the effects that the loop law has on them. There are many other methods that could benefit from this addition. All that is required is that the method 1) be a LP, MILP or QP problem and 2) have a variable for each internal reaction. Some possible candidate methods are: Minimization Of Metabolic Adjustment (MOMA) [88], Regulatory On/Off Minimization (ROOM) [89], gene deletion studies [55], regulatory FBA [90], Flux Coupling Analysis [91] and geometric FBA [92]. Finally, there is a special case worth noting. A recent manuscript [92] uses an iterative linear approach to find the geometric center of the optimal FBA solution space and thereby return a unique solution. One consequence is that usually the solution is also free of loops. While this method is fast computationally, it fails to remove loops in cases where the objective is part of a loop, such as the toy example. By the nature of the problem, it is unlikely that any purely linear programming approach would be able to handle these cases.

4.6 Conclusion

Solutions to the classical FBA problem formulation can violate thermodynamic constraints. The addition of the loop law constraints presented in this manuscript requires no additional parameters or measurements not already present in a typical COBRA model. The benefits of eliminating thermodynamically infeasible solutions are not as great with the loopless approach as some of other thermodynamic methods. However, no additional parameters are required to implement the limitations inherent in the loop law. An easy algorithm is presented here.

The authors thank Ronan Fleming for many insightful discussions on the

topic of thermodynamics.

This work was supported in part by National Institutes of Health Grant GM068837.

The text of Chapter Four, in its entirety, is a reprint of material as it was submitted in J. Schellenberger, B.Ø. Palsson, Elimination of Thermodynamically Infeasible Loops in Steady State Metabolic Models. Biophysical J (2010 Under Review). I was the primary author of the text.

Chapter 5

The BiGG Knowledgebase

5.1 Abstract

Genome-scale metabolic reconstructions under the Constraint Based Reconstruction and Analysis (COBRA) framework are valuable tools for analyzing the metabolic capabilities of organisms and interpreting experimental data. As the number of such reconstructions and analysis methods increases, there is a greater need for data uniformity and ease of distribution and use.

We describe BiGG, a knowledgebase of Biochemically, Genetically and Genomically structured genome-scale metabolic network reconstructions. BiGG integrates several published genome-scale metabolic networks into one resource with standard nomenclature which allows components to be compared across different organisms. BiGG can be used to browse model content, visualize metabolic pathway maps, and export SBML files of the models for further analysis by external software packages. Users may follow links from BiGG to several external databases to obtain additional information on genes, proteins, reactions, metabolites and citations of interest.

BiGG addresses a need in the systems biology community to have access to high quality curated metabolic models and reconstructions. It is freely available for academic use at http://bigg.ucsd.edu.

5.2 Background

Metabolism is the structure and behavior of chemical reaction networks that occur in living organisms in order to maintain life. It is intrinsically linked to many other cellular functions and metabolic abnormalities are implicated as the cause of various diseases. Over the last 100 years, the list of reactions comprising an organisms metabolism has largely been catalogued. This reductionist process has focused on characterizing individual reactions in great detail. However, as the body of metabolic knowledge grew, so did the desire to integrate it into comprehensive models to simulate, predict and ultimately understand its behavior on a systems level. Kinetic models utilizing a system of differential equations are an established method of modeling biochemical pathways [93]. This field is an active area of research with an extensive number of models [94, 95, 96, 97] as well as computational tools [98, 99] available. Kinetic modeling suffers from the difficulty of requiring comprehensive knowledge of kinetic parameters to sufficiently define the system. The parameters have proven difficult to measure in a consistent fashion and are often unknown [100, 26]. A consequence is that the scope of kinetic models tends to be limited.

In contrast, constraint based modeling based on genome-scale metabolic reconstructions aim to include every known reaction for an organism, through the integration of genome annotation and biochemical knowledge. Reactions are defined simply by their reaction stoichiometry, and the networks are easily converted to mathematical models on which constraint-based analysis can be applied. In this paradigm, model predictions depend on constraints through reaction fluxes and an inferred metabolic objective, rather than on precisely defined kinetic parameters. Metabolic reconstructions have proven broadly useful for a number of applications. Case studies have been reviewed in [46].

In recent years, the publication of hundreds of genomes, with various databases such as KEGG [101], Biocyc [102] and Reactome [103] describing their annotation, has simplified the task of creating drafts of genome-scale metabolic reconstructions [104, 105]. This has spurred the development of an ever increasing

number of reconstructions [1, 61, 60, 106, 107, 108, 109] . It is important to note that reconstructions derived directly from genome annotation may contain several gaps or incorrect annotations, leading to errors in model predictions. In order to be useful for prediction, models must undergo multiple rounds of manual curation and testing [39] . A number of widely-used manually-curated, component-by-component (bottom-up) reconstructions of genomic and bibliomic data have been published, creating the need for a systematized biochemically, genetically and genomically structured (BiGG) knowledgebase of metabolic reconstructions.

5.2.1 Model Reconstruction Process

A general bottom-up metabolic reconstruction process has been formulated and detailed in [1, 39]. Initially, a parts list is assembled from existing databases (most notably KEGG [101], EntrezGene [110]) giving a crude reconstruction scaffold. This reconstruction is refined through an extensive review of primary literature, review articles, textbooks, and other specialized databases. A mathematical representation (S matrix) of the reconstruction is created and used to validate network structure by testing functionality, such as growth under some condition or the ability to produce a specific metabolite. Furthermore, gap analysis identifies possible missing reactions by finding so called dead end metabolites which can be produced by the network but not consumed. Failure of network validation tests and the existence of gaps suggest targeted literature searches or experiments, which can be used to improve the model. Each reaction is verified individually and a confidence score can be assigned by the curator. A model may undergo several iterative rounds of validation and changes before it reaches a satisfactory state and is published, a process which can take up to a year of time. Because of the great effort involved, there have been attempts to partially automate the process [111, 112, 113, 114, 115, 115, 116] and split work through collaboration [117].

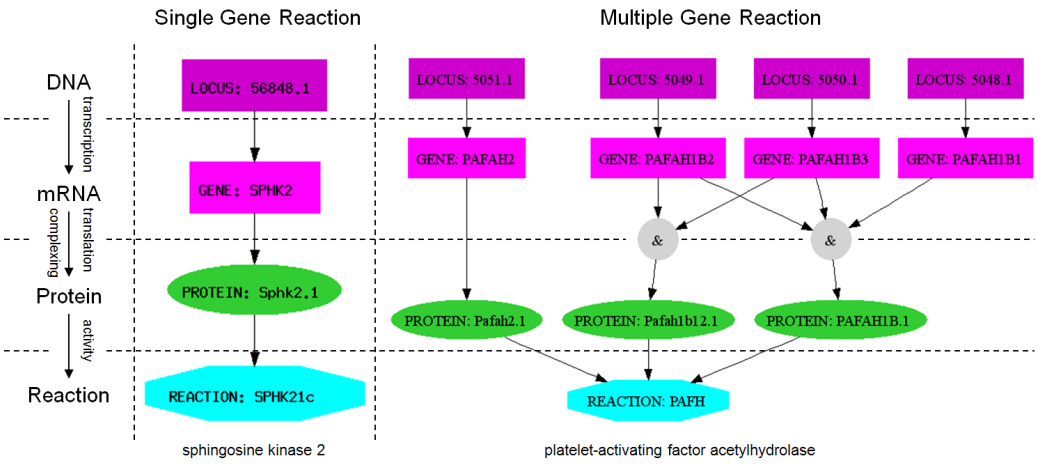


Figure 5.1: Gene Protein Reactions Statements

GPR statements are shown for two Human Recon1 reactions. Each graph indicates the relationship between genes (purple), transcripts (magenta), protein (green), and reaction (teal). A) Sphingosine kinase 2 (SPHK21c) is associated with only one gene. B) Platelet-activating factor acetylhydrolase (PAFH) can be transcribed by either gene PAFAH2 or in combination of genes PAFAH1B1, PAFAH1B2, and PAFAH1B3. The GPR expression for this reaction is (5051.1) or (5049.1 and 5050.1) or (5049.1 and 5050.1 and 5048.1).

5.2.2 Gene-Protein-Reaction associations

Most biological reactions require enzymatic catalysis to occur. Thus the on or off state of each reaction in the network is controlled by the genotype and expression level of associated genes. In the simplest case, a reaction is catalyzed by only a single enzyme which is coded for by a single gene. The expression and translation of that gene implies the feasibility of the reaction, and vice versa. More complex cases involve multiple genes and proteins whose relationship is described using Boolean logic. A single protein may be composed of subunits coded by two (or more) genes. If all of these subunits are required for the catalytic activity of the protein, the activity is modeled as an and logic (gene A and gene B). Alternatively, the model allows for equivalent proteins (isozymes) to catalyze the same reaction. In this case, the presence of either protein is sufficient to establish the activity of the reaction and an or logic is used (protein A or protein B). Other phenomena, which are representable in the Boolean framework, are alternative splicing (or logic) and obligate protein complexes (and logic). Collectively, these Boolean logic statements relating genes, proteins, and reactions are named GPRs. If a GPR statement of a reaction evaluates to true, then its corresponding reaction is said to be feasible. Thus, GPRs may be used to evaluate the effects of gene knockouts and gene regulation on the metabolic reconstructions, ruling out reactions whose genes are not available. GPRs may also be displayed graphically. Figure 5.1 shows two of the possible GPR associations found in BiGG.

5.3 Construction and content

5.3.1 Reconstructions

BiGG is currently capable of browsing and exporting the contents of seven different genome-scale reconstructions of six organisms (see Table 5.1 1): *Homo sapiens* Recon 1 [83], *Escherichia coli* iJR 904 [40] and iAF1260 [82], *Saccharomyces cerevisiae* iND750 [118], *Staphylococcus aureus* iSB619 [81], *Methanosarcina barkeri* iAF692 [119] and *Helicobacter pylori* iIT341 [37]. These reconstructions

Table 5.1: Contents of BiGG

BiGG currently contains 7 reconstructions including two versions of *E. coli*. There are a total of 7234 unique reactions and exchanges in the database. Exchange reactions carry metabolites from the extracellular 'compartment' across the system boundary and are not technically part of the metabolic reconstruction. Translocation reactions carry a metabolite between compartments (possibly performing other transformations). Reactions can be gene associated or not. Every reconstruction contains the 'cytosol' and 'extracellular' compartment. Human and yeast contain 'endoplasmic reticulum', 'mitochondria', 'peroxisome', 'nucleus'. The periplasm in iAF1260, vacuole in yeast, and lysosome in human are unique to these reconstructions.

Organism	**E. coli**		**S cerevisiae**	**H sapiens**	**H pylori**	**M barkeri**	**S aureus**	**Total**
Reconstruction	**iJR904**	**iAF1260**	**Recon 1**	**iIT341**	**iAF6982**	**iSB619**		
Reactions + exchanges	1074	2381	1265	3743	551	689	741	7234
Exchanges	143	299	116	406	74	70	84	626
translocation rxns	203	718	308	1190	82	88	108	2397
non-translocation rxns	728	1364	841	2147	395	531	549	4211
gene associated rxns	873	1944	810	2307	355	509	581	5116
non gene associated	58	138	339	1030	122	110	76	7158
non gene, non transport	40	82	123	335	83	76	49	4153
Metabolites	625	1039	646	1509	412	558	577	2556
Compartments	1+1	2+1	7+1	7+1	1+1	1+1	1+1	9+1
Genes	904	1260	750	1905	339	692	619	5582
Proteins	817	1148	713	2004	339	541	537	4526
Citations	143	325	N/A	1596	N/A	59	N/A	1982

span all three major branches of the tree of life and include two model organisms.

A global reconstruction of the human metabolic network, *H. sapiens* Recon 1, was recently completed [83]. The initial human reconstruction was based on gene information from the KEGG, EntrezGene, and H-Invitational [120] databases and was curated by evaluation of primary literature, reviews, and textbooks. Recon 1 represents a valuable tool as a scaffold for analysis of -omics data sets.

A variety of microorganisms have also been reconstructed. The *E. coli* reconstructions, iJR904 and more recently iAF1260, are the most complete and most used of these reconstructions. iJR904 has been used for the prediction of adaptive evolution endpoints [121] and the engineering of lactate producing *E. coli* strains [122]. *H. pylori*, another Gram-negative enterobacteria that lives in the human stomach and has been shown to cause ulcers and gastritis, has a reconstruction, iIT341. iAF692 is a reconstruction for the methanogenic archaebacteria *M. barkeri*. iSB619 is a reconstruction of the infectious Gram-positive bacteria *S. aureus* of interest due to high rates of infection and increasing resistance to antibiotics. As more reconstructions are published, they will be added to BiGG.

All reconstructions in BiGG were developed on the Genomatica Simpheny (TM) platform. This system includes quality control features to track genes, proteins and reactions, as well as simulation tools to computationally validate models. The models are built from a shared universal database of compounds and reactions. It is therefore not possible to incorporate reconstructions developed with other tools.

The reconstructions are stored on a Genomatica (San Diego, CA) supplied server running an OracleTM database. Access to this database is provided by a read-only client with several tables and views for accessing information on Reactions, Metabolites, Genes, Proteins, Maps and Citations (Figure 5.2). All queries are performed by a Linux/Apache/Perl Server using the CGI and DBI modules.

5.3.2 Browsing

The two main functions of BiGG are browsing content and exporting whole reconstructions. The browser is designed for querying the content and compar-

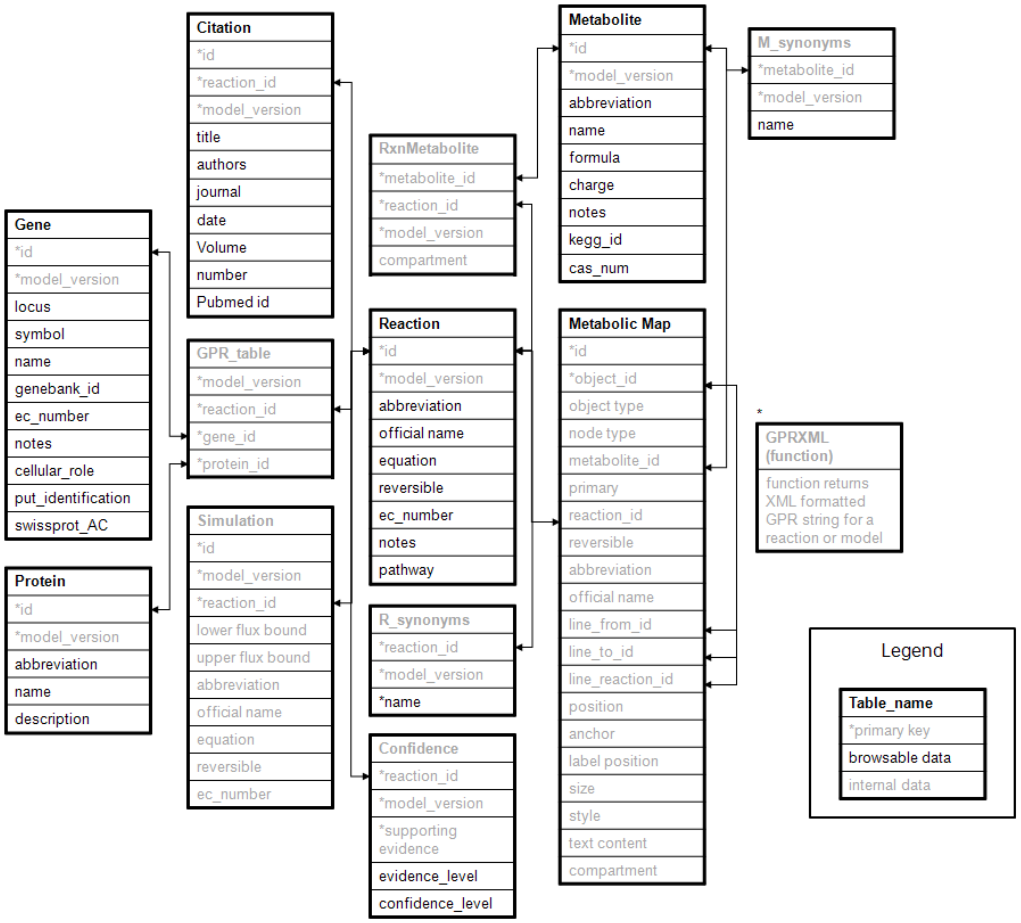


Figure 5.2: The BiGG Schema

BiGG is hosted on a Simpheny server running an Oracle database. Starred columns indicate primary keys. Arrows indicate foreign key relationships. GPR_table stores the relationship between reactions, proteins, and genes. All tables and entries shown in black are directly viewable by the user. Grey entries are used internally only. GPRXML (marked by $*$) is a function which returns the XML formatted GPR string given a reaction ID.

ing different reconstructions whereas the exporter is primarily designed to enable further computational analysis by other software packages.

The BiGG browser contains entries for metabolic reactions, metabolites, genes, proteins, and literature citations (Figure 5.2). Reaction entries contain information such as the balanced equation, compartment localization, EC number [123], reversibility, author comments, and links to references. Metabolite entries contain information such as chemical formula and charge under physiological conditions. The GPR relationships are displayed as text or graphs using the graphviz package (http://graphviz.org). Hyperlinks to other databases are included whenever provided by the authors of the reconstructions. These include NCBI Entrez gene database [110], Uniprot/Swissprot [124] for genes, and KEGG and CAS (www.cas.org) identifications for metabolites.

Reactions and metabolites can be searched through the Search Reactions and Search Metabolites pages. Reactions may be searched for by name, EC number, or associated gene. Alternatively, all reactions in a model may be listed by using the model name as the only search parameter. It is also possible to specify compartment, pathway, or metabolite participation. Results may be limited by only including reactions with known gene associations, high or low confidence, or by excluding transport reactions. In addition, reactions may be searched across reconstructions allowing for model comparison. Lists of reactions matching a set of criteria may be exported as a tab delimited flat file. The exported files can contain information for multiple models, simplifying model comparison.

Metabolites may be searched for by name, KEGG ID, CAS ID, or charge. Just as for reactions, limiting searches by compartment, pathway, and organism is possible. In addition to basic metabolite information such as formula and charge, lists of reactions in which the metabolite participates are listed and categorized by the metabolites role as a reactant or a product. Lists of metabolites matching a set of search criteria may be exported as tab delimited flat file, and contain information such as metabolite name, abbreviation, formula, KEGG ID, and CAS ID. The browser interface is shown in Figure 5.3a.

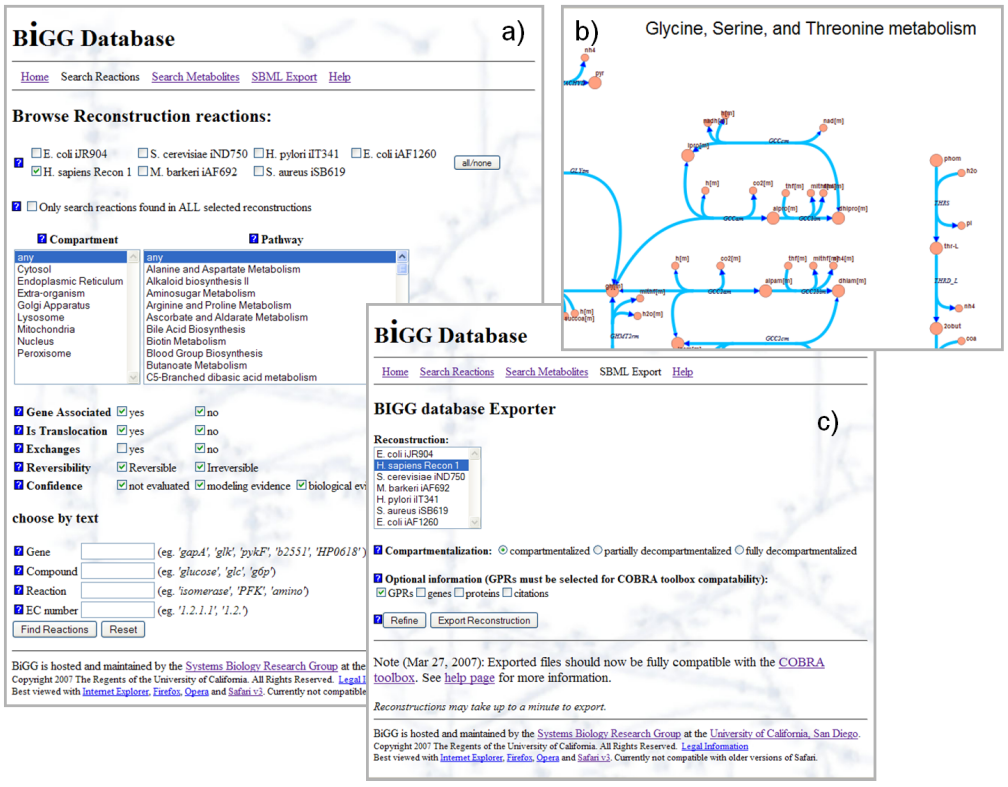


Figure 5.3: BiGG Screenshots

The BiGG knowledgebase can be accessed through a web browser. It has been tested with Mozilla Firefox, Microsoft Internet Explorer, Opera, and Safari. Three screenshots in Firefox show: (a) the content browser, (b) map viewer, and (c) the export tool.

5.3.3 Maps

For visualization, curated metabolic maps are provided for many organisms in BiGG. These maps show metabolites, reactions, and text markup. Some large reconstructions (in particular human Recon 1) have several maps for different metabolic subsystems. Maps are rendered with Scalable Vector Graphics (SVG) for smooth scaling and are hyperlinked back to the entries in the database. A small portion of a metabolic map from the human reconstruction is shown in Figure 5.3b.

5.3.4 Exporting

The second function of the BiGG knowledgebase is exporting reconstructions as Systems Biology Markup Language (SBML) files [125], which are specifically designed to work with the Matlab COBRA toolbox [31] and Systems Biology Research Tool [126] for performing flux balance analysis and other computations. This XML format is widely used for distributing systems biology models [125] . By default, only entries for compartments, metabolites (the ¡species¿ tag), and reactions are included. The user has several options available to customize export as detailed below (see Figure 5.3c for interface).

Decompartmentalization

A reconstruction may be exported as a decompartmentalized model. A compartment in a metabolic reconstruction is a distinct pool of metabolites and their reactions. Metabolites are exchanged among compartments by transporter reactions. All reconstructions included in BiGG have at least two compartments: Cytosol and Extra-organism. Additionally, reconstructions of eukaryotic organisms have internal compartmentalization modeling subcellular organelles. A partially decompartmentalized reconstruction removes these internal compartments and relocates their reactions and metabolites to the Cytosol. A full decompartmentalization removes internal compartments as well as the boundary between the Extraorganism and Cytosol compartments, creating a single-compartment system. In either case, internal transporters are removed.

It should be noted that the utility of decompartmentalization lies in model comparison rather than a basis for simulations. For instance, reactions such as ATP synthase are driven by an electrochemical gradient across compartment boundaries. In the decompartmentalized model there can be no gradients thus the ATP Synthase reaction becomes thermodynamically incorrect, creating unpredictable outcomes with some optimizations. As a rule, decompartmentalization should only be used for comparative purposes. Computational studies should only be performed on the full models.

Associated Genes, Proteins, and Citations

The exported SBML file may include information on genes, proteins and citations. Because the SBML specification does not include fields for this kind of data, this information is stored in the notes field of the reaction entries.

GPR statements. The notes field of the Reaction entries in the exported SBML file can include Boolean strings corresponding to the GPR statements. The GPR field is read and interpreted by the COBRA toolbox but should be ignored by other programs.

From Reconstructions to Models

Reconstructions are the basis for computational models. The process of converting a reconstruction into a model is performed by the curator and is reviewed in [3, 1]. Upper and lower bounds are placed on reaction rates, bounding the space of flux distributions. An objective function is added, often corresponding to the biomass production. The reconstruction, bounds and objective function together comprise the model exported by BiGG.

Most of the simulations are run by default under parameters simulating aerobic growth condition in glucose minimal medium. This is modeled by the constraints on fluxes of the models exchange reactions. For instance, modeling of an aerobic environment with glucose minimal media must allow for glucose, oxygen, ammonium ion, salts, and other ions to be up-taken but other carbon sources only excreted. These bounds are included in the SBML file along with the

objective coefficients of each reaction and flux distribution. For simulating other conditions there is a web based interface for changing the bounds on any reaction by pressing the refine button. In this way, SBML files corresponding to different media compositions can be created.

Compatibility

SBML files conform to the level 2 version 1 specification and are compatible with the COBRA toolbox [31] which contains many computational procedures. Using the COBRA toolbox, the SBML file exported from BiGG may be imported as a network data structure into Matlab. This structure includes the stoichiometric matrix, gene and reaction information, and GPR associations. The toolbox then allows the user to interrogate the models solution space using a variety of tools, including flux balance and flux variability analysis, random sampling, and robustness and gene deletion analysis. Matlab scripting can be used to combine methods or develop new methods not provided in the toolbox. The JAVA based Systems Biology Research tool [126] is another software package tested to work with the SBML files exported from BiGG.

5.4 Utility

5.4.1 Querying General Statistics of Reconstructions

The capability of BiGG to browse and compare multiple reconstructions was used to provide a comparison of the available reconstructions. The seven metabolic reconstructions available via BiGG vary in size from 551 reactions in *H. pylori*, to 3743 reactions in the human reconstruction. The total number of metabolites in BiGG is 2556, of which more than half (1509) are found in the Human Reconstruction (Table 5.1). A set of 96 core reactions is shared by all reconstructions, while most reactions were found in only one reconstruction (Figure 5.4c). Ubiquitous reactions include those involved in central metabolism, nucleotide and amino acid metabolism, and several exchange reactions (results not shown). Translocation

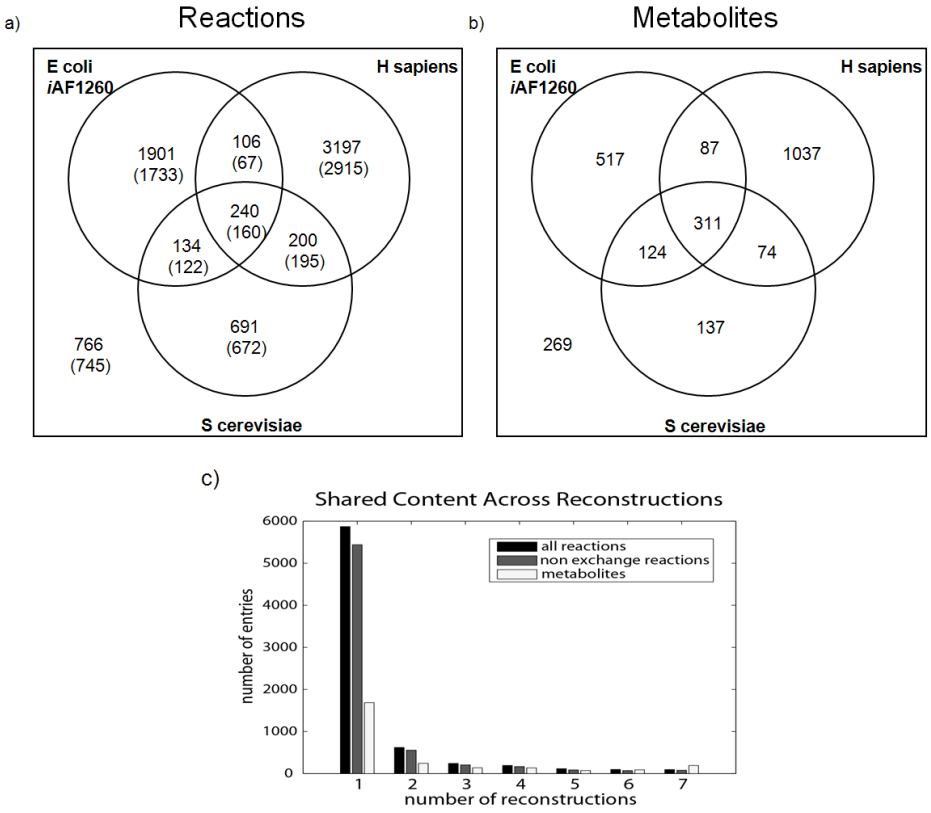


Figure 5.4: Content of BiGG

The three largest reconstructions in BiGG are *E. coli* iAF1260, *S. cerevisiae* iND750, and *H. sapiens* Recon 1. Their shared content can be queried with BiGG: a) The shared reactions. Non-exchange reactions are shown in parenthesis. b) The number of shared metabolites. c) The distribution of reactions and metabolites across all seven reconstructions.

reactions tend to be unique to particular reconstructions. The three largest reconstructions (*H. sapiens*, *E. coli*, *S. cerevisiae*) share a total of 240 reactions, 80 of which are exchange reactions (Figure 5.4a).

The content distribution usage in BiGG is shown in Figure 5.4C. Most reactions are only found in one reconstruction although 1167 are shared between at least two. Metabolites are shared more frequently. A smaller fraction of metabolites is unique to just one reconstruction (Figure 5.4 b,c).

5.4.2 Case Study - Orphan reactions

All reconstructions have knowledge gaps where information on components is not available. One example is orphan reactions which are reactions for which the catalyzing enzyme is unknown. The BiGG knowledgebase can be used to study and help fill in these knowledge gaps by 1) listing all orphan reactions and 2) displaying any other reconstruction that use these reactions. *E. coli* metabolism has been studied extensively and most of the predicted open reading frames have at least putative functional assignments. The *E. coli* metabolic network reconstruction has gone through several iterations and has become more complete [82]. The iJR904 reconstruction contains 58 orphan reactions (Table 5.2). Six are labeled spontaneous, meaning they can proceed without the aid of an enzyme and thus do not require an associated gene. A further reaction is the ATP maintenance requirement which is a virtual reaction representing the turnover of ATP to ADP to maintain cellular functions. A total of seven reactions were removed in iAF1260, including two lumped reactions (reactions which are stoichiometric representations of more complex processes) which are also not gene associated. These two have been replaced with elementary reactions. This leaves 44 reactions with missing gene associations in iJR904. Fourteen now have genes in iAF1260 while the remaining 30 do not. Twelve of these 30 are found in at least one other reconstruction, forming the basis for further searching. Analyses like these provide an overview of the state of reconstructions and can pinpoint areas of future focus. Performing this analysis without the BiGG knowledgebase would be possible although cumbersome.

Table 5.2: Orphan Reactions in *E. coli*

58 *E. coli* iJR904 orphan (non-gene associated) reactions are categorized by current status. Found in 1260 shows whether the reactions are found in reconstruction iAF1260. Y* indicates that the reaction is found in iAF1260 in modified form (usually because of the addition of the periplasm compartment). Notes: spont. -spontaneous, lump lumped reaction, virtual virtual reaction, P *H. pylori*, S *S. aureus*, Y *S. cerevisiae*, H *H. sapiens*

	abbreviation	found in 1260	name	equation	subsystem	compartment	ecnumber	reversible	translocation	Notes
spontaneous	*ATPM*	Y	ATP maintenance requirement	[c] : atp + h2o --> adp + h + pi	Unassigned	Cyt.		N	N	virtual
	G5SADs	Y	L-glutamate 5-semialdehyde dehydratase (spontaneous)	[c] : glu5sa --> 1pyr5c + h + h2o	Arginine and Proline Metabolism	Cyt.		N	N	spont.
	DHPTDCs	Y	4,5-dihydroxy-2,3-pentanedione cyclization (spontaneous)	[c] : dhptd --> h2o + hmfurn	Methionine Metabolism	Cyt.		N	N	spont.
	ACALDt	Y*	acetaldehyde reversible transport	acald[e] <==> acald[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	spont.
	CO2t	Y*	CO2 transporter via diffusion	co2[e] <==> co2[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	spont.
	H2Ot	Y*	H2O transport via diffusion	h2o[e] <==> h2o[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	spont.
	O2t	Y*	o2 transport (diffusion)	o2[e] <==> o2[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	spont.
removed	*AHC*	N	adenosylhomocysteinase	[c] : ahcys + h2o <==> adn + hcys-L	Methionine Metabolism	Cyt.	3.3.1.1	Y	N	B,H
	DKMPPD2	N	2,3-diketo-5-methylthio-1-phosphopentane degradation	[c] : dkmpp + (3) h2o --> 2kmb + for + (6) h + pi	Arginine and Proline Metabolism	Cyt.		N	N	
	PPTGS	N	Peptidoglycan subunit synthesis	[c] : uaagmda --> h + peptido_EC + udcpdp	Cell Envelope Biosynthesis	Cyt.		N	N	
	ECAP_EC	N	ECA polymerization (Ecoli)	[c] : unagamuf --> eca_EC + h + udcpdp	Cell Envelope Biosynthesis	Cyt.		N	N	
	NADDPe	N	NAD diphosphatase (extracellular)	[e] : h2o + nad --> amp + (2) h + nmn	Cofactor & Prosthetic Group Biosynth.	Ext.	3.6.1.22	N	N	
	PAPA_EC	N	Phosphatidate phosphatase	[c] : h2o + (0.02) pa_EC --> (0.02) 12dgr_EC + pi	Cell Envelope Biosynthesis	Cyt.	3.1.3.4	N	N	lump
	PEPT_EC	N	ethanolamine phosphotransferase	[c] : cmp + h + (0.02) pe_EC <==> (0.02) 12dgr_EC + cdpea	Cell Envelope Biosynthesis	Cyt.	2.7.8.1	Y	N	lump
gene found	ABUTD	Y	Aminobutyraldehyde dehydrogenase	[c] : 4abutn + h2o + nad --> 4abut + (2) h + nadh	Arginine and Proline Metabolism	Cyt.	1.2.1.19	N	N	C,H
	PPNCL2	Y	phosphopantothenate-cysteine ligase	[c] : 4ppan + ctp + cys-L --> 4ppcys + cmp + h + ppi	Cofactor & Prosthetic Group Biosynth.	Cyt.	6.3.2.5	N	N	C,B,S
	PPCDC	Y	phosphopantothenoylcysteine decarboxylase	[c] : 4ppcys + h --> co2 + pan4p	Cofactor & Prosthetic Group Biosynth.	Cyt.	4.1.1.36	N	N	C,B,S,H
	PGL	Y	6-phosphogluconolactonase	[c] : 6pgl + h2o --> 6pgc + h	Pentose Phosphate Pathway	Cyt.	3.1.1.31	N	N	P,Y,C,H
	PTRCTA	Y	Putrescine Transaminase	[c] : akg + ptrc --> 4abutn + glu-L	Arginine and Proline Metabolism	Cyt.	2.6.1.29	N	N	C
	SDPTA	Y	succinyldiaminopimelate transaminase	[c] : akg + sl26da <==> glu-L + sl2a6o	Threonine and Lysine Metabolism	Cyt.	2.6.1.17	Y	N	P,C,S
	DPCOAK	Y	dephospho-CoA kinase	[c] : atp + dpcoa --> adp + coa + h	Cofactor & Prosthetic Group Biosynth.	Cyt.	2.7.1.24	N	N	P,C,S,B,S
	TMKr	Y*	thiamine kinase	[c] : atp + thm <==> adp + h + thmmp	Cofactor & Prosthetic Group Biosynth.	Cyt.	2.7.1.89	Y	N	C*
	KDOPP	Y	3-deoxy-manno-octulosonate-8-phosphatase	[c] : h2o + kdo8p --> kdo + pi	Cell Envelope Biosynthesis	Cyt.	3.1.3.45	N	N	C
	UDCPDP	Y	undecaprenyl-diphosphatase	[c] : h2o + udcpdp --> h + pi + udcpp	Cell Envelope Biosynthesis	Cyt.	3.6.1.27	N	N	C
	A5PISO	Y	arabinose-5-phosphate isomerase	[c] : ru5p-D <==> ara5p	Alternate Carbon Metabolism	Cyt.	5.3.1.13	Y	N	C
	AC t2r	Y*	acetate reversible transport via proton symport	ac[e] + h[e] <==> ac[c] + h[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	Y, C*
	ASNt2r	Y*	L-asparagine reversible transport via proton symport	asn-L[e] + h[e] <==> asn-L[c] + h[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	Y, C*
	CYSabc	Y*	L-cysteine transport via ABC system	atp[c] + cys-L[e] + h2o[c] --> adp[c] + cys-L[c] + h[c] + pi[c]	Transport, Extracellular	Cyt., Ext.		N	Y	C*
unresolved reactions	UNK3	Y	2-keto-4-methylthiobutyrate transamination	[c] : 2kmb + glu-L --> akg + met-L	Arginine and Proline Metabolism	Cyt.		N	N	
	PMDPHT	Y	pyrimidine phosphatase	[c] : 5aprbu + h2o --> 4r5au + pi	Cofactor & Prosthetic Group Biosynth.	Cyt.		N	N	S
	MTRI	Y	5-methylthioribose-1-phosphate isomerase	[c] : 5mdr1p <==> 5mdru1p	Arginine and Proline Metabolism	Cyt.	5.3.1.23	Y	N	
	MDRPD	Y	5-Methylthio-5-deoxy-D-ribulose 1-phosphate dehydratase	[c] : 5mdru1p --> dkmpp + h2o	Arginine and Proline Metabolism	Cyt.		N	N	
	MTRK	Y	5-methylthioribose kinase	[c] : 5mtr + atp --> 5mdr1p + adp + h	Arginine and Proline Metabolism	Cyt.	2.7.1.100	N	N	
	ALATA_L	Y	L-alanine transaminase	[c] : akg + ala-L <==> glu-L + pyr	Alanine and Aspartate Metabolism	Cyt.	2.6.1.2	Y	N	Y,B,H
	CMPN	Y	CMP nucleosidase	[c] : cmp + h2o --> csn + r5p	Nucleotide Salvage Pathway	Cyt.	3.2.2.10	N	N	
	DNMPPA	Y	Dihydroneopterin monophosphate dephosphorylase	[c] : dhpmp + h2o --> dhnpt + pi	Cofactor & Prosthetic Group Biosynth.	Cyt.		N	N	P,Y,S
	DKMPPD	Y	2,3-diketo-5-methylthio-1-phosphopentane degradation	[c] : dkmpp + h2o + o2 --> 2kmb + for + (2) h + pi	Arginine and Proline Metabolism	Cyt.		N	N	
	SSALx	Y	succinate-semialdehyde dehydrogenase (NAD)	[c] : h2o + nad + sucsal --> (2) h + nadh + succ	Arginine and Proline Metabolism	Cyt.	1.2.1.24	N	N	B,H
	NADPPPS	Y	NADP phosphatase	[c] : h2o + nadp --> nad + pi	Cofactor & Prosthetic Group Biosynth.	Cyt.		N	N	
	NMNN	Y	NMN nucleosidase	[c] : h2o + nmn --> h + ncam + r5p	Cofactor & Prosthetic Group Biosynth.	Cyt.	3.2.2.14	N	N	
	NMNDA	Y	nicotinamide-nucleotide amidase	[c] : h2o + nmn --> nh4 + nicrnt	Cofactor & Prosthetic Group Biosynth.	Cyt.	3.5.1.42	N	N	
	BPNT	Y	3',5'-bisphosphate nucleotidase	[c] : h2o + bpn --> amp + pi	Cysteine Metabolism	Cyt.	3.1.3.7	N	N	Y,H
	PDXPP	Y	Pyridoxine 5-phosphate phosphatase	[c] : h2o + pdx5p --> pi + pydxn	Cofactor & Prosthetic Group Biosynth.	Cyt.		N	N	H
	HYPOE	Y	hypothetical enyme	[c] : h2o + pyam5p --> pi + pydam	Cofactor & Prosthetic Group Biosynth.	Cyt.		N	N	H
	PYDXPP	Y	Pyridoxal 5-phosphate phosphatase	[c] : h2o + pydx5p --> pi + pydx	Cofactor & Prosthetic Group Biosynth.	Cyt.		N	N	H
	FFSD	Y	beta-fructofuranosidase (invertase, saccharase)	[c] : h2o + suc6p --> fru + g6p	Alternate Carbon Metabolism	Cyt.	3.2.1.26	N	N	
	MME	Y	methylmalonyl-CoA epimerase	[c] : mmcoa-R <==> mmcoa-S	Alternate Carbon Metabolism	Cyt.	5.1.99.1	Y	N	
unresolved transports	12PPDt	Y*	S-Propane-1,2-diol facilitated transport	12ppd-S[e] <==> 12ppd-S[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	
	DAPabc	Y*	M-diaminopimelic acid ABC transport	26dap-M[e] + atp[c] + h2o[c] --> 26dap-M[c] + adp[c] + h[c] + pi[c]	Transport, Extracellular	Cyt., Ext.		N	Y	
	ASNabc	Y*	L-asparagine transport via ABC system	asn-L[e] + atp[c] + h2o[c] --> adp[c] + asn-L[c] + h[c] + pi[c]	Transport, Extracellular	Cyt., Ext.		N	Y	
	DHAt	Y*	Dihydroxyacetone transport via facilitated diffusion	dha[e] <==> dha[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	
	ETOHt2r	Y*	ethanol reversible transport via proton symport	etoh[e] + h[e] <==> etoh[c] + h[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	
	GUAt	Y*	Guanine transport	gua[e] <==> gua[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	H
	PYRt2r	Y*	pyruvate reversible transport via proton symport	h[e] + pyr[e] <==> h[c] + pyr[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	H
	HYXNt	Y*	Hypoxanthine transport	hxan[e] <==> hxan[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	H
	NMNt7	Y*	NMN transport via NMN glycohydrolase	h2o[c] + nmn[e] --> h[c] + ncam[c] + r5p[c]	Transport, Extracellular	Cyt., Ext.	3.2.2.14	N	Y	
	NACUP	Y*	Nicotinic acid uptake	nac[e] --> nac[c]	Transport, Extracellular	Cyt., Ext.		N	Y	
	XANt	Y*	xanthine reversible transport	xan[e] <==> xan[c]	Transport, Extracellular	Cyt., Ext.		Y	Y	S

5.5 Future Development

5.5.1 Additional reconstructions

Two notable reconstructions in development are *Bacillus subtilis* and *Haemophilus influenzae*. As they become available, they will be added to BiGG as well. Currently, only *E. coli* has more than one reconstruction version, but in the future, we plan on hosting different (older) versions of other reconstructions as well. Currently, it is only possible to host reconstructions created within the Simpheny software and at the moment there is no way to import other groups reconstructions. This may change in the future.

5.5.2 Downloadable Maps

The BiGG knowledgebase is designed to work with the COBRA toolbox. Version 2.0 of this toolbox will be released soon and will include a visualization component. The BiGG maps will be downloadable in a custom text format containing coordinates of all metabolites and reaction control points. This is imported into COBRA and displayed in a customizable fashion. Colors and sizes can be changed on a per-reaction basis to visualize biological results.

5.5.3 Pre-made constraints / Media

At present, exported models contain one set of lower and upper flux bounds. Lower bounds of irreversible reactions are automatically set to 0 and upper bounds are either set to arbitrarily large values (eg. 999999) or physiologically determined rates. However, to run meaningful simulations, the bounds of the exchange fluxes must be specified to match the environment. For instance, modeling of an aerobic environment with glucose minimal media must allow for glucose, oxygen, ammonium ion, and salts to be taken up, but not other extracellular species. Currently, this must be done manually via the export "refine" button, but in the future, libraries of bounds (constraint) vectors will be added to the SBML files to allow the user to specify media conditions.

5.6 Discussion and Conclusions

The reconstructions and models in BiGG have several specific features necessary to compute within the COBRA framework:

1. Each reconstruction in BiGG is manually curated. Exotic transformations unique to an organism may be absent from databases and must be pulled from primary literature.

2. BiGG uses both genetics and literature based data to assess whether a reaction is present. If the genetic basis for a reaction is unknown but the reaction is described in the literature, it will be included without associated genes (an orphan reaction).

3. The curators of BiGG reconstructions have the option of providing confidence levels for reactions which can be used when evaluating resultant models. These levels, along with reaction notes, provide an assessment of the confidence that a reaction is correctly included in the model.

4. Boolean relationships between genes, proteins and reactions (GPRs) are described in BiGG. This information is necessary for the proper modeling of mutations or gene knockouts.

5. All reactions in BiGG are mass and charge balanced. In some metabolic databases, simple species such as H^+ and H_2O are simply ignored [127]. Failure to balance reactions can result in unrealistic metabolic predictions.

6. Compartmentalization in BiGG gives an accurate description of reactions involving membrane transporters. This is required for simulation of gradient driven pump [82].

7. BiGG bridges the gap between a reconstruction and a model. The exported SBML files have all been validated and can be used to make predictions about growth rate, predicting the effect of gene deletions (MOMA [88]), and other COBRA framework methods.

Taken together, these 7 features allow BiGG to represent metabolic reconstructions and the underlying chemistry in an accurate way. While individually these features are not unique to BiGG, no other resource including all of these features. The content of other genome-scale metabolic databases cannot be used directly for modeling in the COBRA framework [60].

The advent of genome sequencing has led to an explosion of systems biology methods which attempt to study properties of whole networks rather than individual parts. The results (often referred to as emergent properties) cannot be explained by studying the individual parts separately. Due to the scale of the models used, they are quite time consuming to develop and it is beneficial to share them with other researchers. The BiGG knowledgebase provides the first collection of curated, high quality metabolic reconstructions suitable for study with COBRA methods. We expect it to continue to be a useful resource in the future as new and updated models are added to the database.

5.7 Availability and Requirements

The BiGG knowledgebase is available online at http://bigg.ucsd.edu/. A JavaScript enabled browser is required for browsing and exporting. The map viewer requires SVG support. BiGG data and results require a password which is made freely available for academic use.

The text of Chapter Five, in its entirety, is a reprint of material as it appears in J. Schellenberger, Park, J. O., Conrad, T. C., and Palsson, B. Ø., BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions, BMC Bioinformatics, 11:213, (2010). I was the lead developer, and primary author of this manuscript.

Chapter 6

COBRA Toolbox Version 2.0

6.1 Abstract

We present a major update to the COnstraint-Based Reconstruction and Analysis (COBRA) Toolbox, a MATLAB package for manipulating and analyzing genome-scale metabolic models. Over the last decade, a growing community of researchers has used the COBRA framework to predict metabolic phenotypes such as cell growth rate. Version 2.0 of the Toolbox expands the scope of possible simulations by including methods developed since its original release. New functions include: (1) network gap filling, (2) C13 analysis, (3) metabolic engineering, (4) omics-guided analysis, and (5) visualization. As with the first version, the COBRA Toolbox reads and writes Systems Biology Markup Language (SBML) formatted models. In this release, we improved performance, usability, and the level of documentation. A suite of test scripts can now be used to learn the core functionality of the Toolbox and validate results. This Toolbox lowers the barrier of entry to use powerful COBRA methods and makes them more accessible to non-technical researchers.

6.2 Introduction

COnstraints-Based Reconstruction and Analysis (COBRA) methods have been successfully employed in the field of microbial metabolic engineering [128, 62, 46] and have begun to be extended to modeling transcriptional [33, 34, 129] and signaling [130, 131] networks and the field of public health. Specifically, COBRA methods have been used to guide metabolic pathway engineering, model pathogens [81], model host-pathogen interactions and the impact of disease states on human metabolism [21]. A wide variety of COBRA methods have been developed over the years [3, 31]. COBRA methods have been employed in hundreds of research articles over the years [132, 60, 133].

The COBRA approach focuses on employing physicochemical constraints to enumerate the set of feasible states for a biological network in a given condition (Figure 6.1A). These constraints include compartmentalization, mass conservation, molecular crowding [134], and thermodynamic directionality [86, 87]. More recently, transcriptome data have been used to reduce the size of the set of computed feasible states [135, 136]. Although COBRA methods may not provide a unique solution, they provide a reduced set of solutions that may be used to guide biological hypothesis development [137]. The COBRA Toolbox provides researchers with a high-level interface to a variety of COBRA methods. Detailed descriptions of COBRA methods can be found in a variety of reviews [3, 68, 138].

The biological network models that are analyzed with COBRA methods are constructed in a bottom-up fashion from bibliomic and experimental data and thus represent Biochemically, Genetically, and Genomically (BiGG) consistent knowledgebases [38]. BiGG knowledgebases are manually-curated 2-D genome annotations that relate biological functions, such as metabolic reactions, to the genome through the use of the gene-protein-reaction formalism [38, 1] (Figure 6.1B). Application of the BiGG formalism to metabolism has been particularly successful, and metabolic reconstructions are available for many organisms [82, 139, 140, 141, 142]. A detailed protocol describing the construction of high-quality BiGG knowledgebases for metabolism, and their transformation into mathematical models has been

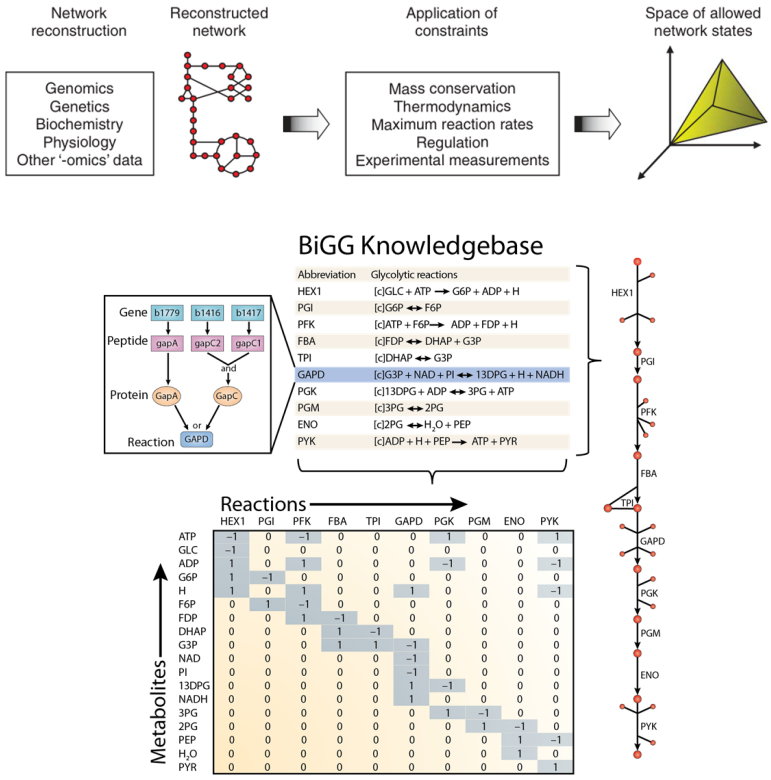


Figure 6.1: The philosophy of Constraints-Based Reconstruction and Analysis (A) Constraints-Based Reconstruction and Analysis of biological networks involves the creation of network models from a variety of biological data sources. The capabilities of the model are then assessed in the context of physical, chemical, regulatory, and omics constraints (Reproduced from Becker et al.). (B) COBRA models are often derived from BiGG knowledgebases which are essentially 2-D annotations of the genome that relate metabolic activity to genomic loci. (left inset) In Escherichia coli, the glyceraldehyde-3-phosphate dehydrogenase (GAPD) activity can be provided by two isozymes (GapA or GapC); GapC is a heteromeric protein that requires genes from two genomic loci. The contents of a BiGG knowledgebase can be converted to a map (right) to facilitate visual interpretation. Or a mathematical modeling formalism to develop and explore hypotheses, such as a stoichiometric matrix (bottom) that can be used to explore mass flow through the network. (Modified reproduction from Reed et al.).

Table 6.1: Features of the COBRA Toolbox 1.0 and 2.0

	COBRA 1.0	New in COBRA 2.0
FBA	• Growth-rate optimization • Robustness analysis • Gene deletion studies • Flux variability • MOMA	• Loop Law • Geometric FBA
Fluxomics	• None	• C13 data fitting and flux estimation • Experimental design
Gap Filling	• None	• detectDeadEnds • gapFind • growthExpMatch
Input / Output	• Read/write SBML (Level 2 Version 1)	• Read/write SBML (Level 2 Version 4)
Metabolic Engineering	• None	• optKnock • optGene • GDLS
Reconstruction	• Model curation tools	• Create sub models (GIMME & Shlomi method)
Sampling	• Artificial centering hit and run (ACHR) sampling	• Updated ACHR sampling (parallel/multi-point)
Test Suite	• None	• Examples • Verify installations

Visualization	• None	• Display maps
		• Overlay data
		  ○ Flux
		  ○ Flux variability
		  ○ Concentration

recently published [39].

The first release of the COBRA Toolbox in 2007 provided access to a variety of methods, including flux balance analysis, gene essentiality analysis, and minimization of metabolic adjustment analysis (Table 6.1). Since the release of the first version of the COBRA Toolbox, many additional COBRA-related methods have been published [92, 143, 144, 145, 146]. In version 2.0 of the COBRA Toolbox, we have extended the capabilities to include: geometric FBA [92], Loop law, creation of context-specific subnetwork models using omics data [31, 136] Monte Carlo sampling [3, 21, 9, 16], C13 fluxomics, gap filling [143, 147], metabolic engineering [144, 145, 146], and visualization of computational models of metabolism (Table 6.1 / Figure 6.2). This protocol aims to provide researchers with the ability to use the *in silico* methods included in the Toolbox with only high-level knowledge of the algorithms. Because of the wide range of creative uses for COBRA methods, not all of the Toolbox's capabilities are described in this protocol; additional

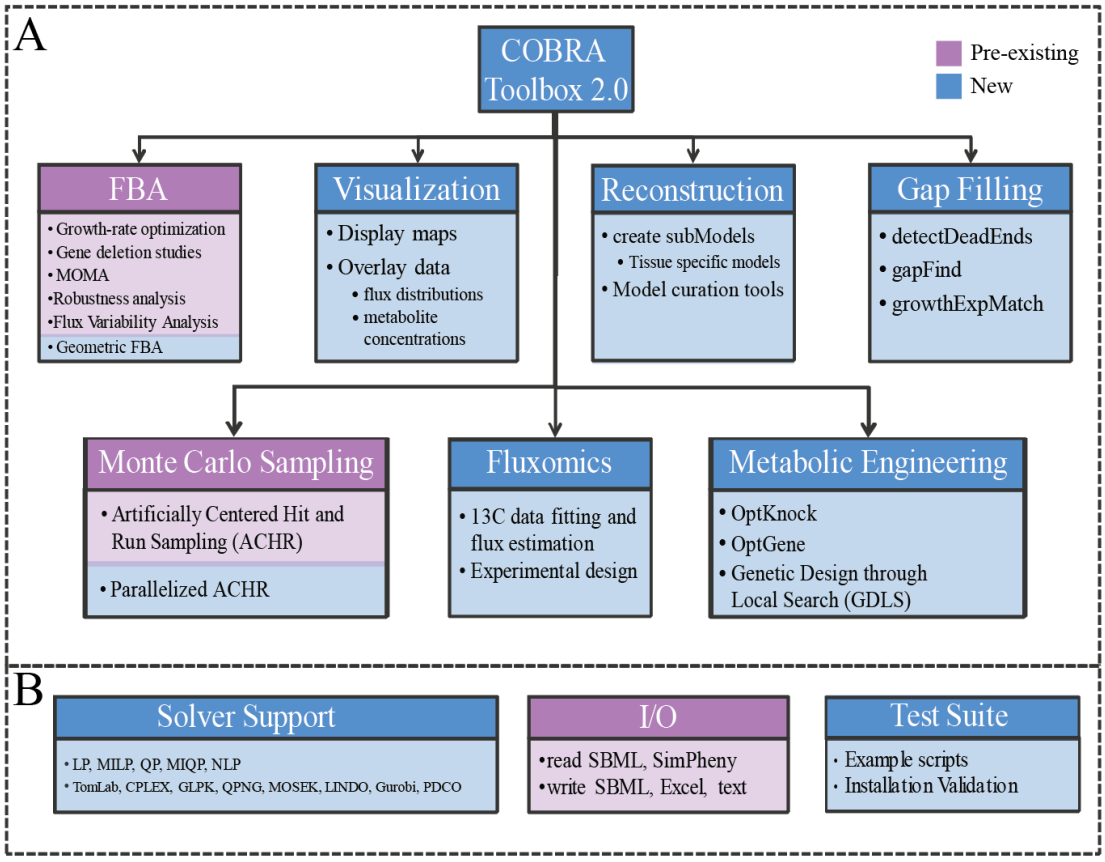


Figure 6.2: Overview of the COBRA toolbox.

(A) Seven categories of COBRA methods contained within version 2.0 of the COBRA Toolbox. (B) The COBRA Toolbox contains solver interface functions for linear, quadratic, mixed integer linear and quadratic, and nonlinear programming problems. Functions to read and write models in several formats are available. A test suite is included to validate installation as well as provide example implementation of many methods.

functionalities are described in the documentation.

The COBRA Toolbox supports models in the Systems Biology Markup Language (SBML) format [125, 148, 84]. Importation of the models into MATLAB is dependent on libSBML [148] and SBML Toolbox [84]. Because SBML does not yet provide complete support for a few key COBRA parameters, we provide an explicit description of the COBRA extensions to SBML below and in the Supplementary Material. The COBRA Toolbox is available for download from our website (www.cobratoolbox.org).

6.3 Materials

6.3.1 Equipment

- A computer capable of running MATLAB
- Version 7.0 or above of MATLAB (Mathworks Inc.) numerical computation and visualization software (http://www.mathworks.com)
- libSBML programming library 4.0.1 or above
- SBML Toolbox version 3.1.1 or above for MATLAB to allow reading and writing models in SBML format
- The COBRA Toolbox version 2.0 or above
- A linear programming (LP) solver. Currently the COBRA Toolbox supports:
  - CPLEX (ILOG Inc.) through Tomlab (Tomlab Optimization Inc. http://tomopt.com)
  - Gurobi (Gurobi Optimization. http://www.gurobi.com) through Gurobi Mex (http://www.caam.rice.edu/ wy1/gurobi_mex)
  - GLPK (http://www.gnu.org/software/glpk) through glpkmex (http://glpkmex.sourceforge.net)
  - Mosek (MosekApS. http://www.mosek.com)

  - LINDO (LINDO Systems Inc. http://www.lindo.com)
  - PDCO (Stanford University http://www.stanford.edu/group/SOL/software/pdco.html)

- A quadratic programming (QP) solver. (optional) Currently the COBRA toolbox supports:
  - CPLEX (ILOG Inc.) through Tomlab (Tomlab Optimization Inc. http://tomopt.com)
  - QPNG (part of GLPK)
  - Mosek (MosekApS. http://www.mosek.com)
  - PDCO (Stanford University http://www.stanford.edu/group/SOL/software/pdco.html)

- A nonlinear programming (NLP) solver. (optional) Currently the COBRA toolbox supports:
  - SNOPT through Tomlab

6.3.2 Equipment Setup

Installation

Microsoft Windows and GNU/Linux: Install MATLAB, libSBML, SBML Toolbox, and one or more of the supported solvers following the software provider's installation instructions. To install the COBRA Toolbox, unzip the COBRA toolbox files to the desired MATLAB accessible local directory.

Mac OS X: run the COBRA toolbox installer for Mac OS X.

COBRA-compliant SBML file

Documentation on the SBML standard is available on the SBML website (http://sbml.org) and in the Supplementary Material. Sample models in COBRA-compliant SBML may be downloaded from the BiGG knowledgebase

(http://bigg.ucsd.edu) [38]. The model files must include the following information for all calculations: stoichiometry of each reaction, upper and lower bounds of each reaction and objective function coefficients for each reaction.

Several functions within the toolbox [145, 146] require information that is not, yet, in the SBML standard or scheduled for removal in SBML 3 and beyond. The gene-reaction associations are essential for relating the metabolic reactions to the genome and the subsystem is useful for ontological classification. Metabolite formulas and charges, are necessary to make sure the model is physically consistent (no generation of mass or energy). Additional annotation parameters, such as KEGG or CAS IDs, should be specified in the notes field.

<reaction>
...
<notes>
<html xmlns=http://www.w3.org/1999/xhtml>
GENE_ASSOCIATION: ((gene1) and (gene2)) or (gene3)
SUBSYSTEM: Transport Inner Membrane
KEGGID:
...
</html>
</notes>
</reaction>
<metabolite>
... <notes>
<html xmlns=http://www.w3.org/1999/xhtml>
FORMULA: C6H12O6
CHARGE: 0
CAS:
</html>
</notes>
</metabolite>

Metabolic map files

The visualization tools require text files of the coordinates for placing metabolites and reactions on a map. Maps of many reconstructions are available from the BiGG knowledgebase. The COBRA Toolbox relates COBRA SBML models to the map coordinate files via the reaction and metabolite ids. A map file for glycolysis may be used with different organism SBML models as long as the identifiers match.

6.4 Procedure

Notes on nomenclature: italics denotes a parameter that is supplied to a function. A bracketed [parameter] is optional. >> denotes the Matlab command line; anything following >> is meant to be entered on the command line. All time estimates for the functions are predicate on a model of about 1200 genes, 2300 reactions, 1800, metabolites, and a 2.4 GHz Intel Core 2 Duo processor.

6.4.1 Initializing the Toolbox

Install MATLAB, the SBML and COBRA Toolbox, initialize paths and check solver availability. Microsoft Windows and GNU/Linux users should open MATLAB, navigate to the directory where you installed the Toolbox:

```
>> initCobraToolbox()
```

and save the paths added if desired.

Mac OS X users: This is performed for you during installation.

6.4.2 Changing COBRA solvers

Set the solvers used by the COBRA Toolbox using the following function:

```
>> changeCobraSolver(solverName, [solverType]);
```

Type the name of the solver to use as the solverName input and the solver type ('LP', 'MILP', 'QP', 'MIQP', 'NLP') for solverType. By default, solvertype is set to

'LP'. The COBRA toolbox currently supports 'LINDO', 'glpk', 'Mosek', 'Gurobi', and 'CPLEX' through Tomlab. When changeCobraSolver is called without any arguments, it will return the names of the current solvers.

6.4.3 Run COBRA Toolbox test suite

The test suit contains scripts that test the functionality of scripts within the toolbox. The scripts in the the testing directory provide useful examples of all the Toolbox's functions.

```
>> testAll()
```

This script sequentially navigates the test suite directory and runs each test. Upon completion, it displays which tests were completed successfully and which failed. If any tests fail see the TROUBLESHOOTING section. [$\sim 10^3$ s]

6.4.4 Read COBRA-compliant SBML models into MATLAB

Load a COBRA-compliant model into MATLAB. To load a model, navigate within MATLAB to the directory containing the model and call the following function from the command window:

```
>> model = readCbModel([filename]);
```

When called with no arguments, readCbModel will prompt the user to select a file using a dialog box. Currently readCbModel fully supports SBML-formatted (Level 2 versions 1 or 4) and SimPheny proprietary format files. SBML files for a variety of organisms are available from the BiGG knowledgebase(http://bigg.ucsd.edu)27. The function returns a COBRA Toolbox model structure containing the necessary fields to describe the model for use with subsequent steps. See Supplementary Material for a description of the fields in a COBRA Toolbox model structure; hereafter, model denotes a COBRA Toolbox model structure. [$\sim 10^2$ s]

Critical Step

If the model is not properly loaded into MATLAB, none of the following functions will work. See TOUBLESHOOTING for tips on solving this issue. Ensure that libSBML and SBML Toolbox are properly installed and accessible by MATLAB and that the SBML file is formatted correctly. See also: testReadWrite().

6.4.5 Modify COBRA toolbox models

Once the model is loaded into MATLAB by readCbModel, the model can be modified to simulate different conditions. Reaction bounds can be modified using the following function:

```
>> model  = changeRxnBounds(model, rxnNameList, value, boundType);
```

The reactions in the cell array of strings rxnNameList will have their bounds changed. boundType specifies which bounds to change for the reactions and can take values of 'l', 'u', or 'b' for lower, upper, or both, respectively.

The objective of the constraint-based model is necessary to determine the flux distribution which optimizes the objective, often a cellular biomass function50. To change the objective function, use the following function:

```
>> model = changeObjective(model, rxnNameList, [objectiveCoeff]);
```

rxnNameList is either a string or a cell array of strings containing reaction(s) that should be included in the objective. The coefficients of the objective vector for the reactions in rxnNameList are set to the value specified in objectiveCoeff. All other values in the objective vector are set to zero. The objectiveCoeff vector contains values corresponding to the reactions in rxnNameList. If left empty, objectiveCoeff is assumed to be 1.

Knock-out or knock-in strains can be simulated by removing or adding the reactions. New reactions can be added to a COBRA toolbox model using the following function:

```
>> [model] = addReaction(model, rxnName, metaboliteList,
  stoichCoeffList, [revFlag], [lowerBound], [upperBound],
  [objCoeff], [subsystem], [grRule], [geneNameList],
  [systNameList], [checkDuplicate]);
```

metaboliteList and stoichCoeffList contain the metabolites and their coefficients, respectively. Additional information for the reaction may be specified using the subsequent input variables. By default the function checks for reactions with the same name or stoichiometic coefficients, however this can be disabled by setting checkDuplicate to false.

To remove a reaction, call the following function:

```
>> [model] = removeRxns(model, rxnRemoveList)
```

Reactions listed in the rxnRemoveList cell array of strings will be removed. By default, the function removes metabolites that are not involved in any reactions. Note that the model may no longer function after reactions have been removed. For more examples on model manipulation see: testModelManipulation().

6.4.6 Saving the model

COBRA Toolbox model structures may be saved as text, xls, or SBML.

```
>> writeCbModel(model, format, [fileName], [compSymbolList],
  [compNameList], [SBMLLevel], [SBMLVersion]);
```

The name of the output file may be set using the fileName input. By default, the function opens a dialogue box, prompting the user to specify where to save and the name of the file. This feature is dependent on the SBML toolbox to generate the xml file. The toolbox is able to output SBML level 2 versions 1 or 4. See also: testReadWrite().

6.4.7 Omics-Guided Creation of Context-Specific Models

An emerging application of genome-scale reconstructions is integrating high-throughput in a systems context [31, 136]. In particular, this procedure is useful for building cell-, tissue-, or condition-specific models. createTissueSpecificModel

is designed to map transcriptomic or proteomic data onto a reconstruction using two established algorithms (GIMME [31] or Shlomi [136]). The GIMME algorithm is an LP procedure that matches high-throughput data to an original flux distribution derived from the full model; thus the algorithm requires a predefined objective function. Alternatively, the Shlomi algorithm is an MILP problem that matches high-throughput data to pathway length, thus avoiding the need for a predefined objective function. Novice users can utilize the GIMME algorithm with two inputs: the COBRA model and expression data; while more experienced users can tweak many more parameters (see Documentation).

```
>> [tissueModel,Rxns] = createTissueSpecificModel(model,expressionData,
  [proceedExp],[orphan],[exRxnRemove],[solver],[options],
  [funcModel]);
```

model is a reconstruction with gene-protein-reaction associations. expressionData is a structure that contains two inputs: .Locus (a vector of GeneIDs matching with the model), and .Data (a vector of presence/absence calls). If using multiple data sets for which the presence/absence calls have not been averaged, proceedExp can be set to 0 for the data to be processed (by default proceedExp = 1). Note: when setting proceedExp = 0, a third input (.Transcript) is required in the expressionData structure (see createTissueSpecificModel Documentation).

Reactions with no known gene-protein-reaction associations are called orphan reactions. orphan controls whether or not these reactions are included with Shlomi-. By default orphan = 1 and orphan reactions are included; orphan reactions are always included in the GIMME algorithm. Select exchange reactions can be excluded from the model by including them in exRxnRemove. solver is either 'GIMME' or 'Shlomi' and defaults to'GIMME'. options is only used with the GIMME algorithm, and it specifies which reaction(s) comprise the objective function; by default, the objective function is chosen by the supplied c vector in the model with a 90% (0.9) threshold. funcModel controls whether the output tissueModel is fully functional (every reaction can carry a flux) or not when using the GIMME algorithm.

tissueModel is the final cell, tissue, or condition specific model generated from the function. Rxns is a structure containing statistics about what reactions were or were not expressed based on the high-throughput data and what reactions were added or removed from the model (see createTissueSpecificModel Documentation). [$\sim 10^2$ s]

6.4.8 Visualization

Visual representation of a metabolic network can aid in understanding the model. Maps for a variety of metabolic pathways are available for many of the models hosted in the BiGG knowledgebase (http://bigg.ucsd.edu). These maps may be used for other organisms that have similar metabolic pathways, given that the user uses the same metabolite and reaction ids as the BiGG model that was used to create the map. After exporting maps from BiGG, the maps can be loaded into MATLAB using the following command:

```
>> map = readCbMap([filename])
```

If readCbMap is called with no arguments, a dialog box will prompt to select a map file. After the map has been read into MATLAB, it can be viewed as a MATLAB figure or a scalable vector graphic (svg). To view the map as a MATLAB figure set the CB map output setting and generate the map by calling the following functions:

```
>> changeCbMapOutput('matlab')
>> drawCbMap(map,[options],[varargin])
```

options is a structure containing the settings for the drawCbMap function such as compound node and reaction arrow size and color. See documentation for description of optional parameters. options is generated using the setMapOptions function:

```
>> options = setMapOptions(options,[varargin])
```

If creating an options structure, set options to []. If adding or modifying parameters within the options structure, pass in the options structure using the options input.

Optional parameters are specified in a parameter, parameter value format. See documentation for more information.

When the output is set to 'matlab', drawCbMap generates a figure in MATLAB of the map. Generating a MATLAB figure may take several minutes for large maps. To output the map as a svg, change the output setting and generate the svg file.

```
>> changeCbMapOutput('svg');
>> drawCbMap(map,[options],[varargin])
```

When the output is set to 'svg', drawCbMap will create the svg file, which can then be viewed using any svg viewer including modern web browsers. By default, the file created is named 'target.svg'. The filename can be set by inputting additional parameters:

```
>> drawCbMap(map,'FileName',filename)
```

On current MATLAB versions, generating svg output is significantly faster than generating a MATLAB figure of the map.

Reaction flux or metabolite concentration data can be overlaid on the maps. A flux variability map can also be generated. Examples of these functions are provided in the ANTICIPATED RESULTS section. See also: testMaps(). [$\sim 10^1$ s]

6.4.9 Simulate optimal growth using flux-balance analysis (FBA)

Simulating optimal growth using FBA is one of the fundamental COBRA phenotypic calculations for metabolic network models. FBA is a method that calculates the flow of metabolites through a metabolic network [68]. Growth is simulated by optimizing the reconstruction for flux through the model's biomass function. The reaction to optimize is set using the c vector within the model structure. Growth conditions are specified by setting the exchange reactions accordingly. The solution returned will have units based on the units used in the model (typically mmol·gDW$^{-1} \cdot h^{-1}$).

```
>> [solution] = optimizeCbModel(model, [osenseStr], [minNorm],
  [allowLoops])
```

The osenseStr is either 'max' or 'min' to maximize or minimize the value of the objective. The minNorm input offers functionality to minimize the Manhattan or Euclidian Norm. Loops can be removed from the final solution using the loop law algorithm by setting allowLoops to false. Note that this function optimizes the model using the objective function and if the objective function is not set to a biomass function, it will not simulate optimal growth.

The function will return a solution structure containing the following: the objective value 'f', the primal solution 'x', the dual solution 'y', the reduced cost 'w', a universal status flag 'stat', a solver specific status flag 'origStat', and the time to compute the solution 'time'. The primal solution, 'x' represents the flux carried by each reaction within the model. The dual solution, 'y' represents the shadow prices for each metabolite and indicates how much the addition of the corresponding metabolite will increase or decrease the objective value [64, 68]. The reduced cost, 'w', indicates how much each reaction affects the objective. A solver status of 1 indicates that an optimal solution was found.

An alternative to optimizeCbModel is geometricFBA [92]. geometricFBA attempts to return the minimal flux distribution central to the bounds of the solution space while still maintaining optimal growth rate. The flux distribution returned should then be reproducible regardless of solver used.

```
>> flux = geometricFBA(model,[varargin])
```

Optional parameters are passed in as optional argument/value pairs. The function returns the vector 'flux' which contains the centered optimal flux distribution.

The optimal flux distribution obtained using optimizeCbModel or geometricFBA can be overlaid onto an existing map of the model using the following function:

```
>> drawFlux(map, model, flux, [options], [varargin])
```

Pass in the map, model, and flux vector using the respective inputs; where map and model were generated by readCbMap and readCbModel, respectively. options

is the drawCbMap options structure. Optional input can also be entered following the options input, in a parameter, parameter value format. To narrow the reaction arrows for reactions which carry no flux, type 'zeroFluxWidth', 1 for varargin. See script header and testMaps.m for additional information and examples. $[10^0 - 10^2 s]$

CRITICAL STEP

The subsequent steps in this protocol rely on the functionality of optimizeCbModel. If optimizeCbModel fails to return a feasible flux distribution for the examples within this protocol, the problem most likely lays with the installation of the LP solver. See TROUBLESHOOTING for tips to remedy this issue. It is not necessary that geometricFBA return a solution for the subsequent steps. See also: testFBA().

6.4.10 Solving COBRA problem structures

The COBRA toolbox has five function calls used for solving different optimization problems. Basic users will not need to call these low level functions directly as higher level functions encapsulate these calls. These functions act as a common interface for different LP, MILP, QP, MIQP, and NLP solvers ensuring that labs can share code even when using different installed solvers.

The five solver functions use a similar input argument structure: problem structure followed by optional argument/value pairs. The required fields in the problem structure vary for each function to supply the required information to solve the type of problem. For example, the mixed integer problem structures require a field which specifies variable type (continuous, integer, binary). A description on the format of COBRA problem structures can be found in Supplementary Material. The COBRA solution structure also provides a common output format regardless of the solver used.

```
>> [solution] = solveCobraLP(LPproblem, [varargin])
>> [solution] = solveCobraMILP(MILPproblem, [varargin])
>> [solution] = solveCobraQP(QPproblem, [varargin])
>> [solution] = solveCobraMIQP(MIQPproblem, [varargin])
```

```
>> [solution] = solveCobraNLP(NLPproblem, [varargin])
```

See also: testSolvers() [10^0 s]

6.4.11 Simulating deletion studies

Deletion studies can be easily simulated with *in silico* models. Gene deletion study methods within the Toolbox are dependent on the proper setup of the gene-reaction matrix as well as the rules defining the Boolean relationship between genes and reactions. Reactions that are affected by a gene deletion have their upper and lower flux bounds set to zero and are therefore not functional. The set of reactions on which a gene deletion has an effect is calculated using the gene reaction association and rules.

The possible results from deletion studies are: 1) unchanged maximal growth, 2) reduced maximal growth, or 3) no growth (lethal). Deletion studies can be used to predict gene/reaction essentiality. Perform a single gene deletion study using the following function:

```
>> [grRatio, grRateKO, grRateWT, hasEffect, delRxns, fluxSolution]
  = singleGeneDeletion(model, method, [geneList])
```

Valid calculation methods are 'FBA', 'MOMA' [88] and linear MOMA ('lMOMA'). By default, all genes are tested; however, a gene list can be specified if desired using the geneList input. The growth rate of the knockouts (grRatioKO) and wild-type (grRateWT) are calculated and the ratio of each knockout growth rate to wild-type growth rate is calculated (grRatio). If the gene deletion has an effect on the model (i.e. a reaction has its bounds constrained to 0), then hasEffect is true for that gene, else hasEffect is false. delRxns contains a list of the reactions, the bounds of which are set to 0 for each gene deletion. The flux solution for each deletion is also returned.

Perform a pair-wise double gene deletion study using the following function:

```
>> [grRatioDble, grRateKO, grRateWT] = doubleGeneDeletion(model,
  method, [geneList1], [geneList2])
```

The calculation method can be set in the same manner as with singleGeneDeletion. By default all gene pairs are tested. The function calculates the growth rates (grRateKO) and ratios (grRatioDble) for the growth rate of each knockout pair compared to wild type growth rate (grRateWT). See also: testDeletionStudy()

6.4.12 Flux Variability Analysis

12 — FBA only returns a single flux distribution that corresponds to maximal growth under given growth conditions. However, alternate optimal solutions may exist which correspond to maximal growth as biological systems contain redundancies that contribute to robustness. FVA calculates the full range of numerical values for each reaction flux within the network [149]. Determine the minimum and maximum flux values reactions within a model can carry while obtaining a specific percentage of optimal growth rate.

```
>> [minFlux maxFlux] = fluxVariability(model, optPercentage,
  [rxnNameList], [verbFlag], [allowLoops])
```

optPercentage can take any value between 0 and 100 and sets the required minimum growth rate . By default the function only allows for optimal solutions (i.e. optPercentage = 100). By default all reactions are optimized but a subset may be passed with rxnNameList. If allowLoops is set to 0, only solutions with no loops are considered. minFlux and maxFlux contain vectors of minimum and maximum rate of each reaction. To better visualize the results from this function, a flux variability map can be generated from an existing reaction map, color coding reactions based on flux directionality. Call the following function generate a flux variability map:

```
>> drawFluxVariability(map, model, minFlux, maxFlux, [options])
```

map is the map structure corresponding to the model read in using readCbMap. model is the corresponding COBRA model structure. minFlux and maxFlux are vectors generated using fluxVariability. options is a structure containing optional parameters such as edge and node color and size. Bi-directional reversible reactions are colored green. Unidirectional reversible reactions that carry flux in the forward

direction are colored magenta. Unidirectional reversible reactions that carry flux only in the reverse direction are colored cyan. Irreversible fluxes are colored blue. When output is set to .svg, a legend is generated at the bottom of the map. See also: testFVA(). [$\sim 10^2$ s]

6.4.13 Sampling the solution space

FBA only returns a single optimal point and thus gives little information about the entire solution space. An alternative approach is to characterize the solution space using sampling [137]. Sample the solution space by calling the following function:

```
>> [sampleStructOut, mixedFrac] = gpSampler(sampleStruct, [nPoints],
  [bias], [maxTime], [maxSteps])
```

The generalized parallel sampler samples any arbitrary linearly-constrained space by moving a fixed number of points in parallel. Here sampleStruct is the COBRA Toolbox problem structure for linear programming problems; see Supplementary Material. The number of sample points is set through nPoints. The maximum sampling time and number of steps can be set (maxTime, maxSteps). The bias structure can be used to impose marginal distributions on reactions. The function returns sampleStructOut, which looks the same as sampleStruct with the addition of the 'points' field containing the solutions. Additionally, mixedFrac gives an estimate of how mixed the sampling solution is relative to the warmup points. A mixedFrac value of 0.5 indicates complete mixing. See also: testSampling().

6.4.14 Fluxomics

Carbon 13 tracing experiments provide the ability to measure internal flux rates in a metabolic network [53]. To use this data, additional information about carbon tracking must be added to the COBRA model. This is stored in the .isotopomer field as described in Supplementary Material Section S.3. In order to use the C13 solver, the functions must be generated:

```
>> [experiment] = generateIsotopomerSolver(model, inputMet,
  [experiment], [FVAflag]) (~101 s)
```

model is the COBRA model with the .isotopomer field. inputMet is a string corresponding to the C13 labeled input. The solver requires a list of metabolites that must be measured and this is provided via experiment. Finally there is an option to remove reactions that cannot carry a flux by setting FVAflag. Two solvers are generated, one based on the cumomer method [56] and one on the faster EMU method [57].

A given flux distribution can be scored against a set of C13 data:

```
>> output = scoreC13Fit(v0,expdata,model)
```

where expdata is one or more sets of experimental data described in Appendix B.3. [10^2 s]

The most optimal flux distribution can be found with a non-linear optimization as such:

```
>> [vout] = fitC13Data(v0,expdata,model, [majorIterationLimit])
```

This function will return the flux with the lowest experimental score found by the NLP solver. Very often it is useful to compute the confidence intervals of reactions which are consistent with C13 data.

```
>> [vs, output, v0] = C13ConfidenceInterval(v0, expdata, model,
  max_score, [directions], [majorIterationLimit])  (~10e2 s)
```

v0 is the initial guess. expdata is the experimental data that must be fit. max_score is the highest acceptable score. directions is the list of reactions and reaction ratios which will be maximized and minimized (by default all reactions). See also: testC13Fitting().

6.4.15 Gap Filling

Due to incomplete knowledge, a metabolic model may possess gaps. A gap is defined as missing biochemical information which can explain discrepancies between model predictions and experimental data. Gaps are typically reactions that

facilitate the conversion of an available metabolite in the model to one necessary to achieve an objective. The COBRA Toolbox has two functions to help identify gaps in metabolic models: detectDeadEnds and gapFind. Analyze the S matrix to identify gaps by using the following function:

```
>> outputMets = detectDeadEnds(model, [removeExternalMets])
```

The detectDeadEnds function searches the S matrix for metabolites that participate in only one reaction (can only be produced or only be consumed) and returns the corresponding indices for the metabolites in the model.mets field. Setting removeExternalMets to true removes external metabolites from the results. Not all gaps can be identified by simply inspecting the S matrix. Perform a more thorough search calling the following function:

```
>> [allGaps, rootGaps, downstreamGaps] = gapFind(model, findNCgaps,
  verbFlag)
```

This function identifies all gaps and all metabolites that are downstream from a gap using the GapFind algrithm [143]. Set the lower bound of all exchange reactions within model to -1. Set the upper bound on all reactions to a relatively large positive number (i.e. 10^5) and the lower bound of all reversible reactions to a relatively large negative number (i.e. -10^5) within model. The appropriate bound magnitude required varies from model to model. If the bound mangnitudes are too small, the algorithm will incorrectly identify many metabolites as gaps; if this occurs, increase the bound magnitudes by 10-fold. Repeat this process as necessary until the algorithm does not identify all metabolites as gaps.

This function can find metabolites that are products of a reaction that relies on a metabolite that cannot be produced. The function returns the metabolite index for all gaps (allGaps), metabolites that cannot be produced (rootGaps), and metabolites that are produced on a reaction that requires a metabolite that cannot be produced (downstreamGaps).

In addition to these two gap identification functions, the Toolbox includes an optimization-based algorithm that predicts missing reactions [147]. growthExpMatch identifies the minimum number of reactions from a universal reaction database

that are required for a metabolic model to grow on a specified substrate. Identify possible sets of reactions to add to allow a model to grow by using the following function:

```
>> [solution] = growthExpMatch(model, KEGGFilename, compartment,
  iterations, dictionary, logFile, threshold)
```

KEGGFilename is the name of the reaction .lst file downloaded from KEGG (http://www.genome.jp/kegg). compartment is a string denoting which compartment to generate exchange reactions for. The iterations variable controls the number iterations to run the function. dictionary is an n by 2 cell array that maps metabolites to KEGG IDs. logFile is the name of the .mat file to save the solution to. threshold is the minimum value that the biomass function can take for the model to be considered growing. Display the growthExpMatch solution by printing the log file using the following function:

```
>> printSolutionGEM(dir, matFile)
```

dir is the directory containing the growthExpMatch solution .mat file. matFile is the name of the growthExpMatch solution .mat file. See also testGrowthExpMatch(). [$\sim 10^2$ s]

6.4.16 Metabolic Engineering

The COBRA Toolbox version 2.0 provides three methods for *in silico* metabolic engineering: OptKnock [144], OptGene [145], and GDLS [146].

Determine a knockout list to optimize for flux through a specific reaction using OptKnock, OptGene, or GDLS.

```
>> [OptKnockSol, biLevelMILPproblem] = OptKnock(model,
  selectedRxnList, options, constrOpt, prevSolutions, verbFlag)
```

OptKnock runs the OptKnock algorithm to determine reaction sets to knockout for the overproduction of a specific product when the model is optimized for internal cellular objectives [144]. selectedRxnList is a list of candidate reactions to knock out. options is a structure which controls the target reaction to maximize flux through, the number of knockouts is and the maximum flux allowed

for reactions. constrOpt is a structure which specifies specific constraints such as a minimum growth rate. Initial solutions can be input using the prevSolutions variable. verbFlag controls level of printing. See ANTICIPATED RESULTS for an example setup of options and constrOpt structures. The OptKnockSol structure contains the best knockout set. biLevelMILPproblem is the MILP problem generated by the algorithm and subsequently solved.

There are several things to take note of when calling the OptKnock function. First the function does not use the upper and lower bounds set within the model that is passed in. The model is first converted into irreversible format, splitting reactions with a lower bound < 0 and upper bound > 0. The resulting set of reactions has its lower bounds set to 0 and upper bounds set to options.vMax. Use the constrOpt structure to apply constraints on reactions, such as a minimal flux through the biomass function or ATP maintenance. Failure to set the proper constraints may lead to incorrect predictions generated by the function.

OptGene is an evolutionary programming-based method to determine gene knockout strategies for overproduction of a specific product [145]. It can handle non-linear objective functions such as product flux multiplied by biomass.

```
>> [x, population, scores, optGeneSol] = OptGene(model, targetRxn,
  substrateRxn, generxnList, maxKOs, population)
```

Pass in the reconstruction using model. targetRxn and substrateRxn specify the reaction to be optimized for and the exchange reaction for the growth substrate respectively. generxnList is a cell array of strings that specifies which genes or reactions are allowed to be deleted. maxKOs sets the maximum number of knockouts. If resuming a previous simulation, the binary matrix (population) can be specified. Scores are determined by the functions optGeneFitness or optGeneFitnessTilt. OptGene returns the best score (x), the binary matrix representing the knockout sets (population), the scores for each individual knockout set (scores), and the structure summarizing the results (optGeneSol). optGeneSol is in the same format as that produced by the OptKnock function.

```
>> [gdlsSolution, biLevelMILPproblem, gdlsSolutionStructs] =
  GDLS(model, targetRxns varargin)
```

Table 6.2: Troubleshooting

Additional troubleshooting solutions can be found at the COBRA toolbox Google group (http://groups.google.com/group/cobra-toolbox).

Issue	Solution
Tests failed in testing suite	Check that all necessary requirements are met for the function to run: • Is the correct solver selected/installed? • Are libSBML 4.0.1+ and SBMLToolbox 3.1.1+ properly installed and accessible by MATLAB? • Are all toolbox function included in MATLAB path? • Are any of the COBRA toolbox functions shadowed by a function with the same name?
SBML file not read in correctly	Check that libSBML 4.0.1+ and SBMLToolbox 3.1.1+ are installed and accessible in MATLAB. Is the file a COBRA compliant SBML file? (see **Supplementary Material**) If compartment abbreviations are not used as compartment ids within the file, specify all compartment abbreviations using compSymbolList input in readCbModel.
Unknown compartment error using writeCbModel	Specify all compartment abbreviations and names using compSymbolList and compNameList respectively when calling writeCbModel.
drawCbMap requires significant time to produce output	Try changing output to SVG. Text rendering in MATLAB figures is consuming.
optimizeCbModel returns an infeasible or infinite solution	For an infeasible solution, check that the constraints on the model allow for experimental growth. For an infinite solution, check that at least one constraint is limiting.
SolveCobraLP/MILP/QP/MIQP fails because csense or vartype is not a character array	Be sure that csense and/or vartype is initialized as a character array. MATLAB may cast the variable as a double instead. Run verifyCobraProblem to verify the format of the problem structure.

This function runs the Genetic Design Local Search (GDLS) algorithm [146] to finding what to knockout to increase *in silico* production of desired metabolites. The type of knockout (gene, gene set, or reaction) can be defined. By default all genes, gene sets, or reactions are able to be deleted. A specific list of genes, gene sets, or reactions that can be knockout out can also be specified. This approach typically runs faster than the global search performed by OptKnock. It returns the knockout solution (gdlsSolution), the bi-level MILP problem for the solution (biLevelMILPproblem), as well as the intermediate solutions (gdlsSolutionStructs).

6.5 Troubleshooting

Troubleshooting for several steps in the protocol is available (Table 6.2). If your problem is not addressed here, the COBRA toolbox Google group is available for discussing all aspects of the toolbox. See also: testOptKnock().

6.6 Anticipated Results

6.6.1 Displaying metabolic maps

Precompiled maps can be read into MATLAB and used to generate to MATLAB figures. Navigate to the directory containing the map file 'ecoli_core_map.txt'.

```
>> map = readCbMap('ecoli_core_map.txt')
>> changeCbMapOutput('matlab')
>> drawCbMap(map);
```

A MATLAB figure containing the *E. coli* core model should be generated. Maps can also be generated in svg format.

```
>> changeCbMapOutput('svg')
>> drawCbMap(map);
```

The file 'target.svg' will be saved in the working directory. This file, when opened with a program capable of opening svg format (such as any modern web browser), is a metabolic map of the *E. coli* core model. Metabolite concentrations and reaction fluxes can also be overlaid onto maps and will be covered in the sections pertaining to the data generation.

6.6.2 Optimal flux distributions and growth rates for *Escherichia coli* core model

To read in the *E. coli* core model and predict a flux distribution for optimal growth, move to a directory containing the *E. coli* core model and map file and sequentially call the following functions:

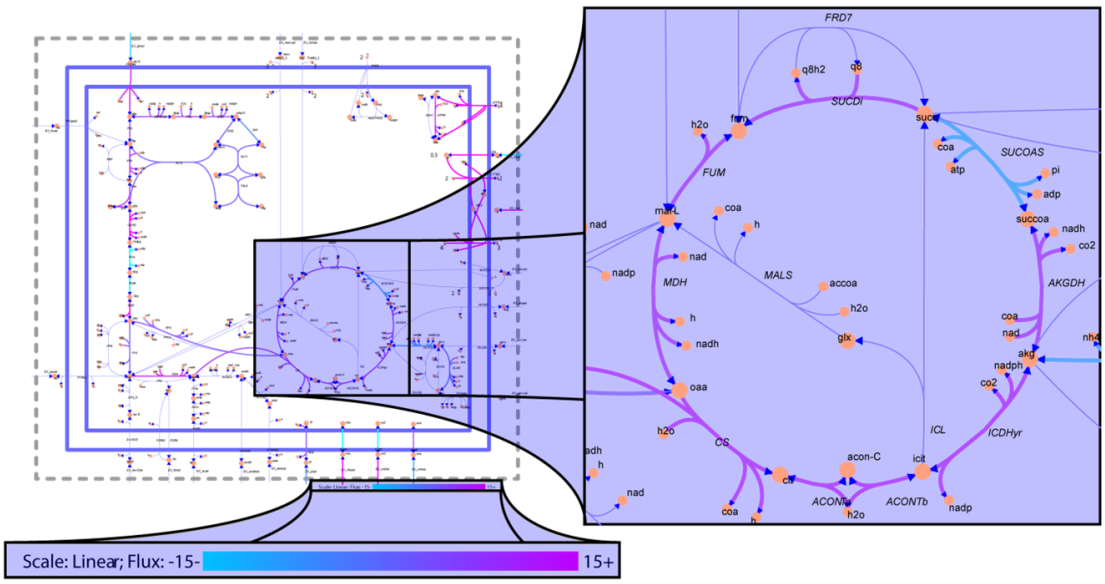


Figure 6.3: Flux balance analysis of *E. coli* core model

(left) Full E. coli core map. (right) Zoom in on the optimal flux distribution map of the citric acid cycle. (bottom) Zoom in on the flux color scale. Reactions are colored according to a scale of cyan (flux of 15 mmol $\cdot$ gDW$^{-1}\cdot$ h^{-1} or greater in the reverse direction) to magenta (flux of 15 mmol $\cdot$ gDW$^{-1} \cdot h^{-1}$ or greater in the forward direction). Reactions carrying zero flux have their corresponding arrows narrowed.

```
>> model = readCbModel('ecoli_core_model.xml');
>> map = readCbMap('ecoli_core_map.txt');
>> changeCbMapOutput('svg');
>> solution = optimizeCbModel(model);
>> drawFlux(map, model, solution.x, [], 'FileName',
  'EcoreOptFlux1.svg');
```

The expected optimal biomass flux is 0.87. The drawFlux function call generates an svg file named EcoreOptFlux1.svg in the working directory. The reactions are color coded using a linear scale from cyan (corresponding to a flux of -29.17) to magenta (corresponding to a flux of 45.51). To more easily extract data from the map, change the width of reactions arrows corresponding to reactions carrying zero flux to 1 point. In addition, set the lower and upper bounds to -15 and 15 respectively.

```
>> drawFlux(map, model, solution.x, [], 'ZeroFluxWidth', 1, 'lb',
  -15, 'ub', 15, 'FileName', 'EcoreOptFlux2.svg');
```

An svg file named EcoreOptFlux2.svg should be written to the working directory with reactions color coded from cyan (flux of -15 or less) to magenta (flux of 15 or greater) and reactions carrying zero flux have their corresponding arrows narrowed (Figure 6.3).

6.6.3 Flux Variability Analysis of *E. coli* core model

To perform FVA for the *E. coli* core model under glucose limited aerobic growth conditions with a minimum cellular growth of 90% of optimal, call the following commands.

```
>> model = readCbModel('ecoli_core_model');
>> map = readCbMap('ecoli_core_map');
>> [minFlux maxFlux] = fluxVariability(model,90);
>> changeCbMapOutput('svg')
>> drawFluxVariability(map, model, minFlux, maxFlux, [], 'fileName',
'EcoreFluxVariability.svg');
```

A reaction map with reactions color coded according to the flux directionality it can carry should be created (Figure 6.4). Bi-directional reversible reactions are

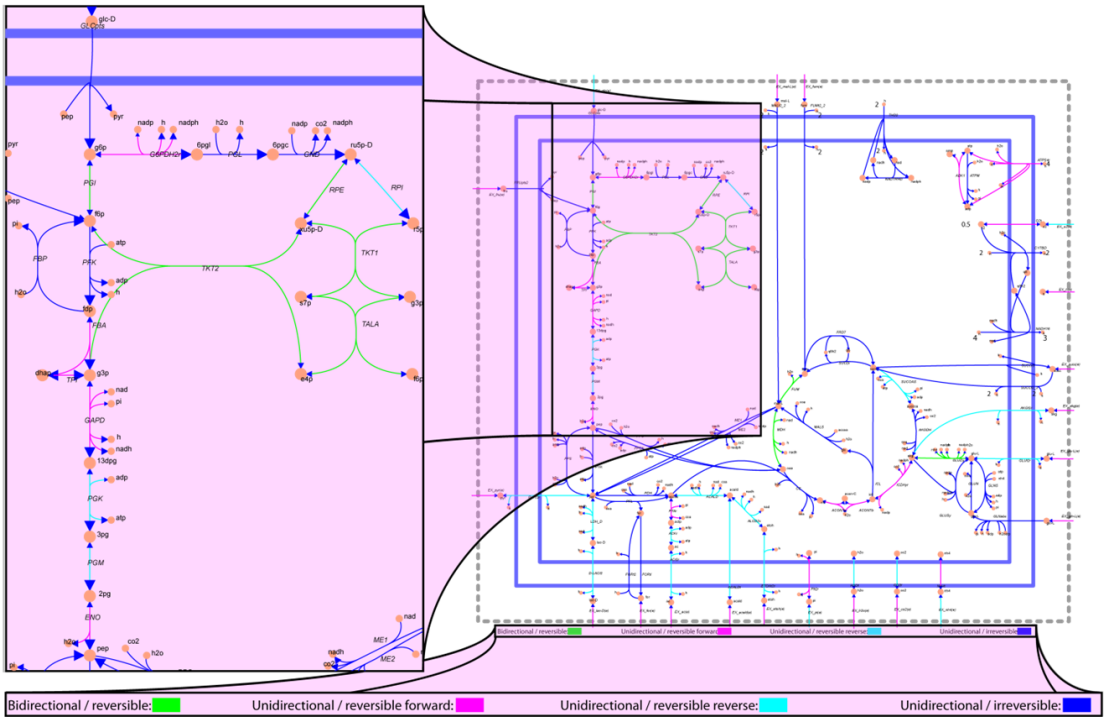


Figure 6.4: Flux variability analysis of E. coli

(right) Reaction map of *E. coli* core model. (left) Flux variability analysis of part of glycolysis and pentose phosphate pathway in the E. coli core model when growth rate is constrained to 90% of optimal. Bi-directional reversible reactions are colored green. Unidirectional reversible reactions which carry flux in the forward direction are colored magenta. Unidirectional reversible reactions which carry flux only in the reverse direction are colored cyan. Irreversible fluxes are colored blue. Unidirectional fluxes have enlarged arrowheads in the direction of the flux.

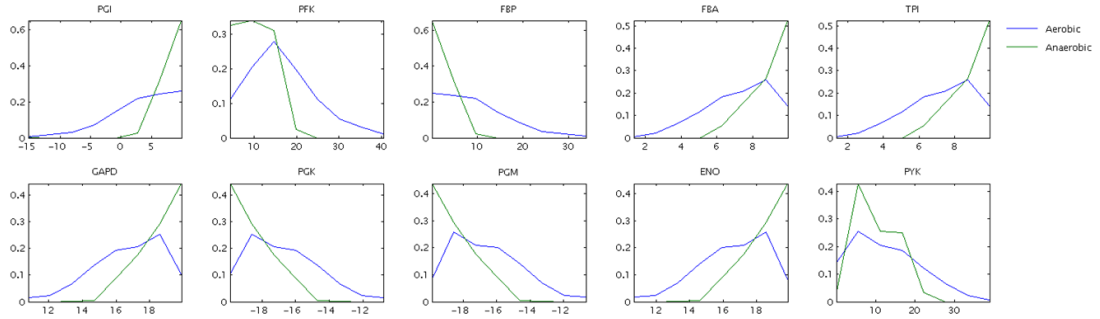


Figure 6.5: Sampling histogram of glycolysis using the *E. coli* core model under aerobic and anaerobic glucose minimal media conditions
There is a large shift in the probable flux through many of the reactions. In general, the range of flux probabilities for each reaction became more constrained. PGI switched from being able to carry flux in either direction with aerobic conditions to only carrying flux in the forward direction with anaerobic conditions.

colored green. Unidirectional reversible reactions which carry flux in the forward direction are colored magenta. Unidirectional reversible reactions which carry flux only in the reverse direction are colored cyan. Irreversible fluxes are colored blue. These colors can be specified within the options structure or optional inputs. Unidirectional fluxes have enlarged arrowheads in the direction of the flux.

6.6.4 Sampling of the solution space of *E. coli* core model aerobic versus anaerobic

To read in the *E. coli* core model and sample its solution space under glucose minimal media and aerobic conditions with 200 points for 2 minutes, call the following commands:

```
>> model_aerobic = readCbModel('ecoli_core_model');
>> sampleStruct_aerobic = gpSampler(model_aerobic,200,[],120);
>> model_anaerobic = changeRxnBounds(model_aerobic,'EX_o2(e)',0,'l');
>> sampleStruct_anaerobic = gpSampler(model_anaerobic,200,[],120);
>> rxnList = {'PGI','PFK','FBP','FBA','TPI','GAPD','PGK','PGM','ENO',
  'PYK'};
>> plotSampleHist(rxnList, {sampleStruct_aerobic.points,
```

```
sampleStruct_anaerobic.points }, {model_aerobic, model_anaerobic},
[],[2,5]);
```

Sampling results will be returned in the two structures sampleStruct_aerobic and sampleStruct_anaerobic within the field points. A MATLAB figure will also be generated showing the histograms for glycolysis with aerobic in blue and anaerobic in green (Figure 6.5).

6.6.5 Identifying gaps in the *E. coli* iJR904 and *E. coli* iAF1260 models

To find gaps in the Ec_iJR904 model using the gapFind function, use the following commands:

```
>> model = readCbModel('Ec_iJR904_glcMM');
>> exchangeRxns = model.rxns(findExcRxns(model));
>> model = changeRxnBounds(model, model.rxns(logical(model.rev)),
  -1e6, 'l');
>> model = changeRxnBounds(model, model.rxns, 1e6, 'u');
>> model = changeRxnBounds(model, exchangeRxns, -1, 'l');
>> [allGaps, rootGaps, downstreamGaps] = gapFind(model);
```

All reversible reactions have their lower bound set to -1e6. All reactions have upper bounds changed to 1e6. All exchange reactions have bounds changed to -1 to allow for uptake. There are a total of 64 metabolites identified as gaps: 28 root gaps and 36 downstream gaps. Repeating the above commands using the Ec_iAF1260 model yields a total of 111 metabolites identified as gaps: 55 root gaps, and 56 downstream gaps.

6.6.6 Filling gaps using growthExpMatch

Remove the PGK reaction from the E. coli core model and use growthExpMatch to propose candidate reactions required to allow growth on glucose. Call the following commands:

```
>> model = readCbModel('ecoli_core_model');
>> modelKO = removeRxns(model,{'PGK'});
```

Table 6.3: growthExpMatch gap filling solutions

Solutions from five iterations of growthExpMatch on a PGK knockout growing on glucose using the E. coli core model. The first solution returned is the knocked out reaction. Solutions from iterations 2, 3, and 5 are reactions to utilize 3-Phospho-D-glycerol phosphate while the solution from iteration 4 bypasses the gap by converting D-Fructose 6-phosphate to D-Erythrose 4-phosphate and Acetyl phosphate.

#	KEGG ID	Reaction Name	Reaction Formula
1	'R01512_b'	phosphoglycerate kinase	ATP + 3-Phospho-D-glycerate <=> ADP + 3-Phospho-D-glyceroyl phosphate
2	'R02188_f'	3-Phospho-D-glyceroyl-phosphate:polyphosphate phosphotransferase	3-Phospho-D-glyceroyl phosphate + (Phosphate)n <=> 3-Phospho-D-glycerate + (Phosphate)n
3	'R01515_f'	acylphosphatase	3-Phospho-D-glyceroyl phosphate + H2O <=> 3-Phospho-D-glycerate + Orthophosphate
4	'R00761_f'	fructose-6-phosphate phosphoketolase	D-Fructose 6-phosphate + Orthophosphate <=> Acetyl phosphate + D-Erythrose 4-phosphate + H2O
5	'R00024_f'	ribulose-bisphosphate carboxylase	D-Ribulose 1,5-bisphosphate + CO2 + H2O <=> 2 3-Phospho-D-glycerate
	'R01523_f'	phosphoribulokinase	ATP + D-Ribulose 5-phosphate <=> ADP + D-Ribulose 1,5-bisphosphate

```
>> KEGGFilename = '2010_7_30_KEGG_reaction.lst';
>> load('Dictionary.mat');
>> growthExpMatch (model, KEGGFilename, '[c]', 5, dictionary);
```

The PGK reaction is removed from the *E. coli* core model, removing the ability of the model to produce biomass from glucose. The KEGG reaction list should be downloaded from the KEGG website. However, a copy of the reaction list as well as a copy of Dictionary.mat is available in the growthExpMatch test folder. The resulting GEMLog file should contain 5 solutions (Table 6.3). The first solution R01512 corresponds to the PGK reaction which was removed previously. The remaining four solutions are alternate reaction sets that when added allow the model to grow on glucose.

Table 6.4: Expected results from OptKnock and GDLS optimizations for lactate, succinate and pyruvate production growing on glucose
The solutions for succinate and pyruvate are the same for both methods. The lactate solutions vary by two reactions; however, both resulting models have the same production and growth rates.

Target	Lactate	Succinate	Pyruvate	Lactate	Succinate	Pyruvate
Knockout List	ALCD2x, FUM, GLUDY, ME2, PYK	ALCD2x, GLUDy, LDH_D, PFL, THD2	ACALD, LDH_D, PTAr	ACALD, FRD7, GLUDy, ME2, PYK	ALCD2x, GLUDy, LDH_D, PFL, THD2	ACALD, LDH_D, PTAr
Product (mmol·gDW^{-1}·h^{-1})	37.7	20.2-26.1	18.9	37.7	20.2-26.1	18.9
Biomass (mmol·gDW^{-1}·h^{-1})	0.142	0.120	0.151	0.142	0.120	0.151
Computational Time (s)	28.1	24.5	5.6	4.9	2.9	1.8

6.6.7 Optimizing for product secretion from glucose using the *E. coli* core model

To optimize for lactate with 5 deletions or less using the optKnock method, use the following commands.

```
>> model = readCbModel('ecoli_core_model');
>> model = changeRxnBounds(model, {'EX_o2(e)', 'EX_glc(e)'}, [0 -20],
  'l');
>> selectedRxns = {mode.rxns{ [1, 3:5, 7:8, 10, 12, 15:16, 18, 40:41,
  44, 46, 48:49, 51, 53:55, 57, 59:62, 64:68, 71:77, 79:83, 85:86,
  89:95]}}';
>> options.targetRxn = 'EX_lac-D(e)';
>> options.vMax = 1000;
>> options.numDel = 5;
>> options.numDelSense = 'L';
>> constrOpt.rxnList = {Biomass_Ecoli_core_N(w/GAM)-Nmet2', 'ATPM'};
>> constrOpt.values = [0.05, 8.39];
>> constrOpt.sense = 'GE';
>> optKnockSol = OptKnock(model, selectedRxns, options, constrOpt);
```

The simulation is run for anaerobic growth with glucose uptake set to 20 mmol $\cdot$ gDW$^{-1} \cdot h^{-1}$. The selectedRxns variable contains a list of the reactions in the model excluding exchange and transport reactions, biomass, and ATP maintenance requirement. The resulting knockout list is alcohol dehydrogenase, fumarase, glu-

tamate dehydrogenase, malic enzyme (NADP), pyruvate kinase. The resulting knockout predicted growth rate of 0.142 and product excretion rate of 37.7. The computational time required for this simulation was 28.1 seconds.

Call the following commands to optimize for the same product using the GDLS algorithm.

```
>> model = readCbModel('ecoli_core_model');
>> model = changeRxnbounds(model, {'EX_o2(e)', 'EX_glc(e)'}, [0 -20],
  'l');
>> selectedRxns = {model.rxns{[1, 3:5, 7:8, 10, 12, 15:16, 18,
  40:41, 44, 46, 48:49, 51, 53:55, 57, 59:62, 64:68, 71:77, 79:83,
  85:86, 89:95]}};
>> [gdlsSolution, bilevelMILPproblem, gdlsSolutionStructs] =
  GDLS(model, 'EX_lac-D(e)', 'minGrowth', 0.05, 'selectedRxns',
  selectedRxns, 'maxKO', 5, 'nbhdsz', 3);
```

The resulting knockout list is acetaldehyde dehydrogenase, fumarate reductase, glutamate dehydrogenase, phosphotransacetylase, and NAD(P) transhydrogenase. The resulting knockout predicted growth rate of 0.142 and product excretion rate of 37.7. The computational time required for this simulation was 4.9 seconds. Both methods were also used to optimize for succinate product with a maximum of 5 knockouts and pyruvate with a maximum of 3 knockouts. The same knockout lists were produced for succinate and pyruvate, however, two reactions were different for lactate (Table 6.4). For lactate, OptKnock chose alcohol dehydrogenase and fumarase, while GDLS chose acetaldehyde dehydrogenase, fumarate reductase. However, both result in the same optimal flux distribution.

We would like to thank individuals who have contributed to or tested COBRA 2.0 Steinn Gudmundsson, Tom Conrad, Nathan Lewis, Neema Jamshidi, Richard Notebaart, Jake Feala.

Funding was provided by NIH grant GM68837-05A1, DE-PS02-08ER08-01, GM057089-12, GM057089-11S1, CalIT2 Summer scholars program and National Institute of Allergy and Infectious Diseases NIH/DHHS through interagency agreement Y1-AI-8401-01. RF and IT were funded by U.S. Department of Energy, Offices of Advanced Scientific Computing Research and the Biological and Environmental Research as part of the Scientific Discovery Through Advanced Computing

program, grant DE-SC0002009.

The text of Chapter Six, is work to be submitted to Nature Methods with authors J. Schellenberger, R. Que, Fleming, R., Thiele, I., Orth, J., Feist, A., Zielinski, D., Bordbar, A., Rahmanian, S., Kang, J. Hyduke, D., and B.Ø. Palsson. I was the primary researcher on this project and oversaw the final document for publication.

Chapter 7

Conclusions

As we gain more knowledge about biology, at some point our minds are no longer able to grasp the big picture into a coherent model. With the advent of genome sequencing, micro-array expression profiling, ChIP on Chip, mass spec, proteomics and many other -omics data types, we have reached a point where the sheer amount of data being generated is impossible to comprehend without powerful computational tools. One approach to dealing with a system that is just too complex is reduction - studying only a small part at a time. This has been the mantra of Biology for the last century and has taught us many things and will undoubtedly continue.

There are, however, biological questions which cannot be answered by a reductionist approach. These are questions that are inherently dependent on many components and their interactions. Studying the parts of a car individually will be able to answer many questions about the car, but not how fast the car can travel. In a similar fashion, studying individual biological components cannot tell you tell you how fast a cell can grow in a mechanistic fashion.

This is where Systems Biology has proven extremely useful. By placing an emphasis on data-driven large-scale model building, it is possible to make predictions about systems that are not obvious from just looking at individual parts. Some have called such results 'emergent properties'. It is the idea that 'the system is more than the sum of its parts.'

Nonetheless, as models grow in complexity two problems begin to emerge. The first is lack of data and the second is, paradoxically, that the available data tends to be inconsistent. This is one area where Constraint Based Reconstruction and Analysis - a sub field of Systems Biology (COBRA) - has proven extremely useful. By attempting to model data as constraints on a system, missing data is simply the lack of constraints. Inconsistent data can be dealt with by assuming experimental error and relaxing constraints. The advantage of this framework is that rather than having a low confidence in computing the exact state of the system, COBRA brings a high confidence of knowing the range of possible states of the system.

Once the solution space is defined, a whole host of methods are available for analysis.

7.1 Looking Forward

This thesis is only the beginning of what Monte Carlo Sampling can do for Systems Biology. Some avenues to consider pursuing in the future include:

1. Algorithmic Development - The GP Sampler developed in this thesis is still quite limited. It can only deal with convex spaces with linear bounds. The set of constraints that can be modeled as such is limited. There is an active community working on algorithms in this area and I suspect they will be very complex in nature and require toolboxes/APIs to be used successfully by biologists.

2. Biological Extensions - The COBRA framework has mainly been used to study flux rates through metabolic networks. There are several reasons why these are well suited, for example the powerful steady state assumption which is a powerful constraint on a network. A few other areas where Monte Carlo Sampling could be useful were proposed in Chapter 1. In particular an integrated framework for incorporating both metabolic concentrations and reaction rates simultaneously would be very powerful. Thermodynamics links the

two quantities in a fundamental way. Algorithmic challenges are formidable though as the resulting solution space is not convex.

Part of this will be driven by new technologies. As new -omics data sets emerge, additional constraints can be placed on models.

3. Visualization - Visualization and data representation are hindered by the same kinds of problems that face biology in general. Some problems are simply so complex that even understanding the *answer* is challenging. A flux distribution is a vector with as many entries as reactions in the model. This is not easy to visualize, even with a map. Data reduction is key - looking at pathways instead of reactions, metabolic pools instead of individual metabolites. Humans are very good at laying out information in a visually pleasing form. Getting machines to do the same is an ongoing challenge.

4. Better Software Tools - The COBRA toolbox is a major step towards bringing together many functions into one coherent package. However, even this tool is still too difficult for an average biologist to use. Most likely it will take a point-and-click interface, perhaps web-based, to really bring these tools to a wider audience. This means that new standards need to be developed for various data types, interfaces, etc. There have been some advances in this area [150, 151] and this will surely continue.

Appendix A

E. coli isotopomer model

A full description of the *E. coli* isotopomer model. Each reaction has entry as follows:

Reaction Name; Range: lb to ub; Formula: Formula

Isotopomer Formula

The isotopomer description consists of a) a chemical formula and b) atom mappings separated by '!'. Multiple mappings are possible if the reaction contains symmetric compounds.

BiomassEcoliGALUi; Range: 0.44 to 0.44; Formula: 0.05 5mthf + 5e-005 accoa + 0.488 alaL + 0.001 amp + 0.281 argL + 0.229 asnL + 0.229 aspL + 45.7318 atp + 0.00645 clpnEC + 6e-006 coa + 0.126 ctp + 0.087 cysL + 0.0247 datp + 0.0254 dctp + 0.0254 dgtp + 0.0247 dttp + 1e-005 fad + 0.003 g1p + 0.25 glnL + 0.25 gluL + 0.582 gly + 0.154 glycogen + 0.203 gtp + 45.5608 h2o + 0.09 hisL + 0.276 ileL + 0.428 leuL + 0.0084 lpsEC + 0.326 lysL + 0.146 metL + 0.00215 nad + 5e-005 nadh + 0.00013 nadp + 0.0004 nadph + 0.09675 peEC + 0.0276 peptidoEC + 0.0232 pgEC + 0.176 pheL + 0.21 proL + 0.0026 psEC + 0.035 ptrc + 0.205 serL + 0.007 spmd + 3e-006 succoa + 0.241 thrL + 0.054 trpL + 0.131

tyrL + 0.139 utp + 0.402 valL -> 45.5608 adp + 45.5574 h + 45.5628 pi + 0.7332 ppi

BiomassEcoliGALUi 0.05 x5mthf 0.00005 xaccoa 0.488 xalaL 0.281 xargL 0.229 xasnL 0.229 xaspL 0.087 xcysL 0.003 xg1p 0.25 xglnL 0.25 xgluL 0.582 xgly 0.154 xglycogen 0.09 xhisL 0.276 xileL 0.428 xleuL 0.326 xlysL 0.146 xmetL 0.176 xpheL 0.21 xproL 0.035 xptrc 0.205 xserL 0.007 xspmd 3e-006 xsuccoa 0.241 xthrL 0.054 xtrpL 0.131 xtyrL 0.402 xvalL > !#a #bc #def #ghijkl #mnop #qrst #uvw #xyzABC #DEFGH #IJKLM #NO #PQRSTU #VWXYZ1 #234567 #89abcd #efghij #klmno #pqrstuvwx #yzABC #DEFG #HIJ #KLMNOPQ #RSTU #VWXY #Z123456789a #bcdefghij #klmno >

EX_ac; Range: 3.5 to 3.5; Formula: ace ->

EX_ac 1 xace > !#ab >

EX_co2; Range: 0 to 100; Formula: co2e ->

EX_co2 1 xco2e > !#a >

EX_etoh; Range: 0 to 100; Formula: etohe ->

EX_etoh 1 xetohe > !#ab >

EX_for; Range: 0 to 100; Formula: fore ->

EX_for 1 xfore > !#a >

EX_fum; Range: 0 to 100; Formula: fume ->

EX_fum 1 xfume > !#abcd > !#dcba >

EX_glc; Range: -9 to -9; Formula: glcDe ->

EX_glc 1 xglcDe > !#abcdef >

EX_glyc; Range: 0 to 100; Formula: glyce ->

EX_glyc 1 xglyce > !#abc > !#cba >

EX_h; Range: -100 to 100; Formula: he ->

EX_h2o; Range: -100 to 100; Formula: h2oe ->

EX_lacD; Range: 0.4 to 0.4; Formula: lacDe ->

EX_lacD 1 xlacDe > !#abc >

EX_lacL; Range: 0 to 100; Formula: lacLe ->

EX_lacL 1 xlacLe > !#abc >

EX_nh4; Range: -100 to 0; Formula: nh4e ->

EX_no2; Range: 0 to 100; Formula: no2e ->

EX_no3; Range: 0 to 100; Formula: no3e ->

EX_o2; Range: -14.9 to -14.9; Formula: o2e ->

EX_pi; Range: -100 to 0; Formula: pie ->

EX_pyr; Range: 0 to 100; Formula: pyre ->

EX_pyr 1 xpyre > !#abc >

EX_so4; Range: -100 to 0; Formula: so4e ->

EX_succ; Range: 0 to 100; Formula: succe ->

EX_succ 1 xsucce > !#abcd > !#dcba >

ABTA; Range: 0 to 100; Formula: 4abut + akg -> gluL + sucsal

ABTA 1 x4abut 1 xakg > 1 xgluL 1 xsucsal !#abcd #efghi > #efghi #abcd

COMBO2; Range: 0 to 100; Formula: akg + h2o + nad + ptrc -> 4abut + gluL + 2 h + nadh

COMBO2 1 xakg 1 xptrc > 1 x4abut 1 xgluL !#abcde #fghi > #fghi #abcde !#abcde #ihgf > #fghi #abcde

ACALDi; Range: 0 to 100; Formula: acald + coa + nad -> accoa + h + nadh

ACALDi 1 xacald > 1 xaccoa !#ab > #ab

COMBO3; Range: 0 to 100; Formula: accoa + atp + gluL -> acg5p + adp + coa + h

COMBO3 1 xaccoa 1 xgluL > 1 xacg5p !#ab #cdefg > #cdefgab

COMBO4; Range: 0 to 100; Formula: 2obut + 2 h + nadph + pyr -> 3mop + co2 + h2o + nadp

COMBO4 1 x2obut 1 xpyr > 1 x3mop 1 xco2 !#abcd #efg > #abfcdg #e

ACKr; Range: 0 to 100; Formula: ac + atp -> actp + adp

ACKr 1 xac > 1 xactp !#ab > #ab

COMBO5; Range: 0 to 100; Formula: 2 h + nadph + 2 pyr -> 23dhmb + co2 + nadp

COMBO5 1 xpyr 1 xpyr > 1 x23dhmb 1 xco2 !#abc #def > #debcf #a

ACODA; Range: 0 to 100; Formula: acorn + h2o -> ac + orn

ACODA 1 xacorn > 1 xac 1 xorn !#abcdefg > #fg #abcde

rACONT; Range: 0 to 100; Formula: cit -> icit

rACONT 1 xcit > 1 xicit !#abcdef > #abcdef

ACOTA; Range: 0 to 100; Formula: acorn + akg -> acg5sa + gluL

ACOTA 1 xacorn 1 xakg > 1 xacg5sa 1 xgluL !#abcdefg #hijkl > #abcdefg #hijkl

ACS; Range: 0 to 100; Formula: ac + atp + coa -> accoa + amp + ppi

ACS 1 xac > 1 xaccoa !#ab > #ab

ACt2r; Range: 0 to 100; Formula: ace + he -> ac + h

ACt2r 1 xace > 1 xac !#ab > #ab

ADHEr; Range: 0 to 100; Formula: accoa + 2 h + 2 nadh -> coa + etoh + 2 nad

ADHEr 1 xaccoa > 1 xetoh !#ab > #ab

ADK1; Range: 0 to 100; Formula: amp + atp -> 2 adp

ADK3; Range: 0 to 100; Formula: amp + gtp -> adp + gdp

COMBOSPMD; Range: 0 to 100; Formula: 2 atp + h2o + metL + prpp + ptrc -> adp + amp + co2 + dkmpp + h + pi + 2 ppi + spmd

COMBOSPMD 1 xmetL 1 xprpp 1 xptrc > 1 xco2 1 xdkmpp 1 xspmd !#abcde #fghij #klmn > #a #fghije #klmndcb !#abcde #fghij #nmlk > #a #fghije #klmndcb

COMBO9; Range: 0 to 100; Formula: 2 atp + gtp + 2 h2o + so4 + trdrd -> adp + amp + gdp + 3 h + 2 pi + ppi + so3 + trdox

COMBO10; Range: 0 to 100; Formula: aspL + gtp + imp -> amp + fum + gdp + 2 h + pi

COMBO10 1 xaspL > 1 xfum !#abcd > #abcd !#abcd > #dcba

AGPR; Range: 0 to 100; Formula: acg5sa + nadp + pi -> acg5p + h + nadph

AGPR 1 xacg5sa > 1 xacg5p !#abcdefg > #abcdefg

ALAR; Range: 0 to 100; Formula: alaL -> alaD

ALAR 1 xalaL > 1 xalaD !#abc > #abc

ALATAL; Range: 0 to 100; Formula: akg + alaL -> gluL + pyr

ALATAL 1 xakg 1 xalaL > 1 xgluL 1 xpyr !#abcde #fgh > #abcde #fgh

ALDD2x; Range: 0 to 100; Formula: acald + h2o + nad -> ac + 2 h + nadh

ALDD2x 1 xacald > 1 xac !#ab > #ab

COMBO15; Range: 0 to 100; Formula: chor + glnL + prpp -> 3ig3p + co2 +

gluL + h2o + ppi + pyr

COMBO15 1 xchor 1 xglnL 1 xprpp > 1 x3ig3p 1 xco2 1 xgluL 1 xpyr !#abcdefghij #klmno #pqrst > #tsrqbgfedcp #a #klmno #ihj

ARGSL; Range: 0 to 100; Formula: argsuc -> argL + fum

ARGSL 1 xargsuc > 1 xargL 1 xfum !#abcdefghij > #abcdef #jghi !#abcdefghij > #abcdef #ihgj

ARGSS; Range: 0 to 100; Formula: aspL + atp + citrL -> amp + argsuc + h + ppi

ARGSS 1 xaspL 1 xcitrL > 1 xargsuc !#abcd #efghij > #efghijbcda

ASAD; Range: 0 to 100; Formula: aspsa + nadp + pi -> 4pasp + h + nadph

ASAD 1 xaspsa > 1 x4pasp !#abcd > #abcd

ASNN; Range: 0 to 100; Formula: asnL + h2o -> aspL + nh4

ASNN 1 xasnL > 1 xaspL !#abcd > #abcd

ASNS1; Range: 0 to 100; Formula: aspL + atp + glnL + h2o -> amp + asnL + gluL + h + ppi

ASNS1 1 xaspL 1 xglnL > 1 xasnL 1 xgluL !#abcd #efghi > #abcd #efghi

ASNS2; Range: 0 to 0; Formula: aspL + atp + nh4 -> amp + asnL + h + ppi

ASNS2 1 xaspL > 1 xasnL !#abcd > #abcd

ASPK; Range: 0 to 100; Formula: aspL + atp -> 4pasp + adp

ASPK 1 xaspL > 1 x4pasp !#abcd > #abcd

ASPTA; Range: 0 to 100; Formula: akg + aspL -> gluL + oaa

ASPTA 1 xakg 1 xaspL > 1 xgluL 1 xoaa !#abcde #fghi > #abcde #fghi

ATPM; Range: 7.6 to 7.6; Formula: atp + h2o -> adp + h + pi

ATPS4r; Range: -100 to 100; Formula: adp + pi + 4 he -> atp + 3 h + h2o

CBPS; Range: 0 to 100; Formula: 2 atp + glnL + h2o + hco3 -> 2 adp + cbp + gluL + 2 h + pi

CBPS 1 xglnL 1 xhco3 > 1 xcbp 1 xgluL !#abcde #f > #f #abcde

CHORM; Range: 0 to 100; Formula: chor -> pphn

CHORM 1 xchor > 1 xpphn !#abcdefghij > #ihjbcdefga

CHORS; Range: 0 to 100; Formula: 3psme -> chor + pi

CHORS 1 x3psme > 1 xchor !#abcdefghij > #abgfedchij

CO2t; Range: 0 to 100; Formula: co2e -> co2

CO2t 1 xco2e > 1 xco2 !#a > #a

CS; Range: 0 to 100; Formula: accoa + h2o + oaa -> cit + coa + h

CS 1 xaccoa 1 xoaa > 1 xcit !#ab #cdef > #fedbac

CYSDS; Range: 0 to 100; Formula: cysL + h2o -> h2s + nh4 + pyr

CYSDS 1 xcysL > 1 xpyr !#abc > #abc

CYSS; Range: 0 to 100; Formula: acser + h2s -> ac + cysL + h

CYSS 1 xacser > 1 xac 1 xcysL !#abcde > #de #abc

COMBO22; Range: 0 to 100; Formula: 5mthf + cysL + h2o + homL + succoa -> coa + h + metL + nh4 + pyr + succ + thf

COMBO22 1 x5mthf 1 xcysL 1 xhomL 1 xsuccoa > 1 xmetL 1 xpyr 1 xsucc !#a #bcd #efgh #ijkl > #efgha #bcd #ijkl !#a #bcd #efgh #ijkl > #efgha #bcd #lkji

CYTBD; Range: 0 to 100; Formula: 2 h + 0.5 o2 + q8h2 -> h2o + q8 + 2 he

CYTBO3; Range: 0 to 100; Formula: 2.5 h + 0.5 o2 + q8h2 -> h2o + q8 + 2.5 he

CYTK1; Range: -100 to 100; Formula: atp + cmp -> adp + cdp

DLACt2; Range: 0 to 100; Formula: he + lacDe -> h + lacD

DLACt2 1 xlacDe > 1 xlacD !#abc > #abc

DAPDC; Range: 0 to 100; Formula: 26dapM + h -> co2 + lysL

DAPDC 1 x26dapM > 1 xco2 1 xlysL !#abcdefg > #g #abcdef !#gfedcba > #g #abcdef

DAPE; Range: 0 to 100; Formula: 26dapLL -> 26dapM

DAPE 1 x26dapLL > 1 x26dapM !#abcdefg > #abcdefg !#gfedcba > #abcdefg !#abcdefg > #gfedcba !#gfedcba > #gfedcba

COMBO25; Range: 0 to 100; Formula: e4p + h2o + pep -> 3dhq + 2 pi

COMBO25 1 xe4p 1 xpep > 1 x3dhq !#abcd #efg > #efdcbag

DHAD1; Range: 0 to 100; Formula: 23dhmb -> 3mob + h2o

DHAD1 1 x23dhmb > 1 x3mob !#abcde > #abcde

DHAPT; Range: 0 to 100; Formula: dha + pep -> dhap + pyr

DHAPT 1 xdha 1 xpep > 1 xdhap 1 xpyr !#abc #def > #abc #def !#cba #def

> #abc #def

COMBO26; Range: 0 to 100; Formula: aspsa + nadph + pyr + succoa -> coa + h2o + nadp + sl2a6o

COMBO26 1 xaspsa 1 xpyr 1 xsuccoa > 1 xsl2a6o !#abcd #efg #hijk > #abcdgfe-hijk

DHQD; Range: 0 to 100; Formula: 3dhq -> 3dhsk + h2o

DHQD 1 x3dhq > 1 x3dhsk !#abcdefg > #abcdefg

DKMPPD; Range: 0 to 100; Formula: dkmpp + h2o + o2 -> 2kmb + formate + 2 h + pi

DKMPPD 1 xdkmpp > 1 x2kmb 1 xformate !#abcdef > #bcdef #a

DKMPPD2; Range: 0 to 100; Formula: dkmpp + 3 h2o -> 2kmb + formate + 6 h + pi

DKMPPD2 1 xdkmpp > 1 x2kmb 1 xformate !#abcdef > #bcdef #a

EDA; Range: 0 to 100; Formula: 2ddg6p -> g3p + pyr

EDA 1 x2ddg6p > 1 xg3p 1 xpyr !#abcdef > #def #abc

EDD; Range: 0 to 100; Formula: 6pgc -> 2ddg6p + h2o

EDD 1 x6pgc > 1 x2ddg6p !#abcdef > #abcdef

ENO; Range: 0 to 100; Formula: 2pg -> h2o + pep

ENO 1 x2pg > 1 xpep !#abc > #abc

ETOHt2r; Range: 0 to 100; Formula: etohe + he -> etoh + h

ETOHt2r 1 xetohe > 1 xetoh !#ab > #ab

F6PA; Range: 0 to 100; Formula: f6p -> dha + g3p

F6PA 1 xf6p > 1 xdha 1 xg3p !#abcdef > #abc #def !#abcdef > #cba #def

FBA; Range: 0 to 100; Formula: fdp -> dhap + g3p

FBA 1 xfdp > 1 xdhap 1 xg3p !#abcdef > #abc #def

FBP; Range: 0 to 100; Formula: fdp + h2o -> f6p + pi

FBP 1 xfdp > 1 xf6p !#abcdef > #abcdef

FDH2; Range: 0 to 100; Formula: formate + 3 h + q8 -> co2 + q8h2 + 2 he

FDH2 1 xformate > 1 xco2 !#a > #a

FDH3; Range: 0 to 100; Formula: formate + 3 h + mqn8 -> co2 + mql8 + 2 he

FDH3 1 xformate > 1 xco2 !#a > #a

FHL; Range: 0 to 100; Formula: formate + h -> co2 + h2

FHL 1 xformate > 1 xco2 !#a > #a

FORt; Range: 0 to 100; Formula: fore -> formate

FORt 1 xfore > 1 xformate !#a > #a

FRD2; Range: 0 to 100; Formula: fum + mql8 -> mqn8 + succ

FRD2 1 xfum > 1 xsucc !#abcd > #dcba !#dcba > #dcba !#abcd > #abcd !#dcba > #abcd

FRD3; Range: 0 to 100; Formula: 2dmmql8 + fum -> 2dmmq8 + succ

FRD3 1 xfum > 1 xsucc !#abcd > #dcba !#dcba > #dcba !#abcd > #abcd !#dcba > #abcd

FTHFD; Range: 0 to 100; Formula: 10fthf + h2o -> formate + h + thf

FTHFD 1 x10fthf > 1 xformate !#a > #a

rFUM; Range: 0 to 100; Formula: fum + h2o -> malL

rFUM 1 xfum > 1 xmalL !#abcd > #abcd !#dcba > #abcd

FUMt22; Range: 0 to 100; Formula: fume + 2 he -> fum + 2 h

FUMt22 1 xfume > 1 xfum !#abcd > #abcd !#dcba > #abcd !#abcd > #dcba !#dcba > #dcba

FUMt23; Range: 0 to 100; Formula: fume + 3 he -> fum + 3 h

FUMt23 1 xfume > 1 xfum !#abcd > #abcd !#dcba > #abcd !#abcd > #dcba !#dcba > #dcba

G1PP; Range: 0 to 100; Formula: g1p + h2o -> glcD + pi

G1PP 1 xg1p > 1 xglcD !#abcdef > #abcdef

G3PD2; Range: 0 to 100; Formula: glyc3p + nadp -> dhap + h + nadph

G3PD2 1 xglyc3p > 1 xdhap !#abc > #abc

G3PD5; Range: 0 to 100; Formula: glyc3p + q8 -> dhap + q8h2

G3PD5 1 xglyc3p > 1 xdhap !#abc > #abc

G3PD6; Range: 0 to 100; Formula: glyc3p + mqn8 -> dhap + mql8

G3PD6 1 xglyc3p > 1 xdhap !#abc > #abc

G3PD7; Range: 0 to 100; Formula: 2dmmq8 + glyc3p -> 2dmmql8 + dhap

G3PD7 1 xglyc3p > 1 xdhap !#abc > #abc

G5SADs; Range: 0 to 100; Formula: glu5sa -> 1pyr5c + h + h2o

G5SADs 1 xglu5sa > 1 x1pyr5c !#abcde > #abcde

COMBO34; Range: 0 to 100; Formula: atp + gluL + h + nadph -> adp + glu5sa + nadp + pi

COMBO34 1 xgluL > 1 xglu5sa !#abcde > #abcde

G6PDH2r; Range: 0 to 100; Formula: g6p + nadp -> 6pgl + h + nadph

G6PDH2r 1 xg6p > 1 x6pgl !#abcdef > #abcdef

GAPD; Range: 0 to 100; Formula: g3p + nad + pi -> 13dpg + h + nadh

GAPD 1 xg3p > 1 x13dpg !#abc > #abc

GHMT2; Range: 0 to 100; Formula: serL + thf -> gly + h2o + mlthf

GHMT2 1 xserL > 1 xgly 1 xmlthf !#abc > #ab #c

GK1; Range: -100 to 100; Formula: atp + gmp -> adp + gdp

GLCP; Range: 0 to 100; Formula: glycogen + pi -> g1p

GLCP 1 xglycogen > 1 xg1p !#abcdef > #abcdef

COMBO36; Range: 0 to 100; Formula: atp + g1p -> adp + glycogen + ppi

COMBO36 1 xg1p > 1 xglycogen !#abcdef > #abcdef

GLCpts; Range: 0 to 100; Formula: pep + glcDe -> g6p + pyr

GLCpts 1 xglcDe 1 xpep > 1 xg6p 1 xpyr !#abcdef #ghi > #abcdef #ghi

GLNS; Range: 0 to 100; Formula: atp + gluL + nh4 -> adp + glnL + h + pi

GLNS 1 xgluL > 1 xglnL !#abcde > #abcde

GLUDC; Range: 0 to 100; Formula: gluL + h -> 4abut + co2

GLUDC 1 xgluL > 1 x4abut 1 xco2 !#abcde > #edcb #a

GLUDy; Range: 0 to 100; Formula: gluL + h2o + nadp -> akg + h + nadph + nh4

GLUDy 1 xgluL > 1 xakg !#abcde > #abcde

GLUN; Range: 0 to 100; Formula: glnL + h2o -> gluL + nh4

GLUN 1 xglnL > 1 xgluL !#abcde > #abcde

GLUSy; Range: 0 to 100; Formula: akg + glnL + h + nadph -> 2 gluL + nadp

GLUSy 1 xakg 1 xglnL > 2 xgluL !#abcde #fghij > #fghij !#abcde #fghij > #abcde

GLXCL; Range: 0 to 100; Formula: 2 glx + h -> 2h3oppan + co2

GLXCL 1 xglx 1 xglx > 1 x2h3oppan 1 xco2 !#ab #cd > #abd #c

COMBO37; Range: 0 to 100; Formula: coa + nad + thrL -> accoa + gly + h + nadh

COMBO37 1 xthrL > 1 xaccoa 1 xgly !#abcd > #cd #ab

GLYCDx; Range: 0 to 100; Formula: glyc + nad -> dha + h + nadh

GLYCDx 1 xglyc > 1 xdha !#abc > #abc !#abc > #cba !#cba > #abc !#cba > #cba

GLYCK; Range: 0 to 100; Formula: atp + glycR -> 3pg + adp + h

GLYCK 1 xglycR > 1 x3pg !#abc > #abc

GLYCL; Range: 0 to 100; Formula: gly + nad + thf -> co2 + mlthf + nadh + nh4

GLYCL 1 xgly > 1 xco2 1 xmlthf !#ab > #a #b

GLYCLTDx; Range: 0 to 100; Formula: glx + h + nadh -> glyclt + nad

GLYCLTDx 1 xglx > 1 xglyclt !#ab > #ab

GLYCLTDy; Range: 0 to 100; Formula: glx + h + nadph -> glyclt + nadp

GLYCLTDy 1 xglx > 1 xglyclt !#ab > #ab

GLYCTO2; Range: 0 to 100; Formula: glyclt + q8 -> glx + q8h2

GLYCTO2 1 xglyclt > 1 xglx !#ab > #ab

GLYCTO3; Range: 0 to 100; Formula: glyclt + mqn8 -> glx + mql8

GLYCTO3 1 xglyclt > 1 xglx !#ab > #ab

GLYCTO4; Range: 0 to 100; Formula: 2dmmq8 + glyclt -> 2dmmql8 + glx

GLYCTO4 1 xglyclt > 1 xglx !#ab > #ab

GLYCt; Range: 0 to 100; Formula: glyc -> glyce

GLYCt 1 xglyc > 1 xglyce !#abc > #abc !#abc > #cba !#cba > #abc !#cba > #cba

GLYK; Range: 0 to 100; Formula: atp + glyc -> adp + glyc3p + h

GLYK 1 xglyc > 1 xglyc3p !#abc > #abc !#cba > #abc

COMBO38; Range: 0 to 100; Formula: dhap + h2o -> h + lacD + pi

COMBO38 1 xdhap > 1 xlacD !#abc > #cba

GMPS2; Range: 0 to 100; Formula: atp + glnL + h2o + xmp -> amp + gluL + gmp + 2 h + ppi

GMPS2 1 xglnL > 1 xgluL !#abcde > #abcde

GND; Range: 0 to 100; Formula: 6pgc + nadp -> co2 + nadph + ru5pD

GND 1 x6pgc > 1 xco2 1 xru5pD !#abcdef > #a #bcdef

H2Ot; Range: -100 to 100; Formula: h2oe -> h2o

HCO3E; Range: 0 to 100; Formula: co2 + h2o -> h + hco3

HCO3E 1 xco2 > 1 xhco3 !#a > #a

HEX1; Range: 0 to 100; Formula: atp + glcD -> adp + g6p + h

HEX1 1 xglcD > 1 xg6p !#abcdef > #abcdef

HSDy; Range: 0 to 100; Formula: homL + nadp -> aspsa + h + nadph

HSDy 1 xhomL > 1 xaspsa !#abcd > #abcd

COMBO41; Range: 0 to 100; Formula: atp + h2o + homL -> adp + h + pi + thrL

COMBO41 1 xhomL > 1 xthrL !#abcd > #abcd

HYD1; Range: 0 to 100; Formula: 2 h + h2 + q8 -> q8h2 + 2 he

HYD2; Range: 0 to 100; Formula: 2 h + h2 + mqn8 -> mql8 + 2 he

HYD3; Range: 0 to 100; Formula: 2dmmq8 + 2 h + h2 -> 2dmmql8 + 2 he

ICDHyr; Range: 0 to 100; Formula: icit + nadp -> akg + co2 + nadph

ICDHyr 1 xicit > 1 xakg 1 xco2 !#abcdef > #abcde #f

ICL; Range: 0 to 100; Formula: icit -> glx + succ

ICL 1 xicit > 1 xglx 1 xsucc !#abcdef > #ab #fcde !#abcdef > #ab #edcf

ILETA; Range: 0 to 100; Formula: akg + ileL -> 3mop + gluL

ILETA 1 xakg 1 xileL > 1 x3mop 1 xgluL !#abcde #fghijk > #fghijk #abcde

IMPD; Range: 0 to 100; Formula: h2o + imp + nad -> h + nadh + xmp

IPMD; Range: 0 to 100; Formula: 3c2hmp + nad -> 3c4mop + h + nadh

IPMD 1 x3c2hmp > 1 x3c4mop !#abcdefg > #abcdefg

IPPMIa; Range: 0 to 100; Formula: 3c2hmp -> 2ippm + h2o

IPPMIa 1 x3c2hmp > 1 x2ippm !#abcdefg > #abcdefg

IPPMIb; Range: 0 to 100; Formula: 2ippm + h2o -> 3c3hmp

IPPMIb 1 x2ippm > 1 x3c3hmp !#abcdefg > #abcdefg

IPPS; Range: 0 to 100; Formula: 3mob + accoa + h2o -> 3c3hmp + coa + h

IPPS 1 x3mob 1 xaccoa > 1 x3c3hmp !#abcde #fg > #fgbcdae

LLACD2; Range: 0 to 100; Formula: lacL + q8 -> pyr + q8h2

LLACD2 1 xlacL > 1 xpyr !#abc > #abc

LLACD3; Range: 0 to 100; Formula: lacL + mqn8 -> mql8 + pyr

LLACD3 1 xlacL > 1 xpyr !#abc > #abc

LLACt2r; Range: 0 to 100; Formula: he + lacLe -> h + lacL

LLACt2r 1 xlacLe > 1 xlacL !#abc > #abc

LDHD; Range: 0 to 100; Formula: lacD + nad -> h + nadh + pyr

LDHD 1 xlacD > 1 xpyr !#abc > #abc

LDHD2; Range: 0 to 100; Formula: lacD + q8 -> pyr + q8h2

LDHD2 1 xlacD > 1 xpyr !#abc > #abc

LEUTAi; Range: 0 to 100; Formula: 4mop + gluL -> akg + leuL

LEUTAi 1 x4mop 1 xgluL > 1 xakg 1 xleuL !#abcdef #ghijk > #ghijk #abcdef

MALS; Range: 0 to 100; Formula: accoa + glx + h2o -> coa + h + malL

MALS 1 xaccoa 1 xglx > 1 xmalL !#ab #cd > #cdba

MDH; Range: 0 to 100; Formula: malL + nad -> h + nadh + oaa

MDH 1 xmalL > 1 xoaa !#abcd > #abcd

MDH2; Range: 0 to 100; Formula: malL + q8 -> oaa + q8h2

MDH2 1 xmalL > 1 xoaa !#abcd > #abcd

MDH3; Range: 0 to 100; Formula: malL + mqn8 -> mql8 + oaa

MDH3 1 xmalL > 1 xoaa !#abcd > #abcd

ME1; Range: 0 to 100; Formula: malL + nad -> co2 + nadh + pyr

ME1 1 xmalL > 1 xco2 1 xpyr !#abcd > #d #abc

ME2; Range: 0 to 100; Formula: malL + nadp -> co2 + nadph + pyr

ME2 1 xmalL > 1 xco2 1 xpyr !#abcd > #d #abc

MTHFC; Range: 0 to 100; Formula: h2o + methf -> 10fthf

MTHFC 1 xmethf > 1 x10fthf !#a > #a

MTHFD; Range: 0 to 100; Formula: mlthf + nadp -> h + methf + nadph

MTHFD 1 xmlthf > 1 xmethf !#a > #a

MTHFR2; Range: 0 to 100; Formula: h + mlthf + nadh -> 5mthf + nad

MTHFR2 1 xmlthf > 1 x5mthf !#a > #a

NACODA; Range: 0 to 100; Formula: acg5sa + h2o -> ac + glu5sa

NACODA 1 xacg5sa > 1 xac 1 xglu5sa !#abcdefg > #fg #abcde

NADH10; Range: 0 to 100; Formula: h + mqn8 + nadh -> mql8 + nad

NADH5; Range: 0 to 100; Formula: h + nadh + q8 -> nad + q8h2

NADH6; Range: 0 to 100; Formula: 4.5 h + nadh + q8 -> nad + q8h2 + 3.5 he

NADH7; Range: 0 to 100; Formula: 3 h + mqn8 + nadh -> mql8 + nad + 2 he

NADH8; Range: 0 to 100; Formula: 2dmmq8 + 3.8 h + nadh -> 2dmmql8 + nad + 2.8 he

NADH9; Range: 0 to 100; Formula: 2dmmq8 + h + nadh -> 2dmmql8 + nad

NADK; Range: 0 to 100; Formula: atp + nad -> adp + h + nadp

NAt31; Range: -100 to 100; Formula: na1 + he -> h + na1e

NAt315; Range: 0 to 100; Formula: 2 na1 + 3 he -> 3 h + 2 na1e

NAt32; Range: 0 to 100; Formula: na1 + 2 he -> 2 h + na1e

NDPK1; Range: 0 to 100; Formula: atp + gdp -> adp + gtp

NDPK2; Range: -100 to 100; Formula: atp + udp -> adp + utp

NDPK3; Range: -100 to 100; Formula: atp + cdp -> adp + ctp

NH4t; Range: -100 to 100; Formula: nh4e -> nh4

NO3R1; Range: 0 to 100; Formula: 2 h + no3 + q8h2 -> h2o + no2 + q8 + 2 he

NO3R2; Range: 0 to 100; Formula: 2 h + mql8 + no3 -> h2o + mqn8 + no2 + 2 he

NO2t2r; Range: -100 to 100; Formula: he + no2e -> h + no2

NO3t7; Range: 0 to 100; Formula: no2 + no3e -> no3 + no2e

NTPP6; Range: 0 to 100; Formula: atp + h2o -> amp + h + ppi

NTRIR2x; Range: 0 to 100; Formula: 5 h + 3 nadh + no2 -> 2 h2o + 3 nad + nh4

O2t; Range: -100 to 100; Formula: o2e -> o2

OCBT; Range: 0 to 100; Formula: cbp + orn -> citrL + h + pi

OCBT 1 xcbp 1 xorn > 1 xcitrL !#a #bcdef > #bcdefa

OMCDC; Range: 0 to 100; Formula: 3c4mop + h -> 4mop + co2

OMCDC 1 x3c4mop > 1 x4mop 1 xco2 !#abcdefg > #abcdeg #f

ORNDC; Range: 0 to 100; Formula: h + orn -> co2 + ptrc

ORNDC 1 xorn > 1 xco2 1 xptrc !#abcde > #a #bcde !#abcde > #a #edcb

P5CD; Range: 0 to 100; Formula: 1pyr5c + 2 h2o + nad -> gluL + h + nadh

P5CD 1 x1pyr5c > 1 xgluL !#abcde > #abcde

P5CR; Range: 0 to 100; Formula: 1pyr5c + 2 h + nadph -> nadp + proL

P5CR 1 x1pyr5c > 1 xproL !#abcde > #abcde

PDH; Range: 0 to 100; Formula: coa + nad + pyr -> accoa + co2 + nadh

PDH 1 xpyr > 1 xaccoa 1 xco2 !#abc > #bc #a

PFK; Range: 0 to 100; Formula: atp + f6p -> adp + fdp + h

PFK 1 xf6p > 1 xfdp !#abcdef > #abcdef

PFL; Range: 0 to 100; Formula: coa + pyr -> accoa + formate

PFL 1 xpyr > 1 xaccoa 1 xformate !#abc > #bc #a

COMBO47; Range: 0 to 100; Formula: 3pg + gluL + h2o + nad -> akg + h + nadh + pi + serL

COMBO47 1 x3pg 1 xgluL > 1 xakg 1 xserL !#abc #defgh > #defgh #abc

PGI; Range: 0 to 100; Formula: g6p -> f6p

PGI 1 xg6p > 1 xf6p !#abcdef > #abcdef

PGK; Range: 0 to 100; Formula: 3pg + atp -> 13dpg + adp

PGK 1 x3pg > 1 x13dpg !#abc > #abc

PGL; Range: 0 to 100; Formula: 6pgl + h2o -> 6pgc + h

PGL 1 x6pgl > 1 x6pgc !#abcdef > #abcdef

PGM; Range: 0 to 100; Formula: 2pg -> 3pg

PGM 1 x2pg > 1 x3pg !#abc > #abc

PGMT; Range: 0 to 100; Formula: g1p -> g6p

PGMT 1 xg1p > 1 xg6p !#abcdef > #abcdef

PHETA1; Range: 0 to 100; Formula: akg + pheL -> gluL + phpyr

PHETA1 1 xakg 1 xpheL > 1 xgluL 1 xphpyr !#abcde #fghijklmn > #abcde #fghijklmn !#abcde #fghinmlkj > #abcde #fghijklmn !#abcde #fghijklmn > #abcde #fghinmlkj !#abcde #fghinmlkj > #abcde #fghinmlkj

PIabc; Range: 0 to 100; Formula: atp + h2o + pie -> adp + h + 2 pi

PIt2r; Range: -100 to 100; Formula: he + pie -> h + pi

POX; Range: 0 to 100; Formula: h2o + pyr + q8 -> ac + co2 + q8h2

POX 1 xpyr > 1 xac 1 xco2 !#abc > #bc #a

rPPA; Range: 0 to 100; Formula: h2o + ppi -> h + 2 pi

PPC; Range: 0 to 100; Formula: co2 + h2o + pep -> h + oaa + pi

PPC 1 xco2 1 xpep > 1 xoaa !#a #bcd > #bcda

PPCK; Range: 0 to 100; Formula: atp + oaa -> adp + co2 + pep

PPCK 1 xoaa > 1 xco2 1 xpep !#abcd > #d #abc

PPND; Range: 0 to 100; Formula: nad + pphn -> 34hpp + co2 + nadh

PPND 1 xpphn > 1 x34hpp 1 xco2 !#abcdefghij > #abcdefghi #j !#abcdefghij > #abcdihgfe #j

PPNDH; Range: 0 to 100; Formula: h + pphn -> co2 + h2o + phpyr

PPNDH 1 xpphn > 1 xco2 1 xphpyr !#abcdefghij > #j #abcdefghi !#abcdefghij > #j #abcdihgfe

PPS; Range: 0 to 100; Formula: atp + h2o + pyr -> amp + 2 h + pep + pi

PPS 1 xpyr > 1 xpep !#abc > #abc

PROD2; Range: 0 to 100; Formula: fad + proL -> 1pyr5c + fadh2 + h

PROD2 1 xproL > 1 x1pyr5c !#abcde > #abcde

PRPPS; Range: 0 to 100; Formula: atp + r5p -> amp + h + prpp

PRPPS 1 xr5p > 1 xprpp !#abcde > #abcde

PSCVT; Range: 0 to 100; Formula: pep + skm5p -> 3psme + pi

PSCVT 1 xpep 1 xskm5p > 1 x3psme !#abc #defghij > #defghijbac

PTAr; Range: 0 to 100; Formula: accoa + pi -> actp + coa

PTAr 1 xaccoa > 1 xactp !#ab > #ab

PYK; Range: 0 to 100; Formula: adp + h + pep -> atp + pyr

PYK 1 xpep > 1 xpyr !#abc > #abc

PYRt2r; Range: 0 to 100; Formula: he + pyre -> h + pyr

PYRt2r 1 xpyre > 1 xpyr !#abc > #abc

RNTR1; Range: 0 to 100; Formula: atp + trdrd -> datp + h2o + trdox

RNTR2; Range: 0 to 100; Formula: gtp + trdrd -> dgtp + h2o + trdox

RNTR3; Range: 0 to 100; Formula: ctp + trdrd -> dctp + h2o + trdox

RPE; Range: 0 to 100; Formula: ru5pD -> xu5pD

RPE 1 xru5pD > 1 xxu5pD !#abcde > #abcde

RPI; Range: 0 to 100; Formula: r5p -> ru5pD

RPI 1 xr5p > 1 xru5pD !#abcde > #abcde

SDPDS; Range: 0 to 100; Formula: h2o + sl26da -> 26dapLL + succ

SDPDS 1 xsl26da > 1 x26dapLL 1 xsucc !#abcdefghijk > #abcdefg #hijk !#abcdefghijk > #gfedcba #hijk !#abcdefghijk > #abcdefg #kjih !#abcdefghijk > #gfedcba #kjih

SDPTA; Range: 0 to 100; Formula: akg + sl26da -> gluL + sl2a6o

SDPTA 1 xakg 1 xsl26da > 1 xgluL 1 xsl2a6o !#abcde #fghijklmnop > #abcde #fghijklmnop

SERAT; Range: 0 to 100; Formula: accoa + serL -> acser + coa

SERAT 1 xaccoa 1 xserL > 1 xacser !#ab #cde > #cdeab

SERDL; Range: 0 to 100; Formula: serL -> nh4 + pyr

SERDL 1 xserL > 1 xpyr !#abc > #abc

SHK3Dr; Range: 0 to 100; Formula: 3dhsk + h + nadph -> nadp + skm

SHK3Dr 1 x3dhsk > 1 xskm !#abcdefg > #abcdefg

SHKK; Range: 0 to 100; Formula: atp + skm -> adp + h + skm5p

SHKK 1 xskm > 1 xskm5p !#abcdefg > #abcdefg

SSALx; Range: 0 to 100; Formula: h2o + nad + sucsal -> 2 h + nadh + succ

SSALx 1 xsucsal > 1 xsucc !#abcd > #abcd !#abcd > #dcba

SSALy; Range: 0 to 100; Formula: h2o + nadp + sucsal -> 2 h + nadph + succ

SSALy 1 xsucsal > 1 xsucc !#abcd > #abcd !#abcd > #dcba

SUCCabc; Range: 0 to 100; Formula: atp + h2o + succe -> adp + h + pi + succ

SUCCabc 1 xsucce > 1 xsucc !#abcd > #abcd !#dcba > #abcd !#abcd > #dcba !#dcba > #dcba

SUCCt22; Range: 0 to 100; Formula: 2 he + succe -> 2 h + succ

SUCCt22 1 xsucce > 1 xsucc !#abcd > #abcd !#dcba > #abcd !#abcd > #dcba !#dcba > #dcba

SUCCt23; Range: 0 to 100; Formula: 3 he + succe -> 3 h + succ

SUCCt23 1 xsucce > 1 xsucc !#abcd > #abcd !#dcba > #abcd !#abcd > #dcba !#dcba > #dcba

SUCCt2b; Range: 0 to 100; Formula: h + succ -> he + succe

SUCCt2b 1 xsucc > 1 xsucce !#abcd > #abcd !#abcd > #dcba !#dcba > #abcd

!#dcba > #dcba

SUCD1i; Range: 0 to 100; Formula: fad + succ -> fadh2 + fum

SUCD1i 1 xsucc > 1 xfum !#abcd > #dcba !#abcd > #abcd !#dcba > #dcba !#dcba > #abcd

SUCD4; Range: -100 to 100; Formula: fadh2 + q8 -> fad + q8h2

SUCFUMt; Range: 0 to 100; Formula: succ + fume -> fum + succe

SUCFUMt 1 xfume 1 xsucc > 1 xfum 1 xsucce !#abcd #efgh > #abcd #efgh !#dcba #efgh > #abcd #efgh !#abcd #efgh > #dcba #efgh !#dcba #efgh > #dcba #efgh !#abcd #efgh > #abcd #hgfe !#dcba #efgh > #abcd #hgfe !#abcd #efgh > #dcba #hgfe !#dcba #efgh > #dcba #hgfe !#abcd #hgfe > #abcd #efgh !#dcba #hgfe > #abcd #efgh !#abcd #hgfe > #dcba #efgh !#dcba #hgfe > #dcba #efgh !#abcd #hgfe > #abcd #hgfe !#dcba #hgfe > #abcd #hgfe !#abcd #hgfe > #dcba #hgfe !#dcba #hgfe > #dcba #hgfe

SUCOAS; Range: 0 to 100; Formula: atp + coa + succ -> adp + pi + succoa

SUCOAS 1 xsucc > 1 xsuccoa !#abcd > #abcd !#dcba > #abcd

SULR; Range: -100 to 100; Formula: 3 h2o + h2s + 3 nadp -> 5 h + 3 nadph + so3

SULabc; Range: 0 to 100; Formula: atp + h2o + so4e -> adp + h + pi + so4

TALA; Range: 0 to 100; Formula: g3p + s7p -> e4p + f6p

TALA 1 xg3p 1 xs7p > 1 xe4p 1 xf6p !#abc #defghij > #ghij #defabc

TESTAKGDH; Range: 0 to 100; Formula: akg + coa + nad -> co2 + nadh + succoa

TESTAKGDH 1 xakg > 1 xco2 1 xsuccoa !#abcde > #a #bcde

TESTNADTRHD; Range: 0 to 100; Formula: nad + nadph -> nadh + nadp

THD2; Range: 0 to 100; Formula: nadh + nadp + 2 he -> 2 h + nad + nadph

THRAr; Range: 0 to 100; Formula: thrL -> acald + gly

THRAr 1 xthrL > 1 xacald 1 xgly !#abcd > #cd #ab

THRDL; Range: 0 to 100; Formula: thrL -> 2obut + nh4

THRDL 1 xthrL > 1 x2obut !#abcd > #abcd

TKT1; Range: 0 to 100; Formula: r5p + xu5pD -> g3p + s7p

TKT1 1 xr5p 1 xxu5pD > 1 xg3p 1 xs7p !#abcde #fghij > #hij #fgabcde

TKT2; Range: 0 to 100; Formula: e4p + xu5pD -> f6p + g3p

TKT2 1 xe4p 1 xxu5pD > 1 xf6p 1 xg3p !#abcd #efghi > #efabcd #ghi

TPI; Range: 0 to 100; Formula: dhap -> g3p

TPI 1 xdhap > 1 xg3p !#abc > #cba

TRDR; Range: 0 to 100; Formula: h + nadph + trdox -> nadp + trdrd

TRPAS2; Range: 0 to 100; Formula: h2o + trpL -> indole + nh4 + pyr

TRPAS2 1 xtrpL > 1 xindole 1 xpyr !#abcdefghijk > #kdefghij #abc

TRPS1; Range: 0 to 100; Formula: 3ig3p + serL -> g3p + h2o + trpL

TRPS1 1 x3ig3p 1 xserL > 1 xg3p 1 xtrpL !#abcdefghijk #lmn > #cba #lmndefghijk

TRPS2; Range: 0 to 100; Formula: indole + serL -> h2o + trpL

TRPS2 1 xindole 1 xserL > 1 xtrpL !#abcdefgh #ijk > #ijkbcdefgha

TRPS3; Range: 0 to 100; Formula: 3ig3p -> g3p + indole

TRPS3 1 x3ig3p > 1 xg3p 1 xindole !#abcdefghijk > #cba #kdefghij

TRSAR; Range: 0 to 100; Formula: 2h3oppan + h + nadh -> glycR + nad

TRSAR 1 x2h3oppan > 1 xglycR !#abc > #abc

TYRTA; Range: 0 to 100; Formula: akg + tyrL -> 34hpp + gluL

TYRTA 1 xakg 1 xtyrL > 1 x34hpp 1 xgluL !#abcde #fghijklmn > #fghijklmn #abcde !#abcde #fghijklmn > #fghinmlkj #abcde !#abcde #fghinmlkj > #fghijklmn #abcde !#abcde #fghinmlkj > #fghinmlkj #abcde

UMPK; Range: -100 to 100; Formula: atp + ump -> adp + udp

UNK3; Range: 0 to 100; Formula: 2kmb + gluL -> akg + metL

UNK3 1 x2kmb 1 xgluL > 1 xakg 1 xmetL !#abcde #fghij > #fghij #abcde

VALTA; Range: 0 to 100; Formula: akg + valL -> 3mob + gluL

VALTA 1 xakg 1 xvalL > 1 x3mob 1 xgluL !#abcde #fghij > #fghij #abcde

HISSYN; Range: 0 to 100; Formula: 10fthf + atp + glnL + 2 h2o + 2 nad + prpp -> akg + 5 h + hisL + imp + 2 nadh + pi + 2 ppi + thf

HISSYN 1 x10fthf 1 xglnL 1 xprpp > 1 xakg 1 xhisL !#a #bcdef #ghijk > #bcdef #kjihga

UMPSYN1; Range: 0 to 100; Formula: aspL + cbp + h + prpp + q8 -> co2 + h2o + pi + ppi + q8h2 + ump

UMPSYN1 1 xcbp 1 xprpp 1 xaspL > 1 xco2 !#a #bcdef #ghij > #g

UMPSYN2; Range: 0 to 100; Formula: aspL + cbp + h + mqn8 + prpp -> co2 + h2o + mql8 + pi + ppi + ump

UMPSYN2 1 xcbp 1 xprpp 1 xaspL > 1 xco2 !#a #bcdef #ghij > #g

CTPSYN; Range: 0 to 100; Formula: atp + glnL + h2o + utp -> adp + ctp + gluL + 2 h + pi

CTPSYN 1 xglnL > 1 xgluL !#abcde > #abcde

IMPSYN1; Range: 0 to 100; Formula: 2 10fthf + aspL + 5 atp + 2 glnL + gly + h2o + hco3 + prpp -> 5 adp + fum + 2 gluL + 7 h + imp + 5 pi + ppi + 2 thf

IMPSYN1 2 x10fthf 1 xgly 1 xhco3 1 xprpp 1 xaspL 2 xglnL > 1 xfum 2 xgluL !#a #bc #d #efghi #jklm #stuvw > #jklm #stuvw !#a #bc #d #efghi #jklm #vutsw > #jklm #stuvw

IMPSYN2; Range: 0 to 100; Formula: 10fthf + aspL + 6 atp + formate + 2 glnL + gly + h2o + hco3 + prpp -> 6 adp + fum + 2 gluL + 7 h + imp + 6 pi + ppi + thf

IMPSYN2 1 x10fthf 1 xformate 1 xgly 1 xhco3 1 xprpp 1 xaspL 2 xglnL > 1 xfum 2 xgluL !#a #b #cd #e #fghij #klmn #opqrs > #klmn #opqrs !#a #b #cd #e #fghij #klmn #opqrs > #klmn #opqrs

dTTPSYN; Range: 0 to 100; Formula: h + mlthf + nadph + trdrd + utp -> dttp + h2o + nadp + thf + trdox

dTTPSYN 1 xmlthf > !#a >

COASYN; Range: 0 to 100; Formula: 3mob + aspL + 4 atp + ctp + cysL + h2o + mlthf + nadph -> 2 adp + amp + cmp + 2 co2 + coa + nadp + 3 ppi + thf

COASYN 1 x3mob 1 xmlthf 1 xaspL 1 xcysL > 2 xco2 !#abcde #f #ghij #klm > #g !#abcde #f #ghij #klm > #k

FADSYN; Range: 0 to 100; Formula: 2 atp + gtp + h2o + nadph + 2 ru5pD -> adp + fad + 3 formate + 2 h + nadp + nh4 + 3 pi + 2 ppi

FADSYN .0000000000001 x10fthf 1 xru5pD 1 xru5pD > 3 xformate !#a #bcdef #ghijk > #e !#a #bcdef #ghijk > #j !#a #bcdef #ghijk > #a

CDPDAGSYN; Range: 0 to 100; Formula: 16.86 accoa + 14.86 atp + ctp + glyc3p + 29.48 h + 14.86 hco3 + 28.48 nadph -> 14.86 adp + cdpdagEC + 14.86

co2 + 16.86 coa + 14.86 h2o + 28.48 nadp + 14.86 pi + ppi

CDPDAGSYN 16.86 xaccoa 1 xglyc3p 14.86 xhco3 > 14.86 xco2 !#ab #cde #f > #f

PSSYN; Range: 0 to 100; Formula: cdpdagEC + serL -> cmp + h + psEC

PSSYN 1 xserL > !#abc >

PESYN; Range: 0 to 100; Formula: cdpdagEC + serL -> cmp + co2 + peEC

PESYN 1 xserL > 1 xco2 !#abc > #a

PGSYN; Range: 0 to 100; Formula: cdpdagEC + glyc3p + h2o -> cmp + h + pgEC + pi

PGSYN 1 xglyc3p > !#abc >

CLPNSYN; Range: 0 to 100; Formula: 2 cdpdagEC + 2 glyc3p + 2 h2o -> clpnEC + 2 cmp + glyc + 2 h + 2 pi

CLPNSYN 2 xglyc3p > 1 xglyc !#abc > #abc !#abc > #cba

LPSSYN; Range: 0 to 100; Formula: 43 accoa + 44 atp + 7 ctp + 2 f6p + 2 g1p + 2 glnL + 52 h + 35 hco3 + 66 nadph + 2 peEC + 5 pep + 5 ru5pD + 3 s7p + 4 utp -> 2 ac + 44 adp + 2 cdp + 2 cdpdagEC + 3 cmp + 35 co2 + 43 coa + 2 gluL + 15 h2o + lpsEC + 66 nadp + 48 pi + 14 ppi + 3 udp + ump

LPSSYN 2 xf6p 2 xg1p 5 xpep 5 xru5pD 3 xs7p 43 xaccoa 2 xglnL 35 xhco3 > 2 xac 35 xco2 2 xgluL !#abcdef #ghijkl #mno #pqrst #uvwxyzA #BC #FGHIJ #P > #BC #P #FGHIJ

PEPTIDOSYN; Range: 0 to 100; Formula: 26dapM + 2 accoa + 2 alaD + alaL + 5 atp + 2 f6p + 2 glnL + h2o + nadph + pep + 2 utp -> 5 adp + 2 coa + gluL + 7 h + nadp + peptidoEC + 7 pi + 2 ppi + udp + ump

PEPTIDOSYN 1 x26dapM 2 xaccoa 2 xalaD 1 xalaL 2 xf6p 1 xpep 2 xglnL > 1 xgluL !#abcdefg #hi #jkl #mno #pqrstu #vwx #yzABC > #yzABC !#gfedcba #hi #jkl #mno #pqrstu #vwx #yzABC > #yzABC

NADSYN1; Range: 0 to 100; Formula: aspL + 2 atp + dhap + h + nh4 + prpp + q8 -> amp + co2 + 2 h2o + nad + pi + 3 ppi + q8h2

NADSYN1 1 xdhap 1 xprpp 1 xaspL > 1 xco2 !#abc #defgh #ijkl > #i

NADSYN2; Range: 0 to 100; Formula: aspL + 2 atp + dhap + h + mqn8 + nh4 + prpp -> amp + co2 + 2 h2o + mql8 + nad + pi + 3 ppi

NADSYN2 1 xdhap 1 xprpp 1 xaspL > 1 xco2 !#abc #defgh #ijkl > #i

THFSYN; Range: 0 to 100; Formula: 2 atp + chor + glnL + gtp + 4 h2o + nad + nadph -> adp + amp + formate + glyclt + 5 h + nadh + nadp + 2 pi + 2 ppi + pyr + thf

THFSYN 1 xglnL .0000000000001 x10fthf 1 xchor .0000000000001 xprpp > 1 xformate 1 xglyclt 1 xpyr !#abcde #f #ghijklmnop #qrstu > #f #tu #onp

G3PP; Range: 0 to 100; Formula: glyc3p + h2o -> glyc + pi

G3PP 1 xglyc3p > 1 xglyc !#def > #def !#fed > #def

EX_co2_r; Range: 0 to 0; Formula: -> co2e

ACKr_r; Range: 0 to 100; Formula: actp + adp -> ac + atp

ACKr_r 1 xactp > 1 xac !#ab > #ab

rACONT_r; Range: 0 to 100; Formula: icit -> cit

rACONT_r 1 xicit > 1 xcit !#abcdef > #abcdef

ACOTA_r; Range: 0 to 100; Formula: acg5sa + gluL -> acorn + akg

ACOTA_r 1 xacg5sa 1 xgluL > 1 xacorn 1 xakg !#abcdefg #hijkl > #abcdefg #hijkl

ACt2r_r; Range: 0 to 100; Formula: ac + h -> ace + he

ACt2r_r 1 xac > 1 xace !#ab > #ab

ADHEr_r; Range: 0 to 100; Formula: coa + etoh + 2 nad -> accoa + 2 h + 2 nadh

ADHEr_r 1 xetoh > 1 xaccoa !#ab > #ab

AGPR_r; Range: 0 to 100; Formula: acg5p + h + nadph -> acg5sa + nadp + pi

AGPR_r 1 xacg5p > 1 xacg5sa !#abcdefg > #abcdefg

ALAR_r; Range: 0 to 100; Formula: alaD -> alaL

ALAR_r 1 xalaD > 1 xalaL !#abc > #abc

ALATAL_r; Range: 0 to 100; Formula: gluL + pyr -> akg + alaL

ALATAL_r 1 xgluL 1 xpyr > 1 xakg 1 xalaL !#abcde #fgh > #abcde #fgh

ARGSL_r; Range: 0 to 100; Formula: argL + fum -> argsuc

ARGSL_r 1 xargL 1 xfum > 1 xargsuc !#abcdef #jghi > #abcdefghij !#abcdef #ihgj > #abcdefghij

ASAD_r; Range: 0 to 100; Formula: 4pasp + h + nadph -> aspsa + nadp + pi

ASAD_r 1 x4pasp > 1 xaspsa !#abcd > #abcd

ASPK_r; Range: 0 to 100; Formula: 4pasp + adp -> aspL + atp

ASPK_r 1 x4pasp > 1 xaspL !#abcd > #abcd

ASPTA_r; Range: 0 to 100; Formula: gluL + oaa -> akg + aspL

ASPTA_r 1 xgluL 1 xoaa > 1 xakg 1 xaspL !#abcde #fghi > #abcde #fghi

CO2t_r; Range: 0 to 100; Formula: co2 -> co2e

CO2t_r 1 xco2 > 1 xco2e !#a > #a

DLACt2_r; Range: 0 to 100; Formula: h + lacD -> he + lacDe

DLACt2_r 1 xlacD > 1 xlacDe !#abc > #abc

DAPE_r; Range: 0 to 100; Formula: 26dapM -> 26dapLL

DAPE_r 1 x26dapM > 1 x26dapLL !#abcdefg > #abcdefg !#abcdefg > #gfedcba !#gfedcba > #abcdefg !#gfedcba > #gfedcba

DHQD_r; Range: 0 to 100; Formula: 3dhsk + h2o -> 3dhq

DHQD_r 1 x3dhsk > 1 x3dhq !#abcdefg > #abcdefg

ENO_r; Range: 0 to 100; Formula: h2o + pep -> 2pg

ENO_r 1 xpep > 1 x2pg !#abc > #abc

ETOHt2r_r; Range: 0 to 100; Formula: etoh + h -> etohe + he

ETOHt2r_r 1 xetoh > 1 xetohe !#ab > #ab

F6PA_r; Range: 0 to 100; Formula: dha + g3p -> f6p

F6PA_r 1 xdha 1 xg3p > 1 xf6p !#abc #def > #abcdef !#cba #def > #abcdef

FBA_r; Range: 0 to 100; Formula: dhap + g3p -> fdp

FBA_r 1 xdhap 1 xg3p > 1 xfdp !#abc #def > #abcdef

FORt_r; Range: 0 to 100; Formula: formate -> fore

FORt_r 1 xformate > 1 xfore !#a > #a

rFUM_r; Range: 0 to 100; Formula: malL -> fum + h2o

rFUM_r 1 xmalL > 1 xfum !#abcd > #abcd !#abcd > #dcba

G3PD2_r; Range: 0 to 100; Formula: dhap + h + nadph -> glyc3p + nadp

G3PD2_r 1 xdhap > 1 xglyc3p !#abc > #abc

G6PDH2r_r; Range: 0 to 100; Formula: 6pgl + h + nadph -> g6p + nadp

G6PDH2r_r 1 x6pgl > 1 xg6p !#abcdef > #abcdef

GAPD_r; Range: 0 to 100; Formula: 13dpg + h + nadh -> g3p + nad + pi

GAPD_r 1 x13dpg > 1 xg3p !#abc > #abc

GLUDy_r; Range: 0 to 100; Formula: akg + h + nadph + nh4 -> gluL + h2o + nadp

GLUDy_r 1 xakg > 1 xgluL !#abcde > #abcde

GLYCt_r; Range: 0 to 100; Formula: glyce -> glyc

GLYCt_r 1 xglyce > 1 xglyc !#abc > #abc !#cba > #abc !#abc > #cba !#cba > #cba

HCO3E_r; Range: 0 to 100; Formula: h + hco3 -> co2 + h2o

HCO3E_r 1 xhco3 > 1 xco2 !#a > #a

HSDy_r; Range: 0 to 100; Formula: aspsa + h + nadph -> homL + nadp

HSDy_r 1 xaspsa > 1 xhomL !#abcd > #abcd

ICDHyr_r; Range: 0 to 100; Formula: akg + co2 + nadph -> icit + nadp

ICDHyr_r 1 xakg 1 xco2 > 1 xicit !#abcde #f > #abcdef

ILETA_r; Range: 0 to 100; Formula: 3mop + gluL -> akg + ileL

ILETA_r 1 x3mop 1 xgluL > 1 xakg 1 xileL !#fghijk #abcde > #abcde #fghijk

IPPMIa_r; Range: 0 to 100; Formula: 2ippm + h2o -> 3c2hmp

IPPMIa_r 1 x2ippm > 1 x3c2hmp !#abcdefg > #abcdefg

IPPMIb_r; Range: 0 to 100; Formula: 3c3hmp -> 2ippm + h2o

IPPMIb_r 1 x3c3hmp > 1 x2ippm !#abcdefg > #abcdefg

LLACt2r_r; Range: 0 to 100; Formula: h + lacL -> he + lacLe

LLACt2r_r 1 xlacL > 1 xlacLe !#abc > #abc

LDHD_r; Range: 0 to 100; Formula: h + nadh + pyr -> lacD + nad

LDHD_r 1 xpyr > 1 xlacD !#abc > #abc

MDH_r; Range: 0 to 100; Formula: h + nadh + oaa -> malL + nad

MDH_r 1 xoaa > 1 xmalL !#abcd > #abcd

MTHFC_r; Range: 0 to 100; Formula: 10fthf -> h2o + methf

MTHFC_r 1 x10fthf > 1 xmethf !#a > #a

MTHFD_r; Range: 0 to 100; Formula: h + methf + nadph -> mlthf + nadp

MTHFD_r 1 xmethf > 1 xmlthf !#a > #a

OCBT_r; Range: 0 to 100; Formula: citrL + h + pi -> cbp + orn

OCBT_r 1 xcitrL > 1 xcbp 1 xorn !#bcdefa > #a #bcdef

PGI_r; Range: 0 to 100; Formula: f6p -> g6p

PGI_r 1 xf6p > 1 xg6p !#abcdef > #abcdef

PGK_r; Range: 0 to 100; Formula: 13dpg + adp -> 3pg + atp

PGK_r 1 x13dpg > 1 x3pg !#abc > #abc

PGM_r; Range: 0 to 100; Formula: 3pg -> 2pg

PGM_r 1 x3pg > 1 x2pg !#abc > #abc

PGMT_r; Range: 0 to 100; Formula: g6p -> g1p

PGMT_r 1 xg6p > 1 xg1p !#abcdef > #abcdef

PHETA1_r; Range: 0 to 100; Formula: gluL + phpyr -> akg + pheL

PHETA1_r 1 xgluL 1 xphpyr > 1 xakg 1 xpheL !#abcde #fghijklmn > #abcde #fghijklmn !#abcde #fghijklmn > #abcde #fghinmlkj !#abcde #fghinmlkj > #abcde #fghijklmn !#abcde #fghinmlkj > #abcde #fghinmlkj

PRPPS_r; Range: 0 to 100; Formula: amp + h + prpp -> atp + r5p

PRPPS_r 1 xprpp > 1 xr5p !#abcde > #abcde

PSCVT_r; Range: 0 to 100; Formula: 3psme + pi -> pep + skm5p

PSCVT_r 1 x3psme > 1 xpep 1 xskm5p !#defghijbac > #abc #defghij

PTAr_r; Range: 0 to 100; Formula: actp + coa -> accoa + pi

PTAr_r 1 xactp > 1 xaccoa !#ab > #ab

PYRt2r_r; Range: 0 to 100; Formula: h + pyr -> he + pyre

PYRt2r_r 1 xpyr > 1 xpyre !#abc > #abc

RPE_r; Range: 0 to 100; Formula: xu5pD -> ru5pD

RPE_r 1 xxu5pD > 1 xru5pD !#abcde > #abcde

RPI_r; Range: 0 to 100; Formula: ru5pD -> r5p

RPI_r 1 xru5pD > 1 xr5p !#abcde > #abcde

SDPTA_r; Range: 0 to 100; Formula: gluL + sl2a6o -> akg + sl26da

SDPTA_r 1 xgluL 1 xsl2a6o > 1 xakg 1 xsl26da !#abcde #fghijklmnop > #abcde #fghijklmnop

SERAT_r; Range: 0 to 100; Formula: acser + coa -> accoa + serL

SERAT_r 1 xacser > 1 xaccoa 1 xserL !#cdeab > #ab #cde

SHK3Dr_r; Range: 0 to 100; Formula: nadp + skm -> 3dhsk + h + nadph

SHK3Dr_r 1 xskm > 1 x3dhsk !#abcdefg > #abcdefg

SUCFUMt_r; Range: 0 to 100; Formula: fum + succe -> succ + fume
SUCFUMt_r 1 xfum 1 xsucce > 1 xfume 1 xsucc !#abcd #efgh > #abcd #efgh !#abcd #efgh > #dcba #efgh !#dcba #efgh > #abcd #efgh !#dcba #efgh > #dcba #efgh !#abcd #hgfe > #abcd #efgh !#abcd #hgfe > #dcba #efgh !#dcba #hgfe > #abcd #efgh !#dcba #hgfe > #dcba #efgh !#abcd #efgh > #abcd #hgfe !#abcd #efgh > #dcba #hgfe !#dcba #efgh > #abcd #hgfe !#dcba #efgh > #dcba #hgfe !#abcd #hgfe > #abcd #hgfe !#abcd #hgfe > #dcba #hgfe !#dcba #hgfe > #abcd #hgfe !#dcba #hgfe > #dcba #hgfe
SUCOAS_r; Range: 0 to 100; Formula: adp + pi + succoa -> atp + coa + succ
SUCOAS_r 1 xsuccoa > 1 xsucc !#abcd > #abcd !#abcd > #dcba
TALA_r; Range: 0 to 100; Formula: e4p + f6p -> g3p + s7p
TALA_r 1 xe4p 1 xf6p > 1 xg3p 1 xs7p !#ghij #defabc > #abc #defghij
THRAr_r; Range: 0 to 100; Formula: acald + gly -> thrL
THRAr_r 1 xacald 1 xgly > 1 xthrL !#cd #ab > #abcd
TKT1_r; Range: 0 to 100; Formula: g3p + s7p -> r5p + xu5pD
TKT1_r 1 xg3p 1 xs7p > 1 xr5p 1 xxu5pD !#hij #fgabcde > #abcde #fghij
TKT2_r; Range: 0 to 100; Formula: f6p + g3p -> e4p + xu5pD
TKT2_r 1 xf6p 1 xg3p > 1 xe4p 1 xxu5pD !#efabcd #ghi > #abcd #efghi
TPI_r; Range: 0 to 100; Formula: g3p -> dhap
TPI_r 1 xg3p > 1 xdhap !#cba > #abc
TRPAS2_r; Range: 0 to 100; Formula: indole + nh4 + pyr -> h2o + trpL
TRPAS2_r 1 xindole 1 xpyr > 1 xtrpL !#kdefghij #abc > #abcdefghijk
TYRTA_r; Range: 0 to 100; Formula: 34hpp + gluL -> akg + tyrL
TYRTA_r 1 x34hpp 1 xgluL > 1 xakg 1 xtyrL !#fghijklmn #abcde > #abcde #fghijklmn !#fghinmlkj #abcde > #abcde #fghijklmn !#fghijklmn #abcde > #abcde #fghinmlkj !#fghinmlkj #abcde > #abcde #fghinmlkj
VALTA_r; Range: 0 to 100; Formula: 3mob + gluL -> akg + valL
VALTA_r 1 x3mob 1 xgluL > 1 xakg 1 xvalL !#fghij #abcde > #abcde #fghij
GHMT2_r; Range: 0 to 100; Formula: gly + h2o + mlthf -> serL + thf
GHMT2_r 1 xgly 1 xmlthf > 1 xserL !#ab #c > #abc

Appendix B

COBRA toolbox v2 additional specifications

B.1 Description of a COBRA compliant SBML file structure

The format of the SBML files used in this work were generated using the standards outlined in http://sbml.org/documents/specifications. COBRA compliant SBML files should contain the following data:

- Unit definitions
- Compartments
- Metabolites (format: M_<metabolite abbreviation>_<compartment id>')
- Metabolite name, compartment, charge, formula
- Reactions (format: R_<reaction abbreviation>')
- Reaction name, reversibility, reaction stoichiometry, gene-protein-reaction (GPR) association, subsystem, E. C. number, lower bound, upper bound, flux value, objective coefficient

Unit definitions are listed in the <listOfUnitDefinitions> section of the SBML file in the following format (example from ecoli_core_model.xml):

<listOfUnitDefintions>
<unitDefinition id=mmol_per_gDW_per_hr>
<listOfUnits>
<unit kind=mole scale=-3/>
<unit kind=gram exponent=1/>
<unit kind=second exponent=-1 multiplier= 0.00027777777777778"/>
</listOfUnits>
</unitDefinition>
</listOfUnitDefinitions>

Compartments are listed in the <listOfCompartments> section of the SBML file in the following format (example from ecoli_core_model.xml):

<listOfCompartments>
<compartment id=c name=Cytoplasm/>
<compartment id=e name=ExtraOrganism/>
</listOfCompartments>

The compartment id should correspond to the compartment abbreviation appended to metabolite names. The full name of the compartment is defined using the name parameter.

Metabolites are listed in the <listOfSpecies> section of the SBML file in the following format (example: acetaldehyde in the cytoplasm):

<listOfSpecies>
...
<species id=M_ACALD_c name=Acetaldehyde compartment=c/>
<notes>
<body xmins=http://www.w3.org/1999/xhtml>
FORMULA: C2H4O
CHARGE: 0
</body>

</notes>
...
</listOfSpecies>

The compartment abbreviation should be appended to the end of the metabolite id. The formula for metabolite is defined in the <notes> section. Note that as the charge parameter was deprecated as of SBML Level 2 Version 2, metabolite charge is now specified in the <notes> section.

Reactions are listed in the <listOfReactions> section of the SBML file in the following format (example: D-lactate dehydrogenase):

<listOfReactions>
...
<reaction id=R_LDH name=D-lactate dehydrogenase>
<notes>
<body xmlns=http://www.w3.org/1999/xhtml>
GENE_ASSOCATION: (b1380) or (b2133)
SUBSYSTEM: Pyruvate Metabolism
EC Number: 1.1.1.28
Confidence level: 0
</body>
</notes>
<listOfReactants>
<speciesReference species=M_lac_DASH_D_c/>
<speciesReference species=M_nad_c/>
</listOfReactants>
<listOfProducts>
<speciesReference species=M_h_c/>
<speciesReference species=M_nadh_c/>
<speciesReference species=M_pyr_c/>
</listOfProducts>
<kineticLaw>

<math xmlns=http://www.w3.org/1998/Math/MathML>
<ci> FLUX_VALUE </ci>
</math>
<listOfParameters>
<parameter id="LOWER_BOUND" value="-1000" units=
"mmol_per_gDW_per_hr"/>
<parameter id="UPPER_BOUND" value="1000" units=
"mmol_per_gDW_per_hr"/>
<parameter id="FLUX_VALUE" value="0" units=
"mmol_per_gDW_per_hr"/>
<parameter id="OBJECTIVE_COEFFICIENT" value="0" units=
"mmol_per_gDW_per_hr"/>
</listOfParameters>
</kineticLaw>
</reaction>
...
</listOfReactions>

The gene-protein-reaction associations and subsystem for a reaction are specified in the <notes> section. Reactants and products for a reaction are listed in <listOfReactants> and <listOfProducts> sections respectively. The lower bound, upper bound, and objective coefficient for a reaction are listed in the <listOfParameters> section within the <kineticLaw> section.

Note that Sid entries in an SBML format file are limited to '0 to 9', 'A to Z', 'a to z', and '_'. To compensate for this, several substitutions are used in place of certain characters. convertSBMLToCobra and convertCobraToSBML are able to handle translating these substitutions back and forth.

B.2 Description of the COBRA toolbox model structure

The model structure contains the following required fields:

- rxns: reaction name abbreviation; reaction ID; order corresponds to S matrix.
- mets: metabolite name abbreviation; metabolite ID; order corresponds to S matrix.
- S: Stoichiometric matrix in sparse format.
- rev: logical array; true for reversible reactions, otherwise false
- lb: lower flux bound for corresponding reactions
- ub: upper flux bound for corresponding reactions
- c: objective coefficient for corresponding reactions
- metCharge: value of charge for corresponding metabolite
- metFormulas: Elemental formula for each metabolite
- rules: Boolean rule for the corresponding reaction which defines gene-reaction relationship.
- genes: List of all genes within the model.
- rxnGeneMat: matrix with rows corresponding to reactions and columns corresponding to genes
- grRules: rules field in a format readable format
- subSystems: subSystem assignment for each reaction
- description: A string describing the model (i.e. model name)

Additional fields which contain supplemental information may also be provided using the following fields:

- rxnNames: Full name of each corresponding reaction
- rxnReferences: Cell array of strings which can contain optional information on references for each specific reaction.
- rxnECNumbers: E. C. number for each reaction
- rxnNotes: Cell array of strings which can contain optional information for each specific reaction.
- confidenceScores: Confidence score for each reaction
- proteins: proteins associated with each reaction
- metNames: Full name of each corresponding metabolite
- metChEBIID: ChEBI ID for each corresponding metabolite
- metKeggID: KEGG ID for each corresponding metabolite
- metPubChemID: Pub Chem ID for each corresponding metabolite
- metInchiString: Inichi String for each corresponding metabolite

B.3 Description of the COBRA format for C13 tracing

B.3.1 Description of Network

The matlab structure must contain separate forward and reverse reactions for every Carbon traced reaction. The Carbon information is stored in the following additional field:

.isotopomer: a vector of strings of length n (number of reactions). For each entry, if the corresponding reaction is carbon tracked, .isotopomer must contain the following string:

Unique_label <space> coefficient/metabolite pairs describing reactants '>' coefficient/metabolite pairs describing products '!' corresponding carbon mapping indicated with '#'.

If there are multiple possible mappings due to compound symmetry, another '!' may follow. All possible mappings are treated with equal probability.

Example: .isotopomeri = 'CYSS 1 xacser > 1 xac 1 xcysL!#abcde > #de #abc'

B.3.2 Experimental data

Each experiment is stored as a structure comprising the following fields.

- .input vector of length $2\hat{}$(#carbons in input) corresponds to cumumer distribution of isotopomers. If only Isotopomer distribution is available then function iso2cdv(input, carbons) will convert to cumomer.
- .inputfrag generated automatically from .input. Contains distribution of C13 in input compound fragments. This field is used by the EMU method.
- .fragments A structure containing the actual measured data. Each field in the structure is one fragment and is identified by a .fragment_name. These fragments contain fields:
  - .met string of metabolite measured
  - .fragment binary vector of which carbons were measured. A 1 indicates inclusion in a fragment, 0 exclusion.
  - .data A vector of length #carbons+1 corresponding to the fraction of carbon labeled 0, 1, #carbons times.
  - .metfrag a string comprising the metabolite name (.met) and .fragment concatenated as a string.
- .std2 The standard deviation of experimental measurements. Used as a scaling factor for goodness of fit testing.

Bibliography

[1] J. L. Reed, I. Famili, I. Thiele, and B. O. Palsson. Towards multidimensional genome annotation. *Nat Rev Genet*, 7(2):130–41, 2006.

[2] Hyun Uk Kim, Tae Yong Kim, and Sang Yup Lee. Metabolic flux analysis and metabolic engineering of microorganisms. *Molecular BioSystems*, 4(2):113–120, 2008.

[3] N. D. Price, J. L. Reed, and B.O. Palsson. Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nat Rev Microbiol*, 2(11):886–897, 2004.

[4] B.O. Palsson. *Systems biology: properties of reconstructed networks*. Cambridge University Press, New York, 2006.

[5] N. D. Price, I. Thiele, and Palsson B.O. Candidate states of helicobacter pylori's genome-scale metabolic network upon application of loop law thermodynamic constraints. *Biophys J*, 90:3919–28, 2006.

[6] A. Varma and B. O. Palsson. Metabolic capabilities of escherichia coli: I. synthesis of biosynthetic precursors and cofactors. *Journal of Theoretical Biology*, 165(4):477–502, 1993.

[7] J. M. Savinell and B. O. Palsson. Optimal selection of metabolic fluxes for in vivo measurement. i. development of mathematical methods. *Journal of Theoretical Biology*, 155(2):201–14, 1992.

[8] J. M. Savinell and B. O. Palsson. Optimal selection of metabolic fluxes for in vivo measurement. ii. application to escherichia coli and hybridoma cell metabolism. *Journal of Theoretical Biology*, 155(2):215–42, 1992.

[9] S. J. Wiback, I. Famili, H.J. Greenberg, and B.O. Palsson. Monte carlo sampling can be used to determine the size and shape of the steady state flux space. *Journal of Theoretical Biology*, 228(4):437–447, 2004.

[10] A. Braunstein, R. Mulet, and A. Pagnani. Estimating the size of the solution space of metabolic networks. *BMC Bioinformatics*, 9(1):240, 2008.

[11] N. D. Price, J. Schellenberger, and B.O. Palsson. Uniform sampling of steady state flux spaces: Means to design experiments and to interpret enzymopathies. *Biophysical Journal*, 87(4):2172–86, 2004.

[12] C.H. Schilling. *On Systems Biology and the Pathway Analysis of Metabolic Networks*. Ph.d., University of California, San Diego, 2000.

[13] 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.

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

[15] D.E. Kaufman and R.L. Smith. Direction choice for accelerated convergence in hit-and-run sampling. *Operations Research*, 46(1):84–96, 1998.

[16] E. Almaas, B. Kovacs, T. Vicsek, Z. N. Oltvai, and A. L. Barabasi. Global organization of metabolic fluxes in the bacterium escherichia coli. *Nature*, 427(6977):839–843, 2004.

[17] E. Almaas, Z. N. Oltvai, and A. L. Barabasi. The activity reaction core and plasticity of metabolic networks. *PLoS Comput Biol*, 1(7):e68, 2005.

[18] Jason A. Papin and Bernhard O. Palsson. The jak-stat signaling network in the human b-cell: An extreme signaling pathway analysis. *Biophysical Journal*, 87(1):37–46, 2004.

[19] N. Jamshidi and B. O. Palsson. Investigating the metabolic capabilities of mycobacterium tuberculosis h37rv using the in silico strain inj661 and proposing alternative drug targets. *BMC Syst Biol*, 1:26, 2007.

[20] Christian L. Barrett, Tae Yong Kim, Hyun Uk Kim, Bernhard O. Palsson, and Sang Yup Lee. Systems biology as a foundation for genome-scale synthetic biology. *Curr Opin Biotechnol*, 17(5):488–492, 2006.

[21] I. Thiele, N. D. Price, T. D. Vo, and B. O. Palsson. Candidate metabolic network states in human mitochondria: Impact of diabetes, ischemia, and diet. *J Biol Chem*, 280(12):11683–95, 2005.

[22] N. Jamshidi, S.J. Wiback, and B. O. Palsson. In silico model-driven assessment of the effects of single nucleotide polymorphisms (snps) on human red blood cell metabolism. *Genome Research*, 12(11):1687–1692, 2002.

[23] R. Occhipinti, M. A. Puchowicz, J. C. LaManna, E. Somersalo, and D. Calvetti. Statistical analysis of metabolic pathways of brain metabolism at steady state. *Ann Biomed Eng*, 35(6):886–902, 2007.

[24] Erkki Somersalo Daniela Calvetti. Large-scale statistical parameter estimation in complex systems wtih an application to metabolic models. *SIAM Model. Simul.*, 5(4):1333–1366, 2006.

[25] B. Teusink, J. Passarge, C. A. Reijenga, E. Esgalhado, C. C. van der Weijden, M. Schepper, M. C. Walsh, B. M. Bakker, K. van Dam, H. V. Westerhoff, and J. L. Snoep. Can yeast glycolysis be understood in terms of in vitro kinetics of the constituent enzymes? testing biochemistry. *Eur J Biochem*, 267(17):5313–29., 2000.

[26] I. Famili, R. Mahadevan, and B. O. Palsson. k-cone analysis: Determining all candidate values for kinetic parameters on a network scale. *Biophys J*, 88(3):1616–25, 2005.

[27] R. Steuer, T. Gross, J. Selbig, and B. Blasius. Structural kinetic modeling of metabolic networks. *Proc Natl Acad Sci U S A*, 103(32):11868–73, 2006.

[28] R. Steuer. Computational approaches to the topology, stability and dynamics of metabolic networks. *Phytochemistry*, 68(16-18):2139–51, 2007.

[29] S. Grimbs, J. Selbig, S. Bulik, H. G. Holzhutter, and R. Steuer. The stability and robustness of metabolic states: identifying stabilizing sites in metabolic networks. *Mol Syst Biol*, 3:146, 2007.

[30] W. Liebermeister and E. Klipp. Biochemical networks with uncertain parameters. *IEE Systems Biology*, 152(3):97–107, 2005.

[31] S.A. Becker, A.M. Feist, M.L. Mo, G. Hannum, B.O. Palsson, and M.J. Herrgard. Quantitative prediction of cellular metabolism with constraint-based models: The cobra toolbox. *Nat. Protocols*, 2(3):727–738, 2007.

[32] E. P. Gianchandani, J. A. Papin, N. D. Price, A. R. Joyce, and B. O. Palsson. Matrix formalism to describe functional states of transcriptional regulatory systems. *PLoS Comput Biol.*, 2(8):e101, 2006.

[33] M. W. Covert, E. M. Knight, J. L. Reed, M. J. Herrgard, and B. O. Palsson. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92–6, 2004.

[34] E. Gianchandani, Andrew R. Joyce, Bernhard . Palsson, and Jason A. Papin. Functional states of the escherichia coli transcriptional regulatory system at the genome-scale. *submitted*, 2008.

[35] A. Kmmel, S. Panke, and M. Heinemann. Putative regulatory sites unraveled by network-embedded thermodynamic analysis of metabolome data. *Mol Syst Biol*, 2:2006.0034, 2006.

[36] Smith RL. Kaufmann DE. Direction choice for accelerated convergence in hit-and-run sampling. *Operations Research*, 46:84–95, 1998.

[37] I. Thiele, T. D. Vo, N. D. Price, and B. Palsson. An expanded metabolic reconstruction of helicobacter pylori (iit341 gsm/gpr): An in silico genome-scale characterization of single and double deletion mutants. *J Bacteriol.*, 187(16):5818–5830, 2005.

[38] J. Schellenberger, J. O. Park, T. M. Conrad, and B. O. Palsson. Bigg: a biochemical genetic and genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinformatics*, 11(1):213, 2010.

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

[40] J.L. Reed, T.D. Vo, C. H. Schilling, and B.O. Palsson. An expanded genome-scale model of escherichia coli k-12 (ijr904 gsm/gpr). *Genome Biology*, 4(9):R54.1–R54.12, 2003.

[41] A. M. Feist and B. O. Palsson. The biomass objective function. *Curr Opin Microbiol*, 2010.

[42] J.S. Edwards and B.O. Palsson. The escherichia coli mg1655 in silico metabolic genotype: Its definition, characteristics, and capabilities. *Proc Natl Acad Sci U S A.*, 97(10):5528–5533, 2000.

[43] Q. Hua, A. R. Joyce, S. S. Fong, and B. O. Palsson. Metabolic analysis of adaptive evolution for in silico designed lactate-producing strains. *Biotechnol Bioeng*, 95(5):992–1002, 2006.

[44] A. P. Burgard, P. Pharkya, and C. D. Maranas. Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnol Bioeng*, 84(6):647–57, 2003.

[45] A. M. Feist, D. C. Zielinski, J. D. Orth, J. Schellenberger, M. J. Herrgard, and B. O. Palsson. Model-driven evaluation of the production potential for growth-coupled products of escherichia coli. *Metab Eng*, 2009.

[46] A. M. Feist and B.O. Palsson. The growing scope of applications of genome-scale metabolic reconstructions using escherichia coli. *Nat Biotech*, 26(6):659–667, 2008.

[47] M. Mollney, W. Wiechert, D. Kownatzki, and A. A. de Graaf. Bidirectional reaction steps in metabolic networks: Iv. optimal design of isotopomer labeling experiments. *Biotechnology and Bioengineering*, 66(2):86–103, 1999.

[48] K. Schmidt, M. Carlsen, J. Nielsen, and J. Villadsen. Modeling isotopomer distributions in biochemical reaction networks using isotopomer mapping matrices. *Biotechnology and Bioengineering*, 55(6):831–40, 1997.

[49] W. Wiechert. 13c metabolic flux analysis. *Metab Eng*, 3(3):195–206, 2001.

[50] P. F. Suthers, A. P. Burgard, M. S. Dasika, F. Nowroozi, S. Van Dien, J. D. Keasling, and C. D. Maranas. Metabolic flux elucidation for large-scale models using 13c labeled isotopes. *Metab Eng*, 9(5-6):387–405, 2007.

[51] N. Zamboni, S. M. Fendt, M. Ruhl, and U. Sauer. (13)c-based metabolic flux analysis. *Nat Protoc*, 4(6):878–92, 2009.

[52] C. M. Metallo, J. L. Walther, and G. Stephanopoulos. Evaluation of 13c isotopic tracers for metabolic flux analysis in mammalian cells. *J Biotechnol*, 144(3):167–74, 2009.

[53] E. Fischer, N. Zamboni, and U. Sauer. High-throughput metabolic flux analysis based on gas chromatography-mass spectrometry derived 13c constraints. *Anal Biochem*, 325(2):308–16, 2004.

[54] J. Zhao and K. Shimizu. Metabolic flux analysis of escherichia coli k12 grown on 13c-labeled acetate and glucose using gc-ms and powerful flux calculation method. *J Biotechnol*, 101(2):101–17, 2003.

[55] J. S. Edwards and B. O. Palsson. Multiple steady states in kinetic models of red cell metabolism. *Journal of Theoretical Biology*, 207(1):125–7, 2000.

[56] W. Wiechert, M. Mollney, N. Isermann, M. Wurzel, and A. A. de Graaf. Bidirectional reaction steps in metabolic networks: Iii. explicit solution and analysis of isotopomer labeling systems. *Biotechnol Bioeng*, 66(2):69–85, 1999.

[57] M. R. Antoniewicz, J. K. Kelleher, and G. Stephanopoulos. Elementary metabolite units (emu): a novel framework for modeling isotopic distributions. *Metab Eng*, 9(1):68–86, 2007.

[58] A. Nanchen, T. Fuhrer, and U. Sauer. Determination of metabolic flux ratios from 13c-experiments and gas chromatography-mass spectrometry data: protocol and principles. *Methods Mol Biol*, 358:177–97, 2007.

[59] W. A. van Winden, C. Wittmann, E. Heinzle, and J. J. Heijnen. Correcting mass isotopomer distributions for naturally occurring isotopes. *Biotechnol Bioeng*, 80(4):477–9, 2002.

[60] M. Durot, P. Y. Bourguignon, and V. Schachter. Genome-scale models of bacterial metabolism: reconstruction and applications. *FEMS Microbiol Rev*, 33(1):164–90, 2009.

[61] A. M. Feist, M. J. Herrgard, I. Thiele, J. L. Reed, and B. O. Palsson. Reconstruction of biochemical networks in microorganisms. *Nat Rev Microbiol*, 7(2):129–43, 2009.

[62] M. A. Oberhardt, B. O. Palsson, and J. A. Papin. Applications of genome-scale metabolic reconstructions. *Mol Syst Biol*, 5:320, 2009.

[63] B. Palsson. Metabolic systems biology. *FEBS Lett*, 583(24):3900–4, 2009.

[64] A. Varma, B. W. Boesch, and B. O. Palsson. Biochemical production capabilities of escherichia coli. *Biotechnology and Bioengineering*, 42(1):59–73, 1993.

[65] R. Schuetz, L. Kuepfer, and U. Sauer. Systematic evaluation of objective functions for predicting intracellular fluxes in escherichia coli. *Mol Syst Biol*, 3(119), 2007.

[66] P. Gianchandani Erwin, K. Chavali Arvind, and A. Papin Jason. The application of flux balance analysis in systems biology.

[67] A. Varma and B. O. Palsson. Metabolic flux balancing: Basic concepts, scientific and practical use. *Nat Biotechnol*, 12:994–998, 1994.

[68] J. D. Orth, I. Thiele, and B. O. Palsson. What is flux balance analysis? *Nat Biotechnol*, 28(3):245–8, 2010.

[69] J. Pramanik and J. D. Keasling. Stoichiometric model of escherichia coli metabolism: Incorporation of growth-rate dependent biomass composition and mechanistic energy requirements. *Biotechnology and Bioengineering*, 56(4):398–421, 1997.

[70] N. D. Price, I. Famili, D. A. Beard, and B. O. Palsson. Extreme pathways and kirchhoff's second law. *Biophys J*, 83(5):2879–82, 2002.

[71] D. A. Beard, E. Babson, E. Curtis, and H. Qian. Thermodynamic constraints for biochemical networks. *J Theor Biol*, 228(3):327–33, 2004.

[72] A. Kmmel, S. Panke, and M. Heinemann. Systematic assignment of thermodynamic constraints in metabolic network models. *BMC Bioinformatics*, 7(512), 2006.

[73] C. S. Henry, L. J. Broadbelt, and V. Hatzimanikatis. Thermodynamics-based metabolic flux analysis. *Biophys. J.*, 92(5):1792–1805, 2007.

[74] H. Qian, D. A. Beard, and S. D. Liang. Stoichiometric network theory for nonequilibrium biochemical systems. *Eur J Biochem*, 270(3):415–21, 2003.

[75] R. M. Fleming, I. Thiele, G. Provan, and H. P. Nasheuer. Integrated stoichiometric, thermodynamic and kinetic modelling of steady state metabolism. *J Theor Biol*, 264(3):683–92, 2010.

[76] P.J. Linstrom Mallard and W.G. Nist chemistry webbook, nist standard reference database number 69, 2010.

[77] M. L. Mavrovouniotis. Estimation of standard gibbs energy changes of biotransformations. *J Biol Chem*, 266(22):14440–5, 1991.

[78] R. A. Alberty. Calculation of standard transformed gibbs energies and standard transformed enthalpies of biochemical reactants. *Arch Biochem Biophys*, 353(1):116–30, 1998.

[79] S Schuster, C Hilgetag, JH Woods, and DA Fell. Elementary modes of functioning in biochemical networks. In R Cuthbertson, M Holcombe, and R Paton, editors, *Computation in Cellular and Molecular Biological Systems*, pages 151–165. World Scientific, London, 1996.

[80] J. D. Orth, R.M.T. Fleming, and B. . Palsson. Reconstruction and use of microbial metabolic networks: the core escherichia coli metabolic model as an educational guide. In P. D. Karp, editor, *EcoSal - Escherichia coli and Salmonelal Cellular and Molecular Biology*. ASM Press, Washington D.C., 2009.

[81] S. A. Becker and B. O. Palsson. Genome-scale reconstruction of the metabolic network in staphylococcus aureus n315: an initial draft to the two-dimensional annotation. *BMC Microbiol*, 5(1):8, 2005.

[82] A. M. Feist, C. S. Henry, J. L. Reed, M. Krummenacker, A. R. Joyce, P. D. Karp, L. J. Broadbelt, V. Hatzimanikatis, and B. O. Palsson. A genome-scale metabolic reconstruction for escherichia coli k-12 mg1655 that accounts for 1260 orfs and thermodynamic information. *Mol Syst Biol*, 3(121), 2007.

[83] N. C. Duarte, S. A. Becker, N. Jamshidi, I. Thiele, M. L. Mo, T. D. Vo, R. Srivas, and B. O. Palsson. Global reconstruction of the human metabolic network based on genomic and bibliomic data. *Proc Natl Acad Sci U S A*, 104(6):1777–82, 2007.

[84] S. M. Keating, B. J. Bornstein, A. Finney, and M. Hucka. Sbmltoolbox: an sbml toolbox for matlab users. *Bioinformatics*, 22(10):1275–7, 2006.

[85] M. L. Mavrovouniotis. Analysis of complex metabolic pathways. In Collado-Vides, J. B. Magasanik, and T. F. Smith, editors, *Integrative approaches to molecular biology*, pages 211–238. MIT Press, Cambridge, Massachusetts, USA; London, England, UK, 1996.

[86] C. S. Henry, M. D. Jankowski, L. J. Broadbelt, and V. Hatzimanikatis. Genome-scale thermodynamic analysis of escherichia coli metabolism. *Biophys J*, 90(4):1453–61, 2006.

[87] R. M. Fleming, I. Thiele, and H. P. Nasheuer. Quantitative assignment of reaction directionality in constraint-based models of metabolism: application to escherichia coli. *Biophys Chem*, 145(2-3):47–56, 2009.

[88] D. Segre, D. Vitkup, and G. M. Church. Analysis of optimality in natural and perturbed metabolic networks. *Proc Natl Acad Sci U S A*, 99(23):15112–7, 2002.

[89] T. Shlomi, O. Berkman, and E. Ruppin. Regulatory on/off minimization of metabolic flux changes after genetic perturbations. *Proc Natl Acad Sci U S A*, 102(21):7695–700, 2005.

[90] M. W. Covert, C. H. Schilling, and B. Palsson. Regulation of gene expression in flux balance models of metabolism. *Journal of Theoretical Biology*, 213(1):73–88, 2001.

[91] A. P. Burgard, E. V. Nikolaev, C. H. Schilling, and C. D. Maranas. Flux coupling analysis of genome-scale metabolic network reconstructions. *Genome. Res.*, 14(2):301–12, 2004.

[92] K. Smallbone and E. Simeonidis. Flux balance analysis: a geometric perspective. *J Theor Biol*, 258(2):311–5, 2009.

[93] Irwin H. Segel. *Enzyme kinetics : behavior and analysis of rapid equilibrium and steady-state enzyme systems*. Wiley, New York, 1975.

[94] R. Heinrich and T. A. Rapoport. Linear theory of enzymatic chains; its application for the analysis of the crossover theorem and of the glycolysis of human erythrocytes. *Acta Biol Med Ger*, 31(4):479–94, 1973.

[95] B. E. Wright and G. L. Gustafson. Expansion of the kinetic model of differentiation in dictyostelium discoideum. *J Biol Chem*, 247(24):7875–84, 1972.

[96] A. Werner and R. Heinrich. A kinetic model for the interaction of energy metabolism and osmotic states of human erythrocytes. analysis of the stationary "in vivo" state and of time dependent variations under blood preservation conditions. *Biomed Biochim Acta*, 44(2):185–212, 1985.

[97] N. Le Novere, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, J. L. Snoep, and M. Hucka. Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res*, 34(Database issue):D689–91, 2006.

[98] M. Tomita, K. Hashimoto, K. Takahashi, T. S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J. C. Venter, and C. A. III Hutchison. E-cell: software environment for whole-cell simulation. *Bioinformatics*, 15(1):72–84, 1999.

[99] M. Ander, P. Beltrao, B. Di Ventura, J. Ferkinghoff-Borg, M. Foglierini, A. Kaplan, C. Lemerle, I. Tomas-Oliveira, and L. Serrano. Smartcell, a framework to simulate cellular processes that combines stochastic approximation with diffusion and localisation: analysis of simple networks. *Syst Biol (Stevenage)*, 1(1):129–38, 2004.

[100] Y. Xia, H. Yu, R. Jansen, M. Seringhaus, S. Baxter, D. Greenbaum, H. Zhao, and M. Gerstein. Analyzing cellular biochemistry in terms of molecular networks. *Annu Rev Biochem*, 73:1051–87, 2004.

[101] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, 27(1):29–34, 1999.

[102] P. D. Karp, C. A. Ouzounis, C. Moore-Kochlacs, L. Goldovsky, P. Kaipa, D. Ahren, S. Tsoka, N. Darzentas, V. Kunin, and N. Lopez-Bigas. Expansion of the biocyc collection of pathway/genome databases to 160 genomes. *Nucleic Acids Res*, 33(19):6083–9, 2005.

[103] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. R. Gopinath, G. R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Res*, 33(Database issue):D428–32, 2005.

[104] J. S. Edwards and B. O. Palsson. Systems properties of the haemophilus influenzae rd metabolic genotype. *Journal of Biological Chemistry*, 274(25):17410–6, 1999.

[105] C. Hodgman, I. Goryanin, and N. Juty. Reconstructing whole-cell models. *Drug Discovery Today*, 6(15):S109–S112, 2001.

[106] A. P. Oliveira, J. Nielsen, and J. Forster. Modeling lactococcus lactis using a genome-scale flux model. *BMC Microbiol*, 5:39, 2005.

[107] M. A. Oberhardt, J. Puchalka, K. E. Fryer, V. A. Martins dos Santos, and J. A. Papin. Genome-scale metabolic network analysis of the opportunistic pathogen pseudomonas aeruginosa pao1. *J Bacteriol*, 190(8):2790–803, 2008.

[108] O. Gonzalez, S. Gronau, M. Falb, F. Pfeiffer, E. Mendoza, R. Zimmer, and D. Oesterhelt. Reconstruction, modeling & analysis of halobacterium salinarum r-1 metabolism. *Mol Biosyst*, 4(2):148–59, 2008.

[109] H. David, I. S. Ozcelik, G. Hofmann, and J. Nielsen. Analysis of aspergillus nidulans metabolism at the genome-scale. *BMC Genomics*, 9:163, 2008.

[110] D. Maglott, J. Ostell, K. D. Pruitt, and T. Tatusova. Entrez gene: gene-centered information at ncbi. *Nucleic Acids Res*, 33(Database issue):D54–8, 2005.

[111] P. D. Karp, S. Paley, and P. Romero. The pathway tools software. *Bioinformatics*, 18 Suppl 1:S225–32, 2002.

[112] R. Overbeek, T. Begley, R. M. Butler, J. V. Choudhuri, H. Y. Chuang, M. Cohoon, V. de Crecy-Lagard, N. Diaz, T. Disz, R. Edwards, M. Fonstein, E. D. Frank, S. Gerdes, E. M. Glass, A. Goesmann, A. Hanson, D. Iwata-Reuyl, R. Jensen, N. Jamshidi, L. Krause, M. Kubal, N. Larsen, B. Linke, A. C. McHardy, F. Meyer, H. Neuweger, G. Olsen, R. Olson, A. Osterman, V. Portnoy, G. D. Pusch, D. A. Rodionov, C. Ruckert, J. Steiner, R. Stevens, I. Thiele, O. Vassieva, Y. Ye, O. Zagnitko, and V. Vonstein. The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic Acids Res*, 33(17):5691–702, 2005.

[113] M. DeJongh, K. Formsma, P. Boillot, J. Gould, M. Rycenga, and A. Best. Toward the automated generation of genome-scale metabolic networks in the seed. *BMC Bioinformatics*, 8:139, 2007.

[114] R. K. Aziz, D. Bartels, A. A. Best, M. Dejongh, T. Disz, R. A. Edwards, K. Formsma, S. Gerdes, E. M. Glass, M. Kubal, F. Meyer, G. J. Olsen, R. Olson, A. L. Osterman, R. A. Overbeek, L. K. McNeil, D. Paarmann, T. Paczian, B. Parrello, G. D. Pusch, C. Reich, R. Stevens, O. Vassieva, V. Vonstein, A. Wilke, and O. Zagnitko. The rast server: Rapid annotations using subsystems technology. *BMC Genomics*, 9(1):75, 2008.

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

[116] Richard A Notebaart, Frank HJ van Enckevort, Christof Francke, Roland J Siezen, and Bas Teusink. Accelerating the reconstruction of genome-scale metabolic networks. *BMC Bioinformatics*, 7(1):296, 2006.

[117] M. J. Herrgard, N. Swainston, P. Dobson, W. B. Dunn, K. Y. Arga, M. Arvas, N. Bluthgen, S. Borger, R. Costenoble, M. Heinemann, M. Hucka, N. Le Novere, P. Li, W. Liebermeister, M. L. Mo, A. P. Oliveira, D. Petranovic, S. Pettifer, E. Simeonidis, K. Smallbone, I. Spasic, D. Weichart, R. Brent, D. S. Broomhead, H. V. Westerhoff, B. Kirdar, M. Penttila, E. Klipp, B. O. Palsson, U. Sauer, S. G. Oliver, P. Mendes, J. Nielsen, and D. B. Kell. A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology. *Nat Biotechnol*, 26(10):1155–60, 2008.

[118] N. C. Duarte, B. Bo Palsson, and P. Fu. Integrated analysis of metabolic phenotypes in saccharomyces cerevisiae. *BMC Genomics*, 5(1):63, 2004.

[119] A. M. Feist, J. C. M. Scholten, B. O. Palsson, F. J. Brockman, and T. Ideker. Modeling methanogenesis with a genome-scale metabolic reconstruction of methanosarcina barkeri. *Mol Syst Biol*, 2(2006.0004):1–14, 2006.

[120] Y. Fujii, T. Imanishi, and T. Gojobori. [h-invitational database: integrated database of human genes]. *Tanpakushitsu Kakusan Koso*, 49(11 Suppl):1937–43, 2004.

[121] A.R. Joyce, S.S. Fong, and B.O. Palsson. Adaptive evolution of e. coli on either lactate or glycerol leads to convergent, generalist phenotypes. In *International E. Coli Alliance Second Annual Meeting*, Banff, Alberta, 2004.

[122] S. S. Fong and B. O. Palsson. Metabolic gene-deletion strains of escherichia coli evolve to computationally predicted growth phenotypes. *Nat Genet*, 36(10):1056–58, 2004.

[123] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic Acids Res*, 32(Database issue):D431–3, 2004.

[124] A. Bairoch, R. Apweiler, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi, and L. S. Yeh. The universal protein resource (uniprot). *Nucleic Acids Res*, 33(Database issue):D154–9, 2005.

[125] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, II Goryanin, W. J. Hedley, T. C. Hodgman, J. H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch,

E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–31, 2003.

[126] J. Wright and A. Wagner. The systems biology research tool: evolvable open-source software. *BMC Syst Biol*, 2:55, 2008.

[127] M. G. Poolman, B. K. Bonde, A. Gevorgyan, H. H. Patel, and D. A. Fell. Challenges to be faced in the reconstruction of metabolic networks from public databases. *Syst Biol (Stevenage)*, 153(5):379–84, 2006.

[128] A. M. Feist, J. D. Orth, D. C. Zielinski, and B. O. Palsson. Construction and evolution of e. coli production strains designed through a model-driven analysis. *In preparation*, 2008.

[129] M. P. Brynildsen, W. W. Wong, and J. C. Liao. Transcriptional regulation and metabolism. *Biochem Soc Trans*, 33(Pt 6):1423–6, 2005.

[130] J. A. Papin, T. Hunter, B. O. Palsson, and S. Subramaniam. Reconstruction of cellular signalling networks and analysis of their properties. *Nat Rev Mol Cell Biol*, 6(2):99–111, 2005.

[131] D. R. Hyduke and B. O. Palsson. Towards genome-scale signalling-network reconstructions. *Nat Rev Genet*, 11(4):297–307, 2010.

[132] R. A. Notebaart, B. Teusink, R. J. Siezen, and B. Papp. Co-regulation of metabolic genes is better explained by flux coupling than by network distance. *PLoS Comput Biol*, 4(1):e26, 2008.

[133] K. Raman, K. Yeturu, and N. Chandra. targettb: a target identification pipeline for mycobacterium tuberculosis through an interactome, reactome and genome-scale structural analysis. *BMC Syst Biol*, 2:109, 2008.

[134] A. Vazquez, Q. K. Beg, M. A. Demenezes, J. Ernst, Z. Bar-Joseph, A. L. Barabasi, L. G. Boros, and Z. N. Oltvai. Impact of the solvent capacity constraint on e. coli metabolism. *BMC Syst Biol*, 2:7, 2008.

[135] S. A. Becker and B. O. Palsson. Context-specific metabolic networks are consistent with experiments. *PLoS Comput Biol*, 4(5):e1000082, 2008.

[136] T. Shlomi, M. N. Cabili, M. J. Herrgard, B. O. Palsson, and E. Ruppin. Network-based prediction of human tissue-specific metabolism. *Nat Biotechnol*, 26(9):1003–10, 2008.

[137] J. Schellenberger and B. O. Palsson. Use of randomized sampling for analysis of metabolic networks. *J Biol Chem*, 284(9):5457–61, 2009.

[138] E.P. Gianchandani, A.K. Chavali, and J. A. Papin. The application of flux balance analysis in systems biology. *Systems Biology and Medicine*, 2009.

[139] Soon Ho Hong, Jin Sik Kim, Sang Yup Lee, Yong Ho In, Sun Shim Choi, Jeong-Keun Rih, Chang Hoon Kim, Haeyoung Jeong, Cheol Goo Hur, and Jae Jong Kim. The genome sequence of the capnophilic rumen bacterium mannheimia succiniciproducens. *Nat Biotech*, 22(10):1275–1281, 2004.

[140] M. L. Mo, B. O. Palsson, and M. J. Herrgard. Connecting extracellular metabolomic measurements to intracellular flux states in yeast. *BMC Syst Biol*, 3:37, 2009.

[141] J. Nogales, B. O. Palsson, and I. Thiele. A genome-scale metabolic reconstruction of pseudomonas putida kt2440: ijn746 as a cell factory. *BMC Syst Biol*, 2:79, 2008.

[142] J. L. Reed and B. O. Palsson. Thirteen years of building constraint-based in silico models of escherichia coli. *J Bacteriol*, 185(9):2692–9, 2003.

[143] V. Satish Kumar, M. S. Dasika, and C. D. Maranas. Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics*, 8:212, 2007.

[144] A. P. Burgard and C. D. Maranas. Optimization-based framework for inferring and testing hypothesized metabolic objective functions. *Biotechnol Bioeng*, 82(6):670–7, 2003.

[145] K. R. Patil and J. Nielsen. Uncovering transcriptional regulation of metabolism by using metabolic network topology. *Proc Natl Acad Sci U S A*, 102(8):2685–9, 2005.

[146] 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. *Mol Syst Biol*, 5:296, 2009.

[147] J. L. Reed, T. R. Patel, K. H. Chen, A. R. Joyce, M. K. Applebee, C. D. Herring, O. T. Bui, E. M. Knight, S. S. Fong, and B. O. Palsson. Systems approach to refining genome annotation. *Proc Natl Acad Sci U S A*, 103(46):17480–4, 2006.

[148] B. J. Bornstein, S. M. Keating, A. Jouraku, and M. Hucka. Libsbml: an api library for sbml. *Bioinformatics*, 24(6):880–1, 2008.

[149] R. Mahadevan and C. H. Schilling. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metab Eng*, 5(4):264–76, 2003.

[150] S. Subramaniam. The biology workbench–a seamless database and analysis environment for the biologist [editorial]. *Proteins*, 32(1):1–2, 1998.

[151] H. M. Sauro, M. Hucka, A. Finney, C. Wellock, H. Bolouri, J. Doyle, and H. Kitano. Next generation simulation tools: the systems biology workbench and biospice integration. *Omics*, 7(4):355–72, 2003.